

Advances in Intelligent Systems and Computing 630

Wilfried Lopuschitz

Munir Merdan

Gottfried Koppensteiner

Richard Balogh

David Obdržálek *Editors*

# Robotics in Education

Latest Results and Developments

 Springer

# **Advances in Intelligent Systems and Computing**

Volume 630

## **Series editor**

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland  
e-mail: [kacprzyk@ibspan.waw.pl](mailto:kacprzyk@ibspan.waw.pl)

### *About this Series*

The series “Advances in Intelligent Systems and Computing” contains publications on theory, applications, and design methods of Intelligent Systems and Intelligent Computing. Virtually all disciplines such as engineering, natural sciences, computer and information science, ICT, economics, business, e-commerce, environment, healthcare, life science are covered. The list of topics spans all the areas of modern intelligent systems and computing.

The publications within “Advances in Intelligent Systems and Computing” are primarily textbooks and proceedings of important conferences, symposia and congresses. They cover significant recent developments in the field, both of a foundational and applicable character. An important characteristic feature of the series is the short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

### *Advisory Board*

#### Chairman

Nikhil R. Pal, Indian Statistical Institute, Kolkata, India

e-mail: [nikhil@isical.ac.in](mailto:nikhil@isical.ac.in)

#### Members

Rafael Bello Perez, Universidad Central “Marta Abreu” de Las Villas, Santa Clara, Cuba

e-mail: [rbellop@uclv.edu.cu](mailto:rbellop@uclv.edu.cu)

Emilio S. Corchado, University of Salamanca, Salamanca, Spain

e-mail: [escorchado@usal.es](mailto:escorchado@usal.es)

Hani Hagrass, University of Essex, Colchester, UK

e-mail: [hani@essex.ac.uk](mailto:hani@essex.ac.uk)

László T. Kóczy, Széchenyi István University, Győr, Hungary

e-mail: [koczy@sze.hu](mailto:koczy@sze.hu)

Vladik Kreinovich, University of Texas at El Paso, El Paso, USA

e-mail: [vladik@utep.edu](mailto:vladik@utep.edu)

Chin-Teng Lin, National Chiao Tung University, Hsinchu, Taiwan

e-mail: [ctlin@mail.nctu.edu.tw](mailto:ctlin@mail.nctu.edu.tw)

Jie Lu, University of Technology, Sydney, Australia

e-mail: [Jie.Lu@uts.edu.au](mailto:Jie.Lu@uts.edu.au)

Patricia Melin, Tijuana Institute of Technology, Tijuana, Mexico

e-mail: [epmelin@hafsamx.org](mailto:epmelin@hafsamx.org)

Nadia Nedjah, State University of Rio de Janeiro, Rio de Janeiro, Brazil

e-mail: [nadia@eng.uerj.br](mailto:nadia@eng.uerj.br)

Ngoc Thanh Nguyen, Wroclaw University of Technology, Wroclaw, Poland

e-mail: [Ngoc-Thanh.Nguyen@pwr.edu.pl](mailto:Ngoc-Thanh.Nguyen@pwr.edu.pl)

Jun Wang, The Chinese University of Hong Kong, Shatin, Hong Kong

e-mail: [jwang@mae.cuhk.edu.hk](mailto:jwang@mae.cuhk.edu.hk)

More information about this series at <http://www.springer.com/series/11156>

Wilfried Lopuschitz · Munir Merdan  
Gottfried Koppensteiner · Richard Balogh  
David Obdržálek  
Editors

# Robotics in Education

Latest Results and Developments

 Springer

*Editors*

Wilfried Lepuschitz  
Practical Robotics Institute Austria (PRIA)  
Vienna  
Austria

Richard Balogh  
Slovak University of Technology (STU)  
Bratislava  
Slovakia

Munir Merdan  
Practical Robotics Institute Austria (PRIA)  
Vienna  
Austria

David Obdržálek  
Charles University  
Prague  
Czech Republic

Gottfried Koppensteiner  
Practical Robotics Institute Austria (PRIA)  
Vienna  
Austria

ISSN 2194-5357

ISSN 2194-5365 (electronic)

Advances in Intelligent Systems and Computing

ISBN 978-3-319-62874-5

ISBN 978-3-319-62875-2 (eBook)

DOI 10.1007/978-3-319-62875-2

Library of Congress Control Number: 2017951140

© Springer International Publishing AG 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature

The registered company is Springer International Publishing AG

The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

We are glad to present the proceedings of the 8th International Conference on Robotics in Education (RiE) held in Sofia, Bulgaria, during 26–28 April 2017. The RiE is organized every year with the goal to provide the opportunity for the presentation of relevant novel research and development in a strongly multidisciplinary context in the educational robotics domain.

Educational robotics is an innovative way for increasing the attractiveness of science education and scientific careers in the view of young people. Robotics represents a multidisciplinary and highly innovative domain encompassing physics, mathematics, informatics and even industrial design as well as social sciences. As a multidisciplinary field, it promotes the development of systems thinking and problem-solving. Due to various application areas, teamwork, creativity and entrepreneurial skills are required for the design, programming and innovative exploitation of robots and robotic services. The fascination for autonomous machines and the variety of fields and topics covered make robotics a powerful idea to engage with. Robotics confronts the learners with the areas of Science, Technology, Engineering, Arts and Mathematics (STEAM) through the design, creation and programming of tangible artifacts for creating personally meaningful objects and addressing real-world societal needs. Thus, young girls and boys can easily connect robots to their personal interests and share their ideas. As a consequence, it is regarded as very beneficial if engineering schools and university programme studies include the teaching of both theoretical and practical knowledge on robotics. In this context, current curricula need to be improved and new didactic approaches for an innovative education need to be developed for improving the STEAM skills among young people. Moreover, an exploration of the multidisciplinary potential of robotics towards an innovative learning approach is required for fostering the pupils' and students' creativity leading to collaborative entrepreneurial, industrial and research careers.

In these proceedings, we present the latest results and development in educational robotics research and application. The book offers a range of methodologies for teaching robotics and presents various educational robotics curricula and activities. Moreover, the book introduces interesting programming approaches as

well as new applications, technologies, systems and components for educational robotics. The presented applications cover the whole educative range, from elementary school to high school, college, university and beyond, for continuing education and possibly outreach and workforce development. In total, 47 papers were submitted and 29 papers are now part of these proceedings after a careful peer review process. We would like to express our thanks to all authors who submitted papers to RiE 2017, and our congratulations to those whose papers were accepted.

This publication would not have been possible without the support of the RiE International Program Committee and the Conference Co-chairs. We also wish to express our gratitude to the volunteer students and staff of the partner organizations, which significantly contributed to the success of the conference. All of them deserve many thanks for having helped to attain the goal of providing a balanced event with a high level of scientific exchange and a pleasant environment. RiE 2017 was greatly supported by SAP Labs Bulgaria, Avitel and Sofia Tech Park, for which we thankfully express our gratitude. We acknowledge the use of the EasyChair conference system for the paper submission and review process. We would also like to thank Dr. Thomas Ditzinger, Jeyashree Kumar and Springer for providing continuous assistance and advice whenever needed.

# Organization

## Committee

### Co-Chairpersons

Richard Balogh	Slovak University of Technology in Bratislava, Slovakia
Wilfried Lopuschitz	Practical Robotics Institute Austria, Austria
David Obdržálek	Charles University in Prague, Czech Republic
George Sharkov	European Software Institute – Center Eastern Europe, Bulgaria

### International Programme Committee

Dimitris Alimisis	EDUMOTIVA-European Lab for Educational Technology, Greece
Julian Angel-Fernandez	Vienna University of Technology, Austria
Ansgar Bredenfeld	Dr. Bredenfeld UG, Germany
Jenny Carter	De Montfort University in Leicester, UK
Dave Catlin	Valiant Technology, UK
G. Barbara Demo	Dipartimento Informatica, Universita Torino, Italy
Jean-Daniel Dessimoz	Western Switzerland University of Applied Sciences and Arts, Switzerland
Nikleia Eteokleous	Dept. of Educational Sciences, Frederick University Cyprus, Cyprus
Hugo Ferreira	Instituto Superior de Engenharia do Porto, Portugal
Paolo Fiorini	University of Verona, Italy
Carina Girvan	Cardiff University, UK
Grzegorz Granosik	Lodz University of Technology, Poland



Ivaylo Gueorguiev	European Software Institute – Center Eastern Europe, Bulgaria
Alexander Hofmann	University of Applied Sciences Technikum Wien, Austria
Martin Kandlhofer	Graz University of Technology, Austria
Boualem Kazed	University of Blida, Algeria
Gottfried Koppensteiner	Practical Robotics Institute Austria, Austria
Tomáš Krajník	University of Lincoln, UK
Miroslav Kulich	Czech Technical University in Prague, Czech Republic
Chronis Kynigos	University of Athens, Greece
Lara Lammer	Vienna University of Technology, Austria
Andrej Lúčný	Comenius University in Bratislava, Slovakia
Martin Mellado	Instituto ai2, Universitat Politècnica de València, Spain
Munir Merdan	Practical Robotics Institute Austria, Austria
Michele Moro	University of Padova, Italy
Margus Pedaste	University of Tartu, Estonia
Pavel Petrovič	Comenius University in Bratislava, Slovakia
Alfredo Pina	Public University of Navarra, Spain
João Machado Santos	University of Lincoln, UK
Fritz Schmöllebeck	University of Applied Sciences Technikum Wien, Austria
Dietmar Schreiner	Vienna University of Technology, Austria
Ugo Solitro	University of Verona, Italy
Roland Stelzer	INNOC – Austrian Society for Innovative Computer Sciences, Austria
Pavel Varbanov	European Software Institute – Center Eastern Europe, Bulgaria
Markus Vincze	Vienna University of Technology, Austria
Igor M. Verner	Technion – Israel Institute of Technology, Israel
Francis Wyffels	Ghent University, Belgium
Nikoleta Yiannoutsou	University of Athens, Greece

### **Local Conference Organization**

Ivaylo Gueorguiev	European Software Institute – Center Eastern Europe, Bulgaria
Wilfried Lepuschitz	Practical Robotics Institute Austria, Austria
Munir Merdan	Practical Robotics Institute Austria, Austria
Christina Todorova	European Software Institute – Center Eastern Europe, Bulgaria
Pavel Varbanov	European Software Institute – Center Eastern Europe, Bulgaria

## Sponsors

SAP Labs Bulgaria



Avitel



Sofia Tech Park



# Contents

<b>Comprehensive Educational Robotics Activities</b>	
<b>TechColleges</b> . . . . .	3
Thomas N. Jambor	
<b>Robotics Peer-to-Peer Teaching Summer School Project Involving University Students, Summer Interns and Middle School Students</b> . . . .	15
Sabrina Rubenzer, Georg Richter, and Alexander Hofmann	
<b>Methods for Managing Student-Driven Robotics Research</b> . . . . .	26
Cem Avsar, Lennart Kryza, and Klaus Bri��	
<b>Robotics Education in Saint Petersburg Secondary School</b> . . . . .	38
Sergey Filippov, Natalia Ten, Alexander Fradkov, and Ilya Shirokolobov	
<b>Workshops, Curricula and Related Aspects</b>	
<b>LEGO WeDo Curriculum for Lower Secondary School</b> . . . . .	53
Michaela Veselovsk�� and Karol��na Mayerov��	
<b>Pythagorean Approximations for LEGO: Merging Educational Robot Construction with Programming and Data Analysis</b> . . . . .	65
Ronald I. Greenberg	
<b>Teaching Robotics Concepts to Elementary School Children</b> . . . . .	77
Mor Friebron Yesharim and Mordechai Ben-Ari	
<b>The Effect of the Programming Interfaces of Robots in Teaching Computer Languages</b> . . . . .	88
B. Baransel Ba��cı, Mustafa Kama��sak, and G��khan Ince	
<b>Creativity and Contextualization Activities in Educational Robotics to Improve Engineering and Computational Thinking</b> . . . . .	100
Albert Valls, Jordi Alb��-Canals, and Xavier Canaleta	

**Educational Robotics for Communication, Collaboration and Digital Fluency** . . . . . 113  
 Ivaylo Gueorguiev, Christina Todorova, Pavel Varbanov, Petar Sharkov, George Sharkov, Carina Girvan, Nikoleta Yiannoutsou, and Marianthi Grizioti

**Using Robotics to Foster Creativity in Early Gifted Education** . . . . . 126  
 Tomislav Jagust, Jasna Cvetkovic-Lay, Ana Sovic Krzic, and Damir Sersic

**The Evaluation of Robotics Activities for Facilitating STEM Learning** . . . . . 132  
 Ronit Ben-Bassat Levy and Mordechai Ben-Ari

**Project-Based Learning Approaches**

**MuseumsBot - An Interdisciplinary Scenario in Robotics Education** . . . . . 141  
 Tanja Heuer, Ina Schiering, and Reinhard Gerndt

**Marine Robotics: An Effective Interdisciplinary Approach to Promote STEM Education** . . . . . 154  
 Saeedeh Ziaefard and Nina Mahmoudian

**Designing Robotics Student Projects from Concept Inventories** . . . . . 166  
 Reinhard Gerndt and Jens Lüssem

**Teaching Research Methodologies with a Robot in a CS Lab Course** . . . . . 180  
 Mathias Landhäußer, Sebastian Weigelt, and Martin Blersch

**Teaching Robotics for Computer Science Students** . . . . . 193  
 Vesna Kirandziska and Nevena Ackovska

**Technologies for Educational Robotics**

**TUC-Bot: A Microcontroller Based Robot for Education** . . . . . 201  
 Sven Lange, Peter Weissig, Andreas Uhlig, and Peter Protzel

**Open Source Robotics Course at Engineering: Infrastructure and Methodology** . . . . . 214  
 Francisco Martín

**The Robobo Project: Bringing Educational Robotics Closer to Real-World Applications** . . . . . 226  
 Francisco Bellas, Martin Naya, Gervasio Varela, Luis Llamas, Abraham Prieto, Juan Carlos Becerra, Moises Bautista, Andres Faiña, and Richard Duro

**Architectural Overview and Hedgehog in Use** . . . . . 238  
 Clemens Koza, Martin Wolff, Daniel Frank, Wilfried Lepuschitz,  
 and Gottfried Koppensteiner

**Open-Source Robotic Manipulator and Sensory Platform** . . . . . 250  
 Luka Čehovin Zajc, Anže Rezelj, and Danijel Skočaj

**OTO – A DIY Platform for Mobile Social Robots in Education** . . . . . 257  
 Thomas Vervisch, Natan Doms, Sander Descamps, Cesar Vandeveldel,  
 Francis wyffels, Steven Verstockt, and Jelle Saldien

**An Elementary Science Class with a Robot Teacher** . . . . . 263  
 Alex Polishuk and Igor Verner

**Design of Robot Teaching Assistants Through Multi-modal  
 Human-Robot Interactions** . . . . . 274  
 Paola Ferrarelli, María T. Lázaro, and Luca Iocchi

**Virtual Environments, Cloud Tools and Artificial Intelligence**

**eduMorse: An Open-Source Framework for Mobile  
 Robotics Education** . . . . . 289  
 Daniele De Martini, Andrea Bonandin, and Tullio Facchinetti

**Teaching Robotics with Cloud Tools** . . . . . 301  
 Igor Zubrycki and Grzegorz Granosik

**Needs, Opportunities and Constraints on the Way  
 to the Wide Introduction of Robotics to Teaching  
 at Secondary Vocational Schools** . . . . . 311  
 Mikulas Hajduk, Zbigniew Pilat, Adrian D. Olaru, and Marek Vagaš

**An Open Robotics Environment Motivates Students  
 to Learn the Key Concepts of Artificial Neural Networks  
 and Reinforcement Learning** . . . . . 317  
 Tapani Toivonen, Ilkka Jormanainen, and Markku Tukiainen

**Author Index** . . . . . 329

# **Comprehensive Educational Robotics Activities**

# TechColleges

## Learn to Teach Using Robots

Thomas N. Jambor<sup>(✉)</sup>

Centre for the Didactics of Engineering, Leibniz Universität Hannover,  
Hanover, Germany  
jambor@zdt.uni-hannover.de

**Abstract.** TechColleges, a project founded by the Federal State of Lower Saxony, aims at inspiring students at vocational schools to become a part of the first generation of academics. These students - being trained to teach at vocational schools - get support during their university courses with mathematical introductory courses and electrotechnical projects using small robots (e.g. Raspberry Pi, Arduino). Robots have a great potential for motivating students excellently by linking electrical engineering with computer science. Positive impressions by using robots were received for projects like robotic summer program or robot lab project. Furthermore, there is a large number of available systems, like Lego Mindstorms or Robotino, which can be used in classrooms. TechColleges uses robots in university and classrooms to motivate students to become a teacher at a vocational school. The project consists of three levels, in which students are confronted with problems in the field of robotics and teaching activities. This paper describes the individual levels of the project. Level one depicts the obtained results. Level two and three give a preview.

## 1 Introduction

Over the last years, the demand of well-trained skilled workers in the fields of electrical engineering and information technology has been significantly increasing (e.g. [1, 2]). Yet, fewer and fewer students opt for the degree program of teaching at vocational schools in the field of electrical engineering even though career perspectives are above average. For example, in the field of electrical engineering for vocational schools, about seven students average began studying for a bachelor's degree during the past years at Leibniz University of Hannover. The situation at other universities is similar. This results in a lack of teachers especially qualified in didactic and pedagogical aspects, who are able to train skilled workers. The project TechColleges, founded by the Federal State of Lower Saxony, aims at inspiring students of vocational schools to become a part of the first generation of academics. These students are trained to teach at vocational

---

This research was funded by Lower Saxonian Ministry for Science and Culture. I am thankful to the TechColleges project staff who provided expertise that greatly assisted the research.

schools using small robots with Raspberry Pi ([3]) or Arduino ([4]) based control unit. During their university courses they are supported by mathematical introductory courses and electrotechnical projects, in which students are soldering, assembling and programming small robots. An important aspect is the evaluation of the measures taken aiming at optimizing and developing the overall concept. A catalog of measures will be created based on the evaluation and the experience gained to pave the way for continuous implementation of those measures at the Faculty of Electrical Engineering and Computer Science at the Leibniz University of Hannover as well as for transferring this approach to other faculties and universities.

## 2 Current Situation of Robotics in Education

The use of robots in the field of teaching has a motivating effect [5]. For this reason, countless robot platforms are available. They differ in terms of price, number of sensors, demands on the learner, appearance, etc. The price for each platform varies between \$ 50 (e.g. AREXX Arduino Roboter [6]), several hundred dollars (e.g. LEGO EV3 [7]), and several thousand dollars (e.g. NAO [8]). Since the target audience (future teachers) is not yet active at a school, we use a budget-priced open platform for the robots. Due to the popularity of Arduino and Raspberry Pi in schools, these boards appear to be very suitable. In addition, Arduino and Raspberry Pi can be used in other projects that are not focused on robots. Finally, it is possible for students to use their own Raspberry Pi as a media center or a favorable game console outside the projects.

A similar diversity can be observed in the field of projects. Projects like to use robots particularly to raise interest in technology. Preschool children (e.g. [9]), girls [10] and engineering students (e.g. [11]) are often focus of these projects. Already employed teachers will also be addressed, in order to encourage the curricular use of robotics (e.g. [12]). A project with future teachers as the target audience of is unknown to the author.

## 3 Education System of Germany

Besides the academic training, vocational education is an important pillar of the German education system ([13]). In this context, it must be distinguished between the dual vocational training and the fully school-based education. In the dual training system, students attend classes at school and receive on-the-job training at a company (e.g. [14]). Here, practical aspects of the future profession are always in the center of interest. For this reason, the theory is to be seen in the context of actual situations where action is required. A distinction is made between three categories. The full school vocational training, which is also preparing for a future professional activity, is most similar to the dual system of training. In the area of electrical engineering and computer science, there are two-year vocational schools, which provide both theoretical and practical



contents. The second category prepares students for a dual training in one-year schools (BVJ, BGJ). Schools belonging to the latter category prepare for academic education and strongly link general and technical education. The age of the students in the respective school types varies due to the different goals of each category. In 2011, for example, students of the dual system of training (BIBB) were in average in their early twenties, whereas the students in classes preparing for academic education were often older, since they had already finished dual training.

Vocational school teachers in all mentioned types of school have a university degree (M.Ed.) ([15]). The diversity of the school types places high demands on their skills. In addition to a theoretical pedagogical background, teachers also need practical knowledge about professional job requirements. For this reason, most teachers have completed a vocational training themselves or have worked as engineers.

A vocational teacher training is divided into two phases. In the first (academic) phase (five years), they study pedagogy and a second subject (such as German, Politics etc.) in addition to their professional specialization (e.g. Electrical Engineering). They do internships at vocational schools in the different study terms. In the second phase (18 months), they already teach at a school while simultaneously attending study seminars to continue their theoretical training. This is comparable to a dual vocational training (from training on-the-job to teaching at a school and from school to study seminar) ([16]).

Having finished the training, an exciting job with young people awaits the graduates. This job also offers good possibilities to reconcile work with family life. Currently, the demand for teachers in the area of electrical engineering is very high (e.g. [17,18]). For this reason, most teachers can freely choose the school they want to work at. Most jobs are offered with a permanent contract and an average gross income of approx. 3650 EURO (unmarried and without children). Promotion prospects are to become head of school department or school principal.

## 4 Concept

Young people often do not feel ready to study directly after graduation. To start a voluntary year or dare an orientation study to bridge the gap between school and university makes sense. The concept of TechColleges based on picking up the students at school and introducing them by means of technical projects to university studies. In this case, technology serves as an instrument to overcome the obstacle to an unknown academic world. The versatile technical projects should increase the motivation of students and reduce threshold fears of taking up university studies for intellectually gifted students less likely to access a university career. Furthermore, the relationship between students and their project supervisor plays an important role. The supervisor is meant to be mentor, consultant and trustworthy contact in one person. Since they know academic life due to their own studies, they may offer help in difficult situations.

Secondly, they might assist in dealing with understanding difficulties regarding the tasks set. The focus is on mentors who are not teachers of (vocational) schools. The mentors are scientific staff from the Leibniz University of Hannover or students, who are studying in the master program of vocational schools or work as research assistants at the Faculty of Electrical Engineering and Computer Science in the Leibniz University of Hannover.

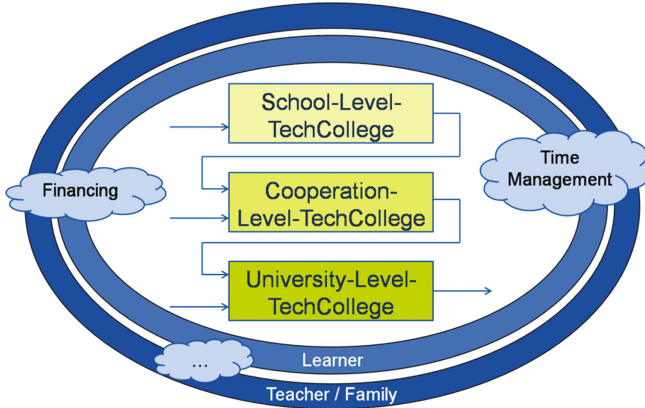
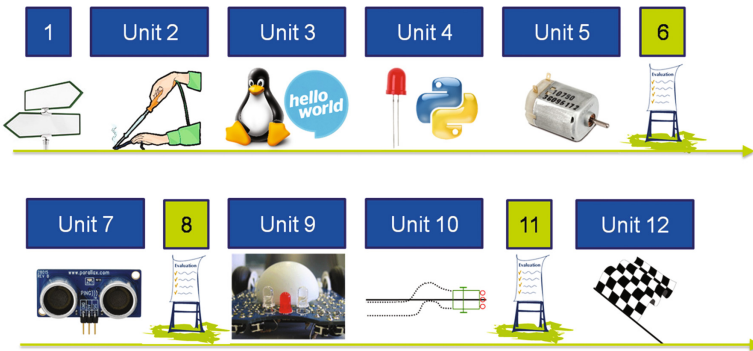


Fig. 1. TechColleges concept

TechColleges is divided into three levels (represented in Fig. 1 as green colored shapes), in which students are confronted with realistic problems in the field of robotics and teaching activities. In the first level (School-Level-TechCollege), a special teaching concept will be developed and performed in the classroom. In the second level (Cooperation-Level-TechCollege), students are supposed to realize a project at the university, thus offering them first contacts with university life and study practice. In the third level (University-Level-TechCollege), students visit an university course and deepen their technical knowledge. Furthermore, students get an insight into the life of a teacher by performing small teaching units. In all three levels, students (represented as learners), teachers and family members of the students visit informational or methodical workshops for free (represented in Fig. 1 as blue colored clouds). Students participate in technical and methodological workshops (e.g. time management). Informational events (e.g. studying or financing) are also offered for teachers and family members in order to arouse their interest in an academic career and to guarantee their support, especially if the student would establish the first generation of academically trained family members.

#### 4.1 School-Level-TechCollege

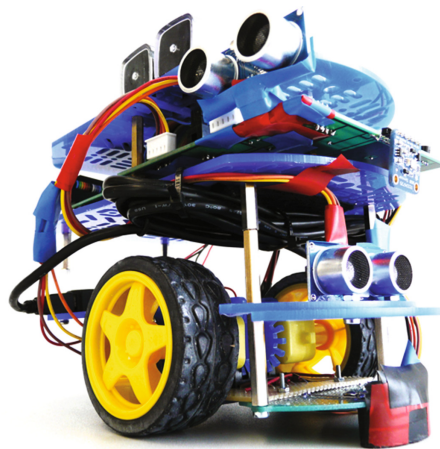
School-Level-TechCollege is divided into twelve units (represented in Fig. 2). Nine blue colored units describe the technical approach of the TechColleges



**Fig. 2.** School-Level-TechCollege - Concept

concept. Three green colored units represent the educational approach of the project. Each unit includes 180 min. This is equivalent to two double lessons at a vocational school in Germany. The first level takes place at vocational schools.

The concept of School-Level-TechCollege is based on developing a moving Raspberry Pi (RPi) robot, extended by line tracking and distance detection (represented in Fig. 3). Students additionally learn simple teacher specific activities, such as designing a worksheet to explain context, presenting work results and independently creating evaluation criteria by solving the task sets. Related seminars such as “Learning”, “Time Management” and “Motivation” facilitate the entry into the study system and are acknowledged as key competences for the teaching profession. Furthermore, counseling and reconnaissance units in form of individual and group interviews are offered to consolidate the knowledge



**Fig. 3.** School-Level-TechCollege - RPi robot

obtained in study and teaching activities resulting in a sustainable improvement of the project and the individual levels. In interviews, the students' sociobiographical background is discussed, the image of the teaching profession is depicted and it was investigated, whether a student is suitable to become a teacher. This review aims mainly at helping students to improve their self-reflection. On one hand, the tutors present the basic concepts and conditions for teaching at a vocational school from a teacher's point of view. On the other hand, the students can experience, what it is like to be a teacher by actually assuming all related tasks. Other questions (such as career opportunities, salary) are also discussed in this context.

## 4.2 School-Level-TechCollege - Units

1. Unit 1: Knowledge (represented in Fig. 5 and explained in Sect. 4.4).
2. Unit 2: The second step is to solder a flip-flop to gain initial experience with soldering. Connecting the entire board of the robot with the necessary sensors and passive and active components is established. The parallel construction of the chassis completes this unit.
3. Unit 3: After the successful construction of the robot, students deal with a few simple commands of the operating system Linux. They take the first program (Hello World) into their programming environment and learn about the development environment, as well as the Python interpreter.
4. Unit 4: Students create the first program to make an LED blink, in which they learn the basic structures of Python.
5. Unit 5: The aim of this unit is to set the robot into motion. The H-bridge and PWM control are discussed.
6. Unit 6: Learner Process (represented in Fig. 5 and explained in Sect. 4.4).
7. Unit 7: In this unit, the forward and backward driving robot from unit 5 is extended. The aim is to avoid collision, which is realized by using the ultrasonic distance sensors.
8. Unit 8: Project (represented in Fig. 5 and explained in Sect. 4.4).
9. Unit 9: In preparation for the next unit, students deal with the detection of a line using phototransistors. To achieve this aim, they need to read the values of the phototransistors via the A/D converter.
10. Unit 10: In this unit, particularly the results of the units 5 and 9 are assembled. The aim is to follow a black line. It is not enough to assemble the existing source code. Due to the disturbing influence of light around, some tweaks need to be made to the robot.
11. Unit 11: Interviews (explained in Chap. 5).
12. Unit 12: In the final unit, the recent improvements are to be made to the robot to drive a race. Each group presents the capabilities of their own robot on a racetrack.

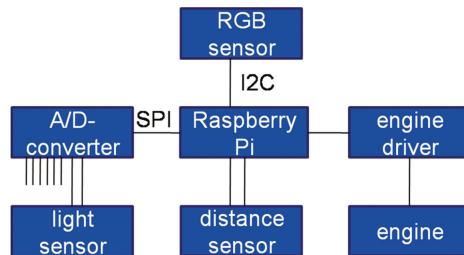
## 4.3 School-Level-TechCollege - Technical Approach

Since the target audience consists of predominantly inexperienced learners and embedded programming is introduced, the electrical engineering and computer

science are directly linked to the microcontroller platform (e.g. Arduino) and a small computer (e.g. RPi). The RPi computer was selected as control unit for the traveling robot. Decisive for this choice was the Linux operating system, which also provides enough space for additional domestic use (e.g. as a multimedia console for the living room). Currently, the first version of the RPi in the B-variant is applied in the project. Therefore, the students use a networkable small computer. The numerous Python libraries allow a comfortable access to the GPIOs and to the associated sensors and actuators. During the development phase, students work on a LAN and a Remote Desktop connection at the RPi. Once the individual robots start to move, a wireless stick can replace LAN.

Control of the motors (represented in Fig. 4 as engine) is obtained via the motor driver (represented in Fig. 4 as engine driver) L293D including two H-bridges. Thus, it is possible to control a motor using only one operating voltage. The motor speed should vary in order to ensure stable driving on the lines. To solve this task, the students have to deal with pulse-width modulation (PWM).

First, the RPi robot orients itself in the room using two ultrasonic sensors. The sensors measure distances between 2 cm and 4 m. At this point, it is necessary to implement a simple communication protocol that is adhered to at all times. Measurement started via a trigger signal. Then, the sensor responds with the setting and subsequent resetting of the echo signal. Based on the time difference between the set and reset signal, the students can determine the measured distance.



**Fig. 4.** School-Level-TechCollege - technical approach

A second way to navigate the robot is the pursuit of a black line, which is mounted on a white background. To facilitate this, two phototransistors are used. A RPi has no analog inputs, which would be able to detect the measured values of the phototransistors. For this reason, the A/D converter MCP3008 is applied. In addition to a 10-bit resolution of the A/D converter, it provides eight channels and a sample rate of 200 kHz. The A/D converter is controlled using the SPI protocol.

The entire circuit is installed on a circuit board that is mounted on a chassis together with the RPi and the individual sensors. A further sensor is mounted on the circuit board with which the color can be detected. Due to time constraints,

however, the color sensor, which deals with yet another protocol (I2C), is not integrated into the School-Level-TechCollege concept.

Nevertheless, the necessary hardware and materials are represented in Fig. 4.

#### 4.4 School-Level-TechCollege - Educational Approach

The School-Level-TechCollege concept follows the didactic approach of self-organized learning ([19]), a three-stage approach divided into “Knowledge”, “Learner Process” and “Project” (represented in Fig. 5).

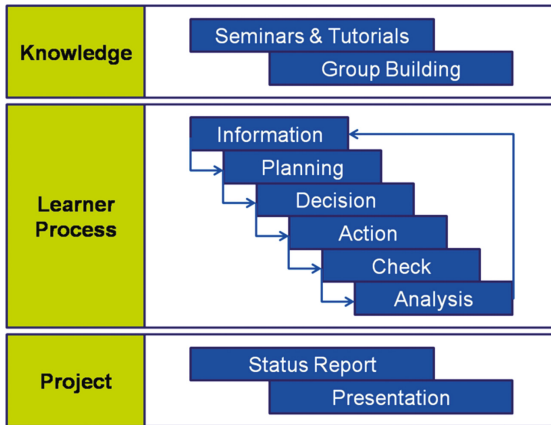


Fig. 5. School-Level-TechCollege - learning approach diagram

Learners are integrated as active participants in a constructive learning process. They work out their own tasks, construct their own robot designs and acquire the knowledge necessary for operating a traveling robot. In the first stage, named “Knowledge”, the participants form groups to exchange their knowledge. Furthermore, independent learning is possible via video tutorials and seminars. The second stage, called “Learner Process”, is divided into six phases ([20]). Working in study groups of up to three people, the students independently pass the following six phases: “Information”, “Planning”, “Decision”, “Action”, “Check” and “Analysis”. Having finished the last phase, they can return to the first phase in order to optimize the results obtained on their robot. In the third stage, named “Project”, all the groups present their results in plenary and give a status report on the difficulties and findings while solving the task.

#### 4.5 Cooperation-/University-Level-TechCollege

The concepts for Cooperation- and University-Level-TechCollege are in the planning stage. Both levels are based on the experience, proposals and suggestions that resulted of the first pass of the School-Level-TechCollege. The technical

projects, such as construction and control of a quadrocopter, zeppelin, segway, and an arm robot or programming a humanoid robot NAO ([21]) are used in the two secondary levels. The technical and conceptual approaches are already in the development phase.

Students receive particular support in their orientation phase by providing mathematical and electrical basic courses. They can expand their mathematical skills and close potential knowledge gaps. The electrical basic courses enable the linking of theoretical contents with practical aspects giving a first insight into teaching. Based on these experiences, students will assume the role of teachers in the third level of TechColleges supporting first level participants. They are confronted with teaching as early as possible, thus being able to reflect on their own suitability for the job.

The aim is the permanent implementation of the School-Level-TechCollege in the Faculty of Electrical Engineering and Computer Science at the Leibniz University of Hannover and the support of young future teachers during their study with an early experience of teaching in vocational schools.

## 5 Evaluation

Prior to the beginning of the project, 12 participants without prior knowledge of electrical engineering tested the technical contents of School-Level-TechCollege in a 5-day workshop. The participants, three of them females, came from different regions of northern Germany and visited the school grades 8 to 9. The workshop served primarily to determine the period, to assess the difficulty of the technical issues, to evaluate the fun factor while performing the task and to verify improvements suggested by the students. After having implemented the suggestions of the test group and having taken into consideration further educational aspects a pilot first level project has been performed at the vocational school of Oldenburg in Germany with a total of five students interested in the overall concept and willing to participate after their regular school time. The analysis of the transcriptions and the survey sheets returned the following results on the following five main subjects: “Socio-biographical background”, “Attitude towards studying”, “Attitude towards the teaching profession”, “Skills in the context of TechColleges” and “Review of School-Level-TechCollege”.

### 5.1 Socio-Biographical Background

The educational level of the parents is rather low (hardly any academic degree). The majority of the parents (60%) have completed professional training and have been working in traditional occupations such as locksmith, elder care or office clerk. The remaining 40% have completed a course of study, but only 20% of them became a teacher. Nevertheless, all participating students get the parental support. Parents support academic training and are willing to help (e.g. assistance in moving to a new city). However, the parents are not the primary point of contact in terms of advising on the decision to study, because they

have little experience in any training. Tips for selecting the studies, the end of the study, the student life etc. are rather preferred to be received from sports colleagues or from friends, who already enrolled in a study. For all parents, the financial support is of not matter anymore. However, they are partially positive about it, expecting a better earning potential after graduation, and are negative in parts, due to the loss of earnings in the period of study. The majority of participants (90%), who take up studies, have no barriers. The minority (10%) feared only the loss of social contacts in their town. Half of the participants are even very positive about the study, due to better earning potential in the private sector, the civil servant status and financial security, and have role models to look up at in mind.

## **5.2 Attitude Towards Studying**

Participants generally have a very positive attitude towards studying. This is reflected in hopes and expectations and in the desire for education, because of intellectual underload, e.g. in the profession as a construction worker, which is more dominated by the physically practiced work. Furthermore, flexible scheduling, a new image and safety aspects due to the academic degree are highly appreciated among the students. The only reason why all participants would hesitate to take up studies, is their fear of failing to meet the challenging requirements.

## **5.3 Attitude Towards the Teaching Profession**

The group discussion regarding the attitude towards the teaching profession resulted in the following findings. All of the participants evaluated job security, personal development and deepening of interests as positive. The low promotion opportunities as in the private sector, a low social status of teachers in society of Germany, exam corrections at home and partly flexible time scheduling based on lessons preparation were evaluated as negative by all participating persons.

## **5.4 Skills in the Context of TechColleges**

All participants have teaching competencies. They had already gained experience in social work, tutoring or as a youth coach. The minority of the respondents (40%) like to explain things and feel competent enough to take up a teacher-training course. The challenges and development needs they see are in self, time and conflict management and their capacity to show empathy as well as in the future change of roles (from student to teacher).

## **5.5 Evaluation of School-Level-TechCollege**

After the project, all participants feel better informed and prepared as to the choice of an adequate study course. The consulting resulted in a significant reduction of their socio-cultural inhibitions. The insight into the teaching practice was very exciting and consequently contributed to the decision for or against



the teaching degree. The technical part of the project was valued by 100% of the respondents as positive. It was very exciting for the participants to build their own moving robot and make them travel around. Students felt valued by the university mentors and often highlighted their positive personal and non-hierarchical supervision. The tasks were understandable and clearly formulated and offered exciting topics from electrical engineering, programming and didactics. All of the participants were very satisfied with the tasks to be solved. Due to the parallel school and the additional homework, the high workload was heavily criticized. However, in some cases, there was a desire for more contents from the educational field. All participants recommended the project and were determined to participate in the second level of the project.

## 6 Conclusion

First experiences gained in the first level (School-Level-TechCollege) show that there are dedicated students, who are interested in technical and pedagogical aspects. Due to the low numbers of participants in the first round of School-Level-TechCollege, the subject teachers at vocational schools significantly enhance marketing activities. Besides that, additional vocational schools could be gained as partner in the region of Hannover. In addition, the Cooperation- and the University-Level-TechCollege will be prepared. The concept of the School-Level-TechCollege should be integrated into the school curriculum at vocational schools at an early stage to enable the university's students from the first semester on get an insight into teaching and to carry out first own projects of the University-Level-TechCollege. An additional goal is to implement the project permanently at the Faculty of Electrical Engineering and Computer Science at the Leibniz University of Hannover and to integrate it into other faculties and universities.

## References

1. Federal Employment Agency: Annual Report 2015 - Sixty-fourth Annual Report of the Federal Employment Agency (2017). <https://www.arbeitsagentur.de/en>
2. Deissinger, T.: The German dual vocational education and training system as 'good practice'? *Local Econ.* **30**(5), 557–567 (2015)
3. Raspberry Pi Foundation: Online Presence of Raspberry Pi platform (2017). <https://www.raspberrypi.org/>
4. Arduino LLC.: Online Presence of Arduino platform (2017). <https://www.arduino.cc/>
5. Khanlari, A.: Robotics integration to create an authentic learning environment in engineering education. In: *Frontiers in Education Conference (FIE)*, pp. 1–4 (2016)
6. AREXX Engineering: Online Presence of AREXX (2017). <http://www.arexx.com/>
7. LEGO Group: Online Presence of Lego Education (2017). <https://education.lego.com/>
8. SoftBank Robotics: Online Presence of SoftBank Robotics (2017). <https://www.ald.softbankrobotics.com/>

9. Pekarova, J.: Using a programmable toy at preschool age: why and how? In: Teaching with Robotics: Didactic Approaches and Experiences. Workshop of International Conference on Simulation, Modeling and Programming Autonomous Robots (2008)
10. Santos, C.B., Ferreira, D.J., do Nascimento Rodrigues, M.C.B., Martins, A.R.: Robotics and programming: attracting girls to technology. In: Advances in Computing, Communications and Informatics (ICACCI), pp. 2052–2056 (2016)
11. Aroca, R.V., Watanabe, F.Y., Avila, M.T.D., Hernandes, A.C.: Mobile robotics integration in introductory undergraduate engineering courses. In: 2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR), pp. 139–144 (2016)
12. Mondada, F., Bonani, M., Riedo, F., Briod, M., Pereyre, L., Rétornaz, P., Magnenat, S.: Bringing Robotics into Formal Education Using the Thymio Open Source Hardware Robot. Institute of Electrical and Electronics Engineers (2017)
13. Frommberger, D., Reinisch, H.: Development of disparate structures of Dutch and German vocational education. In: Proceedings of the First International Conference Towards a History of Vocational Education and Training (VET) in Europe in a Comparative Perspective, pp. 75–87 (2002)
14. Hippach-Schneider, U., Krause, M., Woll, C.: Vocational Education and Training in Germany - Short Description. European Centre for the Development of Vocational Training
15. Euler, D.: Germany's Dual Vocational Training System: A Model for Other Countries? - A Study Commissioned by the Bertelsmann Stiftung. Bertelsmann Stiftung (2013)
16. Ostinelli, G.: Teacher education in Italy, Germany, England, Sweden and Finland. *Euro. J. Educ.* **44**(2), 291–308 (2009)
17. Ministry of Culture of Lower Saxony: Online Presence of Ministry of Culture of Lower Saxony (in German) (2017). <http://www.mk.niedersachsen.de>
18. Ministry for School and Further Education North Rhine-Westphalia Online Presence of Ministry for School and Further Education North Rhine-Westphalia (mainly in German) (2017). <http://www.schulministerium.nrw.de>
19. Wolters, C.A., Pintrich, P.R., Karabenick, S.A.: Assessing Academic Self-Regulated Learning. What Do Children Need to Flourish?, pp. 251–270. Springer, US (2011)
20. Paya-Vaya, G., Jambor, T., Septinus, K., Hesselbarth, S., Flatt, H., Freisfeld, M., Pirsch, P.: ChipDesign: from theory to real world. In: Proceedings of the 2007 Workshop on Computer Architecture Education, pp. 58–64 (2007)
21. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: Mechatronic design of NAO humanoid. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 769–774. IEEE (2009)

# Robotics Peer-to-Peer Teaching Summer School Project Involving University Students, Summer Interns and Middle School Students

Sabrina Rubenzer<sup>(✉)</sup>, Georg Richter, and Alexander Hofmann

University of Applied Sciences Technikum Wien,  
Höchstädtplatz 6, 1200 Wien, Vienna, Austria  
sabrina.rubenzer@technikum-wien.at

**Abstract.** This paper presents a robotics summer school concept involving teaching methods like peer tutoring and teaching in order to meet the demand of robotics summer courses at the University of Applied Sciences Technikum Wien and benefit from its didactical possibilities.

**Keywords:** Robotics in education · Curriculum · Summer school · STEM · Peer-To-Peer tutoring · Peer teaching

## 1 Introduction and Motivation

As part of the RoboCupJunior Austria [1] association, the department of Computer Sciences of the University of Applied Sciences (UAS) Technikum Wien is promoting educational robotics and supporting interested schools, teams and students with robotics kits, introductory courses and arenas to test their robots in a tournament environment since 2007 [2]. However, these supporting measures mostly involve schools and depend on the commitment of teachers. In order to also give students the chance to gather experiences and develop their skills in programming and building robots during summer break, the project “Robots for Kids” arose in 2010. This course was established in cooperation with the “Wiener Kinderfreunde” [3] a non-profit organization based out of Vienna that operates 160 kindergartens in Europe and provides afternoon care as well as holiday entertainment and education for children. In 2013 the course was split into a beginner and an advanced course to offer more experienced students the possibility to improve their skills using different programming languages and hardware.

Moreover, in 2012 the department for the first time offered twelve internships for students aged 15 to 19 years in the field of science and technology promoted by the “Austrian Research Agency (FFG)” [4] using robotics as a hands-on method, not only to increase confidence in using technology and to enlarge their knowledge in programming and science but also to help with tutoring the middle school kids of the “Robots for Kids” course.

Therefore, the project “summer school” was established, organizing and managing the different needs of the involved students as well as developing an adequate curriculum for them.

## 2 Background and Related Research

Many initiatives offer robotics summer camps and courses for students of different age and educational background [5–7]. However, didactic teaching concepts therefore vary a lot.

One method of transferring the knowledge to younger students is peer teaching or tutoring. In that way students learn from other students who have dealt with the same challenges during their education. Inna Pivkina [8] distinguishes between peer learning assistants (PLAs), who help students learn in the courses that they recently took themselves and teaching assistants (TAs), who are students having graduated in the field they are teaching. Furthermore, it is stated that many students prefer PLAs to TAs or even achieved better results [9–11]. The introduction of “Big Brothers” and the resulting informal class environment can mutually benefit all peers [5].

Another perspective onto the topic delivers “Introductory biology course reform: a tale of two courses” [12], showing that the learning transfer is a two-way process. That means that the transfer of learning can be bidirectional – gained knowledge can be applied at work while new knowledge at the same time can be generated and vice versa. Figure 1 is showing that bidirectional learning transfer.

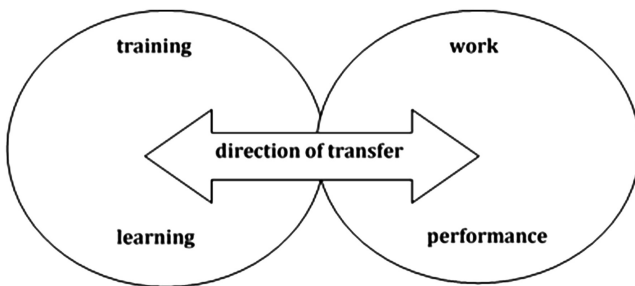


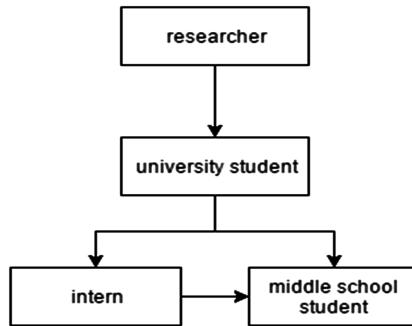
Fig. 1. Bidirectional learning transfer

## 3 Structure

Regarding this background the idea was to benefit from peer teaching by implementing the following structure, shown in Fig. 2.

Researchers and professors of the UAS transfer their knowledge to university students. Those students hired for the summer school then teach the interns (especially in the first two weeks of the internship, as further explained below) as well as the middle school students attending the “Robots for Kids” course. The interns support teaching the middle school students via peer teaching as mentioned above.

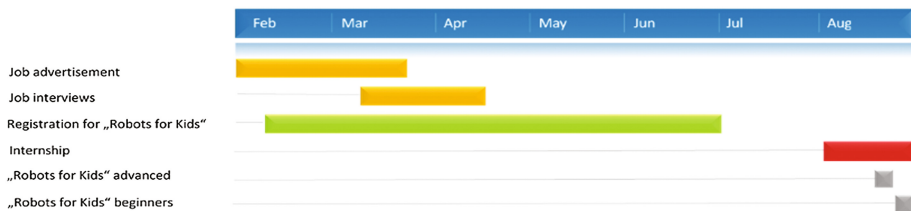
The summer school usually takes place during the long holiday break in the month of August. The interns (high school students) are hired for the whole month, whereas the two week-long courses for the middle school students (one for beginners, the other one for advanced students) are scheduled for the end of the month, generally the last two weeks of August. Experienced university students of the bachelor study program



**Fig. 2.** Implemented teaching transfer

“Computer Science” are hired to train and mentor the interns as well as teach the summer courses.

Figure 2 shows an exemplary timetable containing the recruitment phase of the interns and university students as well as the individual duration time of the summer school courses (Fig. 3).



**Fig. 3.** Summer school timeline

### 3.1 Interns

The summer internship at the UAS Technikum Wien is a one month paid internship for students aged 15 and up. It consists of an average of two weeks of training for the interns followed by two one-week courses for middle school student. Generally, there are between 8–16 internships available.

Table 1 shows the distribution of internships regarding gender and current school type. Involving high school students attending schools without a technical focus or previous technical education was one of the main goals. This was the departments’ choice as to give students interested in the topic who had no previous engagement with robotics or computer science the opportunity to develop their skills and interest in technical professions. Another focus was to hire predominantly female students and to motivate them to apply for technical studies later on.

The hiring process usually starts in February in cooperation with the Austrian Research Agency (FFG). Students can apply via their platform for one or more science related vacant positions. The CVs of the students are forwarded to the institution

**Table 1.** Demographics internships

Year	Project name	Amount of internships	Male	Female	School type with technical focus	School type without technical focus
2012	ROOT	12	6	6	0	12
2013	RADAR	8	5	3	1	7
2014	SPIRIT	16	9	7	5	11
2015	MAZE	12	4	8	4	8
2016	EXPLORE	15	5	10	4	11

offering the vacancies. Moreover, the job advertisement is also sent to partners of the university in order to reach more interested students. The applications are checked for basic requirements and most students are invited to an interview to give all of them the chance to proclaim their suitability as well as the chance to grow their experience with job interviews.

In order to decide whose motivation for the internship is the highest, following questions, amongst others, are part of the interview:

- Why did you apply for the internship?
- Have you ever heard of RoboCupJunior or any other robotic initiative?
- What do you like to do in your free time?
- Can you name a situation you are specifically proud of?
- Which profession do you want to achieve?
- When you are working in a team, which role do you generally engage?
- What is annoying or despairing you?

The motivation behind the detailed interview for the internship is not to test the students but to highlight their strengths to form a team of interns whose teamwork compatibility is rather high.

### 3.2 Middle School Students – “Robots for Kids”

Since 2013 there are two courses available – one beginner course at the end of the month and an advanced one, taking place one week before. 25–60 students are attending each course, varying slightly per year. The registration is operated by the “Wiener Kinderfreunde” who are also promoting the course on their website and other platforms.

The students start on Monday morning with an introduction of the organizer, followed by the decision which group they want to join (categorized into the three RoboCupJunior disciplines [13]). The students are guided through the learning process by three university students (one for each discipline) as well as by the interns and supervised by pedagogues provided by the “Wiener Kinderfreunde” during break time.

Table 2 shows the number of students participating at the beginners and advanced course and the gender distribution. The percentage of girls participating at the course is slightly growing over the years due to marketing activities from both the UAS Technikum Wien and the “Wiener Kinderfreunde”. Nonetheless, the goal is to increase the number of female students up to at least 30% within the next five years. 2016 only 13

**Table 2.** Demographics “Robots for Kids”

Year	Beginners course	Advanced course	Male	Female
2012	62	–	58 (94%)	4 (6%)
2013	45	30	69 (92%)	6 (8%)
2014	42	29	60 (85%)	11 (15%)
2015	42	30	63 (87%)	9 (13%)
2016	45	13	48 (83%)	10 (17%)

students took the advanced course due to the distribution of public holidays and the resulting need to reschedule the course more ahead of time.

## 4 Development of the Curriculum

The main goal of developing the curriculum for the summer school was to create a solid basis of knowledge, followed by practical applications to foster the understanding and at the same time adjust the content in a way that all target groups profit the most.

### 4.1 Curriculum for Summer Interns

The curriculum of the internship has been iteratively developed over the last 5 years. The “Robots for Kids” summer course existed previously, but in 2012 the decision was made to include interns to support it. This addition made it possible to split up the previous single course into an advanced and a beginner course and significantly refine the course contents while also developing this curriculum to introduce the interns to research and development as well as teaching as practiced at the UAS Technikum Wien.

This curriculum for the summer internship is meant to fulfil two main goals: Firstly, it should prepare the interns adequately for the task they need to perform during the internship. Mainly that includes supporting the teachers by solving minor problems of the students during classes as well as continuous support in finding creative ways to improve their programs and robots. As such the interns need to be both technically capable as well as able to effectively deal with frustrated kids and interpersonal problems.

The second goal of the internship is focused on the interns themselves, providing them with the opportunity to experience research and development in an authentic environment. This includes small and practically focused research projects often dealing with integrating different systems with each other or solving specific problems by using unknown sensors or programming techniques.

In summary, the curriculum aims to prepare the interns for their tasks so that the quality of the courses and therefore the continued existence of this internship is ensured but it is also meant to give interns a suitably thorough insight into both teaching and researching in a professional environment.

The current iteration of the curriculum is structured as follows:

- 2–3 days of basic introduction to programming using a variation of C, heavily interspersed with practical exercises

- 1–2 days each of solving the tasks of the RoboCupJunior Rescue Line and Soccer Leagues, respectively
- 5–6 days of focused research by the interns on their own projects
- 1 day of preparation for the courses including an introduction to tutoring/teaching for the interns
- $2 \times 5$  days of summer courses given by university students and supported by interns

The specific order and length of those points slightly varies each year based on the abilities and interests of the interns as well as the distribution of holidays.

Specific breakdown:

- Basic introduction to C (2–3 days)

The interns for this internship generally consist of students from schools without a technical focus and little to no experience in programming and robotics. This is a deliberate choice by the organizers further explained in chapter 3.1. As such, this introduction starts from the very beginning and aims to teach the interns both how and why programming works and is very much inspired by the introductory courses given to university students at the UAS Technikum Wien. The goal is to not just teach the basics of a programming language but to also give the interns a deeper understanding of programming as a whole so that they are able to further expand their knowledge on their own.

- RoboCupJunior Rescue Line & Soccer ( $2 \times 1$ –2 days)

In order to cement the knowledge of the interns, 1–2 days each are spent solving the Rescue Line and Soccer tasks of the RoboCupJunior. The interns work in groups of two mostly independently on the tasks. This provides the opportunity to utilize their knowledge on an unfamiliar task that requires them to both apply their knowledge in a new way on a much more complex problem than their previous exercises as well as introduce the added element of mechanics and robotics to them. For this, the LEGO MINDSTORMS NXT and EV3 sets are used to simplify the construction process so that the interns can focus on solving their problem with generally familiar LEGO Technic hardware rather than having to learn yet another new skill.

Furthermore, this part of the internship functions as a valuable evaluation opportunity for the interns. While they are working in their own groups, cooperating and competing with other teams and experiencing their first major successes, frustrations and failures, the teachers can gain insight into their abilities both pertaining to technical problem solving as well as interpersonal mediation. These observations are later used to find ideal research projects for the interns as well as utilize them optimally during the courses.

- Research Projects (5–6 days)

The research projects are a main focus of the internship and as such there is significant effort put into their viability to provide an enjoyable and enlightening experience for the interns as well as potential benefit for the UAS Technikum Wien. In the current version of the curriculum the interns get a full introduction to scientific work comparable to that of a university student and spend generally 1–2 days



evaluating various topics assisted by the teachers. Typical projects include evaluating the viability of sensors or programming techniques for specific applications, developing integrations for technology into other systems or creating and evaluation solutions for common robotics problems.

The project is chosen in such a way as to provide the interns with both a reasonable challenge as well as a satisfying result. Generally, every intern will have their own project, however, sometimes more complex projects will be worked on by two intern each focusing on different aspects in their work. The research process itself is mostly based on experimentation. Seeking out literature and establishing comparisons is encouraged but due to the limited time available and the relative inexperience of the interns this is usually only a minor part of their work.

Organizationally, the main parts of the research project are completed in the week preceding the two week-long summer courses, however, because there are frequent and lengthy pauses for the younger students, the interns also have the opportunity to do work on their paper during those pauses and after the students leave. The research projects are concluded by a presentation at the end of the internship, based on a bachelor thesis presentation. There, the interns must present their work within 5 min following standard university guidelines after which the course teachers as well as members of the university staff ask them questions about the specifics of their work as well as closely related projects.

- Preparations for “Robots for Kids”

The last part before the summer courses includes both physical preparations of rooms, robots and workplaces as well as an introduction to tutoring/teaching and several other important aspects and previous experiences useful for working with kids. Since each group in the summer course is led by only one university student, the interns capability to support him/her is of paramount importance. (See Sect. 5.1 “Educational Aspects” for more details).

## 4.2 Curriculum for Middle School Students (“Robots for Kids”)

The summer courses consist of two one-week courses organized in cooperation with the “Wiener Kinderfreunde”.

Of the two courses, one is meant for beginners using the graphical programming interface from LEGO for their LEGO MINDSTORM NXT and EV3 robots while the other is an advanced course dealing with the same topics and using the same robotics but programming with the same C-like language used by the interns. The advanced course is taking place before the beginner course to avoid kids coming to both courses. While kids that visited beginner courses in previous years could use their experience to improve their programs and robots, taking part in both of them in succession could be negatively impacting the enjoyment of their peers and themselves because both courses deal with very similar topics.

Both courses are based on disciplines of the RoboCupJunior. Specifically, those disciplines are OnStage, Rescue Line and Soccer Light Weight League. At the start of each course and after a short introduction show of those three disciplines the children will be asked which one they prefer and divided into up to three groups each dealing

with one discipline. When interest in one league is disproportionately high, the group for the respective discipline will be split in half the kids interested in the least popular discipline in that course will be asked to move to other groups. Offering those disciplines in the courses provides natural advantages because the RoboCupJunior is already a focus of the UAS Technikum Wien which means that capable experts are available to teach those courses and the participants are introduced to the RoboCupJunior which might be a way for them to continue their engagement with robotics and programming. It also adds an incentive to return to the advanced course and apply their knowledge to another discipline.

The courses are structured similar to the internship itself beginning with a slightly simplified introduction to programming on the first day followed by more advanced exercises introducing the goals and requirements of the respective discipline. On the third and fourth day, the kids work mostly on their own, usually starting with the construction of their own robots and then beginning to rewrite the programs of their previous exercises. Depending on the progress and motivation of the students the fourth day can also include a short preliminary competition to show the children the remaining flaws in their programs and robots. Finally, the fifth day is concluded by a competition amongst the members of a group. Parents and friends of the students are encouraged to attend to give the students an opportunity to present their achievements as well as provide an additional level of motivation for them to build and program the best robots.

During the course the children work in groups of 2–3 on their robots. The courses are taught by one teacher each supported by the interns. Interns are each assigned 2–3 groups of students and are responsible for solving minor problems so the teacher can concentrate on the group as a whole. In addition, they provide an always present personal contact for their groups and can both help them with problems in construction and programming as well as continually encourage and morally support them.

## 5 Experiences

Since the internship was organized and lead largely by the same personnel, a significant amount of improvements in various aspects could be made, based on the experiences of the previous years.

### 5.1 Educational Aspects

In the current iteration, the internship includes several different educational aspects. The introductions, both to the interns and the students are conducted conventionally but include a large amount of practical exercises to give the opportunity to immediately make use of every new information. Every theoretical unit is also very much focused on explaining why something works the way it does in order to foster a larger understanding of programming and robotics instead of just teaching how to solve specific tasks. Throughout the courses there are also regular comparisons between the tasks at hand and real world applications of sensors and techniques so that the interns and students can develop an understanding of the relevance of their tasks in a professional setting.

In order to strengthen the interns' knowledge and motivate them to expand it on their own, the internship also includes several opportunities for them to teach others. Firstly, after the immediate introduction the interns will be assigned more advanced tasks to research on their own and prepare a short teaching session for the other interns. Those are typically based on what they have previously learned but include new functions or sensors whose specifics they have to research on their own as well as prepare suitable exercises for their fellow interns. They are also encouraged to make their session as interactive as possible and not just deliver a standard presentation.

Secondly, the interns are heavily involved in teaching the "Robots for Kids" courses. While the theoretical part is done by a university student, each of the practical exercises is heavily guided by the interns. Since they have already dealt with similar exercises they can now use those experiences to help the course participants. Knowledge they gain while teaching the kids can be transferred again into their research work.

## 5.2 Other Aspects

In addition to the mentioned educational aspects and the interns' introduction to research and teaching the internship also includes other elements meant to develop skills and awareness helpful for their future careers.

One part of that is the personal job interview every intern has to complete. During the internship itself, the main interviewer as well as the university students that were present during the interviews will conduct an extensive debriefing about best practices and common mistakes made in job interviews. They also more specifically discuss mistakes made by interns and how to improve their curriculum vitae, application and general demeanor during a job interview.

To foster deeper understanding of teaching, regular public feedback discussions will also be held with the interns during the course of the internship. Those not only serve to inform the teachers on the opinions of the interns but are also meant to give them an opportunity to think about how they would improve the proceedings of the internship.

Consolidating the last two points, a personal, private exit interview will be held with each intern discussing their specific shortcomings and successes both during the job interview and the internship itself.

Lastly, to provide the interns with a tangible proof of their performance, each of them receives a personal employment reference letter, outlining all the strengths they presented during the internship both to help them in obtaining further jobs as well as to be an official statement of all their successes and outstanding achievements for themselves.

Despite the corporate social responsibility of the UAS regarding the offer of extra-curricular activities, one goal is to motivate middle and high school students to start a bachelor study program at the UAS later on. First experiences show that the motivation is high and that students do not fear the contact with technology after participating at a course or completing an internship. Moreover, statistic material delivers that already three participants of the "Robots for Kids" course later on applied for an internship.

Additionally, following data was gained from a survey conducted on interns from 2016, three months after the internship. Nine out of fifteen interns answered the survey.

- 50% now consider a technical job, 10% are sure that they want a technical job and 30% already wanted a technical job before the internship.
- 90% of interns found teaching kids to be helpful for their own retention and 10% very helpful.
- 40% of interns found working with kids improved their social skills a little and 20% found them to be improved very much.
- All interns reported, that they found the job interviews and the subsequent discussions of best practices and problems specific to them to be very helpful for their future.

### 5.3 Financial Aspects

The salary of the interns is heavily supported by the Austrian Research Promotion Agency (FFG) while the “Robots for Kids” courses are organized in cooperation with the “Wiener Kinderfreunde”. Profits from the kids courses go towards paying the two university students responsible for training the interns and teaching the separate groups of the kids courses. In order to support up to three groups of children (divided into the three disciplines of the RoboCupJunior initiative), a third university student or an older, more experienced intern will also be hired for the duration of the courses (last weeks of August). Regarding available financial resources of the UAS the summer school itself is quite cost saving.

## 6 Conclusion and Future Work

In its current form the internship and the summer courses harmonize very well with each other, providing both support and benefit to each other as well as being important in their own right. The internship has the major advantage of being fully financed by itself which ensures its continued existence and has growing retention rates both in terms of returning kids as well as interns that return to the UAS Technikum Wien for other jobs, further engagement with robotics in their spare time or to study robotics or computer science.

In order to continue development in this sector the work of the authors, current plans include founding a robotics club in cooperation with the UAS Technikum Wien to further support both the course participants and the interns’ interest and engagement with robotics throughout the year. This includes workshops for specific topics and supervised access to materials such as robots, computer equipment, 3D printers, personal programming and electronics support. This robotics club will be a program to support the already existing activities such as the RoboCupJunior Austrian Open competition and the UAS Technikum Wien’s engagement in various schools in Vienna with introduction courses and classroom support for new robotics and computer science classes.

Lastly, to further improve the effectiveness and knowledge gained, an expansion of long term studies regarding the impact of these activities on student numbers, gender distribution and general interest in robotics and STEM subjects will be made starting this year. A constant evaluation and improvement cycle is the main reason this program has progressed to such a successful state will be a main staple in all future developments.

## References

1. RoboCupJunior Austria Official Site, January 2017. <http://www.robocupjunior.at>
2. Hofmann, A., Steinbauer, G.: The regional center concept for RoboCupJunior in Austria. In: First International Conference on Robotics in Education, Bratislava, p. 1 (2010)
3. Wiener Kinderfreunde Official Site, January 2017. <http://www.wien.kinderfreunde.at/Bundeslaender/Wien>
4. Austrian Research Agency Official Site, January 2017. <http://www.ffg.at/en>
5. He, S., Maldonado, J., Uquillas, A., Cetoute, T.: Teaching K-12 students robotics programming in collaboration with the robotics club. In: 2014 IEEE Integrated STEM Education Conference (ISEC) (2014)
6. Karp, T., Maloney, P.: Exciting young students in grades K-8 about STEM through an afterschool robotics challenge. *Am. J. Eng. Educ.* **4**(1), 39–54 (2013)
7. Morgan, K.P.: Educational Robotics as Leadership Development for Youth, Master thesis, The University of Nebraska (2013)
8. Pivkina, I.: Peer learning assistants in undergraduate computer science courses. In: 2016 IEEE Frontiers in Education Conference (FIE) (2016)
9. Preszler, R.W.: Replacing lecture with peer-led workshops improves student learning. *CBE Life Sci. Educ.* **8**(3), 182–192 (2009)
10. Shuster, M., Preszler, R.: Introductory biology course reform: a tale of two courses. *Int. J. Sch. Teach. Learn.* **8**(2) (2014)
11. Falchikov, N.: Learning Together: Peer Tutoring in Higher Education (2001)
12. Vermeulen, R.C.M.: Narrowing the transfer gap: the advantages of “as if” situations of training. *J. Euro. Ind. Training* **26**(8), 366–374 (2002)
13. Sklar, E.: A long-term approach to improving human-robot interaction: RoboCupJunior rescue. In: 2004 IEEE International Conference on Robotics and Automation, Proceedings ICRA 2004 (2004)

# Methods for Managing Student-Driven Robotics Research

Cem Avsar<sup>1</sup>, Lennart Kryza<sup>2</sup>, and Klaus Briß<sup>2</sup>

<sup>1</sup> beSpace GmbH, Berlin, Germany  
cem.avsar@bespace.de

<sup>2</sup> Department of Aeronautics and Astronautics, Technische Universität Berlin, Berlin, Germany  
{lennart.kryza,klaus.briess}@tu-berlin.de

**Abstract.** Robotics plays a major role in educating engineering students and can be integrated into a curriculum in many ways. Problem-based lecture courses allow only a limited technical complexity due to the number of their corresponding credit points. From all project categories described, long-term student-driven research projects are the most challenging regarding management. Several of such projects have been initiated at the Technische Universität Berlin (TUB). One example is the rover SEAR which participated in space robotics competitions. During many years of experience in hands-on education, the major characteristics and challenges of student-driven projects were identified. Proven methods and good practices for addressing these challenges are discussed. If project leaders carefully respect all conditions of a student project, they can reduce the risks by implementing measures to keep a steady workflow. Thus, the success rate of complex robotics research conducted by a student workforce can be increased significantly.

**Keywords:** Space · Robotics · Education · Hands-on · Students · Research

## 1 Introduction

Problem-based courses have proven to be effective for engineering training and further help students in improving their teamwork and communication skills [1,2]. Robotics play a major role in the conception of problem-based courses. The TUB, Chair of Space Technology has over 50 years experience in educating space systems engineers by utilizing technologies in education like satellites, sounding rockets and robots for planetary exploration [3]. Valuable experience has been accumulated at TUB over time through applying different methods of engineering education and evaluating the processes.

The most common approach to embed hands-on activities into a curriculum is to conduct projects with students in scope of a lecture course. However, these projects come with a number of constraints. For example, they are often timely limited to the academic term. The working hours that students can invest are

limited with respect to the credit points for the lecture course. Even though some degree of freedom in the design of a technical system may be offered, the complexity of such projects is limited.

A unique learning experience is created when students are given the opportunity to participate in a scientific project with practical application. This experience can add value to students by exposing them to a project which resembles conditions that are met in real professional life. On the other hand, the faculty can gain a lot by a highly motivated student workforce. There is a number of examples for educational research projects at TUB which are categorized in Sect. 2.

The characteristics of such projects are different when compared to projects which are tailored for specific lecture courses. On one hand, the characteristics of an educational robotics research project are shaped by the project's requirements and constraints. Typically, the main drivers are the schedule, the available human resources for supervision and the technical complexity. On the other hand, the project work-flow is highly dependent on the fashion in which students are involved in it. A student is under significantly different circumstances than a full-time scientific employee. There are a number of aspects which need to be considered when building a project mainly on student workforce. The main challenges which are described in Sect. 3 are related to many factors, amongst which are the workload, commitment, level of expertise and many more.

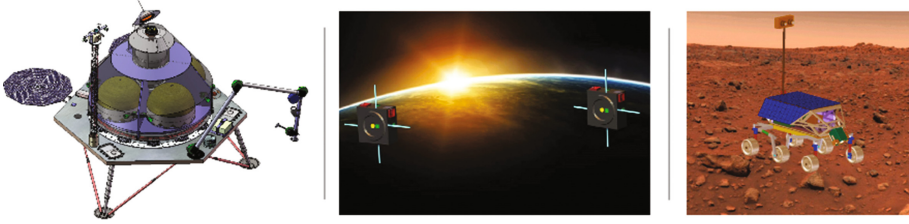
Years of experience in student projects, variations in methods of student project management, feedback from evaluations and regular lessons-learned sessions have led to a number of methods that can be applied in the management of student projects. The methods described in Sect. 3 can help leaders and members of student-driven projects to reflect their own experiences and make adjustments in project management, thus achieving higher satisfaction regarding education and project results.

## 2 Categories of Student-Driven Projects at the Technische Universität Berlin

The projects at the TUB, Chair of Space Technology can be separated into three different categories. Although these categories and the examples mentioned might seem to be specific for the domain of space engineering, they can very well be applied similarly to other fields of engineering. All categories have different characteristics.

### 2.1 Problem-Based Lecture Courses

A typical example for a problem-based lecture course project is a feasibility study on a novel space mission. In this type of project, mission objectives, requirements and constraints are given to a group of students. Work packages are distributed



**Fig. 1.** Examples of feasibility studies conducted at TUB (from left to right): Mars sample return mission, nanosatellite formation flight, micro Mars rover.



**Fig. 2.** Impressions from CanSat projects conducted at TUB (from left to right): CanSat electronics, CanSat launcher, CanSat structure.

among the group and the students adjust their concepts constantly using loop iteration. All aspects of the space mission like e.g. the spacecraft design, the operations and management are being studied. The study results are presented in front of a review board. Figure 1 gives impressions on a variety of feasibility studies conducted at TUB.

Another example with more hands-on elements is the development of CanSats. CanSats are small satellite-like systems in shape of a standard soda can. With this format, they are well suited for a launch on-board a small launcher which typically carries them up to a few hundred meters altitude and pushes them out. The CanSats are attached to a parachute and perform different types of missions like e.g. sending telemetry from different sensors for measuring inertial and atmospheric data. More sophisticated CanSats are able to perform controlled landing at given coordinates. Figure 2 gives impressions on the CanSat activities at TUB. The running time of these projects typically ranges from four weeks to four months at TUB. The average student group size is 15. In these courses, students are expected to bring disciplinary knowledge. The main objectives are to develop personal, inter-personal and professional attributes as well as designing, implementing and operating skills.





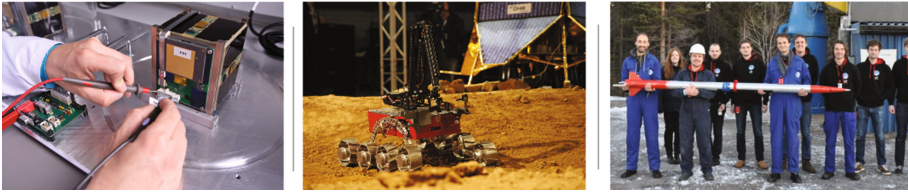
**Fig. 3.** Examples of space-related experiments conducted at TUB (from left to right): Satellite separation test during parabola flight, sun sensor experiment during sounding rocket flight, boom deployment test during parabola flight.

## 2.2 Space-Related Experiments

Often, space-related experiments in scope of a research project or a publicly funded program by the German Aerospace Center (DLR) or the European Space Agency (ESA) are being pursued by students. Typical examples are testing new technologies on-board a sounding rocket, a stratospheric balloon, a zero-gravity drop-tower or on a zero-gravity parabola flight. Figure 3 gives impressions on a variety of space-related experiments which have been conducted at TUB. The team sizes are usually smaller than five students and the learning experience in these projects is very intense. Preparation, execution and post-processing usually takes between one and two years. These types of projects are seldom part of lecture courses. They are project-dependent initiatives for which students are invited to participate as an extra-curricular activity. In many cases, students develop their thesis in scope of such a project. The projects are usually intensely supervised. Besides the scientific goals, the main objectives for students are to develop disciplinary knowledge as well as designing, implementing and operating skills. Since students are often involved with other international students and experts in scope of a funded program, personal, interpersonal and professional attributes are trained as well.

## 2.3 Student-Driven Research Projects

Typical examples of student-driven research projects at the Chair of Space Technology are the development of small satellites, robots and rockets. Figure 4 gives impressions on a variety of student-driven research projects which have been conducted at TUB. These projects usually receive public funding and have scientific goals besides the main goal of enabling practical education for students. The duration of these projects ranges from one year to several years. While typically a batch of 10 to 20 students work on such a project simultaneously, the total student participation over many semesters reaches numbers close to 100. These projects are also embedded in lecture courses and are often conducted through theses and voluntary activities. Besides the scientific goals, the main



**Fig. 4.** Examples of student-driven projects conducted at TUB (from left to right): CubeSat BEESAT-3 [4], autonomous planetary rover SEAR, sounding rocket DECAN [6].



**Fig. 5.** Evolution of SEAR (from left to right): SEAR in 2013, SEAR in 2015 (Image by Christopher-Eyk Hrabia), SEAR 2 for 2017 (Image by Marcus Meuser)

objectives for students are to develop disciplinary knowledge as well as designing, implementing and operating skills. Since students are often involved with other international students and experts in scope of a funded program, personal, interpersonal and professional attributes are trained as well.

Satellites, planetary robots and rockets make up the major standard developments in space engineering. The space robotics projects at TUB shall be shortly described to give more insights into the outline and complexity of a student-driven research activity.

**Space Rover Projects at TUB.** Planetary rovers have been developed at TUB for over a decade now. Major advancement in autonomous operations have been made in 2013 in preparation for the DLR SpaceBot Cup, the first German space robotics competition. Students developed the Small Exploration Assistant Rover (SEAR) within a time frame of less than nine months. In preparation for the competition held by the German Aerospace Center, students started to develop the rover from scratch in scope of a lecture course which was supervised by staff members. The developments included the complete mechanical, electrical and software design as well as manufacturing, test and operations. The main goal

was to demonstrate fully-autonomous operations on a planetary surface test-bed. SEAR also participated successfully in the second competition in 2015. Over 50 students took part in this rover activity at TUB and over 10 thesis topics have been dedicated to this project so far [5]. The result has been an active development of the rover platform for almost four years, with the evolution of the system shown in Fig. 5.

These experiences have led to the decision to participate in further competitions in the future. Currently, the two new rover projects Berlin Exploration Assistant Rover (BEAR) and the PicO LavAtube Rover (POLAR) are being developed.

### 3 Challenges of Student-Driven Robotics Projects and Methods for Managing Them

From the three different project categories described in Sect. 2, the student-driven research projects are the most challenging regarding management. This section describes the major characteristics of student-driven research projects and the corresponding challenges which leaders and members faced at TUB in the past years. Methods which have been applied at the Chair of Space Technology for addressing the challenges are presented.

#### 3.1 Working in Interdisciplinary Teams

**The Challenges.** Developing a project like e.g. an autonomous rover demands for a team of people who cover a set of different engineering skills like electronics, software and mechanical design. Most aerospace study programs have a focus on covering a rather broad range of aerospace specific topics. They do not educate electrical engineers, software designers or mechanical engineers. Although some students might have a certain skill in one of the fields, most aerospace students lack of the proper skills to perform detailed design in the mentioned engineering disciplines. Only by properly injecting disciplinary knowledge within the scope of a complex student-driven project, the participants will be able to contribute to it.

**Method: Assigning Tasks and Responsibilities.** We have found that students are more motivated when they are allowed to pick assignments. Studies on characteristics of an effective team show that all members should have a share in the leadership and ownership of the team's tasks [8]. In the rover projects, the work packages are treated as if they were advertised jobs. Students apply for their three favorite work packages by writing a short motivation letter and adding a CV. The supervisors select the best candidates for a certain work package while respecting the wishes of the students. In most cases, good allocation was possible which was satisfying for all students and the supervisors. An example of a work package description is given below.

JOB DESCRIPTION
-----------------

<p><b>Electrical engineer for design of on-board electronics</b></p>
--

<p>All components on the rover including on-board computer, motors, sensors, actuators etc. have to be supplied with electrical energy. An appropriate energy storage has to be allotted that can provide all systems with energy for at least one hour. For all components, appropriate voltages have to be conditioned. For some electronics, adapter circuits have to be designed.</p>
---

<p>Tasks until course date 4:</p>
-----------------------------------

<p>- <i>Presentation of a block diagram that shows all electrical components, their properties and interfaces.</i></p>
--

**Method: Working With Other Faculties.** Involving students from other faculties has turned out to be a major asset to a project and to the quality of education. Teams benefit from the combination of people with diverse characteristics, skills, knowledge, and capabilities [7]. We recommend to advertise projects at other departments to diversify the student team. It was found that students are glad to help each other out with their strengths when this will advance the project.

### 3.2 Low Funding

**The Challenges.** Whereas scientific research projects are mostly well-funded, student projects have comparably few resources available. In most cases, the funds do not cover scientific researchers, making it necessary that scientific staff and lecturers dedicate their time in addition to their main duties. In best case, the funds for student-driven projects cover the working materials and a few student assistants. However, the number of student assistants is by far not sufficient to conduct a project solely based on them. These projects practically depend on extra student workforce. This creates challenges in the supervision and in keeping a steady workforce for properly conducting the projects. In many cases, students want to continue with a project even though the funding has expired. For these cases, the department should have a long-term plan to allow for continuation, thus keeping know-how and bridging gaps until another funding opportunity comes up.

**Method: Involving Staff Members.** Although most staff members of a faculty are occupied with their research projects, many are gladly willing to support student activities. Student project leaders should make use of experienced experts. In their role as mentors and reviewers they can help to reduce technical and managerial risks in the project. Having the work reviewed by expert staff members can raise motivation in the student team and lead to more accurate results. Some experts might even take off some load from the team by contributing even more. In any way, the experts are helpful when thesis supervision is needed.

### 3.3 Drain of Expertise

**The Challenges.** The typical study duration of a Master student is four semesters, whereat the final semester is often used for preparing the thesis. The scientific student projects listed in the examples in Sect. 2 last over one or more years. Students joining in their second or third semester often leave the projects soon while first semester students or even less experienced undergraduates join the projects. This leads to a loss of expertise and knowledge. This issue needs to be addressed carefully to maintain a steady project work-flow.

**Method: Establishing Knowledge Transfer.** There are different concepts for ensuring that knowledge is sustainably transferred from experienced students to following participants. Students actively teaching or mentoring each other is a widely used method. Design projects at the Purdue University in Alcoa are vertically integrated, consisting of a mix of freshmen, sophomores, juniors and seniors thus benefiting in from the year to year continuity as well as different perspectives and experience levels in the teams [9]. However, there are risks involved when solely building knowledge transfer on mentoring. The mentors will have to teach the basics to every student joining the project, thus spending a lot of time on knowledge transfer. In an ongoing project with a deadline like e.g. a competition, this loss of valuable time of expert team members can be fatal. We recommend to set up a knowledge base to make the basics easier accessible to students. This can be based on a wiki and can include tutorials on necessary basic skills, project-related technical contents and even video lectures. Tutors, supervisors and senior students can still act in guiding and helping new participants, but they won't have to invest too much time on explaining the basics. On the other hand, new participants have full-time access to knowledge and can comment and improve the knowledge base for next generations. Students from other faculties can also get better insights into the project.

**Method: Setting Project Goals and Milestones.** From our experience, students start to increase the workload very prior to a homework submission or an exam. To have a steady progress in a student-driven project, we recommend to set a high mission goal and to define a number of milestones on the way. A very effective method is to set a milestone within the first two weeks of a lecture course. On this initial milestone, students can be demanded to present a summary of the previous work done regarding their newly assigned work package and to present their working plan for the semester. This graded presentation will help the students to dive into the project in the very beginning. The next milestone can be set after another two weeks when they need to present their detailed concept for approaching the problem. Regular graded milestones help the project to pick up pace and help the students to practice their presentation skills. During this process, students should regularly be reminded of the common goal of the project.

### 3.4 Different Levels of Expertise

**The Challenges.** Interdisciplinary courses require a broad range of students with different expertise. Even though project-based lecture courses have a few basic prerequisites, they are open to many students. Naturally, students have different levels of expertise. In a typical research project, the scientific staff is carefully selected before employment. In a student-driven project conducted in a lecture course, basically every student is accepted. The different knowledge and skill levels of students within the team have to be considered when planning execution of the work. There needs to be a way of embedding students from different fields and on different levels without invoking risks to the project.

**Method: Establishing Self-Evaluation.** When multiple students are assigned to one work package, it is often hard to see the contribution of individuals. If individual grading instead of team grading is preferred by the faculty, individual assignments should be clearly confined. Individuals should present their own contribution in reviews. A questionnaire which is filled out by other students to evaluate other teams can help to identify the top performers if team grading is applied. The questions can be related to volume and depth of the work, the communication with other teams and the personal initiative. Some engineering educators even recommend to use web-based peer evaluation tools and use the results as a modifier, because people who work on a project usually know best how people in the project perform [10,11].

**Method: Promoting Student Leadership.** Since it is difficult for supervisors to dedicate time to a student project, they should consider assigning student leaders. The most talented candidates might act in a systems engineering role, thus reducing contact time to solve minor technical issues as supervisor. Students in these positions are highly motivated and deliver promising results. They are able to strengthen the whole team. In addition, they gather valuable experience by acting in a leading position. In many cases, these students have been promoted to scientific researchers. Since students usually don't have a strong level of hierarchy, the scientific supervisors should communicate the role of the leading students very well with the whole team.

### 3.5 Low Commitment and Unpredictable Situations

**The Challenges.** In contrast to employed scientific staff, students have a much more versatile agenda. Lecture courses, homework, student jobs, extra-curricular activities, loss of interest, false expectations and semester dependent short term changes of plan make the short- and long-term schedule of a student hardly predictable. Project leaders at TUB have faced situations in which a student in key role suddenly dropped out of a project due to too heavy workload, an internship opportunity or private matters. The project leaders also can't predict if an experienced student will be able to continue with the project in the following

semester. Although most students were able to make reliable commitments, the circumstance of non-dependability invokes a risk to a project's success.

**Method: Rewarding Students.** Besides getting credit points or writing theses, many students are putting their free time into a project. Rewards can motivate the students to continue with a project in spite of difficulties. A major reward for a student is to be able to participate in a robotics challenge, rocket flight or satellite launch campaign. The benefits from the experience and contacts that can be made are huge. No less, such a thrilling event is kept as a memory for life. Team-building events, get-togethers and certificates can also be useful to keep up the team spirit. Although extrinsic motivation factors like course grade and resume improvement are important to students, the outcome of a project can strongly rely on the intrinsic motivation factors that can be triggered by the elements mentioned above and by constantly communicating the mutual vision. Individuals can be motivated by both curiosity and perks, by both pleasure and reward, by both love and money [10].

### 3.6 Quality Management for Student Projects

**The Challenges.** Quality management can become a major concern. These projects struggle with limited budget and time, making too experimental designs less preferable. This leads to the necessity of establishing a professional quality management apparatus in a less professional environment. The main challenge hereby is to demand for the needed discipline.

**Method: Documentation.** Proper documentation during the whole design process enables all members of a project to get immediate access to up-to-date information. Students in the Space Rover projects are not only asked to document completed design steps, but also failures and mistakes along the way.

**Method: Feedback and Validation.** Less experienced students might not be able to evaluate their results in a way to meet a defined quality level. It is often up to the more experienced team members, especially the systems engineers, to ensure that results are validated. This is best done by a very close feedback loop and repetitive reviews. It means that students will present their findings on a very tight schedule, either during meetings or in fact-to-face conversation with systems engineers.

**Method: Using Modern Tools.** Tools for project management like e.g. Redmine<sup>1</sup> and version control software like e.g. git<sup>2</sup> are widely used in scientific and students projects at TUB. Other educators state that a system like Redmine makes a significant impact on the outcome of project results [12]. Software

<sup>1</sup> Homepage of project management tool Redmine. <http://www.redmine.org/>, last access: 01/17/17.

<sup>2</sup> Homepage of version control system git. <https://git-scm.com/>, last access: 01/17/17.

development in the project should follow established guidelines, such as that each software unit undergoes unit testing or that automated test procedures are used. These methods create a certain overhead. However, the benefits strongly outweigh the needed effort.

**Method: Focusing on Essentials.** An ambitious student group might come up with very creative ideas. Although this admirable, they might not have a clear picture of the feasibility behind a fully innovative approach. We recommend that students set a focus on the essential groundwork first. For example, basic functions like e.g. steering, RF communications, sensor data acquisition, software upload functions etc. should be implemented and verified before attempting to work on more sophisticated features like e.g. autonomous path-planning.

### 3.7 Keeping a Consistent Work-Flow

**The Challenges.** Keeping a consistent project work-flow poses a challenge in student projects, because of interruptions in the lecture free time or periods without funding.

**Method: Having a Regular Schedule.** In order to make continuous progress, a regular and obliging schedule is a core element. Meetings should be held on a regular basis, also during the lecture free time. These meetings are meant to inform everyone of current events and discuss pressing matters. An additional project day is also very beneficial to drive the project work. Project leaders should communicate the vision of the project clearly to keep the motivation level up during dry seasons.

## 4 Conclusion

The number of challenging characteristics of student-driven research projects show that many risks can be invoked when there is lack of awareness. Neglecting the risks can have severe consequences and can lead to failure of a project, especially when hard deadlines for e.g. student challenges are involved. When project leaders initiate a new student-driven activity, careful planning with foresight and respect for contingencies should be executed.

The presented methods have been proven and have led to success in complex student-driven robotics projects conducted at TUB. Since education implies a never-ending learning process for faculty, lecturers and project leaders around the globe are encouraged to carry these methods into practice and to share their experiences with the community.

**Acknowledgments.** Special thanks go to the German Aerospace Center for initiating and promoting space projects that are paying a major contribution to high quality hands-on education at the Chair of Space Technology for several years now. Thanks go to the project leaders and lecturers who are passionate about uniting research and educational activities. Thanks go also to all the dedicated students who have spent countless nights working on their projects.



## References

1. Dym, C.L., et al.: Engineering design thinking, teaching and learning. *J. Eng. Educ.* **94**, 103–120 (2005)
2. Batdi, V.: The effects of a problem based learning approach on students' attitude levels: a meta-analysis. *Educ. Res. Rev.* **9**(9), 272–276 (2014)
3. Brieb, K., et al.: From small satellites to planetary rovers and space probes - a university approach. In: 11th Low Cost Planetary Missions Conference, Berlin, Germany (2015)
4. Barschke, M., et al.: BEESAT-3: a Picosatellite developed by students. In: 61st German Aerospace Congress, Berlin, Germany (2012)
5. Kryza, L., et al.: Developing technologies for space on a terrestrial system: a cost effective approach for planetary robotics research. In: 1st Symposium on Space Educational Activities, Padova, Italy (2015)
6. Nitschke, C., et al.: The DECAN project at TU Berlin. In: 64th German Aerospace Congress, Rostock, Germany (2015)
7. Leiffer, P.R., et al.: Five curriculum tools to enhance interdisciplinary teamwork. In: American Society for Engineering Education Annual Conference & Exposition, Portland, Oregon (2005)
8. Dorf, R.C., Byers, T.H.: *Technology Ventures-From Idea to Enterprise*. McGraw-Hill, New York (2005)
9. Oakes, W.C., et al.: EPICS: experiencing engineering design through community service projects. In: American Society for Engineering Education Annual Conference & Exposition, St. Louis, Missouri (2000)
10. Goldberg, D.E., Sommerville, M.: *A Whole New Engineer*. ThreeJoy Associates Inc., Douglas (2000)
11. Layton, R., et al.: Software for student team formation and peer evaluation: catme incorporates team maker. In: American Society for Engineering Education Annual Conference & Exposition, Honolulu, Hawaii (2007)
12. Kanai, J., et al.: Redmine as a web-based collaboration tool in engineering design courses. In: American Society for Engineering Education Annual Conference & Exposition, Atlanta, Georgia (2013)

# Robotics Education in Saint Petersburg Secondary School

Sergey Filippov<sup>1</sup>, Natalia Ten<sup>1,2(✉)</sup>, Alexander Fradkov<sup>2,3,4</sup>,  
and Ilya Shirokolobov<sup>1,4</sup>

<sup>1</sup> Presidential Lyceum of Physics and Mathematics #239, St. Petersburg, Russia  
safilippov@gmail.com, nataliagten@gmail.com

<sup>2</sup> ITMO University, St. Petersburg, Russia

<sup>3</sup> Institute of Problems in Mechanical Engineering,

The Russian Academy of Sciences, St. Petersburg, Russia

<sup>4</sup> Saint Petersburg State University, St. Petersburg, Russia

**Abstract.** XXI century is the time of information technologies and automation. As the world technologies go forward, the standard forms of education at schools should be also supported by something innovative, complex and technological. So, it is obvious, that it is robotics that meets needs of the current education system and brings good results. Presidential Lyceum of physics and mathematics #239, which is located in St. Petersburg is known to be one of the best secondary school in Russia. Since 2008 there has been functioning Robotics Center for schoolchildren that provides a high quality education for those, who are keen on developing in the fields of technical education. The paper describes the Robotics Center – unique place that has succeed to create from the very beginning a complex education system, that includes robotics courses for schoolchildren (basic robotics, electronics, applied mechanics and programming), competitions and festivals, camps, courses for teachers and other activities that join together children, their parents, teachers and enthusiasts, who are keen on robotics. Robofinist project and festival, that unite people, who are keen on robotics, are shown and the outcomes of the education system that show its efficiency are described.

**Keywords:** Robotics in education · Secondary education

## 1 Introduction

Robotics is becoming a part of our life. The more we live, the more sophisticated it becomes. Great scientists design robots that replace people in hazardous situations, help them in everyday life and even become a part of their bodies. It is from Universities that these scientists graduate. And they all came to Universities from schools, where they had been taught. The sequence shows that it is important to start teaching children robotic skills at schools. Even if they don't become scientists or engineers, all the basic skills robotics gives will help them in future: project based approach, knowledge about control theory, good

algorithmic and calculating skills, patience and accuracy. The world community understands it so it is not a surprise that robotics is popular in educational contexts nowadays. The positive effects of robotic education can be proven by experiences of robotics schools from all over the world.

As it was mentioned in the article [1], robotics offers new benefits in education. LEGO Mindstorms equipment and other robotics kits has positive effects on education and it was clearly shown in [2] and in [3]. The results of teaching robotics at secondary schools described in [4] show the positive impact robotics activities make on schoolchildren. All the above can be supported by the experience of Robotics Center, which is the main topic of the paper. The first results of collaboration among undergraduate students of St. Petersburg State University and students of Robotics Center are described in [5] and some examples of applied methods of control theory and robotics education and description of Robotics Center are presented in [6–10].

As it is mentioned, the paper is about the Robotics Center that started as a small robotics course in 2008 and later grew up into a huge applied science center. It has built its own education system and spreads it over Russia nowadays. It is officially called Robotics Center of Presidential Lyceum of physics and mathematics #239 (RCPML#239). It is one of the largest school robotics centres in Russia now. A complex system, that was created inside the Center is a successful example of a proper organization of the robotics education, that can be provided in 5–11 forms in secondary schools. The system includes the following parts: robotics courses for schoolchildren, competitions and festivals, camps and quests, courses for teachers, unique methods of control education. It includes 22 varied robotics courses and more then 700 schoolchildren master their skills in it. The courses in brief are described in Sect. 3. Robotics Center is a main robotics resource center and it provides courses not only for children, but for teachers, too. They usually last for one week and are free. Such courses help to share and broadcast the robotics experience all over the country.

To make the robotics community united a “Robofinist” project has been organized (Sect. 4) and a huge list of various competitions is held during the year. Moreover Robotics Center organizes summer robotics camps where children from all over the Russia join together for three weeks time and master their robotics skills (Sect. 5).

The whole system gives lots of positive outcomes. Since 2008 Robotics Center students have been awarded nine gold, seven silver and six bronze medals for taking part in International competitions like World Robotics Olympiad and Robotchallenge. One of them is a project, made by students of the Center, that won World Robot Olympiad (WRO) 2016. As a clear example of the results of education it is described in Sect. 6. All the fields of studies are supported by Science education Center of Institute of Problems of Mechanical Engineering Russian Academy of Sciences (IPME RAS) and lead Universities of St. Petersburg: ITMO University, St. Petersburg State University.

## 2 About Robotics Center

Presidential Lyceum of physics and mathematics #239 is a secondary school that is situated in St. Petersburg, Russia. It was founded in 1918 and is famous for a huge list of achievements: International Olympiads in mathematics, physics and informatics, world wide contests. One of the most successful branches that develops there is robotics. Robotics Center was organized in 2008 by teacher of informatics Sergey Filippov and in 2010 it has become the robotics resource center of our city. It was the first place in Russia, where robotics experience was spread fast by a huge list of activities. As a result an unique methods of how to teach children robotics appeared there and now are spread fast all over the country. It happened because of the open work of the Center that involves children, their parents, engineers and enthusiasts from all over the country. It grows up the community of young researchers and helps everyone to do the same.

Attention should be paid on the place, where all the activities are held. A basic robotic class consists of 3 parts: a classroom, a project lab and a teacher room.

A classroom is a place where the lessons are held with large groups of children. There are about 17 computers placed along the walls of the room and a row of a simple wide tables to assembly robots. Teacher has a special table, where all the additional sensors and power supplies are given. Sometimes small competitions and demonstrations of assembling are held there. There is also a place where training fields are put to run robots during the classes.

A project lab is a place, where small groups of children can join together with a teacher and work with their robotics project for a long time. It is usually a small computer class with a huge table for assembling and a row of shelves to keep all the projects. For example, in one of the labs of Robotics Center lives the famous Robot Greta, the winner of the WRO 2012. Greta plays hand clapping game with guests of Robotics Center.

A teacher room is a place, where all the unique sensors, batteries and high valuable equipment is kept. It is also a place where teachers can prepare for the future lesson and have a short piece of rest.

## 3 Courses

As the Robotics Center is a part of a lyceum, its main activity is teaching lyceum students and other schoolchildren robotics. Robotics classes are included in the program of lyceum 5–7 year students and during classes of informatics children get some robotic skills. Of course it is not enough to have just one or two lessons a week to be good at robotics. So there are 22 unique courses, taught in lyceum as a branch of additional education, that develop their robotics skills. All courses are divided into four branches: basic robotics, electronics, programming and applied mechanics. We advice children to study all of them in parallel to become a complex specialist, because each one provides specific knowledge and skills and their combination makes children complex specialists (Fig. 1).



**Fig. 1.** Robot Greta plays hand clapping game

The course hierarchy helps make a system, where each student can choose the way to develop and master engineering skills step by step.

Some of the courses in lyceum have become massive online courses, books, innovative education programs. Unique techniques that are invented by teachers of Robotics Center with the help of lead scientists from Institute of Problems of Mechanical Engineering Russian Academy of Sciences (IPME RAS) help to explain complicated control theory laws and mechatronics skills to schoolchildren in a simple and understandable way.

About 700 schoolchildren study in the RCPML #239 nowadays. Their teachers are programmers, scientists and engineers.

### 3.1 Basic Robotics Courses

At first basic robotics education starts with courses, where LEGO Mindstorms kits are used. In the very beginning LEGO Mindstorms RCX was used. Now NXT and EV3 kits are used. Basic robotics courses based on LEGO last for 3 years. Children are taught the basic principles of robotics: from calculating gear ratios to calculating the trajectory when finding out an optimal path from the maze.

During the first year of studies children acquire some basic programming skills with Robolab, EV3 Software and TRIK Studio. Text programming is gradually introduced during the second year with CeeBot and during the third year of studies children use RobotC for the same purposes. During the 7th year of studying they also make some free style robotics projects and participate in conferences and trade shows. Such kind of three year studies as well as a unique methodology inventions created in the Center make complicated robotic tasks a breathtaking



**Fig. 2.** A pupil of Robotics Center during sumo competition

game, that includes both constructing and programming. It is noticed that those who study programming during robotics classes usually achieve better results at their final exams at school (Fig. 2).

World Robot Olympiad [11] and RoboCup Junior [12] are the most well known international competitions for schoolchildren. One of the most entertaining tasks, which is usually represented there is robot soccer: autonomous robots made of LEGO or any other materials are supposed to play football matches with their opponents. So robot soccer and humanoid robots courses are provided in the Center. Figure 3 shows a RoboCup Junior soccer play between the Robotics Center team and a guest team from Moscow.

Of course LEGO [13] is a leading company that supplies school robotics, but in 2009 it was understood that not all the needs of Robotics Center students LEGO bricks could meet. It was clear that a special tool should be designed that can be used to teach school children robotics from the very beginning and till the time when they'll be able to use cameras, network connecting and so on. Fortunately, a new cybernetic kit "TRIK" [15] was designed in 2011 by a group of developers from St. Petersburg State University (SPbSU). Now TRIK [14] and its programming software TRIK Studio is one of the most successful robotic kits, made in Russia. It includes an unique controller with Linux OS inside, motors, sensors, camera for computer vision and other metal components (see Fig. 4).

And of course one of the best things about TRIK is its software. A TRIK Studio is a powerful development tool to program TRIK, LEGO NXT and LEGO EV3 robots using Javascript, C++ and simple specially designed visual programming language for children. It is a tool, that makes it possible for young children to build programs with the help of blocks and for students to program



Fig. 3. RoboCup junior soccer play

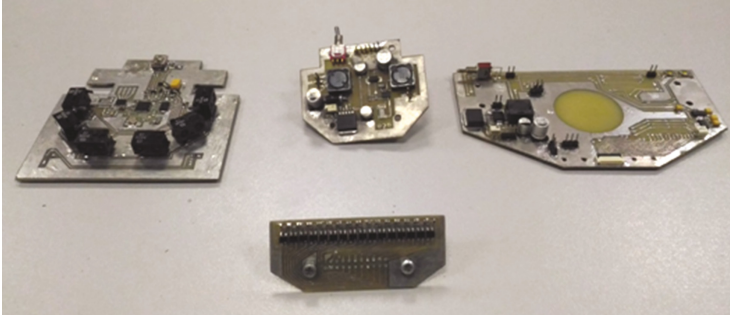


Fig. 4. TRIK cybernetic kit

a quadrotor. It also includes 2D modeling environment which meets educational needs when studying robotics without real kits.

### 3.2 Electronics Courses

It should be mentioned that the use of kits in most of the cases meets educational needs, but of course the best results can be achieved when using task



**Fig. 5.** Self made parts for robotics competitions

oriented materials, motors and sensors. That is why a second largest branch of robotics is being developed in Robotics Center. An electronic control systems is a branch that includes 6 courses: electronics, BEAM robotics, microcontroller based robotics (Arduino, Raspberry Pi). Some examples of the self made robots are shown in Fig. 5. Dip Trace is used to make a PCB board and then with the help of laser cut it is possible to make electronic devices during lessons at school.

### 3.3 Applied mechanics courses

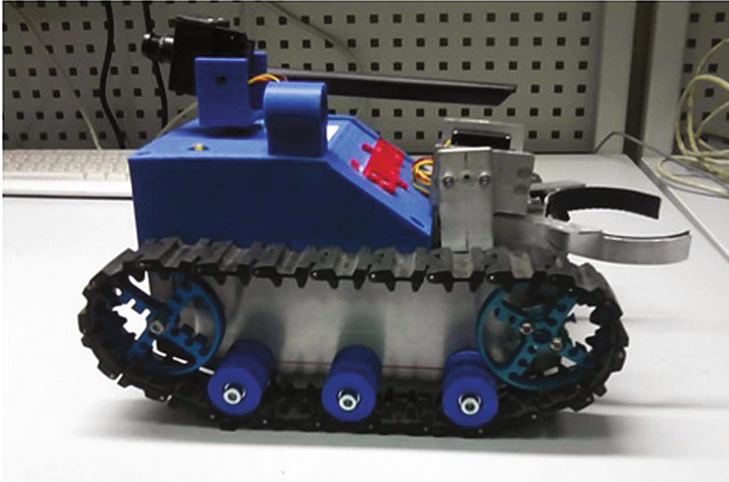
There is often a lack of details when making robots. Sometimes they are too long, sometimes too short, sometimes not strong enough. The branch of applied mechanics, represented by 3D modeling and Engineering Design courses helps children make everything they need to upgrade their robots. During the courses pupils study the basic principles of geometry, designing in CAD systems. They use Solid Works, Autodesk Inventor and Kompas 3D, so everyone can choose the most comfortable software. So if a child needs a part for the robot he or she makes a 3D model and prints it out on a 3D printer or cuts it out with a laser cutting machine for free. Moreover, during the course of Engineering Design children are taught how to prepare technical documentation: descriptions, structural schemes, schematic diagrams, printed circuit boards and so on.

An example of a robot build for RTC Challenge, is shown on Fig. 6.

### 3.4 Programming Courses

Special attention should be payed on developing programming skills. Six courses where children are taught programming teach children solve sophisticated robotics tasks. Some courses start mastering programming skills from the very beginning, like Pascal, but then students learn C++ and Java to create Android applications for their purposes. The most successful programmers come to the most difficult courses, where they are taught how to make quadrotor fly autonomously, mobile robot grab thing with the help of cameras and 3D vision.



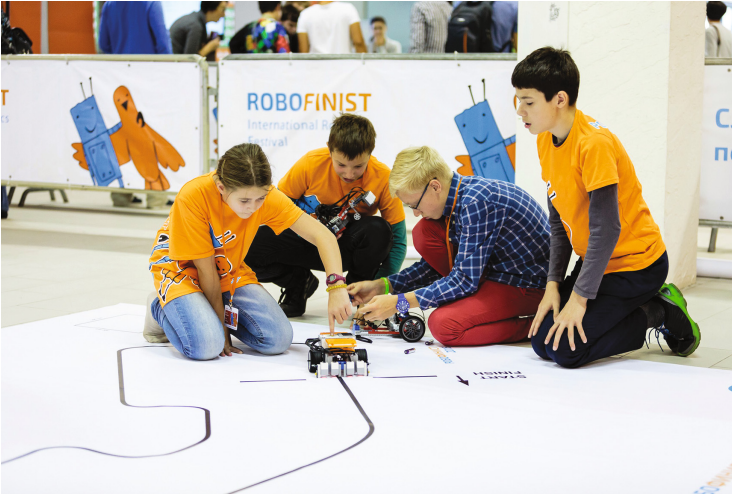


**Fig. 6.** Robot for RTC challenge

## 4 Robofinist

Of course at first it was hard for RCPML #239 to develop on its own. Lack of resources can easily stop the progress. But the Robotics Center has a support nowadays. In 2013 RCPML #239 and charity foundation “Finist”, which was founded by Temur Amindzhanov, head of company “Starline”, started the project, called “Robofinist” [17]. First, it is a website, where children share their knowledge and get registered on the current robotics events (courses, conferences or competitions). This electronic resource allows everybody organize robotics events of various levels: competitions at schools, cities and even regional competitions. It allows to open and control any type of competition, to invite judges and to use the rules that are listed there. It is a comfortable tool that is given to everybody, who wants to develop robotics in Russia for free. Second, it is a support for all the projects, initiated in Robotics Center, starting from small robotics games for children and ending with a support on RoboCup competition for teachers of Robotics Center.

And last, but not least is organizing a huge International robotics festival “Robofinist”, which is usually held in the end of September. The festival is free for all participants and visitors from all over the world. Everybody is invited to take part, to see the beautiful city. The festival is a real fun for children because it is not just a competition, but an event with lots of entertainment, robotics game zones and workshops. Symposiums and round tables for adults are also held during the festival. It is a huge event that joins together the growing robotics community and gives a boost to the most perspective tasks (Fig. 7).



**Fig. 7.** Participants of Robofinist 2016 festival

## 5 Activities: Competitions, Camps

A huge range of robotics events fulfill the year of young roboticists of St. Petersburg. A robotics year starts after the Robofinist festival and ends with it. During the year two competitions of the city level are organized by RCPML #239. There are usually about 20 kinds of competition each time: linefollowing, sumo, soccer, maze, air race, etc. About 500 robots take part in them. With the help of Robotics Center and Robofinist these competitions are run in more than 30 organizations in Russia and the number grows fast.

There are so many things to study in Robotics Center and it is just physically impossible for a child to develop in so many fields of robotics during an academic year. That is why one of the most expected events during the year is a summer robotics camp. The Robotics Center organizes the camp that gathers children from all over the country. They join together in common activities for three weeks time. Each week a pupil takes a course and one week at the camp becomes equal to semester of an academic year. Summer camps also include workshops, competitions and speed up the development of robotics and make the growing community united.

## 6 Project “OWC 2016”

One of the most vivid results of the activities of the Center is a project Ocean waste collector (OWC 2016) - an automated system of robots for collecting trash in the ocean. The project won first prize on the WRO 2016 in Open Category. Three students of Robotics Center who are about 13 years old have designed a unique model of the automated ocean waste cleanup system. OWC consists of

the following parts: surface waste collecting robot, conveyor and lift, underwater robot. Camera, that is located above the aquarium represents satellite or quadrotor. It helps to get the waste location and to send it to the surface robot, that gathers it and then transfers it to the conveyor belt, where sorting, cutting is done. If the waste is located under water, the underwater vehicle finds it and equips it with a special pontoon. It fills up with a gas, when the robot finds the sunken waste and makes it go up from the bottom. After that the surface robot delivers the waste to the lift. The project is an excellent example of how the whole system in Robotics Center helps children solve complex tasks. As they study basic robotics, they have build two mobile robots and conveyor with the help of LEGO Minstorms. When making a surface robot they used their knowledge in mechanics and 3D modelling to make special propellers to make desired movements on the water surface. When making the underwater robot they used their knowledge in electronics either: power supply case for underwater vehicle, special board for electromagnet, etc. Not to mention the fact that computer vision was used for navigation, which was taught during programming courses (Fig. 8).



Fig. 8. OWC team preparing for the contest

## 7 Conclusion

The whole complex of actions taken by Robotics Center and its partners shows good results. The number of children who want to develop their engineering skills

grows up day by day. Each year hundreds of new robotics classes appear all over the country and organizations that teach children for free become partners of Robofinist project. There are not so many people now, who have received this complex robotics education in Robotics Center and have graduated the universities, but those who did got good jobs and became successful. For example, four students of the first robotics courses are now lead engineers in companies that are connected with navigation, aerospace technologies and technical equipment design companies.

**Acknowledgements.** The work was supported by Government of Russian Federation (grant 074-U01) and by the Ministry of Education and Science of Russian Federation (project 14.Z50.31.0031).

## References

1. Johnson, J.: Children, robotics and education. In: Proceedings of 7th International Symposium on Artificial Life and Robotics, vol. 7, pp. 16–21 (2003)
2. Lindh, J., Holgersson, T.: Does LEGO training stimulate pupils ability to solve logical problems? *Comput. Educ.* **49**(4), 1097–1111 (2007). doi:[10.1016/j.compedu.2005.12.008](https://doi.org/10.1016/j.compedu.2005.12.008)
3. Benedettelli, D., Casini, M., Garulli, A., Giannitrapani, A., Vicino, A.: A LEGO mindstorms experimental setup for multi-agent systems. In: Proceedings of IEEE Conference on Control Applications & Intelligent Control. IEEE, pp. 1230–1235 (2009)
4. Dagdilelis, V., Sartatzemi, M., Kagani, K.: Teaching (with) robots in secondary schools: some new and not-so-new pedagogical problems. In: IEEE International Conference on Advanced Learning Technologies (ICALT 2005) (2005)
5. Filippov, S.A., Fradkov, A.L., Andrievsky, B.: Teaching of robotics and control jointly in the university and in the high school based on Lego Mindstorms NXT. In: IFAC Proceedings Volumes (IFAC-PapersOnline), Proceedings of 18th IFAC World Congress, vol. 18, pp. 9824–9829. IFAC, Milano, Italy (2011)
6. Filippov, S.A., Fradkov, A.L.: Control engineering at school: Learning by examples. In: 9th IFAC Symposium Advances in Control Education (ACE 2012), pp. 118–123 (2012)
7. Filippov, S.A.: Robotics for children and parents. Nauka (2013). (in Russian)
8. Filippov, S., Ten, N., Shirokolobov, I., Fradkov, A.: Teaching robotics in secondary school. In: Preparation of the 20th IFAC World Congress. IFAC, Toulouse, France (2017). (accepted)
9. Filippov, S., Ten, N., Fradkov, A.: Teaching robotics in secondary school: examples and outcomes. In: Preparation of the 20th IFAC World Congress. IFAC, Toulouse, France (2017). (accepted)
10. Filippov, S., Nikitin, D., Fradkov, A., Ten, N.: An experience of implementing robotics education in secondary and high schools. In: Preparation of the 25th Mediterranean Conference on Control and Automation (2017). (accepted)
11. World Robot Olympiad official website. <http://www.wroboto.org/>
12. RoboCup Federation official website. <http://www.robocup.org/>
13. LEGO education official website. <https://education.lego.com/>

14. Terekhov, A.N., Luchin, R.M., Filippov, S.A.: Educational cybernetical construction set for schools and universities. In: IFAC Proceedings Volumes, vol. 45(11), pp. 430–435 (2012)
15. TRIK community web-page. <http://www.trikset.com/>
16. Shirokolobov, I., Filippov, S., Luchin, R., Ovchinnikov, K., Fradkov, A., Oblapenko, G.: Control engineering at high schools and universities: project-based learning. In: Handbook of Research on Estimation and Control Techniques in E-Learning Systems, Chap. 11, pp. 141–170. IGI Global, USA (2016)
17. Robofinist festival. <https://robofinist.org/en/>

# **Workshops, Curricula and Related Aspects**

# LEGO WeDo Curriculum for Lower Secondary School

Michaela Veselovská<sup>(✉)</sup> and Karolína Mayerová

Faculty of Mathematics, Physics and Informatics,  
Comenius University in Bratislava, Mlynská dolina, 842 48 Bratislava, Slovakia  
{veselovska, mayerova}@fmph.uniba.sk

**Abstract.** In this paper we characterize curriculum for educational robotics which we iteratively developed within our dissertation research. We developed 11 activities with complex methodical materials, which teachers can productively integrate into their daily teaching and thus prepare attractive introduction for educational programming for pupils. We think that curriculum developed in this way can serve mainly for fulfilling modern educational goals of Informatics. With our new intervention pupils can develop important skills, knowledge and abilities, which they will use not only in school, but in their personal lives as well.

**Keywords:** LEGO WeDo curriculum · Educational robotics · Informatics · Lower secondary school

## 1 Introduction

Results from analysis of several publications indicate positive development of knowledge and skills with the use of educational robotics. Students acquired new knowledge during classes of programming, geography and robotics [1–3] during learning fundamental principles of evolution [4] and during classes of Systems [5]. Students acquired science process skills [5], social interactions [6] and problem solving skills [3]. Authors [7] believe that: “*not only are robots built on advanced technology but they also provide a tangible and physical representation of learning outcomes: a valuable aspect of employing them in education*”. Robotics also offers possibility for real learning, learning that is definable and measurable. Gura in [8] at page 115 also claim: “*When we take a look at almost any set of educational standards, there is a good chance to see connection to robotics. The longer we teach robotics, the more connection we will find.*” In our research we focused mainly on developing curriculum for primary and lower secondary school.

## 2 Methodology

Main aim of our research was design, implementation and verification of curriculum for educational robotics; hence it was integration of educational robotics into compulsory school subject Informatics at lower secondary school. Our curriculum consists of methodical materials for teachers. These materials contain educational goals, developed

competencies by students, selected methodic, content of lessons and they are in accordance with the National Educational Program in Slovakia. Therefore we had one main research question: *What should be specific objectives, content and form of curriculum for educational robotics for 10–12 years old pupils in compulsory school subject Informatics?* We used Design based research with four iterative cycles [9] as a main research strategy and we used qualitative methods of data collection and data analysis [10], including semistructured interview, observation (transcriptions and field notes), focus groups, audio-visual materials (photographs and recorded videos of student’s work and their products). We conducted our research in ordinary lower secondary school in small town near capital city of Slovakia. We cooperated with teacher who taught her pupils within compulsory school subject Informatics according our curriculum. During these classes there were also two researchers who were collecting data. We worked with 101 students aged between 10 and 12, divided into several groups during four years of our dissertation research. In one group there were from 9 to 14 students. These groups included boys and girls in different ratio, see Table 1.

**Table 1.** Pupils participated in our research.

Pupils participated in our research		Boys	Girls	Sum
	<b>1. iteration</b>			
	<b>1. group</b>	7	7	14
	<b>2. group</b>	7	4	11
	<b>3. group</b>	4	6	10
	<b>2. iteration</b>			
	<b>1. group</b>	7	4	11
	<b>2. group</b>	6	5	11
	<b>3. group</b>	6	3	9
	<b>3. iteration</b>			
<b>1. group</b>	7	3	10	
<b>2. group</b>	7	5	12	
<b>4. iteration</b>				
<b>1. group</b>	4	9	13	

During the implementation of our curriculum students worked mostly in pairs of two girls or two boys, with an occasional pair of a girl and a boy. Our curriculum introduces one of possible ways for introduction to programming for lower secondary school students that is appropriate and interesting.



### 3 Curriculum with LEGO WeDo

Our curriculum is based on a number of educational principles [11]. While designing activities for our curriculum we followed *principles of learning activity* [11] and several others recommendations from [8, 12]. In our activities we supported some constructionist ideas [13]: *learning by doing, hands-on activities; genuine achievement and own solutions, problem finding; hard fun and playful learning; learning through designing and creating; freedom to make mistakes; teamwork, communication, collaboration, sharing work and ideas*. Constructionist learning is creating opportunities for various representations of knowledge and it is supporting different types of learning styles [14]. In some activities we used instructions too, because modern cognitive process requires both approaches – instructions and construction of inquiry. These approaches are supporting each other and they are strongly connected in learning process [15]. Instructions often beneficially support construction of inquiry and vice versa [16]. In addition, with our curriculum we tried to engage the attention of all students in our classes. Therefore we followed ideas of “*Creative Science for Everyone*” [12]: *focus on theme, combine art and engineering, support storytelling and organize exhibitions*. We could engage the attention of more students with various interests when we follow mentioned principles [12].

Our curriculum consists of eleven activities for 5<sup>th</sup> and 6<sup>th</sup> grade students in lower secondary school and it contains methodical materials for every activity for teachers. Each activity takes 45 min, which is one school hour. Each methodical material contains even recommendations for teacher, several types of assessment of selected activities [17] and other additions such as worksheets for students, instructions for creating robotic models etc.

#### 3.1 Activity 1: What Is a Robot?

The aim of introduction activity is clarification and classification of students’ previous informal knowledge acquisition about a concept robot via discussion and creation a mind map about mentioned concept. In the end of this class students can formulate their own definition of robot based on created mind map.

#### 3.2 Activity 2: Making a Dream Car

The aim of this activity is familiarization with the content of robotic kit LEGO WeDo and its programming environment. Other aims are the development of ability to follow instruction via construction of attractive robotic model (left part of Fig. 1), familiarization with the axis of rotation of the motor and development of students’ fine motoric skills. Students should also use several selected commands to control robotic model (see right part of Fig. 1) such as *motor turning right (left), motor on for some time, motor power, motor off and play sound*.



**Fig. 1.** Robotic model, which was built according to instructions (on left) and pupils' own program to control it (on right).

### 3.3 Activity 3: Even Cottage on Robotic Foot Can Dance

The aims of this activity are the development of ability to follow instruction via construction of robotic model (see right part of Fig. 2), familiarization with the different type of axis rotation of the motor and development of students' fine motoric skills too, acquisition of knowledge and experiences with programming robotic model using commands for *control motor* and *play sounds* (see left part of Fig. 2). Last aim is the development of collaboration and communication skills via team work. Students are acquainting with commands wait for some *time and count loop* for the first time.



**Fig. 2.** Robotic model, which was built according to instructions (on left) and program from worksheet to control it (on right).

### 3.4 Activity 4: My Own First Assistant

The aim of this activity is the development of creative thinking through building and programming robotic model according students' imagination. The aim is also the development of ability to formulate and to defend their opinion and to agree on one

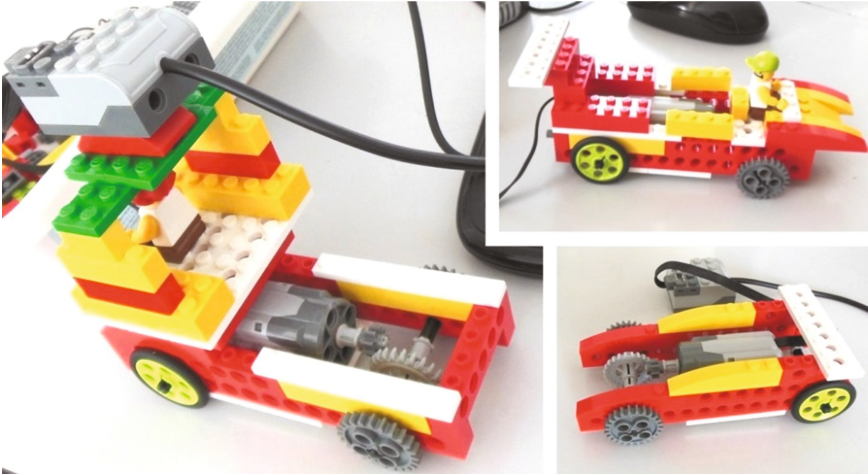


Fig. 3. Robotic models, which pupils created according to their imaginations.



Fig. 4. Programs to control robotic models on Fig. 3.

mutual resolution of possible problem via team work. Examples of pupils' robotic model are on Fig. 3 and examples of programs to control them are on Fig. 4.

### 3.5 Activity 5: Matthew's Hovercraft from the Future

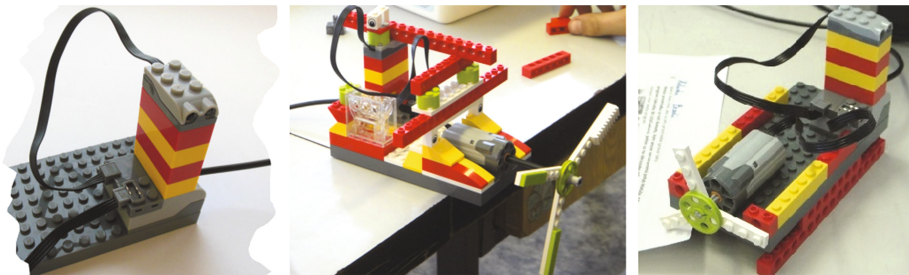
The aim of this activity is familiarization with the use of motion sensor in construction of robotic model (see left part of Fig. 5). Other aims are familiarization with functionality of motion sensor and with several options how to use it to control robotic model. Students are acquainting with the connection of command *wait* with motion sensor parameter. Last aim is the exploration of behaviour of robotic model when it is controlled with tasks from worksheet (see right part of Fig. 5).



**Fig. 5.** Robotic model, which was built according to instructions (on left) and one of the programs which pupils were exploring in activity 5 (on right).

### 3.6 Activity 6: Martina’s Vision of Lost Vehicle

The aim of this activity is development of creative thinking through building robotic model according picture which contains part of this model (see left part of Fig. 6) and finishing it according their imagination (see middle and right part of Fig. 6). Other aims are obtaining more experiences with the use of motion sensor via various types of tasks in worksheet and also clarification of the differences between selected commands and exploration of commands to *display some text* in virtual window.



**Fig. 6.** Examples of two robotic models, which pupils completed from picture (on left).

#### Example of one task from worksheet

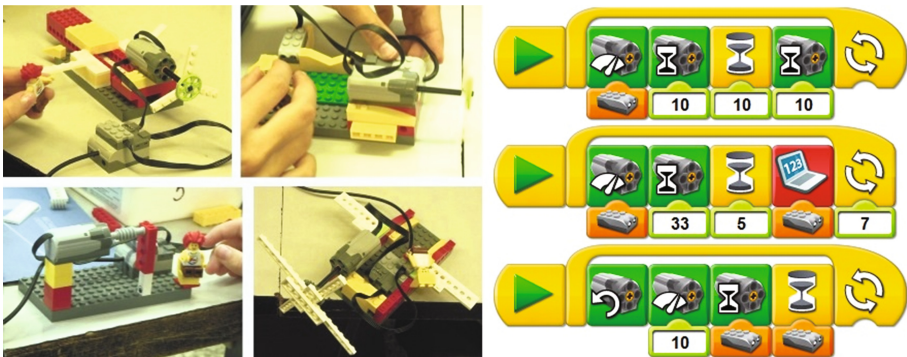
*Describe the differences in the behaviour of the robotic model in the control of program A and control of program B (Fig. 7).*



**Fig. 7.** Example of programs from worksheet, where pupils described differences between these programs.

### 3.7 Activity 7: The Creation of the Best Helper in the World

The aim of this activity is the development of creative thinking during building and programming robotic model created according to students' imagination (examples of students' models are on left part of Fig. 8 and examples of their programs are on right part of Fig. 8) but with some requirements as usage of motor and motion sensor in construction of robotic model and usage of selected commands in program to control this model. The aim is also to obtain more experiences with the use of motion sensor parameter with commands *motor power*, *motor on until*, *play sounds* and command *wait until*. Last aim is development of ability to agree with one mutual solution of problem.



**Fig. 8.** Robotic models, which pupils created according their imaginations.

### 3.8 Activity 8: The Movement of Modern Turtle

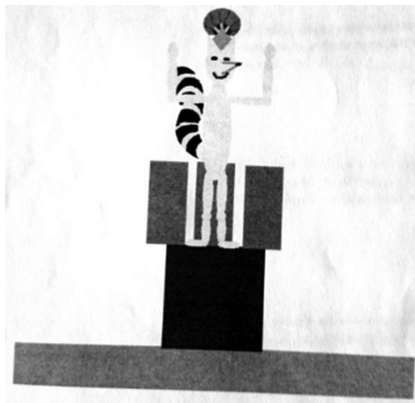
The aim of this activity is familiarization with the use of tilt sensor in construction of simple robotic model (see left part of Fig. 9). Other aims are familiarization with functionality of tilt sensor and with several options how to use it to control robotic model. Students are acquainting with the connection of command *wait* with tilt sensor parameter. The aim is also the exploration of behaviour of robotic model when it is controlled with advanced tasks from worksheet (see left part of Fig. 9).



**Fig. 9.** Robotic model, which was built according to instructions (on left) and one of the programs which pupils were exploring in activity 8 (on right).

### 3.9 Activity 9: Designing a Fairy Tale

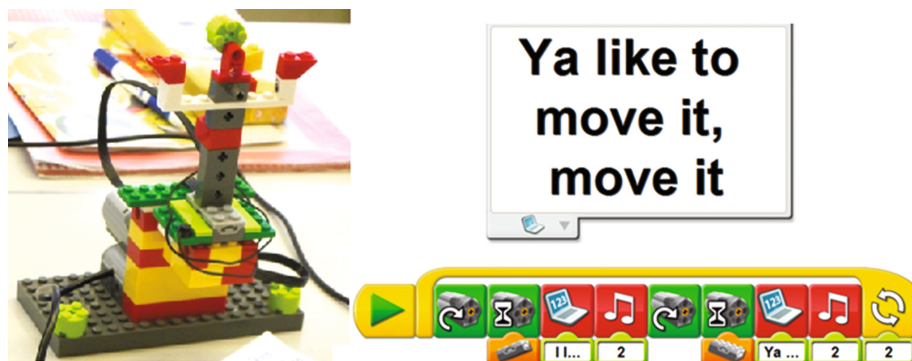
The aim of this activity is the development of ability to think about different possibilities and limitations of construction of robotic model in consideration of content of robotic kit. Other aims are the development of ability to think about different possibilities and limitations of programming of robotic model in consideration of limitations of programming environment for robotic kit LEGO WeDo, the development of creativity and ability to implement students' ideas through creating a sketch of robotic model (see Fig. 10) and describing its future behaviour in writing, using worksheet. Last aim is to demonstrate knowledge and skills acquired during previous activities.



**Fig. 10.** Sketch of construction of robotic model.

### 3.10 Activity 10: Creating a Fairy Tale

The aim of this activity is the implementation (see Fig. 11) of students' designs from previous activity. Other aims are to demonstrate knowledge and skills acquired during previous activities and the development of ability to take responsibility for work created in group.



**Fig. 11.** Robotic model and program created according to design (see Fig. 10). It is a figure lemur king Julien from Fairy tale Madagascar.

### 3.11 Activity 11: Fairy Tale for Everyone

The aim of this activity is the evaluation of students' own work included evaluation of limitation of robotic model and its program. Another aim is the comparison of first imagination about robotic model (its construction and description of its behaviour) with implementation of this imagination (as final product). Last aim is the development of ability to discuss and to defend own opinions. Students were creating presentations about all mentioned issues.

## 4 Findings

In Slovakia pupils start to go to lower secondary school at the age of 10. When we began to create curriculum for pupils this old we were inspired by results and findings from creating curriculum for younger pupils [18]. We implemented group discussion about a term robot as introduction activity. But these older pupils (lower secondary school pupils) did not want to discuss their ideas about term robot as enthusiastically and openly as younger pupils (primary school pupils). According our results [19] older pupils were afraid of their possible incorrect answers. So we **included team work** where pupils were **discussing their ideas about term robot in pairs at first**. Then we conducted **group discussion** during which teacher was creating mind map about robot according pupils ideas.

The next activity was creation of robotic model according to pupils' imagination in our curriculum for primary school pupils [18]. Therefore we implemented similar activity with older pupils. But there appeared several problems. At first pupils took a lot of time to decide what to build and they spent almost whole activity by building construction of robotic models. They spent only last few minutes on programming their models. Also **pupils had unrealistic expectations about construction and behaviour of their robotic models** [20]. During creating own robotic model pairs of pupils had problems with collaboration. There were situations where pupils had to rebuild their robotic models several times (because each member of pair worked on his/her own

robotic model) but in the end every pair finally created one functional robotic model. Therefore we included activities where **pupils created robotic models** and programs to control those models **following several instructions**. Hence pupils could acquire some conception about what robotic model could be created with selected robotic kit.

During first iteration of our research we conducted our activities following mainly constructionist ideas. But pupils took a lot of time to discover required phenomenon and there were several problems with understanding it. Teacher had to explain it to each pair separately, because many misconceptions appeared. Unfortunately we could create curriculum with less than 15 activities, because The National educational program for Informatics prescribes many other topics to teach in lower secondary school. So we decided to **incorporate instructions into introduction of some new phenomenon**. Then pupils could discover how this new phenomenon work during exploration of programs to control robotic models through tasks in worksheets.

Our results indicate that **activities with instructions should be varying with activities with constructionist ideas**. This means that pupils followed instructions to create robotic model at first, then they created their own robotic models and next activity contained instructions again, etc. When pupils were creating robotic models according to their imaginations there appeared several situations:

- We repeatedly reminded pupils, that their robotic model have to include motor or at least one of two sensors so it can be controlled with software.
- Pupils often did not include sensor in their robotic model, but they wanted to use the sensor parameter in their program.
- Pupils included one sensor in their robotic model, but they did not use it in the program that controlled the model.
- Pupils described their imagination about how the robot should work, but not their real program [21].

Based on these situations **we developed mandatory requirements for the content of the construction of a robotic model and its' program**. After we applied mandatory requirements, aforementioned situations did not appear again.

Last activities in our curriculum contain creation of larger robotic project which take three lessons. This project includes several aspects of compulsory school subject Informatics. We divided this project into three parts, where each part takes one school hour (45 min). At first pupils should plan how robotic model should look like and what should it do. It was beneficial to create worksheet for this activity, because pupils did not manage to keep track of their time and they did not distribute the work in the team. In our worksheet pupils could draw a sketch of construction of robotic model with colour pencils and they could write its future behaviour. They could also use graphic and text editor to fill it up. During second part of this project pupils were creating their models following their designs from previous lesson. During the last part of project pupils were evaluating their achieved goals, which they set in first part of project. In conclusion pupils still needed some form of management of their work. But **they manage to proceed according to their designs of robotic models** (they did not have unrealistic expectations about their models) and **to evaluate their work**.



## 5 Conclusion

In this paper we characterized curriculum for educational robotics which we were iteratively developing during four years of our dissertation research. Our curriculum represents one of practical and appropriate ways to integrate educational robotics into compulsory school subject Informatics at common lower secondary school. It was created for ordinary students, not only for gifted ones. This curriculum covers some topics prescribed in the National educational program for Informatics in Slovakia and it introduces one of possible ways for introduction to programming for pupils in selected age that is appropriate and interesting. Learning objectives in this curriculum encompass the development of computational thinking skills through educational robotics. With our curriculum pupils can develop important skills, knowledge and abilities, which they will use not only in school, but in their personal lives as well.

## References

1. Barker, B.S., Ansoorge, J.: Robotics as means to increase achievement scores in an informal learning environment. *J. Res. Technol. Educ.* **39**(3), 229–243 (2007)
2. Nugent, G., et al.: The effect of 4-H robotics and geospatial technologies on science, technology, engineering, and mathematics learning and attitudes. In: *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pp. 447–452. AACE, Vienna (2008)
3. Nugent, G., et al.: The use of digital manipulatives in k-12: robotics, GPS/GIS and programming. In: *Proceedings of the 39th IEEE International Conference on Frontiers in Education Conference, FIE 2009*, pp. 1–6 (2009)
4. Whittier, L.E., Robinson, M.: Teaching evolution to non-English proficient students by using lego robotics. *Am. Secondary Educ.* **35**(3), 19–28 (2007)
5. Sullivan, F.R.: Robotics and science literacy: thinking skills, science process skills and systems understanding. *J. Res. Sci. Teach.* **45**(3), 373–394 (2008)
6. Mitnik, R., Nussabaum, M., Soto, A.: An autonomous educational mobile robot mediator. *Auton. Robots* **25**(4), 367–382 (2008)
7. Mubin, O., et al.: A review of the applicability of robots in education. *J. Technol. Educ. Learn.* **1**, 1–7 (2013)
8. Gura, M.: *Getting Started with LEGO Robotics: A Guide for K-12 Educators*. International Society for Technology in Education, United States of America (2011)
9. Creswell, J.W.: *Educational Research: Planning, Conducting, and Evaluating Quantitative*. Prentice Hall, Upper Saddle River (2002)
10. Lichtman, M.: *Qualitative Research in Education*. SAGE Publications, Thousand Oaks (2013)
11. Pasch, M.: *Teaching as Decision Making: Instructional Practices for the Successful Teacherv*. Addison Wesley Publishing Company, Reading (1991)
12. Rusk, N., et al.: New pathways into robotics: strategies for broadening participation. *J. Sci. Educ. Technol.* **17**(1), 59–69 (2008)
13. Papert, S.: *The Eight Big Ideas of the Constructionist Learning Laboratory*. Unpublished internal document. South Portland, Maine (1999)
14. Kafai, Y., Resnick, M.: *Constructionism in Practice*. Lawrence Erlbaum Associates, Mahwah (1996)

15. Kabátová, M.: Konštrukcionistický prístup vo vyučovaní robotiky v príprave budúcich učiteľov [doctoral thesis]. Faculty of Mathematics, Physics and Informatics of the Comenius University in Bratislava, Bratislava (2010)
16. Feurzeig, W.: Demystifying constructionism. In: Clayson, J., Kalas, I. (eds.) Proceedings of Constructionism 2010. The American University of Paris, Paris (2010)
17. Veselovská, M., Mayerová, K.: Assessing robotics learning at lower secondary school. In: Information and Communication Technology in Education 2015, pp. 240–247. Pedagogical Faculty, University of Ostrava, Ostrava (2015)
18. Mayerová, K., Veselovská, M.: How to teach with LEGO WeDo at primary school. In: Robotics in Education, pp. 55–62. Springer International Publishing, Cham (2017)
19. Mayerová, K., Veselovská, M.: How we did introductory lessons about robot. In: Teaching Robotics, Teaching with Robotics, Padova, pp. 127–134 (2014)
20. Veselovská, M., Mayerová, K.: Pilot study: educational robotics at lower secondary school. In: Constructionism and Creativity Conference, Vienna (2014)
21. Veselovská, M., Mayerová, K.: Programming with motion sensor using LEGO WeDo at lower secondary school. *Int. J. Inf. Commun. Technol. Educ.* **4**, 40–52 (2015). Pedagogical Faculty, University of Ostrava, Ostrava

# Pythagorean Approximations for LEGO: Merging Educational Robot Construction with Programming and Data Analysis

Ronald I. Greenberg<sup>(✉)</sup>

Loyola University, Chicago, IL 60626, USA  
rig@cs.luc.edu

**Abstract.** This paper can be used in two ways. It can provide reference information for incorporating diagonal elements (for bracing or gear meshing) in educational robots built from standard LEGO® kits. Alternatively, it can be used as the basis for an assignment for high school or college students to recreate this information; in the process, students will exercise skills in both computer programming and data analysis. Using the paper in the second way can be an excellent integrative experience to add to an existing course; for example, the Exploring Computer Science high school curriculum concludes with the units “Introduction to Programming”, “Computing and Data Analysis”, and “Robotics”.

**Keywords:** Computer science education · Robotics · Computer programming · Data analysis

## 1 Introduction

Providing students with robotics experiences has become a popular and successful mechanism for broadening participation in computing and STEM more generally, retaining more students in these fields, and improving their learning. Robotics videos were found to be the most popular component of a series of brief computing outreach visits [12], and many studies have reported on successful programs using robots built from LEGO pieces, e.g. [2, 5, 7, 14].

Nonetheless, advice for students on how to build robots with LEGO kits is fairly limited, perhaps because LEGO is assumed to be a familiar medium from youthful play. An exceptional primer has been provided by Fred Martin [11]; in particular, it includes valuable tips regarding use of gears. It remains challenging, however, to incorporate any diagonal elements as opposed to placing each part across one dimension of an underlying regular 3D grid. To mesh gears along a diagonal or construct diagonal bracing, Martin suggests experimentation, though he notes the obvious applicability of the Pythagorean Theorem. In practice, such

---

An abbreviated preliminary version of this paper appeared as [6].

R.I. Greenberg—Supported in part by National Science Foundation grants CNS-1542971 and CNS-1543217.

experimentation with several cycles of building, disassembling, and rebuilding can be very frustrating to youthful robot builders.

This paper provides a more systematic approach to applying the Pythagorean Theorem to robot building. We use a spreadsheet to organize Pythagorean combinations that are close to exact, either with standard integral lengths or with some tricks that can be employed to place parts at spacings of half units (or sometimes even quarter units). Furthermore, the paper shows how to construct the spreadsheet using a simple program, and creating such a program and working with the resulting spreadsheet can be a good exercise in programming and data analysis to assign to students. In this way, students are motivated to complete the programming and data analysis due to its practical application to robot building tasks. This may form a particularly nice integration of the programming, data analysis, and robotics units of the Exploring Computer Science (ECS) curriculum [4], which has been implemented in many high schools in the United States. Initiated in Los Angeles, ECS is now in the seven largest US school districts and many other locations [3], and it has had a particularly strong impact in Chicago [1, 13]; it is also attracting some international attention.

## 2 LEGO Basics

One popular LEGO-based educational robotics program is Botball® [9], which has spread to many locations on four continents [10] and has major culminating tournaments/conferences in both North America (GCER) and Europe (ECER). Participating teams all work with a standard kit of LEGO and other parts assembled from those featured on the Botball web site [8]. Also, extremely widely used is the LEGO MINDSTORMS® kit [17]; for example, this was the primary recommended platform for the final “Robotics” unit of the Exploring Computer Science curriculum prior to the advent of other very inexpensive robots. The base kit for LEGO MINDSTORMS is similar to the Botball kit, though somewhat smaller, and various LEGO extension kits, available from a variety of vendors, also provide parts that are included in the Botball kit. While the MINDSTORMS kit refers to a package of “LEGO Bricks”, there are actually few if any traditional LEGO brick pieces in any of the standard robotics kits; most of the pieces can be ordered directly from [lego.com](http://lego.com) by searching for “Technic” in the brick name [16], with some parts seeming to require a visit to the service portion of the site [15]. The main workhorse pieces are referred to as “beams” at [lego.com](http://lego.com) or “liftarms” in Botball parts lists; most are completely straight or include a  $90^\circ$  angle. The width and height of a straight LEGO liftarm, as well as the space between adjacent holes along the length of a liftarm, are all one LEGO unit, which is 8 mm. Various connectors are available for linking these pieces together.

With standard LEGO robotics parts, the most straightforward approach is to place all pieces along straight lines in the underlying 3D integer grid. This paper initially restricts all parts to be anchored to this integer grid while also considering diagonal components. Later, we also consider some tricks that can yield spacings at fractional LEGO units. Martin also suggests some constructions

involving fractional LEGO units, but these spacings are primarily achieved by using traditional LEGO bricks, which we have noted are not generally very available in current LEGO robotics kits.

Botball names are henceforth preferred, but we also provide information for looking up parts at [lego.com](http://lego.com). The standard gears are listed in Table 1.

**Table 1.** Standard gears in the Botball Lego kit.

Botball description	Botball quantity	LEGO design #	lego.com name	Radius in LEGO units
8 Tooth	12	3647	GEAR WHEEL T = 8, M = 1	0.5
16 Tooth	6	4019	GEAR WHEEL Z = 16, M = 1	1
24 Tooth	8	3648	GEAR WHEEL Z24	1.5
40 Tooth	6	3649	GEAR WHEEL 40T	2.5

In a typical alignment of gears along a liftarm, the gears of radius 1 will only mesh with each other (at a distance of 2), while the other gears may be abutted to one another in various combinations at distances of 1, 2, 3, 4, or 5. At any given time, we will focus on an arbitrary two dimensions of the underlying 3D grid and explore ways of achieving diagonal placement of liftarms and/or gears, while most components run either horizontally or vertically. Employing diagonal liftarms may be a useful bracing mechanism that uses fewer liftarms, or, of potentially greater value, a mechanism to align gears (and transfer motion) along a diagonal. (At least one team in the 2016 Botball competition sought such a design to use gearing and place wheels below and in front of the motor mounts in a standard metal chassis. These design elements were intended to allow the robot to ascend a ramp and straddle certain pieces on the game board.) For diagonals of length at most 5, it may also be possible to place gear centers on the rectilinear grid but with the gears meeting along a diagonal.

### 3 Near-Integral Triples

We begin by exploring ways to construct diagonals that connect to the underlying integer grid. An appropriately constructed spreadsheet provides a convenient way to view near-integral Pythagorean triples that may be helpful in LEGO construction. For example, the macro in Fig. 1 initializes a Microsoft Excel® spreadsheet of relevant data, shown here with appropriate choices for integral lengths only ( $FRAC = 1$  and  $HYPRND = 1$ ).

Additional parameter choices in the code of Fig. 1 were to limit the lengths of all triangle sides to 14 (longest available with a single liftarm), and to limit the slope (long leg divided by second leg) to 5, since constructions with slopes closer to 1 are of greater interest here as opposed to essentially running along a horizontal or vertical. In addition to showing leg lengths, slope, and hypotenuse

```

Sub Initialize()
Dim MAXLEG, MAXHYP, FRAC, HYPRND As Integer
MAXLEG = 14 'Maximum allowed leg length
MAXHYP = 14 'Maximum allowed hypotenuse length
MAXSLOPE = 5 'Maximum allowed slope (2nd leg divided by short leg)
FRAC = 1 'Fractions of Lego units allowed
      '(1 for whole units only, 2 for halves, 4 for quarters, etc.)
HYPRND = 1 'Approximate hypotenuse by rounding to specified fraction
      '(1 for whole units, 2 for halves, 4 for quarters, etc.)
Cells.Clear
Cells(1, 1).Value = "Short Leg" 'A1
Cells(1, 2).Value = "2nd Leg" 'B1
Cells(1, 3).Value = "Hypotenuse" 'C1
Cells(1, 4).Value = "Approx Hyp" 'D1
Cells(1, 5).Value = "Error" 'E1
Cells(1, 6).Value = "Abs Err" 'F1
Cells(1, 7).Value = "Slope" 'G1
Dim shortstep, secondstep As Integer 'Counters for short, 2nd leg lengths
Dim row As Integer: row = 2 'Counter starting at 1st row after headings
For shortstep = 1 To MAXLEG * FRAC
  For secondstep = shortstep To MAXLEG * FRAC
    Cells(row, 1).Value = shortstep/FRAC 'A row
    Cells(row, 2).Value = secondstep/FRAC 'B row
    Cells(row, 3).Formula = "=SQRT(A" & row & "^2 + B" & row & "^2)" 'C row
    Cells(row, 4).Formula = "=MROUND(C" & row & ", " & 1/HYPRND & ")" 'D row
    Cells(row, 5).Formula = "=D" & row & "-C" & row 'E row
    Cells(row, 6).Formula = "=ABS(E" & row & ")" 'F row
    Cells(row, 7).Formula = "=B" & row & "/" & A" & row 'G row
    If Cells(row, 4)<=MAXHYP And Cells(row, 7)<=MAXSLOPE Then row = row + 1
  Next secondstep
Next shortstep
Rows(row).EntireRow.Delete 'Remove last row failing condition at loop end
End Sub

```

**Fig. 1.** A visual basic for applications macro to initialize a Microsoft Excel spreadsheet of Pythagorean triple data.

length, the spreadsheet shows the approximate hypotenuse length (rounded to the nearest multiple of  $1/HYPRND$ ), the error relative to the actual hypotenuse length, and the absolute value of the error.

Constructing a macro and spreadsheet of this sort is a feasible exercise for beginning computing students; the most difficult programming concept involved is nested for-loops. The other rudiments of working with VBA in Excel can be easily conveyed to students. (The task also can be simplified somewhat by generating only data values rather than formulas in all the cells.) This exercise can even be completed in Scratch (<http://scratch.mit.edu>), the environment regularly used for programming in the Exploring Computer Science curriculum and many other beginning computing classrooms. In the Scratch version, one

can generate rows of comma-separated values as items in a list, right click on the list to export to a file, and then read the file into Excel.

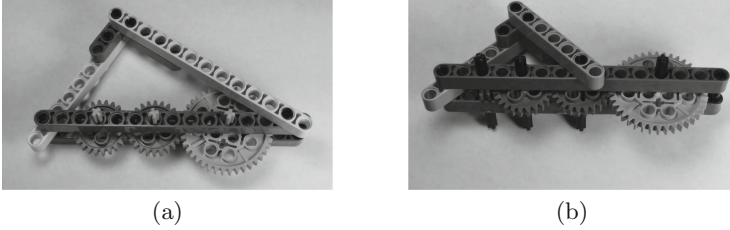
The macro of Fig. 1 generates a spreadsheet with 68 data rows. Sorting by absolute error, we find 11 rows with absolute error of less than 0.1 LEGO units as shown in Table 2. (For presentational convenience, the values here and in further figures are rounded to a small number of decimal places; this also helps provide opportunities to ask for something more on a homework assignment even if students find this paper.) It is actually possible to construct nearly all of the 68 possibilities as verified by constructing cases near the bottom of the sorted spreadsheet, but the ones listed in Table 2 involve less deformation of the pieces. As an example, Fig. 2(a) shows the case with least nonzero absolute error (7-11-13) used to hold a sequence of gears. (At least one combination remains unworkable, e.g., trying to build a 1-1-1 using a bent liftarm for the two legs.)

**Table 2.** The 11 near-integral (or integral) Pythagorean triples with sides  $\leq 14$  with least absolute error.

Short leg	2nd leg	Hypotenuse	Approx. Hyp.	Error	Abs. Error	Slope
3	4	5	5	0	0	1.333
5	12	13	13	0	0	2.4
6	8	10	10	0	0	1.333
7	11	13.038	13	-0.038	0.038	1.571
8	9	12.042	12	-0.042	0.042	1.125
4	8	8.944	9	0.056	0.056	2
4	7	8.062	8	-0.062	0.062	1.75
5	5	7.071	7	-0.071	0.071	1
5	13	13.928	14	0.072	0.072	2.6
5	11	12.083	12	-0.083	0.083	2.2
1	5	5.099	5	-0.099	0.099	5

Probably, the most useful way to use the spreadsheet is to sort by slope after sorting by absolute error. Then when a particular slope is desired along which to align a sequence of gears, one may select the corresponding option with least absolute error. Table 3 shows the triples with least absolute error for slopes from 1 to 5 at intervals of 0.5: As an example of this approach, the tightest construction of slope 1 is a 5-5-7. These dimensions can be used to create a tight construction while actually extending the hypotenuse farther to hold a longer sequence of gears; for example, see Fig. 2(b).

The example constructions of Fig. 2 use liftarms along the hypotenuse to hold gears securely in place, but one can also consider placing liftarms horizontally and vertically only while attempting to place two gear centers on such liftarms so that the gears mesh diagonally along the hypotenuse. In this case, the hypotenuse



**Fig. 2.** Two approximate Pythagorean triples with sides  $\leq 14$ . (a) The one with least absolute error, (7-11-13). (b) The one with least absolute error for a hypotenuse slope of 1, (5-5-7). The hypotenuse in this case is extended to provide more space for gears.

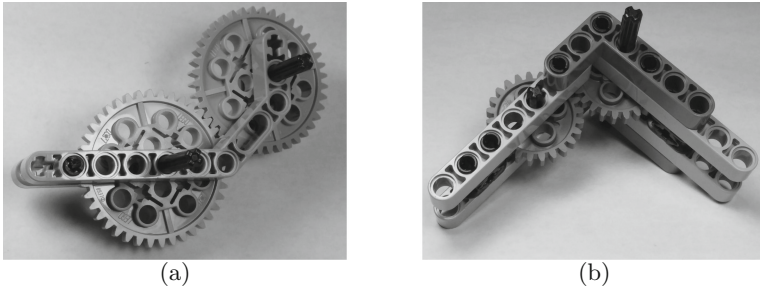
**Table 3.** The approximate Pythagorean triples with sides  $\leq 14$  with least absolute error for slopes of 1 to 5 at intervals of 0.5.

Short leg	2nd leg	Hypotenuse	Approx. Hyp.	Error	Abs. Error	Slope
5	5	7.071	7	-0.071	0.071	1
6	9	10.817	11	0.183	0.183	1.5
4	8	8.944	9	0.056	0.056	2
4	10	10.770	11	0.230	0.230	2.5
1	3	3.162	3	-0.162	0.162	3
2	7	7.280	7	-0.280	0.280	3.5
1	4	4.123	4	-0.123	0.123	4
2	9	9.220	9	-0.220	0.220	4.5
1	5	5.099	5	-0.099	0.099	5

can be no longer than 5, the largest possible separation of adjacent gear centers (using gears of radius 2.5). The most obvious possibility is to use the exact Pythagorean triple 3-4-5. Even this exact triple may leave the gears able to slip past one another due to the possibility of axles wobbling in liftarm holes, etc., but with good bracing, it seems to be feasible also to work with the three triples of small hypotenuse that come next in absolute error: 1-5-5, 1-4-4, or even 1-3-3. Beyond this, the error switches sign and the gears bind, or the error is too great to ensure that the gears do not slip. Figure 3 shows the most extreme pair of cases from the four just mentioned.

(One may also note that the “ $1 \times 11.5$  Liftarm Double Bent” pieces used in Fig. 3(a) can be used to attach liftarms at a slope of 1. Additionally, the “ $1 \times 9$  Liftarm Bent ( $3 \times 7$ )” and the “ $1 \times 7$  Liftarm Bent ( $4 \times 4$ )” form an angle  $90^\circ$  greater than the small angle of a 3-4-5 triangle.)

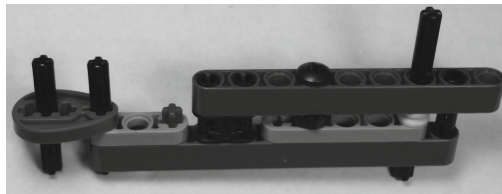




**Fig. 3.** (a) The exact Pythagorean triple 3-4-5 with the hypotenuse formed solely by gears. (b) the approximate Pythagorean triple 1-3-3 that has the greatest absolute error (0.16) that seems workable in practice.

## 4 Half-Unit Spacing

The left-hand side of Fig. 4 shows a simple construction using a Cam and a “ $1 \times 3$  Liftarm Thin” that can extend any ordinary liftarm by 1.5 units (to the leftmost axle center). Looking also at the right-hand side of Fig. 4, we can see that cams, thin lift arms (including “Triangle pieces”), the “Bush 1/2” and “Nut 8-32 Keeps (black)” all can be used to create half-unit spacing perpendicular to a liftarm. All these parts are provided in the standard Botball kit, and all but the nut (and accompanying screws) are available at [lego.com](http://lego.com); see Table 4.



**Fig. 4.** An illustration of various ways to incorporate half-unit spacing along the length of a liftarm or in a perpendicular direction.

Tweaking our macro (Fig. 1) for half-unit spacings up to 15.5 (longest liftarm extended by 1.5 units as in Fig. 4) by setting  $MAXLEG = 15.5$ ,  $MAXHYP = 15.5$ ,  $FRAC = 2$  and  $HYPKND = 2$ , we obtain 301 data rows if we eliminate sides of length just 0.5. Extracting the triples with least absolute error for slopes from 1 to 5 at intervals of 0.5, we obtain Table 5. All but the last of these is new in comparison to Table 3.

As in Sect. 3, we can again consider using only gears (no liftarm) along the hypotenuse. We already know standard gear pairings achieve gear center spacings of 1, 2, 3, 4, or 5, so we can now consider an expanded set of triples with integral hypotenuse from 1 to 5. Note that we can also achieve half-unit gear separations

**Table 4.** Parts in the Botball Lego kit that are useful for half-unit spacings.

Botball description	Botball quantity	LEGO design #	lego.com name
Bush 1/2	42	32123	1/2 BUSH
Nut 8-32 Keeps (black)	85	—	—
1 × 3 Liftarm Thin	16	6632	TECHNIC LEVER 3M
Triangle	4	2905	TRIANGLE
Cam	4	6575	COMB WHEEL

**Table 5.** The approximate Pythagorean triples with sides  $\leq 15.5$  with least absolute error for slopes of 1 to 5 at intervals of 0.5 when half-unit side lengths are allowed.

Short leg	2nd leg	Hypotenuse	Approx. Hyp.	Error	Abs. Error	Slope
6	6	8.485	8.5	0.015	0.015	1
5	7.5	9.014	9	-0.014	0.014	1.5
2	4	4.472	4.5	0.027	0.027	2
5	12.5	13.463	13.5	0.037	0.037	2.5
3	9	9.487	9.5	0.013	0.013	3
4	14	14.560	14.5	-0.060	0.060	3.5
3.5	14	14.431	14.5	0.069	0.069	4
1	4.5	4.610	4.5	-0.110	0.110	4.5
1	5	5.099	5	-0.099	0.099	5

of 1.5, 2.5, or 3.5 by combining the “16 Tooth” gear in Fig. 1 with the other standard gears. We also can use a new trick to place gears with center spacings in half-units; specifically, the double bevel (DB) gears, though recommended by Martin [11] only for changing the axis of rotation, can mesh like traditional gears do as long as we allow new gear center spacings. The three types of double bevel gears shown in Table 6, can mesh with one another at distances of 1.5, 2, 2.5, 3, 3.5, or 4.5. Considering these and the traditional gear spacings as possible hypotenuse values yields 32 data rows. Table 7 shows the 20 rows with the least absolute error, ending with the familiar 1-3-3 considered at the end of Sect. 3, beyond which the error seems too high to be prudent.

**Table 6.** Standard double bevel gears in the Botball Lego kit.

Botball description	Botball quantity	LEGO design #	lego.com name	Radius in LEGO units
12 Tooth DB	4	32270	DOUBLE CONICAL WHEEL Z12,1M	0.75
20 Tooth DB	8	32269	DOUBLE CONICAL WHEEL Z20,1M	1.25
36 Tooth DB	6	32498	DOUBLE CONICAL WHEEL Z36	2.25

**Table 7.** The approximate Pythagorean triples with short hypotenuse with least absolute error while allowing half units.

Short leg	2nd leg	Hypotenuse	Approx. Hyp.	Error	Abs. Error	Slope
1.5	2	2.5	2.5	0	0	1.333
3	4	5	5	0	0	1.333
2	4	4.472	4.5	0.028	0.028	2
2	3.5	4.031	4	-0.031	0.031	1.75
2.5	2.5	3.536	3.5	-0.036	0.036	1
3.5	3.5	4.950	5	0.050	0.050	1
2	4.5	4.924	5	0.076	0.076	2.25
1.5	2.5	2.915	3	0.085	0.085	1.667
1	1	1.414	1.5	0.086	0.086	1
2.5	3	3.905	4	0.095	0.095	1.2
1	5	5.099	5	-0.099	0.099	5
2	3	3.606	3.5	-0.106	0.106	1.5
3	3.5	4.610	4.5	-0.110	0.110	1.167
1	4.5	4.610	4.5	-0.110	0.110	4.5
1.5	1.5	2.121	2	-0.121	0.121	1
1	4	4.123	4	-0.123	0.123	4
1	3.5	3.640	3.5	-0.140	0.140	3.5
1.5	3	3.354	3.5	0.146	0.146	2
2.5	4.5	5.148	5	-0.158	0.158	1.8
1	3	3.162	3	-0.162	0.162	3

## 5 Quarter-Unit Gear Spacing

Quarter-unit spacing is not generally easy to achieve, but we can consider using certain gear combinations to form a hypotenuse that measures in quarter units. Specifically, we can mesh a traditional gear (Table 1) with a double bevel gear (Table 6) to obtain a spacing of 1.25, 1.75, 2.25, 2.75, 3.25, 3.75, 4.25 or 4.75. Tweaking the macro of Fig. 1 as in the previous section but with  $HYP\text{RND} = 4$ , we obtain the approximate Pythagorean triples of Table 8 with a hypotenuse that can be formed by meshing a traditional gear with a double bevel gear.

## 6 Gear Ratios

It can also be helpful to generate a spreadsheet to keep track of the spacings and gear ratios that result from all the different gear pairings that can be considered. Spacings that are integral or in half-units can be used horizontally or vertically,

**Table 8.** The approximate Pythagorean triples with a hypotenuse that can be formed by meshing a traditional gear with a double bevel gear in order of least absolute error.

Short leg	2nd leg	Hypotenuse	Approx. Hyp.	Error	Abs. Error	Slope
1.5	4.5	4.743	4.75	0.007	0.007	3
3	3	4.243	4.25	0.007	0.007	1
1	2	2.236	2.25	0.014	0.014	2
1.5	4	4.272	4.25	-0.022	0.022	2.667
2.5	4	4.717	4.75	0.033	0.033	1.6
2	2.5	3.202	3.25	0.049	0.049	1.25
2.5	3.5	4.301	4.25	-0.051	0.051	1.4
1	1.5	1.803	1.75	-0.053	0.053	1.5
1	2.5	2.693	2.75	0.057	0.057	2.5
1.5	3.5	3.808	3.75	-0.058	0.058	2.333
2	2	2.828	2.75	-0.078	0.078	1
1	3	3.162	3.25	0.088	0.088	3
1.5	3	3.354	3.25	-0.104	0.104	2
1	3.5	3.640	3.75	0.110	0.110	3.5

**Table 9.** All possible pairings of the gears of Table 1 (40, 24, 16, and 8 tooth gears) and Table 6 (36, 20, and 12 tooth double bevel gears) with the resulting spacing and gear ratio, sorted by spacing.

Largest Gear Teeth	Second Gear Teeth	Gear Center Spacing in LEGO Units	Gear Ratio	Largest Gear Teeth	Second Gear Teeth	Gear Center Spacing in LEGO Units	Gear Ratio
40	40	5	1	36	8	2.75	4.5
40	36	4.75	1.111	20	16	2.75	1.25
36	36	4.5	1	24	16	2.5	1.5
40	20	4.25	2	20	12	2.5	1.667
40	24	4	1.667	24	12	2.25	2
36	20	4	1.8	20	8	2.25	2.5
36	24	3.75	1.5	24	8	2	3
40	16	3.5	2.5	16	16	2	1
20	20	3.5	1	16	12	1.75	1.333
40	12	3.25	3.333	16	8	1.5	2
36	16	3.25	2.25	12	12	1.5	1
24	20	3.25	1.2	12	8	1.25	1.5
40	8	3	5	8	8	1	1
36	12	3	3				
24	24	3	1				

or perhaps on a diagonal per Table 7. Quarter-unit spacings can be utilized as per Table 8.

Generating a spreadsheet of gear ratios and spacings also can be assigned to students as an elementary exercise using nested for-loops. An extra programming concept that can be introduced here is creation of an abstract data type. For example, in VBA, one may wish to fill an array of gears that will be paired, with each gear being of the following user-defined type:

```
Type gear
    name As String
    teeth As Integer
    radius As Single
End Type
```

It would also be straightforward to generate a CSV file using this approach in any number of other programming languages, though Scratch would not be a convenient environment for working with an abstract data type.

The data resulting from considering all the gear pairings is perhaps most interesting when sorted by spacing as in Table 9. This exposes a number of ways to build fixed spacing between gears and later tweak the gear ratio with minimal reconstruction, more so than considering the single possibility cited by Martin of being able to interchange a 24-tooth/8-tooth pair with a pair of 16-tooth gears.

## 7 Conclusion

The possible constructions indicated in this paper are by no means exhaustive. Various fractional spacings may also be achieved by interposing metal pieces, bricks, or plates/tiles between liftarms, but the constructions here use the more plentiful pieces in a standard Botball kit and retain a regular grid-based approach, either on the traditional grid or on a half-unit grid. There also are some other types of gears that may be available, for example, the single bevel, crown, and worm gears provided in the Botball kit, but this paper considers the most straightforward usages of gears aligned in the same axis of rotation. The several reference tables in the paper should be useful to robot builders, and the methodological approach provides a good basis for assigning programming and data analysis tasks to students.

## References

1. Dettori, L., Greenberg, R.I., McGee, S., Reed, D.: The impact of the exploring computer science instructional model in Chicago Public Schools. *Comput. Sci. Eng.* **18**(2), 10–17 (2016). (Special Issue: Best of RESPECT 2015). <http://doi.org/10.1109/MCSE.2016.39>
2. Ericson, B., McKlin, T.: Effective and sustainable computing summer camps. In: *SIGCSE 2012*, pp. 289–294. Association for Computing Machinery (2012)
3. Exploring Computer Science: A national program (2016). <http://www.exploringcs.org/about/ecs-now>

4. Goode, J., Chapman, G.: Exploring computer science (version 7.0) (2016). <http://www.exploringcs.org/curriculum>
5. Grabowski, L.M., Brazier, P.: Robots, recruitment and retention: Broadening participation through CS0. In: Proceedings of 2011 Frontiers in Education Conference (FIE), pp. F4H1–F4H5 (2011)
6. Greenberg, R.I.: Pythagorean combinations for LEGO robot building. In: Proceedings of the Global Conference on Educational Robotics (GCER). KISS Institute for Practical Robotics (2016). [http://files.kipr.org/gcer/2016/GCER\\_2016\\_Papers/Pythagorean\\_Combinations\\_for\\_Lego\\_Robot\\_Building.pdf](http://files.kipr.org/gcer/2016/GCER_2016_Papers/Pythagorean_Combinations_for_Lego_Robot_Building.pdf)
7. Kim, S.H., Jeon, J.W.: Introduction for freshmen to embedded systems using LEGO mindstorms. *IEEE Trans. Educ.* **52**(1), 99–108 (2009)
8. KISS Institute for Practical Robotics: Products. <http://botballstore.org/products>. Accessed 21 May 2017
9. KISS Institute for Practical Robotics: Botball® educational robotics program (2015). <http://www.botball.org>. Accessed 8 June 2016
10. KISS Institute for Practical Robotics: Regions & teams (2016). <http://www.botball.org/regions-teams>. Accessed 2 Jan 2017
11. Martin, F.G.: The art of LEGO design. *Robot. Pract. J. Robot Build.* **1**(2), 1–20 (1995)
12. McGee, S., Greenberg, R.I., Reed, D.F., Duck, J.: Evaluation of the IMPACTS computer science presentations. *J. Comput. Teachers* 26–40 (2013). International Society for Technology in Education. <http://www.iste.org/resources/product?id=2853>
13. McGee, S., McGee-Tekula, R., Duck, J., Greenberg, R.I., Dettori, L., Reed, D.F., Wilkerson, B., Yanek, D., Rasmussen, A.M., Chapman, G.: Does a taste of computing increase computer science enrollment? *Comput. Sci. Eng.* **9**(3), 8–18 (2017). (Special Issue: Best of RESPECT 2016)
14. Osborne, R.B., Thomas, A.J., Forbes, J.R.N.: Teaching with robots: a service learning approach to mentor training. In: SIGSCE 2010, pp. 172–176. Association for Computing Machinery (2010)
15. The LEGO Group: Bricks & pieces. <http://service.lego.com/replacementparts>. Accessed 15 Aug 2016
16. The LEGO Group: Pick a brick. <http://shop.lego.com/en-US/Pick-a-Brick>. Accessed 21 May 2017
17. The LEGO Group: MINDSTORMS EV3 (2016). <http://www.lego.com/en-us/mindstorms/products>. Accessed 11 July 2016

# Teaching Robotics Concepts to Elementary School Children

Mor Friebronn Yesharim<sup>(✉)</sup> and Mordechai Ben-Ari

Department of Science Teaching, Weizmann Institute of Science, 76100 Rehovot, Israel

`{mor.friebronn,moti.ben-ari}@weizmann.ac.il`

<http://www.weizmann.ac.il/sci-tea/benari/>

**Abstract.** We taught computer science (CS) with robotics to four second-grade classes of 30 students each (ages 7–8). The lessons were taught using the Thymio robot and the VPL environment. Our goal was to investigate the extent to which students actually learn CS concepts. A taxonomy was developed to characterize the learning levels. The students answered two questionnaires based on the taxonomy and field observations were recorded. We found that students at such an early age were very engaged during the robotics activities and were highly motivated to succeed. Furthermore, these young students do learn CS concepts but find it difficult to create and run their own programs.

**Keywords:** Elementary school · Thymio robot · Braitenberg creatures

## 1 Introduction

Robotics activities are very popular because they reify the abstract behavior of algorithms and programs as concrete artifacts that appeal to young people. We believe that even in activities aimed at young children, an attempt should be made to teach core computer science (CS) concepts, so that they will obtain an insight on what CS truly is, beyond superficial interaction with computers.

This paper reports on the teaching of robotics to elementary school children and on research intended to diagnose the CS concepts that they learned. A robotics course was taught to four second-grade classes (ages 7–8) of 30 children each. We developed a new taxonomy to classify the level of learning achieved and used it to develop questions for the students to solve. Analysis of their solutions enabled us to identify which concepts they learned.

Section 2 reviews existing work in the area of teaching robotics to children. Section 3 describes the robot and its software environment. The research methodology is presented in Sect. 4. The results of the analysis and a discussion of the results appear in Sects. 5 and 6, while Sect. 7 describes our plans for the future.

## 2 Literature Review

Computer activities have been widely used in education. Clements and Sarama [4] survey work on teaching with Logo, but this focused on teaching subjects

such as mathematics and science, not computer science. Meerbaum-Salant et al. [13] were able to characterize the CS concepts learned by middle-school students using Scratch. We are interested in robotics activities that have become popular as a way of engaging students of all ages [5, 8], in particular, we are interested in the extent to which students actually learn CS concepts. Magnenat et al. [10] found that middle-school students were able to understand some, but not all, of the CS concepts that were taught.

What about younger children in elementary school? Martinez et al. [12] taught robotics to elementary-school and even preschool students. They claimed that elementary-school students are capable of understanding and applying CS concepts such as loops, parameters, conditions and sequencing, while preschool students understood fewer concepts. Sullivan and Bers [16] implemented a robotics curriculum in preschool through second grade. They reported that the children were successful in mastering basic programming and robotics concepts. Bers et al. [1] and Sullivan et al. [17] engaged children in robotics activities and claimed that students can learn core concepts of computational thinking.

These positive results on learning by young children apparently contradict the high dropout rate among university students in introductory CS courses [18], as well as the extensive research on misconceptions in learning CS [2, 15]. How can it be that young kids, but not university students, understand CS concepts?!

### 3 The Robot and Its Software Environment

#### 3.1 The Thymio Robot

The Thymio educational robot [9] was designed by EPFL and ECAL in Lausanne, Switzerland (Fig. 1). The robot is small ( $11 \times 11 \times 5$  cm), self-contained and very robust with differential drive, nine infrared proximity sensors, five touch-sensitive buttons, a 3-axis accelerometer, dozens of LEDs, a speaker and a microphone.<sup>1</sup>



**Fig. 1.** Thymio robot

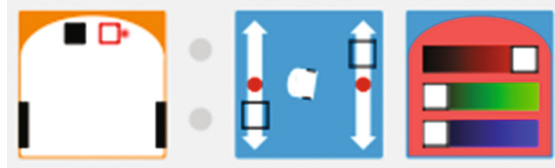
---

<sup>1</sup> The hardware and the software are open source under the CC-BY-SA license. Software, documentation and tutorials can be found at <https://www.thymio.org/>.



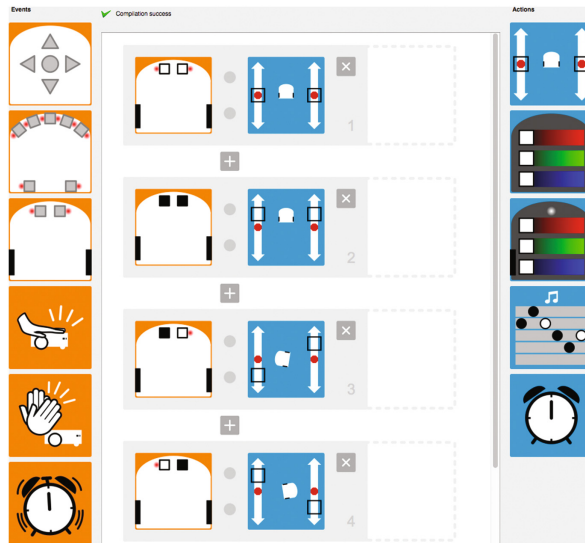
### 3.2 The VPL Graphical Programming Environment

The robot is programmed using the Visual Programming Language (VPL), which was designed for K-12 users of the robot. (VPL is a component of the Aseba programming suite which includes other more advanced programming languages.) Programs are constructed by drag-and-drop of graphical blocks. VPL supports one programming construct: *event-action pairs* (Fig. 2). Event handling is a core CS concept, which has been proposed as the basis of teaching introductory programming [3].



**Fig. 2.** Event-action pair: The event is *the right ground sensor detects the white floor and the left sensor detects the black tape*. There are two actions: the first is *the right motor is set to fast forward and the left motor to fast backwards*, and the second is *the top LEDs display red*. The result is that if the event occurs, the robot turns to the left and displays red.

Figure 3 shows a VPL program for line following.



**Fig. 3.** VPL program for line following. The toolbar for events is on the left and the toolbar for actions on the right. If both ground sensors detect the black line, the robot moves forward. If only one detects the black line, the robot turns in the opposite direction. If the black line is not detected the robot stops.

### 3.3 CS Concepts Taught

The concepts included in our curriculum (although we did not use the technical terminology) included:

- Event handlers.
- Sequential execution of event-action pairs.
- Concurrent execution of event-action pairs, for example, in the line-following program the software simultaneously checks if the robot is leaving the left edge of the line or the right edge of the line.
- Parameters: setting the color of the LEDs and the power applied to each motor.

## 4 Research Methodology

### 4.1 Rationale

Our examination of previous research led us to the conclusion that there is a lack of precision in defining what it means to understand CS concepts. Our research attempts to categorize more formally what young children are able to do in order to achieve a better idea of their level of understanding.

### 4.2 Research Question

- What CS concepts can elementary-school students understand from participation a robotics-based CS course?

### 4.3 The Robotics Class

The research was carried out in a public elementary school. The school had been established only a few years ago and the principal and teachers very receptive to our suggested activities. Four classes (28–30 students each) were taught for one hour a week for 12 weeks. The classes were given during normal school hours, not as an extracurricular activity. The students had no previous formal instruction in CS or robotics. The lessons were taught by the first author who also carried out the research. She was aided by a research assistant from our department and the class teacher was present, primarily to deal with behavioral problems that occasionally arose.

The lessons took place in a computer lab consisting of rows of personal computers. With 30 young students sharing 10 robots this was somewhat inconvenient for working with the robot.

We developed our curriculum by adapting existing curricula and learning materials for the Thymio and VPL for very young students. During the first part of each lesson (about 10 min) the instructor explained and demonstrated a new concept. Following the explanation, the students were given a worksheet that

included several tasks. At the beginning of some lessons, a questionnaire was administered.

The first tasks were intended to familiarize the students with the events generated by the proximity sensors and the buttons, and with the actions of changing the colors of the LEDs on the robots. This was followed by tasks based upon *Braitenberg creatures* [7] which were developed during the Programmable Brick project [11] that was the inspiration for LEGO<sup>®</sup> Mindstorms. We have found these tasks easy to implement and yet highly appropriate for learning about the motion of a robot in response to events generated by the sensors.

#### 4.4 A Taxonomy of Levels of Understanding

A taxonomy is needed in order to characterize the level of learning demonstrated by the students. Taxonomies are difficult to develop for learning CS because learners are asked to create programs from the very beginning of their studies, although creating is considered to be a very high level of understanding, for example, in the Bloom taxonomy. We developed a new taxonomy based upon the one in [13], which in turn was based on both the Bloom and SOLO taxonomies. The taxonomy in [13] was used in research with the Thymio robot but found to be not entirely appropriate for learning robotics [10].

Our taxonomy consists of six different levels:

1. Predicting the behavior of a given program.
2. Choosing the program that gives rise to a specified behavior.
3. Characterizing the difference in the behavior of two similar programs.
4. Completing a partial program in order to achieve a specified behavior.
5. Using a modified programming construct.
6. Creating a program from scratch for a specified behavior.

#### 4.5 The Questionnaires

Each questionnaire contained six multiple-choice questions, one for each level of the taxonomy:

1. Predicting the behavior of a given program: We presented the students with a program and asked what actions are performed when the program is run and a certain event occurs.
2. Choosing the program that gives rise to a given output: We showed the students a video and four programs; the students had to select the program that gives rise to the behavior shown in the video.
3. Characterizing differences in the behavior of two similar programs: We showed the students two videos; the students were asked to select a characterization of the difference between the two behaviors.
4. Completing a partial program: The students had to choose the events or actions needed to add to the program in order that it fulfill its specification. These questions are similar to *Parsons puzzles* [14].

5. Using a modified programming construct: This is similar to the previous question except that we used modified versions of some of the VPL event and action blocks.
6. Creating a program from scratch for a specified behavior: This question was also similar to a Parsons puzzle but the students were required to write a program given a relatively large number of blocks.

Two questionnaires were administered. The first questionnaire asked about simple event-action pairs, while the second one included questions on the motion of the robot. The questionnaires were given during the regular class hour. They looked like the usual worksheets and the students willingly participated in solving the problems. To reduce the load on the young students, the administration of a questionnaire was divided over two or three classes.

Appendix A gives two examples of the questions from the questionnaires.

## 5 Results

### 5.1 Observations

Although the young students were obviously immature, we found that almost all of the them were highly engaged with the instruction and the robotics activities. They were motivated to succeed in the tasks and proud when their programs worked correctly. Incorrect programs did not generate excessive frustration; instead, the students approached the instructors for help. The students tended to use trial and error.

The Thymio robot and the VPL environment were very well received and are clearly appropriate for students of this age group. Students even explored the VPL environment on their own, asking questions on and experimenting with event and action blocks that we had not yet taught.

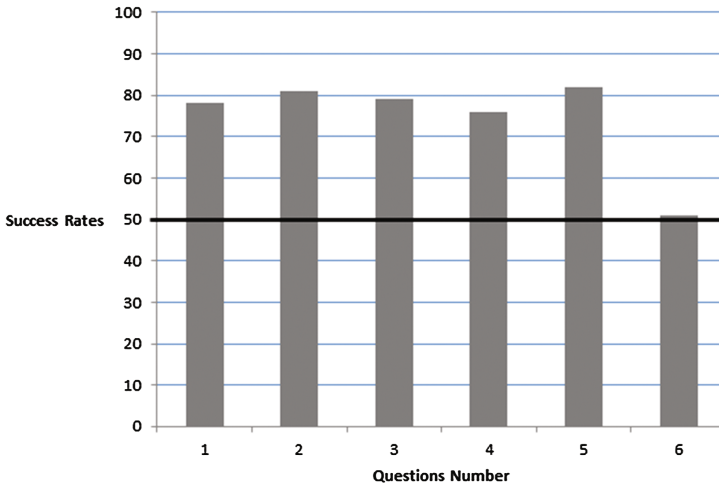
Although we were pleased with the high level of engagement and the success of the students in performing the activities, we encountered several difficulties:

- VPL supports the association of multiple actions with a single event. This proved to be a difficult concept and students tended to write programs with only one action per event even if this did not fulfill the task.
- The students tended to assume that the directional buttons were tied directly to the robot’s motion although any such connection must be explicitly programmed.
- The event blocks for the sensors have “parameters” (white=an object is detected, black=an object is not detected, gray = the sensor is ignored). The students found these difficult to remember.

The difficulties were not serious and disappeared in time.

## 5.2 The Questionnaires

The first questionnaire investigated whether the students can associate an event with an action. It was divided into two parts: the first with four questions and the second with another two questions. Figure 4 shows the percentage of students who answered each question correctly. Recall that each question is intended to check one level of the taxonomy.



**Fig. 4.** First questionnaire: % of students who gave correct answers for each question

The second questionnaire tested the understanding of the connection between the sensors and the motion of the robot. It was administered in three parts. Figure 5 shows the percentage of students who answered each question correctly.

Using the Pearson's chi-square test, we checked that the results were consistent among all four classes and this was true for all questions except question 6. Table 1 shows the results for this question by class.<sup>2</sup>

**Table 1.** Percentage of correct answers for question 6

	Class			
Questionnaire 1	59		63	32
Questionnaire 2	41	54	47	29

<sup>2</sup> The first questionnaire was not administered to class 2 because of logistic difficulties.

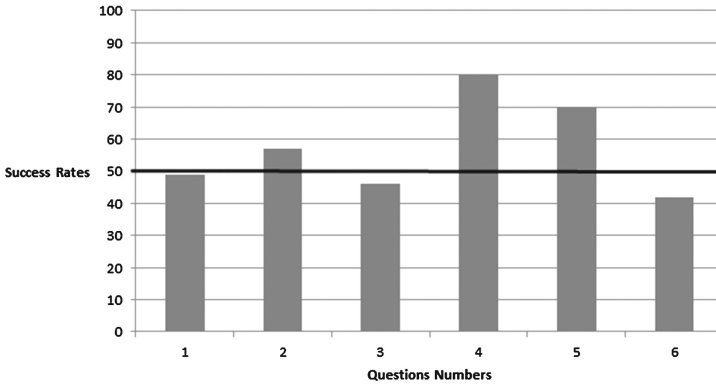


Fig. 5. Second questionnaire: % of students who gave correct answers for each question

## 6 Discussion

To interpret Figs. 4 and 5 as evidence of achievement of levels of the taxonomy, we chose (somewhat arbitrarily) a cutoff number. If 50% or more of the students successfully answer a question, we take this as evidence that the majority of the students achieved the level corresponding to that question. Question three of the second questionnaire which asked about the difference between similar programs showed lower levels of success.

Students displayed some difficulties in constructing a program from scratch, as shown by the results for the sixth question. While they were able to write programs during class using trial and error and with the help of the teaching staff, the questionnaires—which were answered offline—showed that students found it difficult to write programs outside the context of the robot and the VPL environment. At the university level, there is a requirement to demonstrate the ability to implement programs and each program must use multiple constructs in an integrated manner so that the program fulfills a high-level specification. Our investigation provides evidence that young students are not able to do so.

The results for the fifth question of each questionnaire show that the students were able to understand event and action blocks that were similar but not identical to the ones they had learned. This is consistent with *near transfer* of knowledge as defined by Gick and Holyoak [6].

## 7 Conclusions

Robotics activities enable even young students to actually learn CS concepts and programming constructs, and to write and run programs. The robots facilitate a high level of engagement and interest. Students performed well in answering questions on the programming constructs. The VPL environment enabled the students to create programs graphically, thus overcoming the linguistic barriers to programming. Students have the ability to understand what a simple program

does, to choose a program that matches a specification, to characterize differences between programs, to complete a partial program, and even engage with new event or action blocks that are similar to the blocks they had learned.

Robotics activities can be successfully used with very young students, not only to increase their interest and possibly motivation to become engaged with STEM in general and CS in particular—but also to learn CS concepts. Our results showed that young students find it difficult to go from learning concepts and individual programming constructs to being able to construct programs of more than a few lines. This is an area for further research.

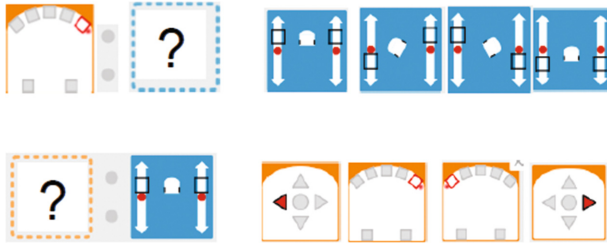
**Acknowledgments.** We would like to thank Stella Khazina for assisting in the classroom. We are grateful to the principal and teachers of the school for their willingness to participate in this research and for their cooperation and support.

## A Questions from the Questionnaires

### Question 4 from the Second Questionnaire

Circle the event or action that will cause the program to implement this behavior:

- When the front right sensor detects an object, the robot turns right.<sup>3</sup>
- When the front left sensor detects an object, the robot moves forwards.



### Question 5 from the First Questionnaire

We invented a new event: The event occurs when the middle button is touched at the same time that the middle front sensor detects an object.



Required behavior:

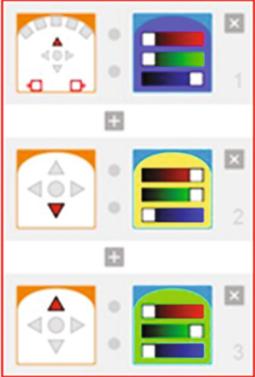
- If an object is detected by the front sensor the top LEDs will display blue.

<sup>3</sup> Note the potential confusion between the button direction and the sensor position.

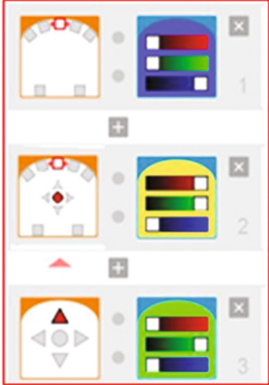
- Touching the middle button will display yellow in the top LEDs.
- Touching the middle button at the same time that an object is detected by the middle front sensor will display green in the top LEDs.

Circle the program that implements this behavior:

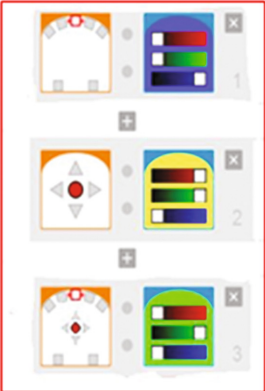
**Option 1**




**Option 3**



**Option 2**



**Option 4**



## References

1. Bers, M.U., Flannery, L., Kazakoff, E.R., Sullivan, A.: Computational thinking and tinkering: exploration of an early childhood robotics curriculum. *Comput. Educ.* **72**, 145–157 (2014)
2. du Boulay, B.: Some difficulties of learning to program. *J. Educ. Comput. Res.* **2**(1), 57–73 (1986)
3. Bruce, K., Danyluk, A., Thomas, M.: *Java: An Eventful Approach*. Prentice Hall (2006)
4. Clements, D.H., Sarama, J.: Research on logo: a decade of progress. *Comput. Schools* **4**(1/2), 9–46 (1997)
5. Druin, A., Hendler, J. (eds.): *Robots for Kids: Exploring New Technologies for Learning*. Morgan Kaufmann (2000)



6. Gick, M.L., Holyoak, K.J.: Analogical problem solving. *Cogn. Psychol.* **12**(3), 306–355 (1980)
7. Hogg, D.W., Martin, F., Resnick, M.: Braitenberg creatures. Tech. Rep. E&L Memo No. 13, MIT Media Lab (1991), [http://cosmo.nyu.edu/hogg/lego/braitenberg\\_vehicles.pdf](http://cosmo.nyu.edu/hogg/lego/braitenberg_vehicles.pdf). Accessed 27 Mar 2017
8. King, K.P., Gura, M. (eds.): *Classroom Robotics: Case Stories of 21st Century Instruction for Millennial Students*. Information Age Publishing (2007)
9. Magnenat, S., Riedo, F., Bonani, M., Mondada., F.: A programming workshop using the robot “Thymio II”: The effect on the understanding by children. In: *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)* (2012)
10. Magnenat, S., Shin, J., Riedo, F., Siegwart, R., Ben-Ari, M.: Teaching a core CS concept through robotics. In: *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education*, pp. 315–320 (2014)
11. Martin, F., Mikhak, B., Resnick, M., Silverman, B., Berg, R.: To Mindstorms and beyond: evolution of a construction kit for magical machines. In: *Druin, A., Hendler, J. (eds.) Robots for Kids*, pp. 9–33. Morgan Kaufmann (2000)
12. Martinez, C., Gomez, M.J., Benotti, L.: A comparison of preschool and elementary school children learning computer science concepts through a multilanguage robot programming platform. In: *Proceedings of the Conference on Innovation and Technology in Computer Science Education*, pp. 159–164 (2015)
13. Meerbaum-Salant, O., Armoni, M., Ben-Ari, M.: Learning computer science concepts with Scratch. *Comput. Sci. Educ.* **23**(3), 239–264 (2013)
14. Parsons, D., Haden, P.: Parson’s programming puzzles: a fun and effective learning tool for first programming courses. In: *Proceedings of the 8th Australasian Conference on Computing Education*, pp. 157–163 (2006)
15. Pea, R.D.: Language-independent conceptual “bugs” in novice programming. *J. Educ. Comput. Res.* **2**(1), 25–36 (1986)
16. Sullivan, A., Bers, M.U.: Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade. *Int. J. Technol. Des. Educ.* **26**(1), 3–20 (2016)
17. Sullivan, A., Elkin, M., Bers, M.U.: KIBO robot demo: Engaging young children in programming and engineering. In: *Proceedings of the 14th International Conference on Interaction Design and Children*, pp. 418–421 (2015)
18. Watson, C., Li, F.W.: Failure rates in introductory programming revisited. In: *Proceedings of the Conference on Innovation and Technology in Computer Science Education*, pp. 39–44 (2014)

# The Effect of the Programming Interfaces of Robots in Teaching Computer Languages

B. Baransel Bağcı, Mustafa Kamaşak, and Gökhan Ince<sup>(✉)</sup>

Faculty of Computer and Informatics Engineering, Istanbul Technical University,  
Maslak, 34469 İstanbul, Turkey  
{baransel,kamasak,gokhan.ince}@itu.edu.tr

**Abstract.** Programming is a popular subject in education and experts emphasize the importance of teaching programming to the children in the high school or even earlier. In this study, we used robots for teaching programming basics to the high school and college students and observed the effects of different interfaces. We used an educational robot called Thymio-II with Aseba Event Script Language (AESL), which is designed specifically for the Thymio. In this work, our hypothesis is that visual programming interfaces are more successful on learning programming and facilitate the learning with other interfaces and languages. In order to teach programming, as well as the interfaces and the robot features, we created a curriculum applicable to all interfaces. We taught students ages ranging from 15 to 24 using lecture content prepared in the form of video recordings. Students were given 30 min of lectures and at the end of each lecture students were expected to write a program according to predefined requirements. Students were divided to groups using different interfaces and we observed the difference of the learning curves of students for each programming interface. In our tests, we used original English AESL, Turkish version of AESL and a graphical interface called Visual Programming Language (VPL). We compared the performance of the students using the graphical icon based against the classical text based programming languages.

**Keywords:** Teaching computer languages · Visual programming · Robots in teaching programming · Programming interfaces

## 1 Introduction

Programming education has long been a controversial issue. For nearly 40 years, effects of the learning programming and the methods of teaching programming languages have been research topics for researchers. In the early days of computers, programming languages are considered as an adjunct to other subjects in teaching, especially for mathematics. The first popular example of this approach is LOGO project, which is a programming language specifically designed for teaching mathematics to children [1]. With the popularization of personal computers, programming languages became a major subject in education.

Most of the researches focused on effects of learning programming languages on children's cognitive style, metacognitive abilities and cognitive development [2–4]. More recent studies suggest that teaching programming languages increase students' problem solving abilities, natural language skills, creativity and ability of working in multidisciplinary subjects [5]. Because of these benefits, teaching programming languages became popular in modern education.

Nowadays, teaching computer languages has been considered as an essential part of the basic sciences and it is recommended that computer languages should be included in the secondary school curriculum or even in primary school curriculum. When teaching computer languages in primary or secondary schools, it is important to note that the main aim is to help to development of students' abilities; not to make students programming experts in early ages which is impossible even for college students in four years of college education [6]. According to a report from European Schoolnet, which is a network organization of 30 European Ministries of Education, computer languages and programming is already part of the curriculum in 12 countries which includes Bulgaria, Cyprus, Czech Republic, Denmark, Estonia, Greece, Ireland, Italy, Lithuania, Poland, Portugal and the UK. Also 7 countries which includes Belgium Flanders, Spain, Finland, France, Luxembourg, the Netherlands and Turkey are working on to integrate computer programming in curriculum [7].

When we consider all these developments, it is not arguable anymore whether programming should be part of the curriculum or not. The main focus is on the methods of teaching computer programming. According to a survey of literature, significant researches on teaching computer programming are focused on either curriculum, pedagogy, computer languages or programming tools with majority of languages and tools [8].

For programming languages, there are a lot of discussions on which one is the most appropriate to expose the students as their first programming language. While popular languages such as C, C++, Java and Python are generally used for learning programming in college level, there exist a lot of languages for learning programming [9–11]. Moreover, some researchers proposed mother-tongue based programming languages, which have keywords in native languages or in native sentence structure; but these proposed languages had limited impact on the general teaching of programming languages [12,13].

When teaching programming languages started to focus on younger students, proposed languages and interfaces began to shape around younger minds. To teach programming languages to younger children, visual programming languages were created. These visual languages have more flexible syntax and are mostly based on the graphical icons and their relations. Scratch is a popular example for visual programming languages and it is designed for children ages between 8 and 16 to teach programming while they working on more complex and creative projects [14].

Another approach for teaching programming languages to children emerged from the field of robotics. Instead of focusing on languages and programs on the computer, robots have been used as a tool in teaching programming.

By programming robots, constructivist approach of teaching has been used as a more actively in teaching programming languages because of using robots support the basic tenets of constructionism by providing more active learning to students, manipulating robots to think with, providing more powerful and significant projects and giving more change to selfreflection [15, 16].

In this study, we focused on the effects of the programming languages and interfaces in teaching programming with educational robots. We used robots as programming objects because of the reasons given above and we worked with precollege students to college students. Our study aimed to clarify the benefits of the visual programming interfaces as either main programming interfaces or an introduction interface before others. At the end, we presented the test result of different interfaces.

## 2 Proposed Methods

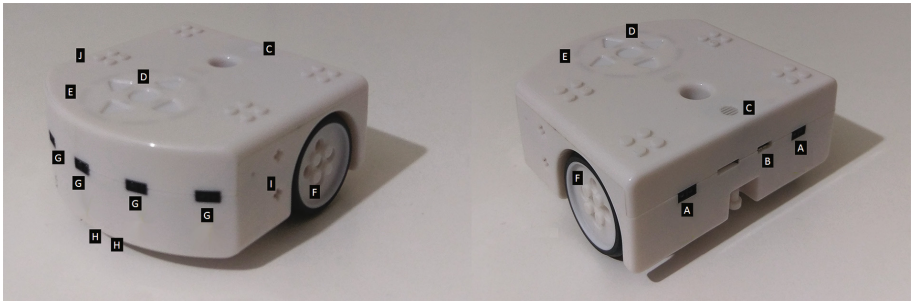
### 2.1 Educational Robot Platform

When we designed our teaching and test environment, we evaluated different robotic platforms for educational usage such as Lego Mindstorm, Arduino and Thymio-II. While Lego Mindstorm is the most popular platform and has a lot of functionality, it requires a hardware preparation, which is building robot using the necessary sensors, motors and other extensions. Arduino is effective developing board and it supports much more complex hardware and software structures but it requires significant electronic knowledge before the programming. When comparing other platforms with Thymio, Thymio is simpler and better suited to children. It does not require any electronic knowledge beforehand and it is in no need hardware related constructions before the programming. At the end we decided to select Thymio-II because it is simple structured, has enough number of sensors on it and has a special programming language and development environment for different operating systems and more importantly it is more accessible and suitable for children because of it is made simple and with limited number of sensors and actuators [17].

Thymio-II has 9 infrared (IR) proximity sensors (5 on the front, 2 on the back and 2 on the bottom), 5 capacitive touch buttons (on the top), 1 three-axis accelerometer, 1 thermometer, 1 microphone (for recording or detecting noise level), 1 IR receiver (for remote control or communication), 39 LEDs (on the front, back, top and bottom), 2 DC motors (which have speed control and speed sensors) and 1 loudspeaker [18]. Therefore, it can be used for different task such as path tracking, following, escaping, balancing etc.

In our work, different robot features are used for teaching programming controls. Microphone and touch button sensors are used in lectures for understanding sensor data inputs and taking actions according to them. For the actions, we used LEDs to change color of the robot and wheels to move robot in the test environment. While robot is moving and microphone sensor is in use, some problems occurred because of the sound of the motors and wheels. When sound

sensing is too sensitive, robot sense the sound of the motors and its own movement and takes action to this input. To avoid this problem without giving an extra instructions about it, we adjusted the default sensor sensitivity in our interfaces according to environment and we adapted our end of lecture tests for further isolation. While introducing decision making structures in the robot, we used proximity sensors at the front and the rear of robot for sensing and escaping from obstacles. Also the ground sensors at under the robot are used for understanding the environment. Loudspeaker component is used in lectures as an alert method for the sensor inputs and the states. Similar to previous problem, using microphone and loudspeaker together caused unwanted side effects. Thus, we designed our test objectives to avoid this specific situation. Also for sensing the robots own movement and the location, we used speed sensor from wheels and accelerometer. While accelerometer is in use, for preventing the negative acceleration effect on the beginning of the movement, we limited the speed of the wheels and accelerometer threshold to be compatible with each other (Fig. 1).



**Fig. 1.** The Thymio-II robot and components. (A) Rear proximity sensors (B) USB connection port (C) Loudspeaker (D) Capacitive touch buttons (E) Status leds (F) Wheels and speed sensors (G) Front proximity sensor (H) Ground sensors (I) Temperature sensor (J) 3-axis accelerometer

## 2.2 Programming Interfaces

For programming environment, Thymio-II takes advantage of the Aseba Tools. It support 3 different types of programming interfaces which can be summarized as text programming, visual programming and blockly interface, which is a mixture of visual and text programming interfaces [19]. Visual programming interface, which is called VPL, consists of only icons and symbols and independent from any human language. Text programming language, which is called Aseba Event Scripting Language (AESL), is English keyword based scripting language like many programming languages. Blockly interface is a combination of text and visual blocks, which are prewritten code blocks. To separate the visual and text programming, we did not include the Blockly interfaces in our work. Because of

these reasons, we used text programming and visual programming interfaces in our study.

**Programming via Text Interface.** Thymio-II is programmed with Aseba Event Scripting Language (AESL) which is a scripting language of a modular, distributed and event-based architecture ASEBA [20,21]. In text programming interface, AESL consists of the English based keywords and native functions on programming. In AESL, program flows are defined by event definitions and each event block can be run in same time if the necessary event is occurs. Event definitions are made with *onevent* keyword and can be supported by decision blocks with *if* and *when* keywords and if necessary with loops. Actions and other changes are made mostly by assigning appropriate variables and array values.

In addition to original AESL, we developed a version of AESL with Turkish based keywords and native functions. In our tests we use these two versions of AESL to compare effects of the native language in programming learning. The syntax of the AESL was kept the same with the original AESL but the keywords were changed with Turkish words and all native functions renamed with Turkish function names.

**Programming via Visual Interface.** In visual programming interface, we used Visual Programming Language (VPL) which is developed for Thymio-II and is suited for children in learning computer languages [22]. VPL does not include any keywords but it only has icons as programming elements. Therefore, there is no language effects on the visual programming interface. Example code in VPL is shown in the Fig. 2 below. A part of the code marked with a yellow background in both VPL and its AESL equivalent. In VPL, first two icons in each line represent the conditions and they correspond to *onevent*, *when* and *if* keywords in its AESL equivalent. Other icons in VPL defines the actions and generally represented as variable assignment in AESL.

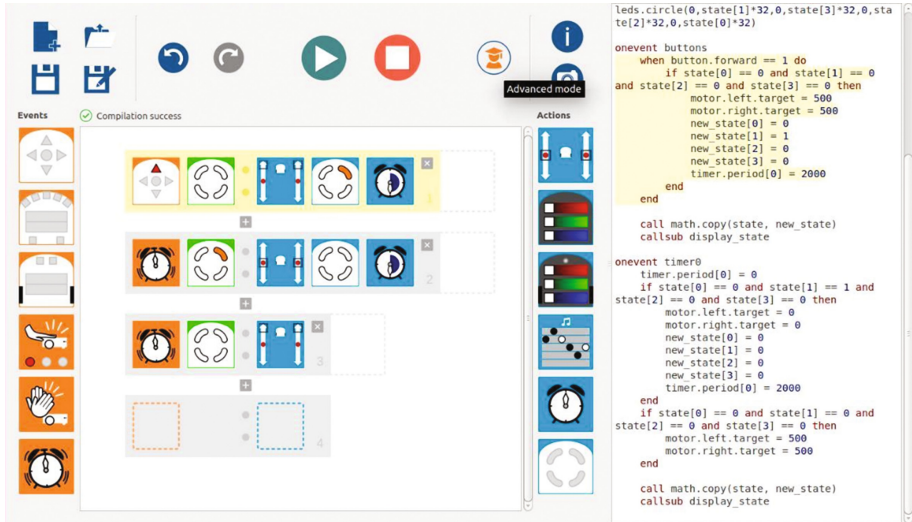
### 2.3 Teaching and Assessment Methodology

We prepared three lessons<sup>1</sup> according to AESL and VPL interface structures and Thymio's features. First lesson generally covered introduction subjects to programming and development environment. Also in first lesson, a brief description of Thymio-II and its features are also included. Course topics can be summarized as follows:

- Introduction to Thymio-II and its components
- Introduction to development environment and interface
- Event definition and event listening
- Usage of the top-buttons
- Usage of the touch and sound sensors

---

<sup>1</sup> Lectures can be accessed via <https://youtu.be/JchvD4sMqBk>.



**Fig. 2.** VPL interface and its equivalent AESL code

- Usage of the wheels
- LED lights and basic data operation.

In the second lesson, we focused on the decision mechanism with minor sensors and actuators of Thymio-II. Topics can be summarized as follows:

- Usage of the front proximity sensors for obstacle detection
- Usage of the rear proximity sensors for escaping
- Usage of the ground sensors for to avoid falling off
- Decision mechanism with *if* and *when* keywords
- Usage of loudspeaker output for sound alert.

In the third lesson, we covered complementary elements to robot components and programming features. In addition to new topics, we have reinforced previous topics to reach a better understanding of algorithms as well as introducing new topics. Third lesson topics can be summarized as follows:

- Variables and tracking robot states
- Usage of the status LEDs
- Delayed or repetitive jobs
- Usage of the accelerometer.

Our teaching strategy as well as the consequent assessment experiments focused on basic programming subjects. We kept the subjects quite simple, so that we can ignore effects of personal skills and cognitive levels of the students. Also the languages, VPL and AESL, are do not need long code part to control the robot, so we escaped from complex algorithms and long code parts. Almost all of

the examples are a few lines of code. This minimized the complexity of teaching and thus we ignored the algorithm creation, debugging and testing effects on our evaluation.

For course materials, in order to reduce the effects of lecturer, we prepared video recorded lectures for each lesson. In addition to that, we organized the lectures as one to one training, thus we minimized the effects of the classroom environments in teaching. During lectures, the students watched training videos and later they asked their questions to the lecturer in a given time.

Each lecture was planned to be a classic 45-minutes long session. First 20 min of the lesson is devoted to the lecture part. In this section, students watched the lecture videos and asked their questions. Next 10 min are reserved for free practicing with the robot. The learning part of the each session limited to 30 min, because our video lecture contents are structured to short periods and in that time students have thinking time and interaction time with the lecturer. Also with short lesson periods, we tried to keep the students focused. The final 15 min section is dedicated to the exam part. Each lesson has an exam which includes 10 objectives on the discussed topic on that day. Students are expected to solve and implement these objectives with Thymio-II robots using the respective interface. Each objective designed independently and generally requires a movement or output to a given input. For example, in first exam objectives, it's expected that the robot stops its movement and ends other outputs when it sense a tap on it. All three exam objectives are given in Tables 1, 2 and 3. After the exam, codes are collected for evaluation.

At evaluation phase, collected codes were tested by our team in predesigned evaluation environment which has obstacles and cliffs on necessary locations. Each objective was assessed in the given order and considered successful only if the code produces the expected outcome. Successful objectives were considered 1 point, while other results were 0 point. Thus, our evaluation focused on the outcome and became independent from the methods and algorithms or how the code is written. While interpreting the result of the experiment, we focused on the two main outcomes: success rate of the exam objectives and time spent to finalize exam.

**Table 1.** Exam-1 objectives

Condition	Expected outcome
When forward button is touched	Robot should start a forward movement at normal speed
When backward button is touched	Robot should start a backward movement at double speed
When center button is touched	Robot should turn in clock-wise
When center button is touched	Robot should turn on the top LEDs to red
When left button is touched	Robot should start to escape to left rear side
When right button is touched	Robot should start to escape to right front side
When right button is touched	Robot should turn on the bottom LEDs to green
When robot senses a tap on it	Robot should stop all movement immediately
When robot senses a tap on it	Robot should turn off the top LEDs
When robot senses a tap on it	Robot should turn off the bottom LEDs



**Table 2.** Exam-2 objectives

Condition	Expected outcome
When forward button is touched	Robot should start a forward movement at normal speed
When center button is touched	Robot should stop all movement immediately
When an object is detected in front left side	Robot should start to escape to right rear side
When an object is detected in front right side	Robot should start to escape to left rear side
When an object in detected in front of the robot	Robot should start to escape to backward
When an object in detected in rear of the robot	Robot should start to escape to forward
While robot is moving forward, if it detects a cliff in front of it	Robot should stop all movement immediately
While robot is moving forward, if it detects a cliff in front of it	Robot should turn on the top LEDs to red
When backward button is touched	Robot should start a backward movement at normal speed
When backward button is touched	Robot should turn off the all top LEDs

**Table 3.** Exam-3 objectives

Condition	Expected outcome
While moving forward, if robot senses a tap on it	Robot should stop all movement immediately
While moving forward, if robot senses a tap on it	Robot should turn off the all top LEDs
When stopped, if robot senses a tap on it	Robot should start a forward movement at normal speed
When stopped, if robot senses a tap on it	Robot should turn on the only front-left status LED
While moving forward, if forward button is touched	Robot should double the speed
While moving forward, if robot senses a slope	Robot should stop all movement immediately
While moving forward, if robot senses a slope	Robot should turn off the all top LEDs
When center button is touched	Robot should turn on the all four status LEDs
When center button is touched	Robot should delay the other actions 1 second
When center button is touched	Robot should stop all the movement after the delay and disable itself

## 2.4 Experiments

We prepared two different interfaces: visual programming language VPL and text programming with AESL. For two interfaces, we prepared a common curriculum, same timetable and same exam questions, so that we can compare them within

each group of students. In our experiments, 17 students, within ages of 15 to 24 with an average 18.6 and standard deviation 2.76 participated. The majority of the students have no experience with programming before and rest of them have vaguely information on programming. Their computer skills focused on the everyday usage and they have intermediate level English proficiency. Younger students' professional interests tends to be any way and the others' professional orientations are not related to programming. We divided students into 3 different groups as below:

- Students who studied VPL (Group 1)
- Students who studied AESL (Group 2)
- Students who studied AESL after VPL (Group 3).

When determining the groups, we took into account that nearly all advanced programming languages are text-based. So, we created Group 3 to evaluate the effects of visual languages on learning text-based languages. But we didn't see a reason to create a group who studied VPL after AESL, because our main objective was to teach students the programming before moving to the advanced languages, which are text-based (Fig. 3).

Test result for each group on each lesson and their average is summarized in Table 4 with the completion time of each lesson in their respective interfaces. Maximum time for the each test is 15 min.



**Fig. 3.** (a) Students in lecture (b) A student in test

**Table 4.** Average results of the experiments

Group	Attendees	Success rate (out of 10)				Test time (in seconds)			
		1st test	2rnd test	3rd test	Average	1st test	2nd test	3rd test	Average
Group 1	11	9.82	9.64	9.27	9.58	348.36	349.00	683.27	460.21
Group 2	6	8.50	9.00	2.83	6.78	695.00	730.33	900.00	775.11
Group 3	6	9.83	9.67	5.67	8.39	498.00	723.17	900.00	707.06

According to the result of the experiment, students who studied with VPL have been more successful than students who studied only AESL. Group 1's success rate is distinctively higher than Group 2 in each lesson and this difference is more explicit in the third lesson which has more complicated content than the first two lessons. In average, Group 1 is 41% more successful than Group 2 in terms of the success rate. So, it can be said that visual programming languages are a lot easier for learning programming. Also, visual programming interfaces are quicker to program than text-based programming languages. Group 1's test times are lower for each lesson than Group 2. In average, Group 2 takes 68% more time than Group 1 for tests. Hence, it can be said that visual programming languages are faster for learning programming.

When we compare students who studied only AESL with students who studied VPL first then AESL, we saw that Group 3 has been 23% more successful than Group 2. These comparisons show that learning programming with visual languages helps to learn text-based programming languages later. When the complexity of the lessons and the tests increased, success rates became more distinct between Group 2 and Group 3. At the 3rd test, Group 3 doubles the score against Group 2. We can say that while visual programming interfaces are faster and more successful on learning programming, they also increase the success rate of the text programming interfaces if they are taught first.

Another measurement criterion on our test between Group 2 and Group 3 is the time spent on each interface. The effects of the visual programming languages also can be seen on the time spent for programming between students who studied VPL first then AESL. Group 3 used 9% less time than Group 2. According to these results we can say that learning programming with visual languages helps to learn text-based programming faster.

In our study, we also took into account the effects of the keywords and native function names of the programming languages. Instead of English-based keywords and function names, we tested our subjects with a native language version of AESL based on Turkish keywords and function names. But our test results neither prove nor disprove the success of mother-tongue based programming languages on learning programming because of that there is no statistically significant difference in between the success rates of original AESL and Turkish AESL. Average success rates are nearly identical between original AESL and Turkish AESL but we didn't include Turkish AESL on overall assessment because of that we used a small group for testing Turkish AESL.

### 3 Conclusion

In this study, we tried to compare text-based programming interfaces against visual programming interfaces on learning programming first time. According to result of our experiments, visual programming languages are easier and quicker to learn. Also if visual programming languages are used for learning programming in first time, students can adapt other programming languages easier.

In future works, other visual programming languages can be included in the experiments and can be compared with VPL. Also mixture of the text programming and visual programming interfaces can be different approach to teaching programming. In addition to that, these programming interfaces can be tested with younger students in order to observe the effects of the age on learning computer languages.




**Acknowledgment.** We would like to thank all the people who participate in this study and our team members Ihsan Halici and M. Adem Celebi for their efforts in teaching and test assessments.

### References

1. Feurzeig, W.: Programming-Languages as a Conceptual Framework for Teaching Mathematics. Final Report on the First Fifteen Months of the LOGO Project (1969)
2. Clements, D.H., Gullo, D.F.: Effects of computer programming on young children's cognition. *J. Educ. Psychol.* **76**(6), 1051–1058 (1984)
3. Pea, R.D., Kurland, D.M.: On the cognitive effects of learning computer programming. *New Ideas Psychol.* **2**(2), 137–168 (1984)
4. Dalbey, J., Linn, M.C.: The demands and requirements of computer programming: a literature review. *J. Educ. Comput. Res.* **1**(3), 253–274 (1985)
5. Saeli, M., Perrenet, J., Jochems, W.M.G., Zwnaeveld, B.: Teaching programming in secondary school: a pedagogical content knowledge perspective. *Inf. Educ.* **10**(1), 73–88 (2010)
6. Winslow, L.E.: Programming pedagogy - a psychological over-view. *ACM SIGCSE Bull.* **28**(3), 17–22 (1996)
7. Computing our future Computer programming and coding - Priorities, school curricula and initiatives across Europe. [http://www.eun.org/c/document\\_library/get\\_file?uuid=521cb928-6ec4-4a86-b522-9d8fd5cf60ce&groupId=43887](http://www.eun.org/c/document_library/get_file?uuid=521cb928-6ec4-4a86-b522-9d8fd5cf60ce&groupId=43887). Accessed 30 Mar 2017
8. Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Paterson, J.: A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bull.* **39**(4), 204 (2007)
9. Duke, R., Salzman, E., Burmeister, J., Poon, J., Murray, L.: Teaching programming to beginners - choosing the language is just the first step. In: *ACE 2000*, Melbourne, Australia, pp. 79–86 (2000)
10. Kölling, M., Rosenberg, J.: Blue - a language for teaching object-oriented programming. *ACM SIGCSE Bull.* **28**(1), 190–194 (1996)
11. Brilliant, S.S., Wiseman, T.R.: The first programming paradigm and language dilemma. *ACM SIGCSE Bull.* **28**(1), 338–342 (1996)

12. Al-A'Ali, M., Hamid, M.: Design of an arabic programming language (ARABLAN). *Comput. Lang.* **21**(3-4), 191-201 (1995)
13. Annamalai, M.: Ezhil: a tamil programming language. eprint [arXiv:0907.4960](https://arxiv.org/abs/0907.4960) (2009)
14. Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E.: The scratch programming language and environment. *ACM Trans. Comput. Educ.* **10**(4), 1-15 (2010)
15. Doolittle, P.E., Camp, W.G.: Constructivism: the career and technical education perspective. *J. Vocat. Techn. Educ.* **16**(1) (1999). <http://scholar.lib.vt.edu/ejournals/JVTE/v16n1/doolittle.html>. Accessed 30 Mar 2017
16. Bers, M.U., Ponte, I., Juelich, C., Viera, A., Schenker, J.: Teachers as designers: Integrating robotics in early childhood education. *Inf. Technol. Child. Educ. Annu.* **2002**(1), 123-145 (2002)
17. Riedo, F., Chevalier, M., Magnenat, S., Mondada, F.: Thymio II, a robot that grows wiser with children. In: 2013 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO) (2013)
18. Specifications - Thymio & Aseba. <https://www.thymio.org/en/thymiospecifications>. Accessed 30 Mar 2017
19. Magnenat, S., Rettornaz, P., Bonani, M., Longchamp, V., Mondada, F.: ASEBA: a modular architecture for event-based control of complex robots. *IEEE/ASME Trans. Mechatron.* **16**(2), 321-329 (2011)
20. Shin, J., Siegart, R., Magnenat, S.: Visual programming language for Thymio II robot. In: *Interaction Design and Children (IDC)* (2014)
21. Magnenat, S., Longchamp, V., Mondada, F.: Aseba, an event-based middleware for distributed robot control. In: *IEEE-RSJ 2007 International Conference on Intelligent Robots and Systems, Workshops and Tutorials CD* (2007)
22. Overview & Download - Thymio & Aseba. <https://www.thymio.org/en/start>. Accessed 30 Mar 2017

# Creativity and Contextualization Activities in Educational Robotics to Improve Engineering and Computational Thinking

Albert Valls<sup>1</sup> , Jordi Albó-Canals<sup>1</sup> , and Xavier Canaleta<sup>2</sup> 

<sup>1</sup> GR-SETAD. Electronic Systems, Telecommunication and Data Analysis, La Salle-Ramon Llull University, Barcelona, Spain  
{avalls, jalbo}@salleurl.edu

<sup>2</sup> GRETEL. Technology Enhanced Learning, La Salle-Ramon Llull University, Barcelona, Spain  
xavic@salleurl.edu

**Abstract.** One of the objectives of the introduction of educational robotics in the schools is the need to adapt the curriculum of the technology to the today's requirements of the students and the development of the skills, competencies and disciplines involved of STEAM. In this paper cover related aspects of the computational thinking, the engineering thinking required to develop the context-oriented activities through technological platforms based on educational robotics. The contextualization of the activities worked with Scratch and LEGO Mindstorms are the basis of two study groups. Different methodologies of learning of the technological platforms are used in these groups.

The methodology developed during several sessions of the academic course is the main argument to introduce the Educational Robotics and the development of the STEAM in a traditional school of Barcelona.

**Keywords:** Educational robotics · STEAM · Engineering thinking · Computational thinking · Learning process · Creativity

## 1 Introduction

What today is understood by educational robotics involves several trends, tools, and methodologies to teach science and technology contents in the schools.

The educational institutions have introduced educational robots in their curriculum and their technical applications in the last 15 or 20 years, through all the different levels [1]. Consequently, the educational robotics has become an integrative discipline representative of novel teaching methodologies and that indicates technological progress. The use of robots in education goes beyond more research to develop a new technology; integration of robotics in education and research in the field of teaching STEM (Science, Technology, and Mathematics Engineering) or STEAM (Science, Technology, Engineering, Arts and Mathematics) to become a reality [2–4].

Furthermore, robotics as a subject itself is increasingly present in the curriculum organization, and there are many schools that introduce robotics as another subject as their curriculum. The reality, however, is that both trends are very intertwined in the

daily school; which applies robotics as a subject in isolation and is only focused mainly on learning the construction, installation and programming of the robot. This will leak other teaching opportunities and take advantage of the potential of robotics as a learning method [5], both connected to the content linked to STEAM.

Although the current robotics technological reality offers many resources, educational robotics skills need some improvements to cover all stages of education, whether in Early Childhood Education [6] Primary or Secondary Education [7, 8] and is covering applications in different disciplines and learning activities.

The evaluation forms and learning resources based on the development of engineering and computational think are based on the constructionism, such as LOGO [9] have evolved in teaching structures of the coding based on multiple platforms such as programmable block Scratch [10] and LEGO Mindstorms [11]. These learning platforms are used in several schools to introduce the teaching of subjects such as coding, technical, science, and robotics.

Finally, the emergence of new teaching methodologies such as Problem Based Learning or Project Based Learning, both with a vision very close to Constructivism [12] promotes the integration of educational robotics and its entire environment learning in the curriculum.

The next study is focused on the analysis of how they apply these methodologies in learning robotics and STEAM, particularly emphasizing creativeness and context of the activities to improve the structures of thought the development of computational thinking and engineering. The computational thinking is beyond the simple fact of coding or the interaction with computers [13]. This thinking like engineering thinking is related to the ability of analysis, and problem-solving skills. These skills are the ones that students must acquire to be ready for the company today.

## 2 Framework

La Salle Bonanova School, a 125 years-old institution, is in the upper part of Barcelona. The teaching that is carried out can be considered traditional but in the route map of the school exists an intention to shift in the education system towards the use of innovative techniques to enhance learning. To do so, the School has committed to the integration STEAM of the educational robotics and Problem Based Learning and Project Based Learning as a catalyst of this methodological change to all education stages, from pre-school, primary, secondary and high schools covering a total of almost 2000 students.

Within the context of Spain and Catalonia in particular, the content about digital content has a customized design according to each center [14]. This subject is developed along the three years of secondary education with a distribution of 2 h per week throughout the year. The distribution of subjects and teaching units does not exist the subject of robotics explicitly, but knowledge related to technology, computer science, coding, and the use of information technologies in general.

The ratio of Spanish students is about 35 students per classroom, so turns to be challenging to focus on attention to diversity. Thus, the desire to introduce educational robotics as a teaching methodology transverse perfectly fits the trend of methodological

change that is proposed from the Catalan government; where the capacities, skills and competencies are acquired from students becoming the center of teaching methodologies.

### 3 Learning Goals

There are several goals of the introduction of educational robotics in this school. These include promote a series of skills and abilities linked to the STEAM disciplines, solve the challenge of improve engineering thinking, the development of computational thinking, promote cooperative work and have technical knowledge of technology platforms. These objectives are linked to the development of the skills of the students throughout the secondary school program. Is for this reason that in this first year, the aim goal is to establish the bases so that the school crated its own methodology of work with educational robotics.

The main goals during 2015-2016 course, as the first year of implementation of educational robotics program are, to get the first knowledge of various educational IT platforms and enhance skills and abilities of students in technology and science. These two goals are quite different, for this reason and in order to differentiate them clearly, they have been separated in the development of assessments both in the analysis of the results.

The learning goals of the technological platform are focused on the knowledge of Scratch and LEGO Mindstorms. This knowledge, although that will be an initial level, will be large enough for the two platforms to be used in the upcoming courses and use them for learning other STEAM concepts and continue developing engineering and computational thinking.

Although that the learning goals are the same as all educational levels, the difficulty of the activities has been modulated depending on the requirement of the course.

The other main goal is the development of skills and competence in the curriculum of the secondary school. These competences are developed in an own framework [14]. In the case of this study, the authors have made an adaptation of other experiences, in other educational stage and environments that can be found in the literature [15, 16].

And finally, the two last goals of this study are to establish the need to include creativity and artistic aspects as an important factor in the learning process; and contextualize the activities approaches and improves learning technological concepts.

### 4 Methodology

As already mentioned, the subject of Technology in Secondary Education does not specify that teaching robotics is required. However, the curriculum does contain contents and knowledge related to this field. The curriculum organizes the Technology subject in 2 h of class per week during the academic year. The school and the authors of this paper are committed to introducing methodological, logistical and contents changes in this subject.

First performing a separation into two types of contents; the first one is dedicated to the teaching of technological processes of the industrial world, building and design



process. These includes the official curriculum content and is properly distributed throughout the 1st, 2nd and 3rd year of secondary [14]. Although, they are not part of this study, they are imparting weekly to all students of one hour during the academic course. The second types of content are those who are directly related to the use of in Information and Communication Technologies (ICT), and Educational Robotics (ER). It is in this block where the proposal of this study to achieve the learning objectives mentioned.

In the second type of content is a division into two small groups. While one group is doing, sessions focused on knowledge of ER, the other group receives lessons of ICT. Arriving in the middle of the academic year two groups exchanged. Therefore, considering that this study was conducted simultaneously in the first, second and third year of secondary distribution of classes and students is shown in Tables 1 and 2.

**Table 1.** Types of contents and groups distribution

Section 1	Lessons based on technology curriculum - All students	
Section 2	Group 1 ER* - Group 2 ICT	Group 2 ER* - Group 1 ICT
	Course begins	Half course
		End of course

\* The groups studied in this paper are the Group 1 ER and Group 2 ER, which from now on were defined as Group 1 and Group 2.

**Table 2.** Students distribution

	1st Secondary		2nd Secondary		3rd Secondary		
	Group1	Group2	Group1	Group2	Group1	Group2	
1A	17	16	2A	16	18	3A	17
1B	17	16	2B	18	18	3B	17
1C	17	17	2C	17	17	3C	18
1D	17	17	2D	18	17	3D	18
1E	17	17	-	-	-	-	-
	85	83		69	70		70
							68

The division of the class is done to facilitate the work in the robotics classroom. This situation helps us to organize better the material resources and the human resources for all students. So, this way we can work with small working groups, furthermore we can observe that in STEAM subjects and in ER, using creativity, motivation and contextualization the activities, help and encourages to problem solving [17]. This separation in smaller working groups is the reason why there is a small variation in the methodology between Group 1 and Group 2.

What this paper would like to show, beyond the consolidation of the educational robotics methodology as an excellent learning resource, is developed the context to use activities and the creativity to be a fundamental pillar in the teaching STEAM.

Of a total of 445 students, 224 students in the first group receive the classes of ER in the period from September to January; and the second group of 221 pupils performed it from February to June, both terms correspond 15 teaching weeks, and therefore there has been an organized in 15 sessions. Once this division between groups the classroom is organized in small groups of 3-4 students, to achieve the cooperative work and peer-to-peer relation.

#### 4.1 Methodological Differences

The authors wanted to show that the desire of changing educational model that seeks to La Salle Bonanova has a fundamental reason: to improve skills and clever minds to ensure that students meet the current demand of society. That is why to verify this, we have followed two trends that are currently in operation in the school and have led to methodological differences for teaching robotics block corresponding to educative.

With the background of technological learning platforms, LEGO Mindstorms and Scratch generated work dynamics and different teaching methods. The main methodological differences between the groups are as follows:

- Group 1:
  - In this group, each session has been very structured and can be based on traditional teaching, where the teacher conveys his whole focused where students and what you learn and how you learn is much guidance.
  - At its introduction to Scratch, the knowledge platform has been scheduled always and with little freedom of research; programs have been conducted following tutorials and defined structures and objects.
  - The teaching of the Scratch features such as loops, variables, objects movement or conditional has been making small programs where all students were the same task. The final evaluation is done by creating a free video game.
  - At LEGO Mindstorms sessions, students have learned to use the platform in a very structured session. Explained each block separately, making small LEGO Mindstorms EV3 robot applications already assembled. Once the students have an overview of the platform, has been challenged to make geometric shapes.
- Group 2:
  - In this group has been a contextualization of the activities always and entered creativity, art, motivation and collaboration among peers as an engine of creation. Been teaching real applications in robotics and space has been left to the imagination of science fiction films screened and creating fantastic stories.
  - Prior to the Scratch sessions, each group has written three little stories and presented to the rest of the class. Among the three little stories, students choose the best one.
  - Teaching features like Scratch loops, variables, objects or conditional movement has been doing quite a lot of freedom with small programs, although there are some recommended. The final evaluation is done by creating a video game based on one of the stories submitted.
  - In the LEGO Mindstorms platform, there has been an understanding of the working environment but has previously contextualized learning that the

application of the movement of the robot to do geometric shapes known. These shapes are contextualized in the real world, looking for similarities in design, architecture or engineering objects in the immediate environment to the students.

Tables 3 and 4 shows how distributed teaching Scratch and LEGO, how many sessions are needed for each content conceived and what are the teams that are working.

One aspect to highlight is the difference in the learning of some technical aspects. The number of sessions used to show the features of Scratch or Lego Mindstorms are not the same. This is because learning to use the platform has been more agile and faster in the second group. In this case, the second group received instruction on the use of creativity, there have been a work of communication and motivation before using Scratch. Similarly, there has been a contextualization of contents before using LEGO Mindstorms. These two aspects lead to changes in the learning process between the two groups and together with the analysis of the corresponding assessment results, leading to a several considerations about this. These issues are set out in Sect. 6 corresponding to Results.

**Table 3.** Group 1. Sessions distribution

Session name	Sessions	Concepts developed
Learning Scratch	1, 2	Create user account Scratch. Know the framework, libraries, characters, objects, background and operation of the blocks. Observe programs already made
Learning Scratch	3	Observe programs already made and make the first programs following a tutorial
Program 'Pong'	4	Apply motion control blocks to perform the program 'Pong' following a tutorial. Loops and Conditional apply. The 'Pong' game is based on tried bouncing a ball that does not fall on the ground, to prevent it controls bar
Program 'A little story'	5, 6	Make a little story by blocks of dialogue, control of movement and change background and costumes
Video Game	7, 8, 9	Define the concept of variable. Make a free-form game
Learning LEGO Mindstorms	10, 11, 12	Perform basic robot assembly following the instructions manual. Get functioning of Brick. Get workplace software. Interacting with the engine control units and display
Program 'Geometric shapes'	13, 14, 15	Do the challenges go online 1 m straight, make a square, rectangle, circle and number 8

**Table 4.** Group 2. Sessions distribution

Session name	Sessions	Concepts developed
Creativity and motivation	1, 2	Talk about what creativity. Viewing videos on art and creativity. Get students through dialogue, former pressure tastes or hobbies. Teaching applications, videos and films of robotics and technology in today's world
Creativity and communication	3, 4	Writing three little stories by following the structure of introduction, middle and end where several characters appear, go in several different scenarios and can have several different endings. Presentation of three little stories aloud to the whole class support with a brief presentation and class vote on which of the three little stories is the best
Learning Scratch	5	Create user account Scratch. Know the framework
Program 'Pong' or 'Cat-Mouse'	6	There is an exhibition of some programs. Explain 'Pong' and 'Cat-Mouse' games, and then allowed that freedom to choose one and try to do. The 'Pong' game is based on tried bouncing a ball that does not fall on the ground, to prevent it controls bar. The 'Cat-Mouse' game is based on creating two animals chasing each other, depending on whether the cat touch your mouse or mouse touches the cheese is added or subtracted points
Video Game	7, 8, 9	Based on the story chosen the game takes place where you can introduce small variations that have a structure of game
Define and contextualize geometric shapes	10	Definition straight line, square, rectangle, triangle, circle and see in real life applications
Learning LEGO Mindstorms	11, 12, 13	Perform basic robot assembly following the instructions manual. Get functioning of Brick. Get workplace software. Interacting with the engine control units and display
Program 'Geometric shapes'	14, 15	Do the challenges go online 1 m straight, make a square, rectangle, circle and number 8

## 5 Evaluation

The evaluation group has made measuring some issues throughout the sessions under the same parameters for both study groups; it is generated rubrics. It should be mentioned that the notes or evaluations obtained from the rubrics are due to assessment throughout the process, and therefore in the last session activity where you expect a better note. However, we must differentiate two types of evaluation:

## 5.1 Assessment of Competence

The first section corresponded to the type of analysis and evaluation aspects competencies and included in the learning objectives in five areas: Communication, Collaboration and Community Building, Context Creation, Creativity and Conduct or Behavior. These areas are based on a rubric applied to kindergarten competence [15] and that the authors of this paper have adapted to the environment and the context of this study.

For each of these five areas, there are some skills or abilities assessed. This assessment is carried out at each session. However, the authors consider that the evaluation takes more importance in the last sessions, where is the resolution of the activity or challenge. The process of learning the skills is cumulative, so at the last sessions the assessment takes more importance.

An observation of behavior and the development of the activity of each student are done during each session. This observation is marked in the rubrics that shows on Table 5 and follows the 1-5 grading of Likert scale.

**Table 5.** Competent aspect evaluated

C1 Communication	C1.1 Exchange of ideas among group members
	C1.2 Expression of ideas and debate them
	C1.3 Demand for teacher support and is beneficial for the project
C2 Collaboration and Community Building	C2.1 Help peer group
	C2.2 The individual contributions make the group advance
	C2.3 Different work roles/Tasks diversity
C3 Context Creation	C3.1 The activity follows a structure designed
	C3.2 Analysis of the errors in the process
	C3.3 Justification of the solution
	C3.4 Write the process of solution to the challenge
C4 Creativity	C4.1 Initiative to make further steps in programs
	C4.2 Use of various elements outside environment platform
	C4.3 Application of concepts from other disciplines
C5 Conduct	C5.1 Concentration activity
	C5.2 Following the rules of the classroom
	C5.3 Responsible use of the material
	C5.4 Behavior with classmates and teacher
	C5.5 Motivation towards activity

In general, in all items the score of 1 corresponds to a very low level of competence or ability. On the other hand, the score of 5 corresponds to a complete integration of the skills in the development of activities were justified and argued the process and the takes of decisions.

## 5.2 Assessment of Contents

The other section is used to evaluate the relevant concepts and contents related platforms Scratch and LEGO Mindstorms. In this case, each platform specific concepts have been evaluated. Is for this reason that has been separated into two different rubrics. The items evaluated in each of these rubrics are shown in Tables 6 and 7.

**Table 6.** Evaluated items on Scratch activity

Conditional structures	- Understand the concept of conditional structures - Use different types of conditional structures
Loops	- Understand the concept or loop iteration - Use loops within the structure of the game
Objects	- Use various objects - Import objects outside environment Scratch - The motion control is done several ways - The use of objects follow criteria established
Scenario/Dresses	- Use different scenarios - Make changes in objects dresses
Bloc Posts	- Use the blog post to give orders objects
Text	- Use language structures - Dialogues appear
Variables	- Use variables to make a counter + - Use variables to make a counter - - Conditions certain actions variables
Music/Sounds	- Use music blog - Use varied sounds - Use block sound conditioning in another action

**Table 7.** Evaluated items on LEGO activity

Blocks movement	- Apply different types of engine blocks movement - Understand the various parameters that make up the blocks
Conditional structures	- Understand the concept of conditional structures - Use different types of conditional structures
Loops	- Understand the concept or loop iteration - Use meaningful use of the geometric shapes in the loop
Geometric shapes	- Apply different solutions depending on the geometric shapes - Understand the characteristics of shapes, and this is reflected in the solution of the program
Process	- There is a preliminary approach in solving geometric shapes - Express reasons solving each geometric shapes

The evaluation of these aspects is made through the delivery of activities and resolution of the challenges. For Scratch the evaluation is based on the video game, and in the case of LEGO Mindstorms evaluation is based on the resolution of the challenge of the geometric shapes. For each of the items is a graduation from 0 to 10, where 0 corresponds to no knowledge of the concept or item, and it is not used significantly in the resolution of the challenge; 10 corresponds to a high level of understanding of the concept and used perfectly within the platform.

## 6 Results

Following the structure of rubrics for assessment, the results are presented in two groups. The first group corresponds to the mean and standard deviation of the five competency areas, as shown in Table 8. The other group corresponds to the results of the evaluation of Scratch and LEGO platforms, which has also been made the mean and the standard deviation of all items and the results are grouped by levels and groups as shown in Table 9.

**Table 8.** Mean and Standard Deviation of competent areas for each course and group

		C1		C2		C3		C4		C5	
		Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Group 1	1st	2.68	0.50	2.48	0.64	2.25	0.77	1.66	0.65	3.73	0.86
	2nd	2.67	0.46	2.56	0.57	2.27	0.72	1.68	0.57	3.75	0.68
	3rd	2.84	0.42	2.54	0.58	2.45	0.61	1.95	0.56	3.79	0.69
Group2	1st	3.29	0.69	2.96	0.74	2.64	0.80	2.57	0.80	3.95	0.77
	2nd	3.06	0.66	3.01	0.67	2.66	0.66	2.61	0.80	4.04	0.54
	3rd	3.14	0.66	3.00	0.68	2.71	0.62	2.74	0.75	4.07	0.50

**Table 9.** Mean and Standard Deviation of Scratch and LEGO task for each course and group

		Scratch		LEGO	
		Mean	SD	Mean	SD
Group 1	1st	5.31	1.04	5.17	0.93
	2nd	5.44	1.08	5.11	0.91
	3rd	5.89	1.48	6.26	1.24
Group 2	1st	6.00	1.02	6.53	1.08
	2nd	5.92	1.19	6.22	1.09
	3rd	6.79	1.32	6.99	1.05

The results obtained in the competent areas have several readings. It studies began with the same knowledge to all groups and levels, and that the work is based on the same activity. That is why we should wait for the results obtained significant upper-mind better than other courses. However, in most of the skills, the mean is not significantly higher up there, and in some cases, the 3rd course does not get the best results.

On the other hand, if we compare the results between the two groups studied, observed in almost all skills are assessed improvement in group 2 compared to group 1. These differences in the average range between 0.22 and 0.61 in competences C1, C3 and C5. This is more relevant on skill C2. Collaboration and Community Building, and C4. Creativity, where differences in the average were 0.45 to 0.93.

The results of the evaluation of Scratch and LEGO platforms follow different trends and results of skills evaluation. Because these platforms have worked more technical and theoretical concepts, the results reflect the idea of getting some higher scores in the higher grades. The observed difference is not much when compared with the 1st to 2nd Secondary, but instead, highlights the increase in ratings to 3rd

## 7 Conclusions

The analysis of the results based on the difference method is one of the objectives described in this paper. The overall trend in the world of education is to enhance the skills and capabilities over to put emphasis on the theoretical aspects of the technology. We can see how the Group 2 obtained better results in the areas of competence and knowledge of technology platforms in comparison to Group 1. This validates the new approach proposed by the authors.

Another lecture that can be extracted from this study is how to direct lessons and activities. Beyond based on competence issues, the activities to be developed in the second group have been contextualized in situations close to students. Creativity and freedom to continue the learning process of students in the second group has encouraged creativity and learning more participatory. This has led to perform a task where solutions have been more creative and varied. Programs have been longer and more complex, therefore developed better computational thinking and engineering thinking. The goal of improving the technical knowledge of some of the technological platforms in the educational robotics also has been fulfilled. While both groups are treated so satisfaction these concepts, is in the second group where there is a significant improvement in the average and therefore an increase knowledge technical platform

One aspect observed by the authors that have not been mentioned in the process of obtaining the results is that the total duration of the course has not been homogeneous for both groups. Initially, the program was the course of 15 sessions, since the first group need to complete all 15 sessions learning, and in some cases, the low score is caused by not being able to complete the learning sessions. In contrast, the second group, usually need 14 sessions to complete the process and activities. We consider this very important when one of the most important aspects of the development of school curricula is the duration of the course and the needs to improve the learning process. To improve the analysis of results in future experiences, the authors study to use videos and other metrics, such as surveys or interviews to students. With these tools, we could get other data that the current use of rubrics cannot be analyzable.

The development of skills and abilities of the subjects STEAM should help these students to develop in the world of science and technology. Therefore, the main learning goal of this paper was to establish methodological bases in this school, to promote and improve learning skills and knowledge of some technological platform



has been fulfilled. Future challenges for in coming years are focused into two aspects. The first relates to improving the design of activities. Especially standardize the environment in which contextualizes and promote a more active creative aspect. Another aspect to consider is the use of the technical concepts of programming. and thus, improve the use of platforms used in educational robotics, whether Scratch, LEGO Mindstorms or others.

**Acknowledgment.** Special acknowledgment to La Salle Bonanova School, to believe in the need to include Educational Robotics as an important aspect of students' education.

La Salle-Ramon Llull University, to promote the use of Educational Robotics in school environments.

Colleagues, to the help received during the study and make this paper possible.

And finally, the most significant actor in this study, the students of La Salle Bonanova School. With enthusiasm and patient, they have enabled this project.

## References

1. Danahy, E., Wang, E., Brockman, J., Carbery, A., Shapiro, B, Rogers, C.: LEGO-based Robotics in Higher Education: 15 years of student creativity invited review. *Int. J. Adv. Robot. Syst.* (2014)
2. Brown, J.: The current status of STEM education research. *J. STEM Educ.* **13**, 7–12 (2012)
3. Madden, M., Baxter, M., Beauchamp, H., Habermas, D., Huff, M., Ladd, B., Pearson, J., Plague, G.: Rethinking STEM education: an Interdisciplinary STEAM curriculum. *Procedia Comput. Sci.* **20**, 541–546 (2013)
4. Stohlmann, M., Moore, T., Roehrog, G.: Considerations for Teaching Integrated STEM Education. *J. Pre-College Eng. Educ. Res.* **2**(1), 28–34 (2012)
5. Alimisis, D.: Educational robotics: Open questions and new challenges. *Themes Sci. Technol. Educ.* **6**, 63–71 (2013)
6. Bers, M.U.: *Blocks, Robots and Computers: Learning about Technology in Early Childhood.* Teacher's College Press, NY (2008)
7. Kee, D.: Educational robotics. primary and secondary education robot. *Autom. Mag.* **18**(4), 16–19 (2011). IEEE
8. Sans-Cope, O., Albó-canals, J., Barco, A., Diaz, M., Angulo, C: Robotics Montserrat: A case of learning through robotics community in a primary and secondary school (2014)
9. Papert, S., Seidel, R.J, Rubin, M.: A learning environment for children. In: *Computers and Communication: Implications for Education*, pp. 271–278. Academic Press, New York (1977)
10. Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: programming for all. *Commun. ACM* **52**(11), 60–67 (2009)
11. Papert, S.: *Mindstorms: Children, Computers, and powerful ideas.* Basic Books, New York (1980)
12. Piaget, J.: *To Understand is to Invent. The future of Education.* Grossman Publishers, The Viking Press, Inc.
13. Wing, J.M.: Computational thinking. *Commun. ACM* **49**(3), 33–35 (2006)

14. Generalitat of Catalonia Department of Education. Curriculum of Secondary Education. Basic skills in the digital realm. Identification and deployment in compulsory secondary education (2013)
15. Bers, M.U.: (2010) Beyond computer literacy: Supporting youth's positive development through technology. *New Dir. Young Dev.* **128**, 13–23 (2010)
16. Bers, M.U., Flannery, L., Kasakoff, E.R., Sullivan, A.: Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Comput. Educ.* **72**, 145–157 (2014)
17. Alonso, J.: *Motivation and learning in the classroom. How to teach to think*, Santillana (1991). ISBN: 84-294-3334-1

# Educational Robotics for Communication, Collaboration and Digital Fluency

Ivaylo Gueorguiev<sup>1</sup>(✉), Christina Todorova<sup>1</sup>, Pavel Varbanov<sup>1</sup>,  
Petar Sharkov<sup>1</sup>, George Sharkov<sup>2</sup>, Carina Girvan<sup>3</sup>,  
Nikoleta Yiannoutsou<sup>4</sup>, and Marianthi Grizioti<sup>4</sup>

<sup>1</sup> European Software Institute – Center Eastern Europe (ESI CEE),  
Sofia, Bulgaria

{ivo,tina,pavel}@esicenter.bg, petarsharkov@gmail.com  
<sup>2</sup> Faculty of Mathematics and Informatics, Plovdiv University, Plovdiv, Bulgaria  
gesha@esicenter.bg

<sup>3</sup> Cardiff University, Cardiff, Wales, UK  
girvanc@cardiff.ac.uk

<sup>4</sup> UoA ETL, National and Kapodistrian University of Athens, Athens, Greece  
{nyannoutsou,mgriziot}@ppp.uoa.gr

**Abstract.** This paper is the experience-based summary of the work with the design, implementation and results from an “Educational Robotics and Creativity Workshop” under the EU funded Horizon 2020 project “ER4STEM – Educational Robotics for Science, Technology, Engineering and Mathematics”. This paper gives an overview of the empirical data obtained from the post-workshop questionnaires, completed by the participants from 13 educational robotics workshops, performed in 7 schools (public and private) in Bulgaria with 312 students (142 girls and 170 boys) in the time period from February 16, 2016 until May 31, 2016. The students were between 7 and 14 years old with the majority of them aged between 9 and 10 years old.

**Keywords:** Educational robotics · Creativity · Collaboration · Communication · Digital fluency · Arduino · Visual programming · Robotics in education

## 1 Background

Today, especially within the context of the rapidly developing technological environment, we hardly ever fail to realize the need of an educational system built on powerful ideas with personal meaning to keep students motivated to study the STEM disciplines. Most children are learners by nature, meaning they manifest interest to learn more about the world they live in - about how things work, about mechanics and technology. To keep this interest alive, especially when it comes to science, technology, engineering and mathematics, we believe that we have to cultivate in students a learning attitude of excitement, personal interest and social meaning.

As a result of the EU funded Horizon 2020 project, “ER4STEM – Educational Robotics for Science, Technology, Engineering and Mathematics”, the European

Software Institute – Center Eastern Europe, with the direct support and contribution of the project consortium, was able to develop and implement with more than 600 students an educational robotics workshop with the aim to promote and prove in practice the value of those beliefs.

The authors base this article on the empirical data obtained from the post-workshop questionnaires, completed by the participants from 13 educational robotics workshops, performed in 7 schools (public and private) in Bulgaria with 312 students (142 girls and 170 boys) in the time period from February 16, 2016 until May 31, 2016. The students were between 7 and 14 years old with the majority of them aged between 9 and 10 years old.

This paper aims to present in brief the experience of working on this educational robotics workshop in its entirety, including the necessary overview of the methodology underlying the workshop. The article explores the feedback received in respect to the workshop objectives to encourage the development of skills in the participants in the field of creativity, digital fluency (technology, engineering and science), communication and collaboration. With this paper, we aim to contribute to the promotion of the importance of such activities within the academic life of both students and pedagogues, based on the results received throughout an extensive evaluation process and interviews with teachers.

Last, but not least, this paper represents the combined effort of a group of robot-enthusiasts to share their experience with robotics as a tool to teach 21<sup>st</sup> century skills, along with subjects of digital fluency, and to share the joy of applying robotics as a way to keep students curious to learn about the world.

## 2 Workshop Design

As the nature of this paper requires a certain level of knowledge on the multi-layered methodological background of this workshop, this paper will give a brief overview of the pedagogical approach, underlying the design of the educational robotics workshop. A review of the social orchestration will follow, along with a description of the basic hardware and software solutions, designed to fit the pedagogical purposes of this workshop. Lastly, this chapter will provide information about the goals of this workshop, related to promote a set of 21<sup>st</sup> century skills, as well as a presentation of the authors' understanding of those skills.

### 2.1 Pedagogical Approach

The basic pedagogical theory underlying the design of all educational robotics workshops within the ER4STEM project, including ESI CEE's educational robotics workshop for creativity, is constructionism. According to the project's understanding of constructionism, a constructionist pedagogical setting is one, where learning is connected to powerful ideas inherent in constructions with personal meaning for the students [1]. Furthermore, ER4STEM places special emphasis on the social dimension of the construction process aiming to introduce the maker culture to the students

(i.e. sharing, discussing, reflecting around constructions) cultivating a learning attitude growing out of collaboration-based experiences [2, 3].

The pedagogical approach, as well as the background for the elaboration of the educational robotics workshops within the ER4STEM project are coordinated and structured with the means of an Activity plan template. The template provides a generic design instrument that identifies critical elements of teaching and learning with robotics based in theory and practice and is expected to contribute to the description of effective learning and teaching with robotics.

With the above considerations in mind, the Activity plan template developed for designing Robotics activities for the ER4STEM workshops, addresses the following aspects: (a) Focus and resources: reference to the different domains involved, different types of objectives, duration and necessary material; (b) contextual information regarding space and characteristics of the participants; (c) social orchestration of the activity (i.e. group or individual work, formulation of groups etc.); (d) a description of the teaching and learning procedures where the influence of the pedagogical theory is mostly demonstrated; (e) expected student constructions; (f) description of the sequencing and the focus of activities; (g) means of evaluation [4].

## 2.2 Workshop Setting

ESI – CEE, designed and developed an activity plan focusing on the use of educational robotics for creativity and addresses boys and girls within the age group 8–12 years old. This activity plan is implemented in a set of 13 different workshops taking place in the school with the status of extra-curricular activity. This means that the workshop does not need to be aligned to the curriculum – although it can be – and student participation in the workshop is not connected in anyway (e.g. in terms of grades) to the subjects they are following in school. The workshop takes place during school hours; the duration of each workshop is not more than eight hours in total and it is usually divided into two consecutive sessions of equal duration.

Two main challenging requirements were identified by the development team. On the one hand, it was of utmost importance that the workshops are designed to adequately align to the environment of a regular, in-school and at-class context of public general education schools. In Sofia, Bulgaria, where most of the workshops were conducted, a regular school class consists of students, anywhere within the range of 24 to 28 students. That number of students, supposed to work simultaneously, presented a challenge to the design of the workshop.

On the other hand, a feasible workshop for this context, had to be of a duration, not surpassing 8 h. As mentioned above, the design team, in most cases, chose to conduct the workshops in two consecutive sessions of equal duration. The brevity of this time frame was challenging for the design of an adequate creativity program. This posed the need for the team to optimize the other sessions in their entity, including materials and pedagogical approach, so that sufficient time and attention could be dedicated to the so called “soft sessions” of the workshop, namely MODULE 4: ROBOT’S TOUCH and MODULE 5: LET’S IMAGINE. A solution identified by the team, was to lead the soft sessions, between the technical sessions (MODULE 3: CONSTRUCTING A ROBOT and MODULE 6: PROGRAMMING A ROBOT), thusly providing sufficient time for

the tutors to check the robots for short circuits and other dangerous mistakes, and provides students with the opportunity to think creatively and not overburden them with consecutive technical sessions.

The workshop is structured into 7/seven/modules:

### **MODULE 1: Introduction and Pre-Evaluation**

The tutors introduce themselves and explain what they do and why they came to the school. They also explain what they find fascinating about robots and what the task is for the specific day. Next, they discuss with the students whether they like robots, if they have had any experience with them in order to informally introduce themselves by their interest within the topic of robotics. The purpose of this module is for the tutors “to break the ice”, trigger the students’ interest in robotics, become familiar with the students, get to know some names and show that they are interested to learn with whom they are going to work with. This first module is important for the specific setting in which the workshop is implemented: i.e. external tutors, collaborating, for 8 h only, with students and teachers they haven’t seen before, which means they are not acquainted with the norms of the specific school and they are not familiar with the specifics of the classrooms participating in the workshop.

### **MODULE 2: What Is a Robot**

The tutors ask students “what is a robot” to generate ideas what are the key components of the robots. Once a student generates an idea, the tutors encourage the others to comment and contribute. This module aims to engage students in a meaningful discussion about robotics and inspire their imagination, thus providing basic information about robotics and influencing positively their attitudes about the workshop.

### **MODULE 3: Constructing a Robot**

In this module, students are introduced to the robotic kit. Tutors show the different elements in the kit and say a few words about their purpose. They show, as an example, the assembled kit, to motivate the students and give them an idea towards what they are working on. Students are encouraged to shift roles within their team, so that everyone could learn and everybody participates. This module is intended for students to collaborate with their teams, gain practical experience and become confident that building a robot is not difficult when in a team.

### **MODULE 4: Robot’s Touch**

Once students build their robots, the tutors demonstrate in action different types of robots such as NAO, VGo, omnidirectional robots, the Finch robot and facilitate a Question and Answer session. The purpose of this module is to showcase different robots, with various applications, but, regardless, similar elements to what they already know, after assembling their robot. Showing similarities between the project, that the participants have just completed, and more complex robots, encourages students’ confidence in their skills and knowledge about robots. In addition, students are inspired to think about and imagine different robots and their application in the real life.

### **MODULE 5: Let’s Imagine**

The tutors engage the participants in a discussion on what creativity is and how important it is in everyday life. They are using a set of predefined games aiming to

demonstrate different aspects of creativity and logical thinking. The goal of those game is to put participants in a creative mood and liberate them from some predefined notions on how the world functions in order to stimulate them to find “out-of-the box” and “crazy” solutions to a given problem. The purpose of this module is to inspire students to believe in the value of their ideas and to boost their confidence to openly share with others their ideas on various applications of robotics.

### **MODULE 6: Programming a Robot**

The tutors say a few words about the basics of programing with Scratch and how specific blocks are used to control the motors and the sensor. Each team uses the “set up” block to switch on the robot. The teams are left to experiment with the blocks, their functions, and the ultrasonic sensor (to measure the distance between the sensor and obstacles). The students have to “discover” how to program the robot, so that it would turn left and right, around its center, forwards and backwards at a different speed. The purpose of this module is for the students to gain programing skills and to have fun programming.

### **MODULE 7: Final Evaluation**

Evaluation session is held for students to present their achievements and evaluate their experience. Group and/or individual interviews are conducted and students fill out Post-Workshops Questionnaires.

The workshop implementation team consists of 3–4 tutors for a workshop with the maximum number of 30 participants. Furthermore, the above activity is designed for indoor implementation only within an adapted, but yet regular school setting. The adaptation usually involves the positioning of furniture – chairs, tables, computers, etc. – aiming to create a makers’ space for the participating students and facilitate group work.

Groups of 3–5 students are formed under no specific criteria. Workshop tutors delegate the responsibility of choice of teammates to the students themselves. A reason for this is the belief that if students are able to sit together, based on pre-established friendships, they would generally manifest a more positive attitude towards the workshop and the educational activity. On a rare occasion, schoolteachers would form groups, mainly due to disciplinary concerns. In cases of students with disabilities, the implementation team responds according to their special education needs.

## **2.3 Robotics Artefacts**

For the purposes of this educational robotics workshop, and in alignment to its pedagogical purposes, the European Software Institute – Center Eastern Europe has developed a custom Arduino-based robotics set.

In order to encourage the implementation of innovative technology in the education process on a local level, ESI CEE aimed at creating a cost-effective kit with easily replaceable and adaptable standard components.

The elements in the robotics kit are connected through a breadboard using wires. Moreover, the mechanical parts are fixed together with plastic pins which allows for them to be easily assembled and disassembled many times. The assembled robot is a small tank that can be controlled with either a PC using USB cable, or a Bluetooth module.

Furthermore, the robot can run autonomously through a program code, uploaded to the controller.

Among the advantages of the created platform, in comparison to other robotics platforms for young students in which the electronic components are not directly visible (i.e. “black box”), the ESI CEE Arduino robotics kit uses, with some minor modifications, original engineering elements such as Arduino Uno board or other adapted compatible microcontrollers, ultrasonic sensors, motor drivers, LEDs and resistors. This way, young researchers are enabled to make their first steps in electronics and robotics by experiencing technology in a way they most likely rarely see it - as a white box. The final artefact is a robot-tank, assembled with the use of a visual guide, consisting of photos only, allowing children to learn by doing and experimenting.

## 2.4 Digital Artefacts

In addition to the Arduino IDE, ESI CEE adapted a program code that allowed younger students to control the robot using visual languages such as Scratch and Snap.

The implementation team chose visual programming software as an appropriate version for younger learners to overcome some of the predominantly age-based difficulties of programming, while still enabling children to experience and learn the logic and concepts behind programming.

ESI CEE chose the desktop version of Scratch for most of the workshops, reason being that this way, avoiding internet connectivity issues interfering with the workshop’s implementation process, becomes easier. Furthermore, limited internet connectivity mitigates safety risks for children during the workshop.

ESI CEE used s2a\_fm software to control the Arduino Uno board through Scratch or Snap. The team developed custom blocks to set up the robot for work, to visualize data from the sonar sensor and to control each of the motors through numerical values between  $-100$  (max speed of the motor backward) to  $100$  (maximum speed of the motor forward) while  $0$  value stops the motor.

## 2.5 21<sup>st</sup> Century Skills

Based on an extensive research of the literature for the 21<sup>st</sup> century skills, the ER4STEM partners focused on the development of a specific set of skills through the project activities. In particular, ESI CEE, through this educational robotics workshop, aims to use robotics as a tool for the cultivation and development of communication, collaboration and digital fluency, which are among the most important 21<sup>st</sup> century learning skills. Bellow we present an overview of these three skills and how we approach them through the ER4STEM project.

By **collaboration skill**, we refer to the ability of students to work effectively and respectfully with others. More specifically to (a) contribute constructively to project teams (b) be helpful and make necessary compromises to accomplish a common goal (c) assume shared responsibility and value the individual contributions when working in a team (d) use collaborative technologies to connect and work with others (i.e. peers, experts or community members, etc.) globally. For the development of these skills, through the robotics workshops, we aim to create situations where students will have to



work in teams, distribute roles and build a public artefact that will trigger discussions and argumentation. Having this pedagogical approach in mind, the authors of this workshop designed the activities correspondingly. More precisely, students in each team are changing roles during the implementation of the tasks, which enables them to learn more about working effectively and respectfully with others in order and to build relevantly complex robotics system.

Similarly, considering **communication as a 21<sup>st</sup> century skill**, students should be able to communicate with others effectively. This includes the ability to (a) articulate thoughts and ideas effectively using oral, written or nonverbal communication skills (b) communicate complex ideas clearly and effectively (c) publish or present content that customizes the message and medium for their intended audience (d) utilize multiple media and technologies in order to communicate and know how to judge their effectiveness (e) communicate effectively in diverse environments. The essence of the workshop presented here, requires from students to clearly communicate their plans, ideas and feelings to the other members of their group.

Finally, by the skill of **digital fluency**, we refer to the technological and science-related knowledge of the students. Thus, digital fluency includes the ability to understand the fundamental concepts of technology operations and to know how to use digital technology and media as tools to research, organize, evaluate and communicate information. Through the activities of this workshop, students learn more about the core elements of a robot (technology), they construct a robot (technology & engineering) and develop a visual program to control the robot in order to execute tasks (technology).

### 3 Evaluation

In order to evaluate ER4STEM activities, a mixed-method multiple-case study design was used. Data collection consisted of questionnaires, observations, reflections, interviews and artefacts of learning. For the purposes of this paper, we will focus on data collected through questionnaires at the end of the workshops, in order to answer the following questions:

- (1) Do educational robotics workshops inspire students' interests in STEAM?
- (2) Do educational robotics workshops support learners to develop digital fluency, communication and collaboration skills?

Although the focus-group interview data that was also collected at the end of each workshop, could be used to answer these questions, the questionnaire provides us with a broad understanding of all learners' perceptions of the workshops, rather than just a select few participants. The questionnaire primarily utilizes Likert-scale and yes/no questions and allows for analysis of the data between age groups, workshops and by gender.

The questionnaire was given to every student who had parental informed consent to participate in the research, at the end of the workshop. In total 381 students participated in the workshops and 312 students completed the questionnaire. Of these 170 were boys and 142 were girls. Not every question was answered by every student and, therefore, we also report in the Findings below the number of times a question was unanswered, for clarity.

## 4 Findings

In this section, we present the findings of the analysis of the questionnaires in relation to the research questions posed.

**(1) Students liked the educational workshop activities and were inspired to do more educational robotics activities in future.**

The vast majority of the students reported the educational robotics workshop activities as interesting, fun and not very difficult (Table 1).

**Table 1.** Aggregated students' feedback related to problems and their work with robots.

	Strongly disagree	Disagree	Neither agree Nor disagree	Agree	Strongly agree	Blank
The problems we had to solve were:						
Interesting	0%	0%	1%	9%	87%	3%
Difficult	40%	24%	18%	8%	7%	4%
Fun	1%	1%	2%	11%	84%	2%
Working with robots was:						
Interesting	0%	0%	1%	7%	89%	3%
Difficult	39%	26%	16%	6%	6%	6%
Fun	0%	0%	1%	7%	88%	4%

The students reported a very high level of overall satisfaction from the workshops (4.9 of max 5.0), 90% of the students reported that they “would like to try to solve more challenges like this one” and the same share (90%) of the students “would like to do more activities like this one”.

**(2) The majority of students, who participated in the workshops reported an improvement in their skills in technology and science and consider robotics as an interesting and important subject.**

Technology and science were reported as the leading knowledge fields applied and further developed during the workshops followed by “How the things work” (Table 2).

Moreover, students reported in the questionnaires an increased interest towards studying science and learning about how things work (Table 3).

It is interesting to mention that although encouraging the development of mathematical skills and interest was not directly targeted as workshops goals, the majority of the students – 74% reported that they “**understand how important mathematics is**”. Another 54% of the students stated that they applied their knowledge in mathematics to solve the workshop tasks. A further 36% reported, “**Working with robots has helped me to learn more about mathematics**”.

**Table 2.** Knowledge applied during the workshops

Working with robots I have used my knowledge of...	
Science	73%
Technology	84%
Art	29%
How things work	66%
Mathematics	54%
Working with robots has helped me to learn more about...	
Science	69%
Technology	87%
Art	24%
How things work	64%
Mathematics	36%

**Table 3.** Interest towards STEM

I am now more interested in studying science	89%
I am now more interested in learning about how things work	91%
I would like to build robots to solve problems in the future	80%
I would like to use robots to learn in the future	90%
Now I understand how important mathematics is	74%
Now I understand how important science is	84%
I would like to learn more about programming	88%
I understand how robots can be used to solve important problems	77%

### (3) Good cooperation and collaboration.

The majority of students reported working in a team as interesting, fun and not difficult (Table 4). Most of the students manifested positive attitudes towards aspects of teamwork, such as communication and collaboration.

Students generally enjoyed working as a part of a team, helping others and felt encouraged by their teams. To support that, students mostly showed disagreement with attitudes that do not support teamwork, communication and collaborations such as “working on my own”, giving up quickly, or being bored.

**Table 4.** Working in a team, communication and collaboration

	Strongly disagree	Disagree	Neither agree Nor disagree	Agree	Strongly agree	Blank
Working in a team was:						
Interesting	1%	1%	5%	14%	75%	4%
Difficult	45%	20%	16%	4%	9%	5%
Fun	2%	1%	4%	14%	76%	4%
During the workshop...-						
I worked as part of a team	2%	0%	5%	11%	78%	4%
I worked on my own	65%	18%	6%	3%	5%	3%
I helped design a robot	6%	3%	10%	20%	56%	5%
I helped create a robot	3%	4%	2%	18%	67%	5%
I helped program a robot	3%	1%	6%	16%	71%	4%
I was able to choose what I wanted to do	14%	7%	18%	15%	42%	4%
I feel that other people did not listen to me	38%	13%	16%	9%	21%	4%
I did most of the work	29%	19%	25%	9%	13%	5%
I was encouraged by my team	7%	5%	14%	22%	49%	3%
I was bored	71%	13%	5%	2%	3%	6%
I liked sharing what I had done with other people	2%	1%	9%	16%	68%	3%
I helped someone	6%	3%	18%	22%	46%	5%
I gave up too quickly	71%	13%	3%	4%	6%	4%

## 5 Conclusions

In this article, we presented our experience with applying educational robotics as a tool to enhance the learning of science, technology, engineering subjects and mathematics, as well as a tool to cultivate 21<sup>st</sup> century skills in a constructionist setting. This brief overview of a fraction of the data received by students was shared, to serve as a positive example of the opportunities from the application of educational robotics as a tool for

introducing general education subjects. The following research questions were derived and postulated:

**Do educational robotics workshops inspire students' interests in STEAM?**

According to the data received, **students feel more inclined to study the science, technology, engineering subjects and mathematics following the workshops.** The educational robotics workshops showcase the combined product of the application of the above-mentioned subjects in an appealing and intriguing way. Assembling a complex robot provides the space necessary for students to construct their own knowledge-based structures, thus enabling them to explore new ideas and express their creativity [5].

**Do educational robotics workshops support learners to develop digital fluency, communication and collaboration skills?**

This paper overviewed the design of an educational robotics workshop, formulated around the pedagogical concept of constructivism to support the development of 21<sup>st</sup> century skills. Namely, the target skills, which this educational robotics workshop aims to develop are creativity, communication, collaboration and digital fluency. Empirical data to support the conclusion that educational robotics has the capacity to positively influence, and thusly to support, young learners to cultivate skills and knowledge in digital fluency, communication and collaboration, was presented.

The design of the evaluation of this workshop presents an opportunity for the participants to reflect on particular concepts, related to 21<sup>st</sup> century values, for example teamwork. **Students show a tendency to enjoy the teamwork aspects of the workshop.**

The workshop's aims to support a **constructionist setting of working in groups to engage students in tasks, requiring abilities, such as clearly communicating ideas, voicing out concerns, proposing solutions,** based on pre-existing knowledge. The workshop's educational plan, by design, involves covering main stepping-stones of digital fluency.

The empirical data collected through the post workshop questionnaires does not provide information about if the workshop develop creativity. The authors are researching the artefacts and are analysing interviews and observation to answer this question, which will be discussed in a separate paper.

## **5.1 Feedback from Schools and Educational Institutions**

Among the reasons to determine the constructionist approach in education as one of an increasing appreciation, is the positive feedback on the learning methodology from schools, educational authorities and academic partners.

Most of the schools, where the workshops were implemented, are now strong supporters of the idea of constructionist education and the application of robotics tools in education. Receiving good feedback from schools on the activities and establishing sustainable partnerships with the schools could serve as an indicator for the positive results from the educational activities.

Among the examples is 125 High School “Boyan Penev”, which, inspired by the outcomes of the workshop and the positive feedback from students and parents, used the gained experience to formally become an innovative school in Bulgaria and embrace the constructionist approach as an inseparable part of their educational curricula.

Partners from the Mathematics Gymnasium in Kiustendil, with the support of the city mayor and the municipality, decided on applying and further developing this educational robotics workshop in their school. Teachers and representatives from the school attended educational robotics workshops in Sofia on multiple occasions and decided upon organizing it locally at the gymnasium. For this purpose, they are currently training their teachers on the particularities of the workshop with the support of ESI CEE.

The European Software Institute - Center Eastern Europe has also received positive feedback from other counties on the educational robotics workshop. Moldovan partners are currently in the process of organizing train-the-trainer activities in Moldova aiming to implement this educational technology in more than 10 schools.

## 5.2 Current Work and Future Prospects

Based on tutor reflections and students’ feedback, the European Software Institute - Center Eastern Europe improved some hardware issues, making the assembly of the robot more pleasant. Such improvements are enhancing the stability of the battery holders by changing the design of the pins, ensuring stability of the ultrasonic sensors and the microcontroller, when the robot is in operation.

Various aspects of the visual guide for the assembly of the robot were changed to make it clearer and easier to navigate through in order to further facilitate the collaboration among the team members. Furthermore, software solutions were researched and implemented to make fixing issues during operation quicker, as well as to improve the functionalities, along with the usability for both tutors and students.

The improved activity plans and evaluation protocol, kit and tools made the process of obtaining feedback from the students easier, quicker and more effective in terms of the ER4STEM project goals.

Moreover, based on the positive experience gained throughout the organization and successful implementation of this series of educational robotics workshops, ESI CEE was inspired to create another educational robotics workshop. “Visualizing mathematics with the Mathbot” already has more than 160 successfully trained participants with even higher level of students’ appreciation. This workshop aims to build on the educational robotics for creativity workshop. Using the Finch Robot by BirdBrain Technologies LLC., and by applying more advanced programming tasks it aims to encourage positive attitudes towards learning mathematics by teaching, demonstrating and exercising in practice students’ knowledge on the basic mathematical concepts, in support to the Bulgarian national educational curriculum on mathematics for the 4<sup>th</sup> grade.

The sustainable partnerships, established with our academic partner, allow us to continue improving and developing new educational technologies, based on robotics and programming to support general education.

With multiple international stakeholders showing interest and appreciation of the technology, we firmly believe that educational robotics has the power to reform insurrectionism and result in the implementation of constructionist practices for improving learning outcomes and contributing to quality education.

**Acknowledgements.** The project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 665972. Project Educational Robotics for STEM: ER4STEM

## References

1. Papert, S.: *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc. (1980)
2. Kafai, Y.B., Burke, Q.: Constructionist gaming: understanding the benefits of making games for learning. *Educ. Psychol.* **50**(4), 313–334 (2015). doi:[10.1080/00461520.2015.1124022](https://doi.org/10.1080/00461520.2015.1124022)
3. Kafai, Y.B., Burke, Q., Mote, C.: What makes competitions fun to participate?: the role of audience for middle school game designers. In: *Proceedings of the 11th International Conference on Interaction Design and Children*, pp. 284–287. ACM (2012). <http://dl.acm.org/citation.cfm?id=2307146>. Accessed
4. Yiannoutsou, N., Nikitopoulou, S., Kynigos, C., Gueorguiev, I., Fernandez, J.A.: *Activity plan template: a mediating tool for supporting learning design with robotics*, pp. 3–13. Springer International Publishing, Cham (2017)
5. I.R. Management Association, *Robotics: Concepts, methodologies, tools, and applications: Concepts, methodologies, tools, and applications*, Information Science Reference (2013)

# Using Robotics to Foster Creativity in Early Gifted Education

Tomislav Jagust<sup>1,2(✉)</sup>, Jasna Cvetkovic-Lay<sup>2</sup>, Ana Sovic Krzic<sup>1,2</sup>,  
and Damir Sersic<sup>1</sup>

<sup>1</sup> Faculty of Electrical Engineering and Computing,  
University of Zagreb, Unska 3, 10000 Zagreb, Croatia  
{tomislav.jagust, ana.sovic.krzic, damir.sersic}@fer.hr

<sup>2</sup> Center for Gifted Child Development “Bistric”,  
Nikole Pavića 5, 10000 Zagreb, Croatia  
nadarenost@gmail.com

**Abstract.** This paper presents our experiences from workshops with gifted primary school students (grades 2–4) especially in programming with robotics sets (Lego Mindstorms EV3) and other technology. As a part of extracurricular enriched program at the Center for Gifted Child Development in Zagreb, Croatia, we organized a number of robotics and ICT workshops. Main goal of these workshops was to introduce gifted primary school students to computer programming and robotics, teach them some basic programming and mechanics skills, and develop their algorithmic thinking, problem solving and creativity. However, through lessons, students showed unexpected productive giftedness in specific domains of creativity, with children experimenting with different ideas and designs, discussing inventions or alternative approaches to the given problems, or expressing their visual arts or music talents through robots and programming tasks.

**Keywords:** Gifted education · Productive giftedness · Creativity · Digital natives · Skills in ICT · Robotics · Lego mindstorms

## 1 Introduction

Today’s generations of children and youth were born and grew up surrounded with the technology. Motivated by flexibility and resourcefulness of ICT around them, they show very high levels of creativity and interactivity. The common term used to describe children and youth who have interacted with digital technology from an early age is “Digital Natives” [1], “n-gen” [2], or “Millennial Generation” [3]. Some authors ([1, 4]) suggest that for these students, a different learning environment and procedures should be created, to accommodate to their specific way of living (e.g. multitasking, nonlinear information processing, shorter attention span, communication over social media and text messages, etc.).

Advanced technology is very strong media and educational tools, capable to explain and visualize complicated abstract concepts, help students in developing their competencies, and at the same time keeping every student on the challenging upper



limit of her own capabilities. It is also the useful tool to make giftedness productive in domain specific skills and abilities connected with ICT and robotics.

This makes computers and technology particularly suitable for gifted children, since their capabilities, specific interests and motivation are on a higher level than their peers. Well-tailored computer applications enable a free, playful and fearless approach to learning and thinking, often in a very creative and imaginative environment. This too, makes them a great educational tool for gifted children who often stand out of their surroundings because of their creativity and personal traits.

It is a common prejudice that gifted children do not need additional support for developing domain specific abilities and creativity. For that reason, many gifted children are in danger of becoming the so-called “unguided missiles” [5]. Key focuses of ICT and robotics workshops at the Center for Gifted Child Development is early identification and educational support of young creative talents but also professional mentoring aiming to make creativity and giftedness productive. The role of mentoring involves coaching, expertise, guidance, instruction, knowledge, or skill-building in domain-specific area of creativity in ICT.

Working in small groups, we strive to utilize creative skills of our young talented digital natives with the educational objectives in the fields of robotics and ICT, at the same time directing their development on legal and ethical path, and for the common good. Through the lessons, LEGO Mindstorms EV3 sets were used predominantly, but sometimes different sets or tools were utilized as well, e.g. LittleBits, LEGO WeDo, Arduino, Drones etc.

## 2 Theoretical Background/State of the Art

Creative thinking skills are one of the most important skills of the future, especially if they are combined with gifted and talented children [6]. Giftedness includes the above average abilities and task commitment as well as the evidence of creativity (products) [7]. Walberg and Paik find out that productive giftedness is not only potential but also achievement and accomplishment [8]. In robotics workshops, children are actively engaged, they collaborate and contribute on meaningful outcomes-products. And that is the best way of learning [9]. Taucei, Stoltz and Gabardo’s study focused on one gifted student and his interactive relationships with teachers in regular school as well as special education for gifted. He has difficulty in making friends with students, but he expresses himself mathematically in a creative manner [10]. In an open-ended project, Samuel combined usage of robotic kits (LEGO Mindstorms NXT) with mathematical simulation environment GeoGebra. Presented method noted especially good results with autistic students [11]. Lykke, Coto, Mora, Vandel and Jantzen compared three learning designs in programming teaching lessons: a problem based learning, a combination of problem-based learning with LEGO Mindstorms and a control group. The results show that robots can be an effective educational tool, but to achieve that goal, the project tasks and theoretical background must be well-defined and prepared [12]. In [13] robotics activities for K-12 students and teachers are presented. The activities are based on 5-Step Active Learning Cycle model: concepts, models, application, problems and design. In our previous papers [14, 15], we showed how to use LEGO Mindstorms robots to teach

children some of the basic science concepts, e.g. friction or “Why do we move faster on the ice than on the asphalt?”, ultrasonic sensors or “How do bats see?” etc.

### 3 Participants

The participants in this study are members of Croatian NGO - Center for Gifted Child Development – Bistrić (en. “Smartie”)<sup>1</sup>. The Center offers different extracurricular enriched programs organized in a form of workshops throughout a school year (Mathematics, Logic, Informatics, Programming and Robotics), as well as parent counseling and in-service teacher trainings for educational professionals. All of the participating students were, before attending our workshops, professionally identified as intellectually gifted, at least two mental years above average some of them even more. The authors organize and conduct robotics and programming workshops, usually monthly or bimonthly, with a group of 4–6 professionally identified gifted students, lasting 90 min each. Roughly 15 students, aged 8 to 10 years participated in our workshops in the last two years. Typically, students work in pairs, cooperating or collaborating on a solution to the given task.

### 4 Lessons Design

Most of the lessons are organized as a “learning by doing” or “learning through experiment” activities. At the beginning of the lesson, students receive a worksheet which serves as a guide throughout the workshop. Worksheet usually contains several examples and exercises and a short explanation of important concepts. Although it is not necessary for students to follow the worksheet “step by step”, they can always refer to it if they get stuck in any point during the lesson. For most of the lessons, a simple and easily expandable LEGO rover robot – Riley rover is used [16]. It can be assembled rather quickly, which is useful in more complicated lessons or lessons that have more emphasize on programming than on robot design.

Lessons are organized to gradually introduce more complex tasks, but also to show-case different science fields or problems that can be solved using robots or programming and foster creativity. The lessons are divided into several small tasks, giving students a sense of progress, and essentially gamifying the learning experience. For example, the introductory lesson, instead of explaining different “driving” modes of LEGO motors, ask students to compare them, write test results in a provided table, and draw conclusions on their own. All lessons are open-ended. Students need to use their creativity and communicate ideas with colleagues to find possible solutions. Gradually, different concepts from mathematics and other fields are introduced and incorporated in lessons.

In previous work we argued that first graders, and especially gifted students, can understand and even master some of the university-level programming concepts [17]. During the “programming workshops” students were introduced to loops, conditional

---

<sup>1</sup> <http://www.bistic.info/index.php/home>.



**Fig. 1.** Left: A young gifted student programs “Rock-Paper-Scissors” game for robot. Right: What is a robot? Example of gifted child’s drawing product.

statements, variables, etc. They created a number of simple games (like “Guess the number?” or “Rock-Paper-Scissors”) and played them against robots (Fig. 1 left).

Many lessons were focused on support of creativity process with productive outcomes, such as: students built a robot that created different drawings or played songs that they composed. Sometimes, before they built the robot, students were given a task to imagine and explain what “a robot” means to them (Fig. 1 right).

The third group of lessons was intended for practice of acquired specific knowledge. For that purpose, the FIRST LEGO League 2017<sup>2</sup> (FLL) field materials, models and mission problems were used. Applying acquired knowledge, students had to upgrade and program their robot to autonomously carry out different missions. Through these lessons children practiced teamwork, collaboration and cooperation skills, problem solving and trial and error approach. Although the FLL is intended for ages 9 to 16, even younger gifted children successfully solve some of the challenges.

## 5 Results and Student Reception Discussion

During the workshops, the following observations were made:

**Gifted students are creatively productive** - In a number of occasions, students’ solutions to the given tasks were different than the solutions we prepared. Their divergent and out-of-box creative thinking allowed them to imagine many possible (or impossible) ways to solve a common problem, e.g. build a robot that can move faster than its motors. The brainstorming sessions sometimes went in the unfeasible direction, at these occasions the pros and cons of the proposed solution were discussed and the workshop was reverted on the intended path. Another way of expressing creative productivity was through various creative products (drawings, constructions, composing and recording music, etc.), resulting in different robot designs (Fig. 1). Many students expressed other talents: composing music for robots or simply decorating robot with different visual add-ons.

<sup>2</sup> <http://www.firstlegoleague.org/>.

**Gifted students are focused and task oriented “speed learners”** - As a part of our outreach activities through the ŠUZA Program<sup>3</sup> at the University of Zagreb Faculty of Electrical Engineering and Computing, we organized a number of similar workshops with “regular” students of all ages. Typically, gifted students are much faster than regular students in adoption and application of acquired specific knowledge and skills, and need less time to solve given problems. Also, they can relatively easily understand complicated mathematical and physical laws. For that reason, lessons for gifted should be more content-rich and accompanied with at least several mental challenges and various possible problem solutions. Same approach in the “regular” student environment could have a negative effect on the workshop, with students giving up from task, or constantly asking for guidance from teacher.

**Gifted students are more motivated and independent learners** - If they encounter difficulties during the task, gifted students tend not to give up easily, but instead try until they come out with an answer, or ask for help. Further, in some cases, students didn't want to “go home” until they solved all of the problems in a worksheet. On the other hand, if the task was not challenging enough, gifted children would give up and take another task.

**Gifted students like to collaborate, if they share the same specific interests and domain specific skills** - They usually like working in pairs with other students, by splitting the task in two or working together on every detail of the problem. Since usually both students have their own idea how to solve a given problem, they learn how to debate with arguments. In “regular” student population, an “alpha” pair member is elected easier. The best teamwork results with gifted students were achieved when teams were able to meaningfully split the task into two loosely connected parts.

## 6 Conclusion

In this paper we presented our experience in special educational support to domain specific abilities – talents of creatively productive gifted students through ICT and robotic workshops. We have selected tools which inspire our creatively productive gifted to create independent projects and other various creative outcomes, which would otherwise be impossible to achieve without the use of ICT skills. Robots, especially LEGO Mindstorms, have proven to be an excellent learning medium, due to their attractiveness, ease-of-use, and almost unlimited creative building and combining possibilities. During the workshops, different observations and students' reactions were noticed, and discussed in more detail in this paper. Creatively productively gifted students are fast and independent learners, think divergently and are motivated solve demanding tasks. The main duty of a mentor is to enhance the independence and the creativity of a gifted child, in relation to child's skills and capabilities. Tasks should be complex, challenging and open-ended, further boosting already high level of creativity and resulting with various outcomes (products). Teamwork can additionally improve the quality of learning.

---

<sup>3</sup> <http://suza.fer.hr/>.

## References

1. Prensky, M.: Digital natives, digital immigrants. *On the Horiz.* **9**(5), 1–6 (2001). MCB University Press
2. Downes, S.: E-learning 2.0. *eLearn Mag.* **10** (2005)
3. Schorn, D.: The “Millennials” Are Coming. CBS News (2009)
4. Rosen, L.D.: *Rewired: Understanding the iGeneration and the Way They Learn.* St. Martin’s Griffin, New York (2010)
5. Cvetković-Lay, J., Sekulić – Majurec, A.: Darovito je, što ću s njim?, Priručnik za odgoj i obrazovanje darovite djece predškolske dobi, Alinea i Centar za poticanje darovitosti djeteta «Bistrić», Zagreb (1998)
6. Jackson, J.: Making learning ‘real’ for gifted and talented students with robotics. *Digital Learning and Teaching Victoria*
7. Renzulli, J.S., Callahan, C.M.: Product Assessment; Introduction and Rationale for Product Assessment. In: *Alternative Assessments With Gifted and Talented Students, The Critical Issues in Equity and Excellence in Gifted Education Series*, pp. 259–285. National Association for Gifted Children, Prufrock Press Inc, Texas (2008)
8. Walberg, H.J., Paik, S.J.: Making giftedness productive. In: *Conceptions of Giftedness*, pp. 395–410. Cambridge University Press (2005)
9. Papert, S.: What’s the big idea? toward a pedagogy of idea power. *IBM Syst. J.* **39**(3–4), 861–879 (2000)
10. dos Reis Taucei, J., Stoltz, T., Gabardo, C.V.: Creativity and education: interactive teaching practices with a gifted student. *Creative Educ.* **6**, 2263–2273 (2015)
11. Samuels, P.: Developing extended real and virtual robotics enhancement classes with years 10–13. In: *Robotics in Education: Research and Practices for Robotics in STEM Education*, pp. 69–81. Springer (2016)
12. Lykke, M., Coto, M., Mora, S., Vandell, N., Jantzen, C.: Motivating programming students by problem based learning and LEGO robots. In: *IEEE Global Engineering Education Conference (EDUCON)*, pp. 544–555 (2014)
13. Saygin, C., Yuen, T., Shipley, H., Wan, H.D., Akopian, D.: Design, Development, and implementation of educational robotics activities for K-12 students. In: *ASEE Annual Conference and Exposition, San Antonio, Texas*, pp. 1–17 (2012)
14. Savić, A., Jaguš, T., Seršić, D.: Using lego mindstorms robots in science education. In: *Proceedings of the International Science Education Conference, Singapore*, pp. 1656–1684 (2014)
15. Cvetković-Lay, J.: Extra –curricular enriched program for gifted students – individual projects in informatics and robotics, invited lecture, In: *Programme and Abstract Book of 14th International ECHA Conference, European Council for High Ability, University of Ljubljana, Faculty of Education, MIB, Ljubljana* (2014)
16. Kee, D.: *Classroom activities for the busy teacher: EV3, CreateSpace Independent Publishing Platform* (2013)
17. Savić, A., Jaguš, T., Seršić, D.: How to teach basic university-level programming concepts to first graders? In: *IEEE Integrated STEM Education Conference*, pp. 1–6, March 2014

# The Evaluation of Robotics Activities for Facilitating STEM Learning

Ronit Ben-Bassat Levy and Mordechai Ben-Ari<sup>(✉)</sup>

Department of Science Teaching, Weizmann Institute of Science,  
76100 Rehovot, Israel

{ronit.ben-bassat,moti.ben-ari}@weizmann.ac.il

<http://www.weizmann.ac.il/sci-tea/benari/>

**Abstract.** We used the theory of planned behavior to predict students' intentions to choose STEM (science, technology, engineering and mathematics) in the transition from middle school to high school after participating in robotics activities. We found that students' attitudes towards STEM were not as high as expected, although most of them expressed an intention to choose future study of STEM. Then we interviewed teachers on their attitudes on the effect of robotics activities on choosing to study STEM, and checked if the activities actually led to an increase in students choosing STEM. We found positive results for both questions.

**Keywords:** Robotics · Theory of planned behavior · STEM

## 1 Introduction

Many factors discourage students from studying science, technology, engineering, and mathematics (STEM), for example the perception of STEM as boring, only appropriate for nerds [4] and not for female students [8]. The Israeli Ministry of Education is attempting to increase the number of students studying STEM. A major program, called AMT, is aimed at strengthening the learning of STEM in middle-schools [11]. The subjects taught include mathematics, physics, computer science (CS), robotics and computer security.

Attitudes concerning STEM are formed as early as middle-school [5] so students' must be influenced early. One approach is to use kinesthetic activities, such Computer Science Unplugged. Another is to use programming environments designed for young students such as Scratch and Alice. A third approach is to engage students in robotics activities. This became feasible with the appearance of LEGO<sup>®</sup> Mindstorms. Recent advances in technology have made educational robotics even more accessible. We asked whether engaging in robotics transcends fun and leads to significant positive changes in their attitudes towards STEM, as well as in their intentions to study STEM. This question is important because of the time, money and effort required for robotics activities, an investment that can be justified only if the above goals are achieved.

## 2 Previous Work

Students participating in the FIRST LEGO<sup>®</sup> League (FLL) competitions maintained high attitudes and achieved meaningful learning of STEM [7]. On the other hand, the learning was sub-optimal because of the pressures of the competition [6]. It follows that it is preferable to engage in robotics activities within a non-competitive curricular environment.

Two research projects investigated students' motivation to learn CS in the context of robotics activities. Markham and King [9] found that the robotics group devoted more effort when compared with non-robotics classes, and claimed that this extra effort implied increased intrinsic motivation. McGill [10] studied changes in motivation through robotics activities in a preliminary CS course for non-majors. She found an improvement in students' attitudes towards programming, but little effect on other measures such as confidence.

Our research significantly extends previous work in several ways:

- We investigated attitudes towards STEM in general and not just towards CS or robotics.
- We went beyond measuring attitudes and looked into the intentions that are engendered by attitudes; this is important because it is intentions that directly affect future behavior.
- By carrying out the research in middle-schools, we checked the effect of robotics before students make firm decisions on their future studies.

A preliminary report on this research was published in [3]. There we described the construction and administration of the questionnaire, and our conjecture that the results would lead more students to study STEM. Here we report quantitative results showing that this in fact did take place. We also report results from our interviews with the teachers.

## 3 Theoretical Background

The research used the theory of planned behavior (TPB) [1]. This is both a theoretical research framework and a quantitative methodology. TPB models human behavior using attitudes, subjective norms and perceived behavioral control (PBC) to predict intentions to perform a behavior, in this case to study STEM. Our research group has used TPB before within the context of educational technology [2] and it proved effective in understanding the causal links from attitudes to intentions to behavior.

Here are short definitions of the elements that appear in TPB (full definitions can be found in [3]): Behavior is the observed human action that is a response to a given situation. Intention is an indication of a person's readiness to perform a given behavior. An attitude towards a behavior is the degree to which the performance of the behavior is positively or negatively valued. Evaluation of the behavior is assumed to have two components: (1) behavioral beliefs about the consequences of the behavior, and (2) the outcome evaluation of this behavior's

consequence to be positive or negative. Subjective norms about the behavior are a person's estimate of the social pressure to perform or not to perform the target behavior. Perceived behavioral control of the behavior is the extent to which a person feels that he or she can control the behavior.

TPB questionnaires are built after taking field notes and interviews. Since they are based on issues that arise in practice, the results from TPB tend to be more valid than questionnaires based solely upon the researchers' experience.

## 4 Methodology

### 4.1 Research Questions

1. (a) To what extent does participation in robotics activities influence the attitudes of students towards STEM and their intentions concerning STEM studies in the future? (b) Do they really choose STEM in high school?
2. What are the teachers' attitudes on the role of robotics' activities towards the intentions of their students to choose STEM?

### 4.2 Context and Populations

We investigated two contexts of robotics activities in Israeli schools: The first population consisted of participants in the FIRST LEGO<sup>®</sup> League (FLL) competitions intended for grades 4–8. The characteristics of this population were that the activities were extracurricular and the participants self-selected.

The second population consisted of middle-school students in the AMT program. Unlike the first population, their activities were part of the school curriculum and the students were selected by teachers and principals. Therefore, these students were likely to display a more diverse set of attitudes and intentions.

The control group consisted of students in the MOFET program which is also aimed at excellent students of STEM, but they focus on physics and mathematics with no robotics activities.

For the second research question we interviewed ten teachers of robotics.

### 4.3 Research Instruments and Data Analysis

The first year of the research was devoted to field observation and interviews; these were used to construct a 44-question TPB questionnaire. We sent more than 700 questionnaires and received back 350. We terminated our analysis after 106 questionnaires at which point additional analysis did not change the results.

Interviews with the teachers were conducted throughout the entire project.



## 5 Results and Discussion

### 5.1 The First Research Question: (a) Changes in Attitudes

We found that most students were enthusiastic towards robotics at the beginning of the year when the subject was new. They carried out the assignments given by teachers and they collaborated on the construction of robots. The FLL students collaborated more than the AMT students because they had a concrete goal. The robots often malfunctioned which led to frustration. The interviews showed that the students felt good when they received respect and support from the teachers and the school staff, as well as from their parents. The interviews revealed a problem in scheduling: robotics classes are usually given in the afternoon after all the other students have gone home.

Several questions of the TBP questionnaire dealt with the students’ attitudes towards science. Their answers showed that the experience of trial and error made them feel like “real” scientists and that engaging with science in the future would be considered a success, both by themselves and by their teachers and parents. We noted that students’ did not mind failing in their robotics activities and that they readily accepted the challenge of correcting their errors.

We analyzed the data from the questionnaire according to the TPB methodology we developed [2]. We divided the values calculated for each TPB predictor into quartiles (Table 1). The analysis showed that: (1) Students are roughly uniformly distributed in the quartiles for attitudes, which is somewhat disappointing. (2) Most students fall into the two middle quartiles for subjective norms, which means that they can be influenced to choose STEM by the school and home environments. (3) The relatively high scores for perceived behavioral control mean that students feel that they can control their future choices to study STEM. (4) The scores for intentions are very high, indicating that they are likely to choose STEM.

**Table 1.** Results of the TPB questionnaire (n = 106)

	Attitudes	Subj. Norms	PBC	Intentions
First quartile	27	12	34	58
Second quartile	24	36	33	32
Third quartile	31	53	37	12
Fourth quartile	24	5	2	4
Total	106	106	106	106

While the attitudes were not as high as we expected, the results for the subjective norms are of particular importance, because they show that students can be motivated by the respect and support they receive from their teachers and parents. The PBC results show that the students feel that they can control their choice of STEM at high school.

The high scores for intentions were consistent with the answers to a question about the student's intentions to study STEM that was asked at the beginning of the questionnaire before the TPB items.

## 5.2 The First Research Question: (b) Do Students Choose STEM?

From one of the cities in which the activities took place we obtained the numbers of students in the AMT and MOFET programs. In 2012, before robotics activities were introduced, only 10% of the students chose STEM subjects. In 2015, after robotics activities were introduced, this percentage increased to 13% and in 2016 increased again to 18%, almost double the percentage from four years before.

Table 2 gives the number of students in that city's AMT (the ones who studied robotics) and the number of students in MOFET (who did not study robotics). From 2015 to 2017, the number of students in AMT increased, exceeding the number of MOFET students.

**Table 2.** Number of students in 2015–17

Year	2015	2016	2017
MOFET	300	283	272
AMT	316	372	446

The AMT program has three topics: Scratch, robotics and computer security. Before AMT, Scratch was taught middle-schools, yet the numbers of students choosing STEM did not increase. We interpret the data in Table 2 as justifying the use of robotics activities, because robotics is a central topic of the AMT program, while robotics is not taught in the MOFET program. Since AMT students are *not* self-selected, this strengthens the claim that robotics was significant in their decision to study STEM.

## 5.3 The Second Research Question

Most of the teachers said that their students enjoyed engaging with robots to the extent that many did not want to go home at the end of the activities. We believe that this enjoyment positively influences motivation. The FLL teachers from outside the school had difficulties controlling their students, but this was improved by assigning teachers from the school to assist. The teachers said that students are more motivated and responsible than they are in regular classes. In their opinion, participation in robotics activities caused students to choose to study STEM, in general, and CS, in particular, in high school. Their explanation was that students became addicted to working with robots and enjoyed programming them. They emphasized that many were students who wouldn't otherwise have chosen STEM. High school teachers explicitly said that their students recognize CS topics from middle-school CS and robotics activities.

## 6 Conclusions

Our research provides evidence that supports the claim that robotics activities motivate students to further study of STEM. We believe that the advantage of using robots in classrooms is that the students experience kinesthetic activities: The robots give a concrete feedback in contrast to the virtual world of a software-only environment. Moreover since the students' must learn from mistakes that lead to incorrect concrete behavior of the robots, engaging with robots imitates game-like learning that attracts students and this in turn reduces the fear from STEM. Students obtain experience similar to that of scientists and this encourages them to study STEM. We found that students, as well as their parents and teachers, consider science professions as success in life, and we believe that robotics activities can mediate this success.

**Acknowledgments.** This research was supported by the Israel Science Foundation (grant 912/13).

## References

1. Ajzen, I.: Perceived behavioral control, self-efficacy, locus of control, and the theory of planned behavior. *J. Appl. Soc. Psychol.* **32**(4), 665–683 (2002)
2. Ben-Bassat Levy, R., Ben-Ari, M.: Adapting and merging methodologies in doctoral research. *Comput. Sci. Educ.* **19**(2), 51–67 (2009)
3. Ben-Bassat Levy, R., Ben-Ari, M.: Robotics activities-is the investment worthwhile? In: 8th International Conference on Informatics in Schools, pp. 22–31 (2015)
4. Carter, L.: Why students with an apparent aptitude for computer science don't choose to major in computer science. *SIGCSE Bull.* **38**(1), 27–31 (2006)
5. Gibbons, S.J., Hirsh, L.S., Kimmel, H., Rockland, R., Bloom, J.: Middle school students' attitudes to knowledge about engineering. In: International Conference on Engineering Education, pp. 1–6 (2004)
6. Kaloti-Hallak, F., Armoni, M., Ben-Ari, M.: The effectiveness of robotics competitions on students' learning of computer science. *Olympiads Inf.* **9**, 89–112 (2015)
7. Kaloti-Hallak, F., Armoni, M., Ben-Ari, M.: Students' attitudes and motivation during robotics activities. In: Workshop in Primary and Secondary Computing Education, pp. 102–110 (2015)
8. Margolis, J., Fisher, A.: *Unlocking the Clubhouse: Women in Computing*. MIT Press, Cambridge (2003)
9. Markham, S., King, K.: Experiences, outcomes, and attitudinal influences. In: 15th Annual Conference on Innovation and Technology in Computer Science Education, pp. 204–208 (2010)
10. McGill, M.: Learning to program with personal robots: influences on student motivation. *ACM Trans. Comput. Educ.* **12**(1), 1–32 (2012). Article 4
11. Zur Barguri, I.: A new curriculum for junior-high in computer science. In: 17th Annual Conference on Innovation and Technology in Computer Science Education, pp. 204–208 (2012)

# **Project-Based Learning Approaches**

# MuseumsBot - An Interdisciplinary Scenario in Robotics Education

Tanja Heuer<sup>(✉)</sup>, Ina Schiering, and Reinhard Gerndt

Ostfalia University of Applied Sciences, Wolfenbüttel, Germany  
{ta.heuer,i.schiering,r.gerndt}@ostfalia.de

**Abstract.** This paper introduces a project based interdisciplinary course in computer science. The students of the course Robotics should create an autonomous robot for a museum. Therefore various soft and technical skills were measured, the goals were determined and the results of the course are compared to earlier courses. The aim of the project based approach in combination with a real life scenario was to deepen a holistic view of different topics in robotics and to strengthen the motivation of the students.

**Keywords:** Autonomous robot · Robotics · Student course · Museum robot · MuseumsBot · Education · Computer science

## 1 Introduction

Robots have already entered our world in every part of daily life. From huge robots in the industry working together with people to small robots helping at home, for example cleaning the floor. In the health care area robots can help to take care of older or ill people. Robotics taught in education is a highly relevant topic, as can be inferred from various teaching approaches. The traditional approach is lecture based [11, 13, 30]. A more common approach is project based, which is about working with virtual and real robots [15, 25, 26, 28, 31]. According to Beer et al. [10] autonomous robotics is an important topic to teach.

The course investigated here is the second part of a robotics course in a Computer Science Master program. In the first part, the mathematical and physical background is presented whereas in the second part investigated here, practical aspects of robotics are the focus. To teach these practical aspects two different approaches have been used: In a model based approach the single areas were presented in lectures with accompanying practical exercises with ROS [24, 27] using e.g. gazebo, a Baxter [6] and a self-built chefBot (see Fig. 1). To address the lack of motivation of the students and to foster a holistic view on robotics, as a second approach project based learning was used. In cooperation with a nature experience center the students should create a MuseumsBot. This MuseumsBot should be autonomous and shall guide the visitors through the museum while talking and interacting with them. Examples here are e.g. Thrun et al. [29], Urbano [9] and Rato and Showbot [21].

There are various existing project based courses in computer science but most of them focus on the programming part of robotics. Those who also care about design for example build small robots with Lego Mindstorms [20, 32]. This time the students get the opportunity to handle a problem where the museum is really interested in the results to use them afterwards. Therefore a robot should be created right from the start. This course should show if the students work more independently and the support of a holistic view of robotics is given more than in a usual course.

The following chapter introduces the course, the museum and the competences to be examined. Afterwards the results are evaluated. At the end the two different approaches will be compared and discussed.

## 2 Related Work

There are already different kinds of robots for education. A popular platform is Lego Mindstorms NXT [4]. Humanoid robots for educational purpose are e.g. the Aldebaran NAO robots [7] or Robovie [8]. A more complex, industrial robot is the Baxter [6]. With those ready to use robot kits different competences of programming in robotics as e.g. path planning, obstacle avoidance can be taught.

In the scenario of a museum, the number of robots is increasing. Because of that Lupetti et al. [22] evaluated different kinds of museum robots. One kind are the telepresence robots that allow a guided walk without being in the museum. Visitors can use the robot to move around the museum while being at home. Some existing prototypes for that are already in use [2, 3, 17]. Another kind of robots are guiding robots. In the scenario of Thrun et al. [29] the robot is used to interact with visitors within the museum. He behaves autonomous and acts as an educational help. Other examples for those guiding bots are Urbano [9], CiceRobot [23], Rato and Showbot [21] and a prototype presented by Schraft et al. [14].

Most of the courses building a robot from scratch use Lego Mindstorms [10, 20, 31, 32]. The students have to build a robot based on their own ideas or for a given scenario and program the robot fitting to the scenario. In our case the students should create a robot suitable for the project in the museum.

## 3 Creating a Robot from Scratch

To address the creativity and motivation of our students we tried a **project based learning approach (PBL)** [19]. According to Krajcik et al. [19] students are more engaged if they handle a real problem. Typically between 10 and 20 students attend the lecture *Robotics* in Computer Science over a period of two semesters.

The first course started with a **lecture based** part included the mathematical aspects needed for robotics [11, 30]. With physical robots like the Baxter [6] and ROS [24, 27] the students got the chance to deepen the understanding

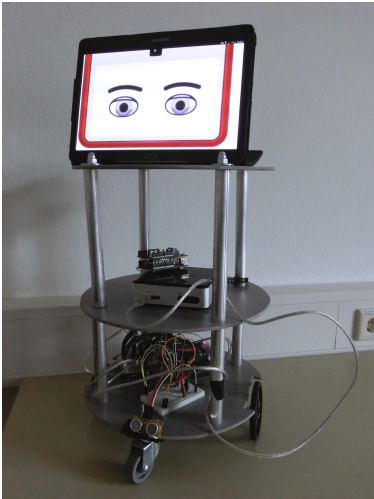


Fig. 1. ChefBot

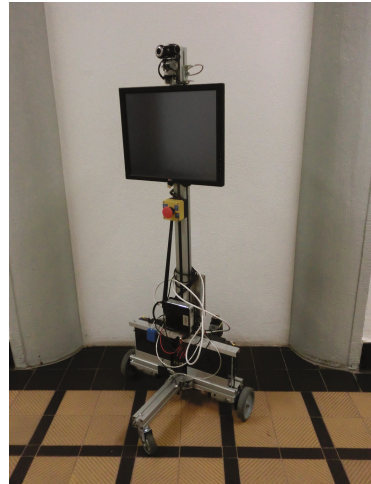


Fig. 2. Telepresence robot

because theory by itself didn't sufficiently motivate the students. In the second part of the course we followed a **model based approach** [12]. Here, the students could rework theoretical aspects and learned to work with an industrial robot. To foster a holistic view on robotics a project based approach [19] was tried in the second iteration. There students should create an autonomous robot called MuseumsBot from scratch for a museum. The students were allowed to work together but should write a research paper at the end everyone on his own. With that project description the exercise was held open and with a real life scenario the students were supposed to be more motivated by working on a useful project. At the end of that course various aspects were investigated. The topics are divided into soft skills and a technical part. The selected soft skills are partially chosen from Hobsen [18] and the technical skills are partially selected from Tosello et al. [31]. They were adapted to our necessities and other skills were associated. They are listed and described in Sects. 3.2 and 3.3.

### 3.1 Museum Scenario

The museum is a nature experience center. It deals with environmental and nature conservation and exists since about one year now. During that time it was perceived that only adults are interested in the topic of the center. The form of presentation mostly addresses older people. With the idea of an autonomous educating robot (MuseumsBot) the museum should also be made more attractive for the younger generation. The topic was introduced like that and if the students wanted to have more information, they were allowed to visit the museum to ask for more information.



**Fig. 3.** Show room of the nature experience center

The robot should act as a guide. The museum has a small entry area and a single show room where the tour is done. The MuseumsBot should interact with the visitors in different ways. There are three possibilities to do a round walk depending on the ages. For smaller guests the robot talks in an easy language and maybe shows pictures on the screen in accordance to the topics. Teenagers can answer questions and may follow a text-based interaction. For adults the guide is an extension to the information material in the museum. The appearance could either fit with the general appearance of the center or could be completely abstract contrast to it. The museum shows natural history exhibition in a green-white ambient (Fig. 3). The museum laid down the requirements but had no opinion about the looking and the realization of the Human-Robot-Interaction. At the end of the course they should get an overview of different possibilities to have a more precise idea of the robot they want to have. But following only the goals of the project based approach are mentioned and evaluated and not if the specific goals of the museum were reached.

### 3.2 Soft Skills

The soft skills category covers various important skills listed on Table 1. For our research we focused on five different skills. The students need the ability to talk to each other and with non-computer scientists to get along with the project ('social interaction'). In addition to that they have to bring in a lot of creativity and self-engagement. Because it is an open held task, they have to figure out important things and what need to be done at first. In a second step the students have to work on further technical aspects like building the robot, designing the robot and functionality of the robot. Therefore it can be necessary to learn mathematical or technical stuff. Compared to the traditional course, the students only need to occupy with the theoretical part they want to, instead of everything. One of the main aspects in the project based approach is the motivation of the students. They have the chance to create a robot prototype



**Table 1.** List of the examined soft skills

Soft skills	Description	Source
Creativity	<b>Goal:</b> Develop new ideas <b>Measurement:</b> Own ideas for the robot independent of the skill-topic	[18]
Social interaction	<b>Goal:</b> Students interact with others <b>Measurement:</b> Communication with non computer scientists	[18]
Self learning	<b>Goal:</b> Willingness to learn new things they are interested in <b>Measurement:</b> Using different tools for the work	[18]
Self motivation	<b>Goal:</b> Students are able to discipline themselves <b>Measurement:</b> Appearance of result, number of processed tasks	
Organization	<b>Goal:</b> Students are able to structure their own work <b>Measurement:</b> Structured operation, target-oriented mode	

for a real life scenario. This hopefully encourages the students gladly to work on the robot for the museum. At best the course improves their social skills during the semester as a result of the project.

### 3.3 Technical Skills

Next to the soft skills the technical skills are an important part. The evaluated tasks for that are listed in Table 2. The most important part at the beginning is the requirements analysis. This analysis has the aim to determine the requirements of the users for the project. Those requirements need to be structured and checked to get a fitting and accurate robot for the project. The already existing telepresence robot (Fig. 2) or the chefBot (Fig. 1) offer a first suggestion for a robot. Based on those, the students have a first idea and can expand and adapt it fitting to their conception. To realize a human-robot-interaction (HRI) also sensors are needed. The students are free in the choice of using sensors. There is for example the possibility of a camera for the vision, touch sensors for interaction, buttons for communication, distance sensors for obstacle avoidance. The students shall describe the sensors they use and why they use those. Furthermore, there are no a priori restrictions concerning of the appearance of the robot. The robot could have an abstract, modern, nature-inspired or technological appearance. At the end there is the question about usability of the designed robot that needs to be considered in sight of detailed use cases and user stories of the robot.

**Table 2.** List of the examined technical skills

Technical skills	Description	Source
Requirements analysis	<b>Goal:</b> Detection of important requirements <b>Measurement:</b> Ask for requirements, check if requirements are executable	
Hardware architecture	<b>Goal:</b> Prototyping a guiding robot <b>Measurement:</b> Think about the construction of a robot	
Sensors and actors	<b>Goal:</b> Description of Sensors and Functionality <b>Measurement:</b> Find useful sensors and actors to support, Sensors needs to fit the requirements	[31]
Motion and path planning	<b>Goal:</b> Understanding of whole movement background <b>Measurement:</b> General movement of the robot, Path Planning through the museum	[31]
Appearance	<b>Goal:</b> A considerable prototype <b>Measurement:</b> Development of a surface for the robot	
Usability	Are the requirements fulfilled? <b>Goal:</b> Critical view of the own work and ideas <b>Measurement:</b> Put the question if the created prototype is useful	

### 3.4 Objectives

With the project based approach we want the students to work with fun in the lecture. They get the opportunity to work on a serious MuseumsBot project which should be used for real. In addition this offers a practical part which is useful to deepen the understanding of the topic of robotics. Depending on the interests of the students, each can lay the focus on the relevant subjects. To get into the tools they got small exercises. The course should support independent thinking for students. The measurements for the competences are listed in the Tables 1 and 2.

Table 3 lists the goals for the two learning approaches. While the focus for model based learning is more theoretical, the project based approach sets the focus on more practical tasks and on the social competences. The project should show, if the willingness to learn is higher with such a free held project than with small exercises. The presentation at the end of the semester will be evaluated with different criteria concerning soft and technical skills.

**Table 3.** Learning goals of the different approaches

	Model based approach	Project based approach
Creativity	o	+
Social interaction	o	+
Self learning	+	+
Self motivation	o	+
Organization	o	+
Requirements analysis	o	+
Hardware architecture	+	o
Sensors and actors	o	+
Motion & path planning	+	+
Appearance	o	o
Usability	o	+

## 4 Evaluation of MuseumsBot Project

Here the two sections **Soft Skills** and **Technical Skills** are evaluated based on the seminar papers of the students.

### 4.1 Soft Skills

In the following paragraph the results of the measurements are listed and some of the students ideas are presented.

**Creativity** is measured with own ideas for the MuseumsBot. Some students for example created a new robot model, some had interesting ideas for the appearance and the sensors. Some of the ideas are presented in the evaluation of the technical skills. 20% of the students didn't get a point for creativity because they only copied exactly the existing robot and had no own concepts. **Social Interaction** goes along with the requirements analysis. Those students who talked to workers of the museum did a requirements analysis. They asked among others about user group, functionality and appearance. According to that they extracted different requirements. On average there were listed about eight specifications for the robot. **Self Learning** means the willingness to learn. Different tools were presented for different tasks and they could decide on their own what they want to use. Every student worked at least with one of these tools. Summarized the used tools were Blender, Meshlab, Gazebo, LibreCAD and rviz from ROS. For graphics like Fig. 4 the students used e.g. LibreCAD and Blender to create the robots and Meshlab. Meshlab could be used to set a case around the robot for the appearance. **Self Motivation** is a combination of the other social parts, how the result looks at the end and how many of the aspects were processed. This part was graded relatively bad. Half of the students only handled half of the tasks categorized in this paper. **Organization** means the structure

of the students work. Only half of the students had a structure in their work and the paper. Some didn't had a requirements analysis. Others had a not conclusively work because they described different topics not compatible to each other.

## 4.2 Technical Skills

**Requirements Analysis** is one of the most important things to do, before creating and building a robot. Nearly 70% of the students made an analysis of the requirements before starting with further work. On average the students elaborated 8 requirements. **Hardware Architecture** was handled by 85% of the students. Some of them only adopted one of the presented models. Others created new robots and built a prototype with Blender. Figure 4 shows two variants of prototyped robots. The left robot should be about 160 cm tall, has a touch screen on the top for adults and one on the back side of the robot for children. The holder system can be used e.g. for advertisement. The right robot is smaller than the first one, the head should consist of a touch display. **Sensors and Actors** is the subject everybody worked on. The students listed their chosen sensors and actors with an explanation why they used those things. A wide variety of sensors were introduced. The used sensors for the environment for example are a laser range scanner, a lidar sensor, a Kinect camera and a ultrasonic sensor. For the interaction with a person, students chose a touchscreen, loudspeakers and the Kinect camera. **Motion and Path Planning** is divided into two parts. All robots were created with wheels to move around. Only one student worked on the velocity of the robot and how fast or slow he has to move or what happens if there is an obstacle. The path planning was a wide spread topic. Like mentioned in the topic before, a lot of students listed sensors for the environmental recognition. Some referred to known search algorithms like A\* or lowest cost search. Other ideas dealt with lighthouse navigation [5] or iBeacons [1]. **Appearance** is an

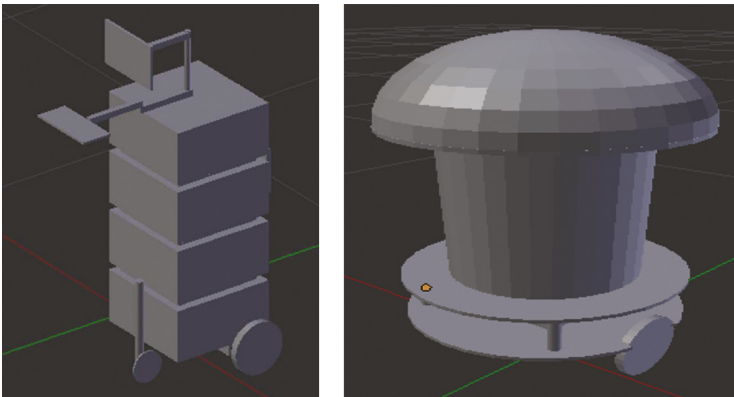


Fig. 4. Two variants of a hardware architecture



**Fig. 5.** Two variants of a hardware architecture

issue one third of the students worked on. Figure 5 shows two possibilities. The left one is painted in the colors of the museum, the right one is plastered with a thematic photo of the museum. Other concepts were a wood paneling or a small mushroom (see Fig. 4 (right)). **Usability** is the least edited area of the project. Only 20% wrote anything to that topic. A lot of students build a robot with the height of about 1 m. That is a possible height for younger guests but older visitors cannot use such a small guide.

## 5 Comparison of Approaches and Lessons Learned

Geronimo et al. [16] listed the advantages of PBL as follows:

- It allows the students to gain real experience in a topic.
- It brings together theoretical and practical concepts with a single goal, which enhances students' motivation.
- It allows the students to discover these new concepts for themselves if the course is well designed.

In our PBL approach the three advantages apply to the course. The Fig. 6 shows the soft skills and how many students reached each of them. Compared to the defined goals in Table 3 the parts organization and self motivation didn't work fine. It was proved to be difficult for the students to organize all the different subjects for the project because they didn't know where to start, what were the most important parts and what was not so important to handle. Because of that, huge amount of different topics the motivation of the students decreased during the semester. The goals of the technical skills 2 also fit only partly to the results showed in the Fig. 7. Concerning the technical aspect the analysis and hardware part worked fine. Positively the students worked also on the hardware architecture. But since usability and appearance are not the main subject in

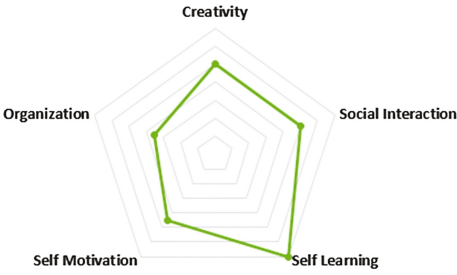


Fig. 6. Results of soft skills

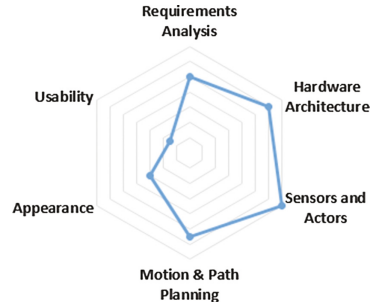


Fig. 7. Results of technical skills

computer science, almost none dealt with that. The approach was an experiment to try if the students are also interested in these topics.

In comparison to the model based approach, the project based approach fulfilled more criteria than the other one. Table 3 lists the goals for each variant. To work on small exercises the students don't need to be creative. They fulfill the given tasks based on the theoretical knowledge from the lecture. In the project based phase the students were able to show a creative part in their projects. The social interaction is only in the second approach an important part. The communication between students and users of the museum worked fine. Self learning must exist in both approaches to reach a satisfactory solution. Self motivation is part of the project based approach. The skill organization is not interesting for model based teaching. The lecture with exercises is organized by the professor and the students don't need to take care of that in contrast to project based learning. The quality of both approaches is good, but measured with different criteria. The first approach is measured solely by the quality of the results, whereas the second approach measures not wrong or right but the way to solve a task.

The technical part shows differentiated handled tasks. While model based learning focused on deepening of the learned subjects, during the project based approach the students were free to choose the subjects to work on. The earlier lecture addressed the behavior like e.g. moving, obstacle avoidance and path planning. Some of those topics could partly be simulated with small exercises, others were only learned theoretical. The project based approach should give a further insight into robotics. The evaluation shows that there were to many topics to work on. Students picked out about 3 tasks and neglected the rest. In this case the topics hardware architecture and sensors worked fine. As well the subject requirements was handled in a satisfactory way. The tasks appearance and usability not belonging to computer science were excluded from most of the students. Motion & path planning was only taken into account in a few sentences. This topic has different issues like e.g. moving, obstacle avoidance and path planning and most of the students only mentioned what sensors may be used as an aid for those various tasks but not deepened those tasks.

The students worked more on already known topics. Some aspects like usability or the appearance were almost completely neglected. The students were overexerted, so all components were only partly processed. The question in this case is, if the students didn't work enough for a successful work or if they didn't understand the parts they didn't work on.

To avoid that, for further courses there are various ways to change the task description. One possibility is to concentrate on specific tasks. The whole work can be split up into three or more parts for example and the students are able to decide on what they want to work on. A second approach can be a combination of virtual and real autonomous robots. Such a combination was already examined by Tosello et al. [31]. But everyone has to decide on his own, if it is a better way to teach in a simulation or a real context [32] or to find a combination. Another option is an interdisciplinary project with students from a social sciences. They can work in interdisciplinary teams to exchange specific knowledge and split the work. So all in all the process during the semester was interesting, but the course should be more focused on the realization competence.

## 6 Conclusion

Our project based learning approach is a good alternative to the previous way of lecturing robotics. Viewed as a whole, the approach worked fine for interdisciplinary competences but the technical expertise misses out. Referred to the Figs. 6 and 7 compared with the intended goals of Table 3 seven from nine goals were reached. Independent from the goals, the topic hardware architecture was handled by almost all students. However, in a next iteration the task description has to be more precisely, such that the students get the possibility to focus on specific important tasks and maybe work in small groups with two people.

## References

1. Apple Inc. <https://support.apple.com/en-us/HT202880>
2. CSIRO. <https://www.csiro.au/en/Research/D61/Areas/Robotics-and-autonomous-systems/Telepresence/Visiting-the-National-Museum?ref=/CSIRO/Website/Research/Technology/Robots/Museum-robots>
3. Droïds company. <https://www.csiro.au/en/Research/D61/Areas/Robotics-and-autonomous-systems/Telepresence/Visiting-the-National-Museum?ref=/CSIRO/Website/Research/Technology/Robots/Museum-robots>
4. Lego mindstorms. <https://www.lego.com/en-us/mindstorms/>
5. Lighthouse Navigation Co., Ltd. <http://www.lighthousenavigation.com/>
6. Softbank robotics. <http://www.rethinkrobotics.com/de/baxter/>
7. Softbank robotics. <https://www.ald.softbankrobotics.com/en/cool-robots/nao>
8. Vstone Co., Ltd. <https://www.vstone.co.jp/english/products/robovie.x/>
9. Álvarez, M., Galán, R., Matía, F., Rodríguez-Losada, D., Jiménez, A.: An emotional model for a guide robot. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **40**(5), 982 (2010)

10. Beer, R.D., Chiel, H.J., Drushel, R.F.: Using autonomous robotics to teach science and engineering. *Commun. ACM* **42**(6), 85–92 (1999)
11. Bekey, G.A.: *Autonomous Robots: From Biological Inspiration to Implementation and Control*. MIT Press, Cambridge (2005)
12. Blumschein, P., Hung, W., Jonassen, D., Strobel, J.: Model-based approaches to learning. In: *Using Systems Models and Simulations to Improve Understanding and Problem Solving in Complex Domains*. Sense Publishers, Rotterdam (2009)
13. Dautenhahn, K., Billard, A.: Bringing up robots or the psychology of socially intelligent robots: from theory to implementation. In: *Proceedings of the Third Annual Conference on Autonomous Agents*, pp. 366–367. ACM (1999)
14. Dieter Schraft, R., Graf, B., Traub, A., John, D.: A mobile robot platform for assistance and entertainment. *Ind. Robot Int. J.* **28**(1), 29–35 (2001)
15. Gerecke, U., Wagner, B.: The challenges and benefits of using robots in higher education. *Intell. Autom. Soft Comput.* **13**(1), 29–43 (2007)
16. Gerónimo, D., Serrat, J., López, A.M., Baldrich, R.: Traffic sign recognition for computer vision project-based learning. *IEEE Trans. Educ.* **56**(3), 364–371 (2013)
17. Giuliano, L., Ng, M.E.K., Lupetti, M.L., Germak, C.: Virgil, robot for museum experience: study on the opportunity given by robot capability to integrate the actual museum visit. In: *2015 7th International Conference on Intelligent Technologies for Interactive Entertainment (INTETAIN)*, pp. 222–223. IEEE (2015)
18. Hobson, R.S.: The changing face of classroom instructional methods: service learning and design in a robotics course. In: *30th Annual Frontiers in Education Conference, FIE 2000*, vol. 2, pp. 3–20. IEEE (2000)
19. Krajcik, J.S., Blumenfeld, P.C.: *Project-based learning*. na (2006)
20. Liu, E.Z.F., Lin, C.H., Chang, C.S.: Student satisfaction and self-efficacy in a cooperative robotics course. *Soc. Behav. Pers. Int. J.* **38**(8), 1135–1146 (2010)
21. López, J., Pérez, D., Santos, M., Cacho, M.: Guidebot. A tour guide system based on mobile robots. *Int. J. Adv. Robot. Syst.* **10**, 381 (2013)
22. Lupetti, M.L., Germak, C., Giuliano, L.: Robots and cultural heritage: new museum experiences. In: *Proceedings of the Conference on Electronic Visualisation and the Arts*, pp. 322–329. British Computer Society (2015)
23. Macaluso, I., Ardizzone, E., Chella, A., Cossentino, M., Gentile, A., Gradino, R., Infantino, I., Liotta, M., Rizzo, R., Scardino, G.: Experiences with cicerobot, a museum guide cognitive robot. In: *Congress of the Italian Association for Artificial Intelligence*, pp. 474–482. Springer, Heidelberg (2005)
24. Martínez, A., Fernández, E.: *Learning ROS for robotics programming*. Packt Publishing Ltd. (2013)
25. Nourbakhsh, I.R., Crowley, K., Bhawe, A., Hamner, E., Hsiu, T., Perez-Bergquist, A., Richards, S., Wilkinson, K.: The robotic autonomy mobile robotics course: robot design, curriculum design and educational assessment. *Auton. Robots* **18**(1), 103–127 (2005)
26. Polishuk, A., Verner, I., Klein, Y., Inbar, E., Mir, R., Wertheim, I.: The challenge of robotics education in science museums. In: *The 4th Knowledge Cities World Summit*, p. 319 (2011)
27. Quigley, M., Gerkey, B., Smart, W.D.: *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. O’Reilly Media, Inc., Sebastopol (2015)
28. Stein, C.: *Botball: Autonomous students engineering autonomous robots*, p. 7, 1 (2002)



29. Thrun, S., Bennewitz, M., Burgard, W., Cremers, A.B., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J., et al.: Minerva: a second-generation museum tour-guide robot. In: Proceedings of the 1999 IEEE International Conference on Robotics and Automation, vol. 3. IEEE (1999)
30. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
31. Tosello, E., Michieletto, S., Pagello, E.: Training master students to program both virtual and real autonomous robots in a teaching laboratory. In: 2016 IEEE Global Engineering Education Conference (EDUCON), pp. 621–630. IEEE (2016)
32. Wu, C.C., Tseng, I.C., Huang, S.L.: Visualization of program behaviors: physical robots versus robot simulators. In: International Conference on Informatics in Secondary Schools-Evolution and Perspectives, pp. 53–62. Springer, Heidelberg (2008)

# Marine Robotics: An Effective Interdisciplinary Approach to Promote STEM Education

Saeedeh Ziaeefard and Nina Mahmoudian<sup>(✉)</sup>

Mechanical Engineering Department, Michigan Technological University,  
1400 Townsend Dr., Houghton, MI 49931, USA  
{sziaeefa,ninam}@mtu.edu  
<http://me.sites.mtu.edu/mahmoudian/>

**Abstract.** GUPPIE, a Glider for Underwater Problem-solving and Promotion of Interest in Engineering was developed in Nonlinear and Autonomous System Laboratory at Michigan Technological University to be used as an educational tool to broaden the impact of Science, Technology, Engineering, and Mathematics (STEM) learning. The GUPPIE educational program utilizes high-interest themes, meaningful contexts, and hands-on activities to engage students as early as 4th grade and sustain their interest and learning to and through college. The program has engaged over 2000 students since 2013. The interdisciplinary nature of GUPPIE and hands-on activities in diverse areas from hardware development, and programming to gathering and interpreting data will improve students' ability for critical, creative problem solving, and ultimately increase individual motivation for pursuing STEM academic and career pathways.

**Keywords:** STEM · Engineering education · Marine robotic · Hands-on Activity

## 1 Introduction

It is a general belief that robotics provides an exceptional source of excitement that can be used to motivate student learning [1]. Robotics have a potential impact [1,2] on students' STEM learning and personal development including cognitive, social skills, creative thinking, decision making, problem solving, communication, and team working skills, all of them being crucial skills necessary in the workplace of the 21st century [3]. Robotics has a hybrid interdisciplinary nature and engages students in solving problems in multiple disciplines, by integrating different fields such as electrical engineering, computer science, and mechanical engineering.

Robotics helps educators to ensure that the learning experience is relevant to students' lives and prior experience [4]. Biomedical, environmental, aeronautical, and computer engineering are four examples of engineering branches that are

blooming to educate 21st century generation of engineers to find solutions to new challenges requiring a diverse set of knowledge and skills [5].

There are few examples of marine robotic in STEM education as a means to explore the environment and teach students the underwater physics and dynamics. Current available underwater robots are SeaGlide [6] from MIT, LEGO Waterbotics [7], MarineTech Project [8] with Old Dominion University and Norfolk State University, and the nationwide project known as SeaPerch [9] initiated by Nelson with the Society of Naval Architects and Marine Engineers (SNAME) and supported by the Office of Naval Research (ONR). SeaPerch was developed to introduce marine robotic to STEM educators to teach fundamentals of underwater robotics to young students. SeaPerch, a Remotely Operated Underwater Vehicle (ROV), teaches students basic skills in ship and submarine design and introduces possible career options in naval related fields. The teach, build, become approach of SeaPerch has resulted in national models for: teacher training, hands-on curriculum that aligns with national science standards, broad resource dissemination, and challenge events that celebrate 5–12th grade student interest and ability in STEM.

Because SeaPerch has been so well received, the next step is to build a more sophisticated or advanced curriculum model that continues to engage and sustain student interest in STEM. Figure 1 illustrates how teachers can scaffold student learning from SeaPerch’ underwater robotic program to GUPPIE’s modular platform. Unlike SeaPerch, GUPPIE is an autonomous system and students learn how to control a robot without a human in the loop during the operation.

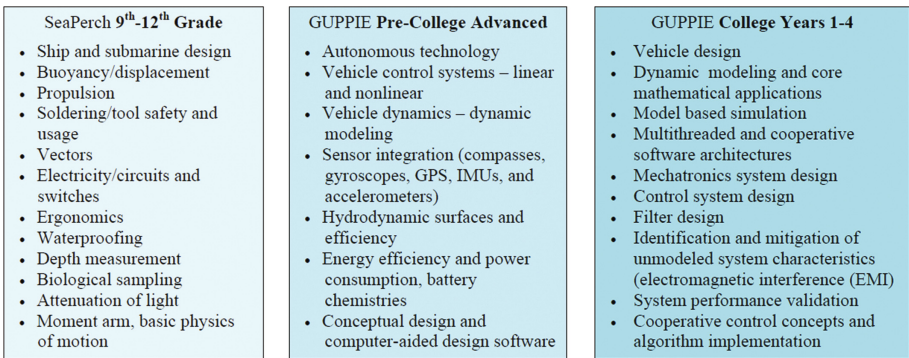


Fig. 1. Concept learning from SeaPerch to GUPPIE

GUPPIE [10], a Glider for Underwater Problem-solving and Promotion of Interest in Engineering is developed in Nonlinear and Autonomous System Laboratory at Michigan Technological University. A course has been developed to teach fundamentals of autonomous robots in underwater environment to STEM students with fun hands-on activities. Figure 2 illustrates a scholar student playing with GUPPIE in the swimming pool.

One of the goals of GUPPIE curricula is to advance the current work of SeaPerch by providing a modular educational platform that scaffolds learning from early ages in middle school to pre-college and undergraduate level. In general, educational robots should be designed to consider students age, their requirements, and be adaptable in real life scenarios [11].



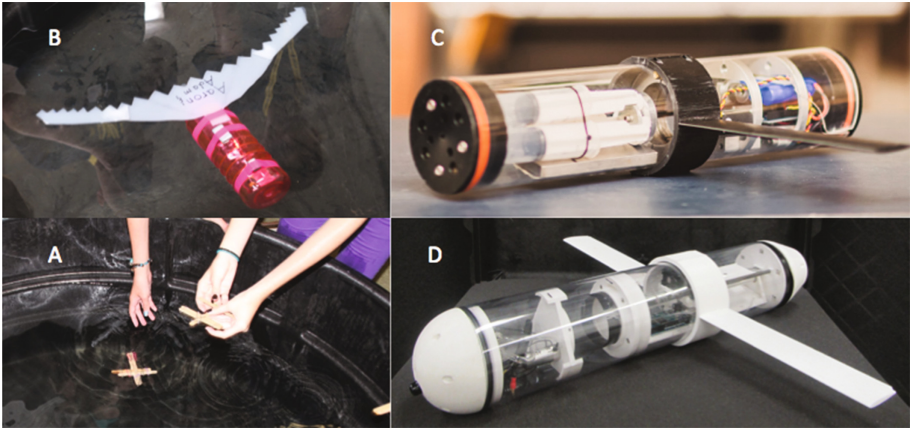
**Fig. 2.** Summer Youth Program student observes GUPPIE in the swimming pool

The GUPPIE design can be modified to fit the needs of different educational activities, age groups, and learning levels. GUPPIE will provide an easily duplicated, inexpensive modular educational platform to use nationally in outreach, middle school and high school classes, and college engineering courses to inspire and rigorously prepare a more robust workforce. The GUPPIE helps students continually advance their knowledge in physics, mathematics, mechanics, electrical engineering, computer science, and oceanic engineering.

## 2 GUPPIE Evolution

GUPPIE shares the buoyancy-driven concept with the commercial underwater gliders. Underwater gliders use change of buoyancy by taking or expelling water into buoyancy tank to travel in the water to perform a saw-tooth pattern. The wing generates lift force and causes the vertical motion to translate into forward motion. Gliders can collect temperature, salinity, pressure, and ocean floor mapping data as they fly through water.

The GUPPIE was developed using practical, familiar, and off-the-shelf components such as syringes for ballast system, hull made of acrylic tube or water bottle, straw for plumping, and servo motors used in toys for actuation. Students start the GUPPIE project with a demonstration of interaction of gravity versus buoyancy force acting on submerged bodies using wood sticks and paper clips



**Fig. 3.** (A) Micro Glider used for elementary school and outreach day program, (B) Mini GUPPIE used in middle school level, (C) GUPPIE used in high school level, (D) Super GUPPIE used in undergraduate level

to build Micro Glider [12] as depicted in Fig. 3A. The wood sticks represent the effect of buoyancy and paper clips represent the gravity. This activity is also used during one day outreach programs. We observed that the Micro Glider activity is interesting for teachers and parents during the outreach programs as they would participate in this short activity to make their Micro Glider work.

The mini GUPPIE [13] illustrated in Fig. 3B was used in middle school level where students are comfortable to play with the familiar objects such as water bottle, syringes, batteries and cutting a tail from plastic sheets. The buoyancy engine in mini GUPPIE is composed of a single syringe, servo motor, and a lead screw. Electronics and micro processor are the new components for this age group. During mini GUPPIE course students learn how to solder, how to build a circuit, and how to design a part with engineering design software simplified for their age. 3D printing as an emerging technology is introduced to them in this program. Students can 3D print the parts designed with the CAD software to experience this new technology first hand.

The control system for GUPPIE had to be easily understood by non-technically educated students, and the platform had to be useful as an educational tool as their knowledge increases. To satisfy these constraints, a simple bang-bang control system was selected for the GUPPIE for younger students. In mini GUPPIE the amount of water intake in the syringe is controlled by a push button. When the piston reaches its maximum travel it hits the switch and reverses the travel direction. The glider estimates depth based on time and glide angle, and engages the buoyancy engine when the syringe is empty or full.

In the high school level we used GUPPIE, depicted in Fig. 3C. This version uses mainly 3D printed parts for structural supports and a wing. Students learn CAD design software such as Solid Works in more depth. They are required to

design and 3D print the end plates of the glider. They also use more sophisticated circuitry to use in their robot. In this version the GUPPIE uses three syringes as the buoyancy engine in conjunction with the servo-pinion actuation system. Soldering and wiring is part of the production stage that students have the opportunity to practice and improve their hands-on skills using Printed Circuit Board (PCB) instead of the bread board. The GUPPIE uses the time based control system for depth control and limit switch for buoyancy control.

The Super GUPPIE illustrated in Fig. 3D is utilized in college and undergraduate level. This version of the GUPPIE not only is capable of performing the traditional saw tooth motion of the glider but also has the mechanism to turn. The roll actuator is added to this version to increase the complexity of this robot and introduces new challenges for this group of learners. Super GUPPIE uses a pump instead of the servo and ballast tank in place of the syringes in the buoyancy system. Therefore, the control system would require position feedback on the piston's position. The Super GUPPIE uses linear potentiometer to measure the amount of travel the piston head takes. This simple modification would allow the Super GUPPIE to be used to demonstrate other linear and nonlinear control systems.

As for the processor the family of Arduino board were used in GUPPIE projects. The open source hardware and firmware allows students to work with the glider at their level of expertise. Arduino Nano was selected for mini GUPPIE due to simplicity of the circuitry and limited space in the water bottle. Arduino Uno was selected for GUPPIE as an upgrade comparing to the mini version which requires more processing capability and suitable number of I/O pins to be used in the electrical circuitry. Arduino Mega was used in Super GUPPIE to allow two Pulse Width Module (PWM) servos and a pump to actuate the internal mechanism while integrating sensors, including compasses, gyroscopes, GPS, inertial measurement unit (IMU), and accelerometers.

In undergraduate level, students use Super GUPPIE to learn about the control system in more depth. To model their vehicle they use Sim-Mechanic in MATLAB environment by importing the CAD model for example from Solid Works software. Sim-Scape, a package in Sim-Mechanic, has predefined blocks that define different linking joints, gear and coupling, and drive trains. Students can modify their model and add appropriate forces, torques and environmental elements to the model and run the simulation to obtain the rendered model of the vehicle. By manipulating the control signals one can manipulate the actuators and study their effect on the vehicle motion.

Figure 4 illustrates students attempt to use Super GUPPIE as a hardware in the loop platform to study the effect of hydrodynamic forces and the behaviour of the glider during saw-tooth motion. Other options that the simulation offers to students are changing the position of the glider wing and study the effect of hydrodynamic forces, mainly lift, on the vehicle motion [10].

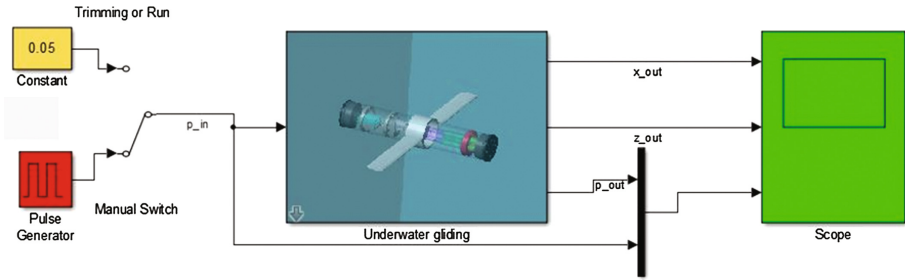


Fig. 4. Super GUPPIE in simulation as a hardware in the loop platform

### 3 GUPPIE Curriculum Development and Implementation

One of the outcomes of the GUPPIE program is the multi-disciplinary curriculum development that combines elements of electrical, mechanical, mechatronics engineering, and computer science and use it as a tool to teach STEM concepts. As mentioned in literature the fundamental issue of teaching STEM through robotics is the curriculum development that determines the success of learning and alignment of technology with STEM learning [3].

To engage the students in a real world problem solving scenario, we attempted to teach them how to approach the problem similar to the real world professionals. To this end, from the beginning of the course students were introduced to the engineering practice process. Engineering practice as depicted in Fig. 5, is composed of six stages of developing engineering process solutions. This process includes ask, imagine, plan/brainstorm, create/build, test/validation, and improve/redesign.

The curriculum developed for GUPPIE is aligned with six beyonds that Perkins [14] suggested. Main attributions of the GUPPIE curriculum with respect to Perkins' criteria are: (1) it goes beyond the basic skills and creates a learning environment that helps improving students' 21st century skills such as critical and creative thinking, problem solving, decision making, communication, and collaboration skills; (2) it does not just focus on discrete disciplines rather includes interdisciplinary topics and problems; (3) it is a Worth Learning subject that is not a subject without any application in the real word, it connects the content with real life situations and problems.

#### 3.1 Program Content

The GUPPIE curriculum in all three levels, middle school, high school and college, follows engineering practice cycle and starts with an introduction to robotics and showcases the current state-of-the-art robots through available promotional videos on the World Wide Web. Students discuss how robots can affect human life and the surrounding environment. To familiarize students with the different components of the GUPPIE and their functions, students explore a pre-assembled GUPPIE. This experience helps students to explore the robots before

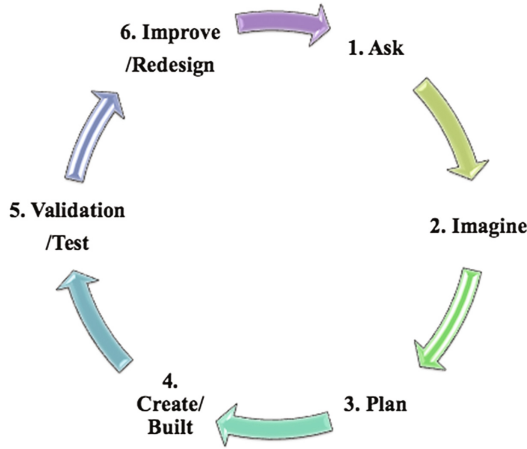


Fig. 5. Students learn engineering practice process during GUPPIE program

engaging in the design process and make a connection between experience and scientific concept.

The GUPPIE program includes mechanical design, physics of mechanics, basics of electronics and circuits, and C programming environment as depicted in Fig. 6. In GUPPIE curriculum, regardless of the age group, we paid special attention to the programming related activities. Since coding could be intimidating for some students, we adopted what we call *play and learn* and *parallel learning* approaches with this aspect of STEM learning.

Integrating programming with electronics in the GUPPIE curricula through small projects, helps students to understand how programming can manipulate the data received from the sensors or to make the circuits work autonomously. With this approach students can see how their code (software) interacts with the components (hardware), achieving satisfaction in each step, which in turn builds

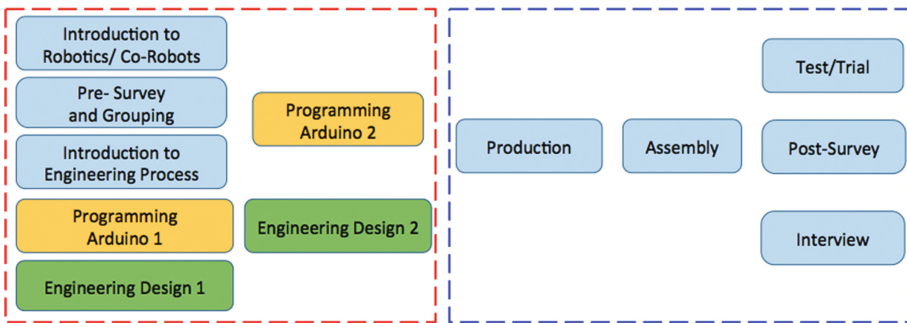
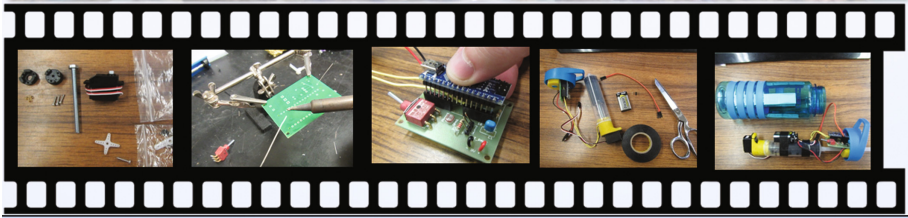


Fig. 6. GUPPIE program curriculum plan



their confidence in learning less familiar concepts. When they write a code to move a servo motor they observe the outcome of the programming in a physical sense and relate it to the similar phenomenal in the real life such as moving parts in robots.

Most of the experiments involving robotics activities are not part of the curriculum and are integrated into after-school programs or in summer camps [1]. One of the main advantages of conducting informal courses with educational robots over formal curriculum is that they are short term, require minimum curriculum design and the robot experts can be available during the program. Also minimum training is required for the teaching staff if the school decides to run the session with their own teaching staff. If the program were to be conducted over the course of few weeks and as part of the official curriculum, then a systematic teacher training is required. These types of programs are suitable to study the longitudinal effects of the curriculum [11].



**Fig. 7.** Mini GUPPIE assembling process

Although the informal programs such as summer camps are not sufficient to integrate robotics in K-12 but these courses are very effective to develop the curriculum and adjust it to the needs of the students in different age group as part of a bigger plan. The informal curriculum is a snapshot of the long term program with a fast fruiting process. This approach helps foster both the robot and the curriculum from prototype to final product. For example mini GUPPIE program in the informal curriculum is composed of two days of fundamentals and three days of speciality. In fundamentals students learn about building circuits in general and basics of coding by completing small projects. In the specialty days, they build the mini GUPPIE circuitry, write the program that moves GUPPIE in a saw-tooth pattern. For example day one speciality is assembly. Assembly process is a combination of structural, mechanical, and electrical component assembly. A snapshot of this process is depicted in Fig. 7. Following this stage they perform functionality and leak test. At this stage they learn about troubleshooting and how to rectify raised problems. By day five if they are confident that the mini GUPPIE performs as expected they take the glider to the swimming pool and test the performance. Figure 8 illustrates middle school students effort to test their mini GUPPIE in the swimming pool.



**Fig. 8.** Middle school students test their Mini GUPPIE in the swimming pool

### 3.2 Program Evaluation

The GUPPIE program was introduced to high school students from summer of 2013 through Michigan Tech's Center for Pre-College Outreach and Summer Youth Program (SYP) under Women In Engineering and Engineering Scholars Program, attracting scholar students nationwide. This robotic platform was used in several outreach programs and water festivals in the Upper Peninsula of Michigan from 2013 to present time, exposing just a little over 2000 local students to STEM concepts. Previously the SeaPerch was used with the high school students in SYP. In 2013 both SeaPerch and GUPPIE were used in summer camp to teach fundamentals of underwater vehicles to students and to transition between SeaPerch and GUPPIE program. The mini GUPPIE program was offered in two consecutive summer (2015-16) to middle school students in collaboration with Western Upper Peninsula's Center for Science, Mathematics, and Environmental Education (WUPC) and Michigan Tech's SYP. The third year of the middle school program is scheduled for July 2017. GUPPIE was also used as college educational tool with the community college program during summer 2015-16. The Super GUPPIE was used to prepare undergraduate and graduate student to work on underwater glider projects from 2013 to present time. In 2014 the Super GUPPIE was used in undergraduate control class MEEM 4700 as a hardware in the loop. Table 1 illustrates the participation of the students in the SeaPerch and GUPPIE program.

While the GUPPIE program was initiated to engage students in robotics in early ages and increase the excitement of STEM related subjects among students through hands-on activities, we used the opportunity to developed a curriculum

**Table 1.** GUPPIE program participation statistics

Year	Middle school			High school			Outreach	College
	Total	Boys	Girls	Total	Boys	Girls	Total	Total
2012	–	–	–	24	12	12	400	–
2013	–	–	–	26	15	11	500	6
2014	–	–	–	23	12	11	550	44
2015	20	18	2	24	0	24	500	4
2016	31	20	11	50	20	30	550	4

for a systematic informal program that is usable by teachers. We also assessed the effects of a hands-on course on students attitude and confidence towards STEM related subjects during this program.

To this end and to evaluate the GUPPIE course, we adopted four different assessment methods: survey, in-class assignments and challenges, observation, and interview. The first sets of surveys are collected before the students attended the program including writing an essay to answer questions about their background in robotics, their level of computer/coding skills, and their comfort level with teamwork. They are also required to complete an on line survey on the first day of the class. In the pre-course survey students rate the level of their interest in different STEM related subjects in scale of 1 (not interested at all) to 7 (very interested).

After completing the course, students complete a post-course survey and participate in a post program interview. The post-course survey assesses students' interest in each course activity and how the activities increase their confidence and interest in engineering and robotics. The post-course interview is useful for gaining more knowledge of their experience with the program. As an example Table 2 illustrates the results of the pre and post survey on middle school students interest in STEM related subject in 2015. The trend shows that the level of interest in use of computer, programming, and robotics increased over the week long program.

**Table 2.** Students interest in STEM, Middle school survey result in 2015

Topic	Pre average	Post average	Pre median	Post median
Robotics	5.55	6	6.5	7
Using computers	6.0	6	6.1	6.5
Computer programming	5.4	6	5.5	6.5
Making things	6.55	7	6.2	6.5
Science	6.6	7	6.9	7
Math	6.4	7	6.2	7

**Table 3.** Students confidence in activities, Middle school survey result in 2015

Daily activity	Median-start	Median-End	Average-start	Average-End
Circuits 101	4.5	5.5	4.1	5.6
CAD Design	4	5	3.9	4.5
Programming	5	6	4.6	5.9
GUPPIE assembly	5	6	5.1	6
GUPPIE Programming	6	6	5.1	5.8

As another example of the survey results, Table 3 shows students' confidence in different activities before and after daily GUPPIE activities in middle school level in 2015. Students feel more confident to assemble and program GUPPIE after learning how to write codes and wire a circuit throughout the week.

The results of the post-course survey and post-program interview are compared to the pre-survey controlled data as an outcome of this work. The comparison and other evaluations assist us in recognizing the successful pedagogical practices and the ones that need improvement.

## 4 Discussion, Conclusion and Future Work

This paper presents Glider for Underwater Problem-solving and Promotion of Interest in Engineering (GUPPIE) as an educational tool to engage students in STEM learning as early as 4th grade. The GUPPIE program has a real world focus on environmental contexts that are meaningful and *make a difference*. It offers continuous design potential and engagement for pre-college (4th through 12th grade) to college students through use of a platform that integrates design with engineering.

GUPPIE is easy and inexpensive to manufacture, with readily available light-weight and durable components, and modular to accommodate a variety of learning activities. Its interdisciplinary nature makes it an excellent pedagogical platform for teaching core science and engineering concepts through hands-on meaningful contexts. In particular, GUPPIE blends physics, computer science, electrical engineering, mechanical engineering, and environmental engineering. The GUPPIE program scaffolds and stimulates inquiry-based problem-solving learning to and through college.

Since 2013, the GUPPIE program has engaged over 2000 students to 4th–12th grade and college level students. To increase the impact of the program and reach more students, the next focus is pre-college school teachers. We know teachers play an important role in encouraging STEM interest and engaging students in activities that will translate interest into academic and career pathways. Teachers and mentors will be trained to engage students as designers of marine robots that can be used to analyze socially meaningful problems related to water and environmental engineering. Through workshops and summer institutes, teachers

will be coached to how to build GUPPIEs and how to utilize GUPPIE to instruct interdisciplinary STEM concepts in middle school and high school students either in class or after school programs.

**Acknowledgement.** This work was funded by National Science Foundation and worked under grant numbers 1426989 and 1453886.

## References

1. Benitti, F.B.V.: Exploring the educational potential of robotics in schools: a systematic review. *Comput. Educ.* **58**(3), 978–988 (2012)
2. Eguchi, A.: What is educational robotics? Theories behind it and practical implementation. In: *Society for Information Technology and Teacher Education International Conference*, San Diego, USA (2010)
3. Alimisis, D.: Educational robotics: open questions and new challenges. *Themes Sci. Technol. Educ.* **6**(1), 63–71 (2013)
4. Khanlari, A.: Robotics integration to create an authentic learning environment in engineering education. In: *IEEE Frontiers in Education Conference (FIE)- Erie, USA* (2016)
5. Padir, T., Chernova, S.: Guest editorial special issue on robotics education. *IEEE Trans. Educ.* **56**(1), 1–2 (2013)
6. Britt-Crane, M.: Small-scale underwater glider (2013). <http://www.seaglide.net/about/>
7. Stambaugh, B.: Waterbotics: Pooling students to stem. In: *EGU General Assembly Conference Abstracts*, vol. 17 (2015)
8. Verma, A.K., Dickerson, D., McKinney, S.: Engaging students in STEM careers with project-based learning-marine tech project. *Technol. Eng. Teacher* **71**, 25–31 (2011)
9. Nelson, S.G., Cooper, K.B., Djapic, V.: Seaperch: how a start-up hands-on robotics activity grew into a national program. In: *MTS/IEEE Oceans*, Genova, Italy (2015)
10. Ziaeefard, S., Ribeiro, G.A., Mahmoudian, N.: GUPPIE, underwater 3d printed robot a game changer in control design education. In: *American Control Conference (ACC)*, Chicago, USA (2015)
11. Mubin, O., Stevens, C.J., Shahid, S., Al Mahmud, A., Dong, J.-J.: A review of the applicability of robots in education. *J. Technol. Educ. Learn.* **1**, 209-0015 (2013)
12. Ziaeefard, S., Mahmoudian, N.: Building micro underwater gliders: lesson plan for exploring engineering design and understanding forces and interaction. *Michigan Sci. Teachers Assoc. MSTA* **61**(2), 49–56 (2016)
13. Ziaeefard, S., Mahmoudian, N., Miller, M., Rastgaar, M.: Engaging students in stem learning through co-robotic hands-on activities (evaluation). In: *American Society for Engineering Education Conference (ASEE)*, New Orleans, USA (2016)
14. Perkins, D.: *Future Wise: Educating Our Children for a Changing World*. Wiley, Hoboken (2014)

# Designing Robotics Student Projects from Concept Inventories

Reinhard Gerndt<sup>1</sup> and Jens Lüsses<sup>2(✉)</sup>

<sup>1</sup> Ostfalia University, Wolfenbuettel, Germany  
r.gerndt@ostfalia.de

<sup>2</sup> University of Applied Sciences Kiel, Kiel, Germany  
jens.luessem@fh-kiel.de

**Abstract.** Student projects play a crucial role in current tertiary education. Projects help students to verify their understanding of technological and scientific concepts by applying them to practical problems. Typically they represent a phase between a consumptive and active learning or between acquiring and applying knowledge. This phase is of paramount importance to education, especially in science and engineering. However, there is no systematic way of designing robotic student projects. With this paper we want to propose a method of deriving student projects from concept inventories (CI), listing the concepts that are necessary to comprehend in order to actively contribute to a scientific or engineering domain.

**Keywords:** Robotics education · Concept inventory · Student project

## 1 Introduction

Student projects play an important role in modern education. This holds for the engineering studies and specifically for teaching robotics, which is a highly trans-disciplinary field of education. Typically the projects are centered on a specific application or robot or robot kit that is available for a project.

Researchers emphasize the importance of student projects in order to increase the self-motivation of students and improve their ability to handle the engineering design process, consisting of the following steps: problem definition, invention, evaluation, decision, implementation and review [1]. Projects should help to encourage students to enter scientific and engineering careers [2]. These clear statements underline the necessity for a systematic approach to design student projects.

Student projects are part of almost every engineering curriculum. They serve as means for motivating young students [1]. Attached to a lecture and often more like a lab exercise than a complex task that stretches over several days or weeks and requires planning, they serve as a measure to foster cooperation among students in teams or with others, external to the home university during an internship or company placement. They allow digging deeply into a scientific or engineering problem as a final project.

Still the design of projects often appears to be random. Final projects and those for mature students are derived from, possibly changing, research projects or projects with

companies. Early projects, typically for a larger number of students, are driven by available infrastructure and optimized for an effective assessment.

However, accounting for the importance of projects in modern STEM education, the design of at least part of the projects should be guided by the objectives of the course of studies, as listed in the concept inventories [3, 4].

The remaining part of this paper is organized as follows: After this introduction we will detail the role student projects play in university curricula. Then we will revisit concept inventories and present two sets of projects that were derived from concept inventories. Finally we will present our conclusions.

## 2 Projects in University Curricula

In our curricula, robotics projects appear in different forms. We have Robotics projects:

- As an integral part of a robotics module (typically done in student groups)
- As a module on its own (typically done in student groups)
- As part of a working group (done alone or in student groups)
- In the form of the bachelor or master thesis (typically done alone)

In this paper we would like to concentrate on robotics projects as part of a lecture as well as a module on its own.

Robotics projects as part of a lecture - In our curricula, a robotics module consists of a lecture (typically 2 h per week) and a lab (2 h per week as well). In this context, the robotics project is part of the lab – besides the repetition of course content, deepening discussions, and exercises. In addition to the mentioned 4 h per week, students are asked to work additionally at home or at the lab. Altogether, the workload is around 150 h per semester.

Robotics projects as a module on its own - Students have the opportunity to choose a robotics project in terms of an elective module. The workload of such a module is also around 150 h per semester – this kind of project has a far greater complexity than the first project type.

Projects play a certain role in our curricula – but our curricula are far away from a curriculum completely based on project-based learning ideas.

## 3 Concept Inventories – The Robotics CI Example

This section shortly repeats the ideas behind concept inventories. Concept inventories list the relevant concepts of specific scientific fields and a single or a series of multiple-choice tests, related to those fields. Concept inventories serve as orientation for teaching and as a pedagogical measuring device to measure the levels of students and the gain of understanding, independent of the student's background and the actual method of teaching. This section will only give a very short introduction to the domain of concept inventories in general and to robotics concept inventories. For more details on concept inventories, please refer to the respective publications, e.g. [5, 6].

The list of concepts is assembled from the feedback of teachers and practitioners. It undergoes a lengthy period of revision and quality checks to make sure, all relevant concepts are covered. Within a scientific domain concept inventories may be adjusted for the level of students (e.g. school or university) and for sub-domains (e.g. analogue and digital).

The following list gives an overview of the different domains of concept inventories with the number of available concept inventories [3]:

- General Science and Learning (15)
- Force, Mechanics, and Materials (17)
- Electricity and Magnetism (12)
- Geosciences and Astronomy (5)
- Optics and Waves (4)
- Thermodynamics (8)
- Chemistry and Biology (7)
- Mathematics (10)
- Electrical Engineering (3)
- Computer Science (3)
- Other (6)

Central components are the concept inventory tests, one for every concept inventory, that allow assessing the understanding of students of the relevant concepts independently of the actual knowledge. Students typically undergo the test twice, first as a ‘pre-test’ at the beginning of the course and second as a ‘post-test’ at the end.

After an initial development phase, the tests typically do not change over some time or only develop slowly. This way, a single test can be used to measure the relative level of students of different years. In combination with a post-test after a class it allows assessing the concept learning gain of a specific course.

Noteworthy to mention is the fact that concept inventory tests must not be used for grading. Otherwise there may be strong incentive for students to specifically prepare for the tests, e.g. by memorizing the answers without working on the concepts. Therefore, test are carried out anonymously.

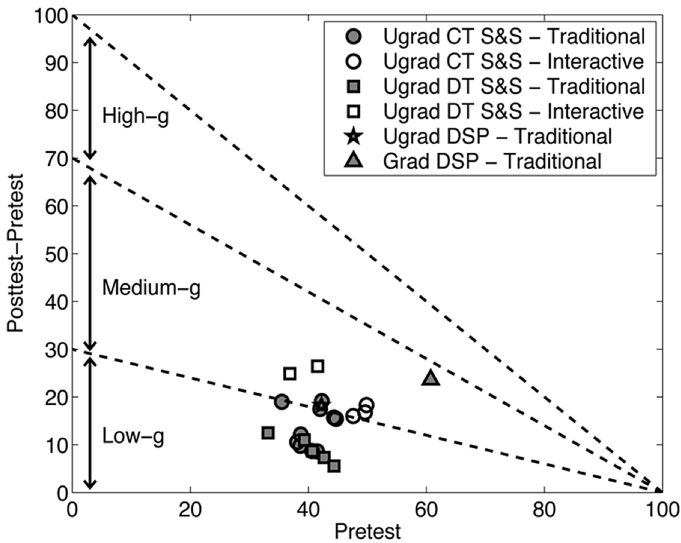
Evaluation of the test results can be very informative. Above the aggregated numerical values, the overall spread of answers and possible clusters of wrong or right answers often are interesting for teachers to know. With a second test development of understanding can be inferred. Figure 1 visualizes the gain for different teaching approaches in ‘Signals and Systems’ courses [7]. The learning gain is calculated by the following formula:

$$gain = \frac{post - pre}{100 - pre}$$

with post and pre representing the aggregated results of pre- and post tests.

Furthermore, the development of individual students may be interesting to follow. For this, a ‘magic value’ that does not allow to refer the two tests to a specific person, but to each other, is used. It may show improvements on some concepts and acquiring miss-concepts on others.



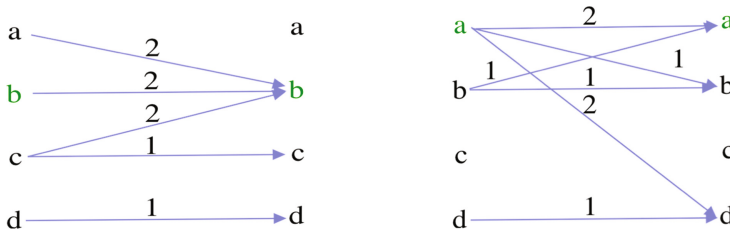


**Fig. 1.** Gain for different teaching approaches [7]

Figure 2a shows the development of understanding on a specific question. Whilst answers in the pre-test were more or less randomly distributed, in the post test the students demonstrated the understanding of the targeted concept by choosing the correct answer (b), which can be considered an outcome of the course.

A negative example is given in Fig. 2b. In this case, answer “a” was the correct one. This example shows how student, after answering the question correctly in the pre-test, got distracted and picked a wrong answer in the post-test. This result may indicate a wrong approach to teach specific concepts. However, this kind of ‘unlearning’ may also be a necessary step for students to overcome incorrect concepts.

A tentative list of categories for a robotics concept inventory is given in Table 1 [5]. The ‘math’ and ‘numerical methods’ category covers the mathematical foundation, which typically is related to linear algebra, differential equations and representation of multi-parameter properties by means of vectors and tensors. The ‘mechanics’ category covers all aspects of Newtonian mechanics. ‘Stability’ covers mechanical stability that keeps a robot from falling, control theoretic stability that keeps systems from un-intended oscillation and the stability notion of decision making, to consequently follow a plan. ‘Kinematic’ concepts are required for intentional behaviours of complex mechatronic systems and have some theoretical links with systems of linear equations and trigonometry. ‘Dynamics’ addresses the field of rigid body dynamics and dynamic behaviour. ‘Sensing’ of physical parameters requires signal estimation and filtering. ‘Perception’ can be seen as the level above sensing that turns sensor data into information for planning and decision-making. ‘Planning’ includes tasks like path planning. Concepts of artificial or computational intelligence are part of a separated concept inventory.



**Fig. 2.** Differences between pre- and post-test answers, (a – left) improving, (b – right) ‘unlearning’

**Table 1.** Tentative list of robotics concept inventory category list

#	Category	Concepts
1	Math	Transformation between different coordinate systems: select the transformation matrix that transfers a point from one coordinate system to another
2	Math	Time shift: given a plot of $p[n]$ , select the plot of $p[n + 1]$
3	Math/Numerical methods	Difference equations: Given a sequence of equidistant distance measurements, select the values for speed and acceleration
4	Numerical methods	Linearization: Given a curve, select a suitable stepwise linear representation
5	Mechanics	Spring-mass-damper system: give a specific configuration, select the steady-state configuration
6	Mechanics	Robot control: select a suitable configuration of a differential drive wheeled robot that would follow a specific trajectory
7	Control theory	Control parameters: Identify the a most suitable control response for a specific task
8	Stability	Static stability: Given a set of rigid bodies on different slopes, select the (un) stable one
9	Kinematics	Trajectory: given a differential drive robot with both wheels rotating at different speed with a fixed ratio, select the trajectory the robot takes
10	Kinematics	Building space: Given a specific robot arm configuration, select the sketch of the space the robot can reach with its tool
11	Dynamics	Motor momentum: given four robot configurations, select the one that requires the lowest motor momentum for a given task
12	Sensing	Drift: Assume a measuring system that adds a fixed, ever increasing value to the measured value, determine the time after which the measurement will be unreliable
13	Perception	Object properties: Given a four different objects, determine the number of properties to identify the objects
14	Planning	Path planning: given a specific environmental configuration (obstacles and path), derive a suitable cost function that describes the situation
...		

## 4 Requirements for Student Projects Derived from Concept Inventories

Whilst concept inventories must not be used directly for teaching, they may serve as a good orientation for defining student projects, which we will demonstrate with a small example. Assume the partial list of concepts listed in Table 2.

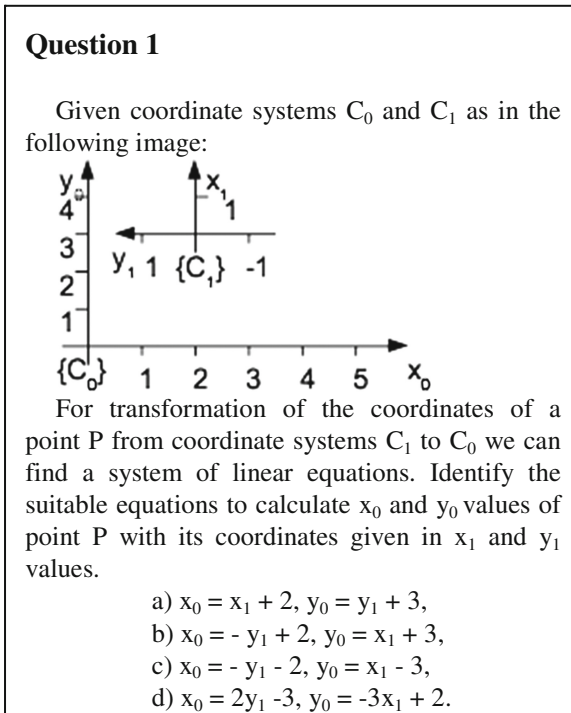
**Table 2.** Partial list of concepts resp. questions

	Concept/Question	Refers to category (Table 1)
1	Transformation	Math
2	Time shift	Math
3	Acceleration	Math
4	Small-signal/linearization	Numerical methods
5	Mass-Damper	Mechanics
6	Segway	Kinematics
7	Maze	Planing
8	M-Bot	Dynamics

The first question is related to the coordinate transformation between different coordinate systems or frames of reference, e.g. an inertial coordinate system and one that is statically or dynamically transformed with respect to the inertial coordinate system. The transformation concept has different levels of complexity. In mechanics it could be a displacement along a single or multiple axes, a rotation around a single or multiple axes or a combination thereof. Image acquisition and processing adds aspects like scaling and perspective. Depending on the maturity of students, the project may make use of different levels of complexity. The transformation between two coordinate systems may depend on a single parameter, e.g. the angle of a single robot arm experiment – like an actuated pendulum or it could be a mobile robot with a local (mobile) and a global (fixed) frame of reference. Figure 3 shows details of the question related to transformations.

The time shift question is related to the delay of signals. This is a common problem in processing sensor data. Pre-processing and evaluating sensor data may result in significant delays between the occurrence of an event and its perception by a control system. Similarly, actions may be delayed due to processing and communication times. However, delays may be neglected by using an ideal simulator, such that the related concepts can be excluded from the learning process by choosing a respective project setup.

The third question is related to changes in speed of mobile systems. Typically, there is at least one instance when mobile systems experience an acceleration, which is when they start or stop moving. Often, the moment can be neglected, if there is a simple on-off-control of robot locomotion. However, there are scenarios, e.g. a soccer-playing robot to avoid interception by an opponent that may require consideration of deceleration and acceleration.



**Fig. 3.** Example question 1 [6]

The small-signal and linearization question is related to the concept of acceptable error. With high computation power or long processing times, imprecise computations can be avoided. However, small computers and hard real-time requirements may need to sacrifice accuracy.

The mass-damper question is related to the concept of stability and oscillation. However, oscillation requires specific classes of systems and setups, which are subject to the design of the project.

The Segway and the M-Bot question relate to the kinematic concepts of mobile robots, as well as to the concept of friction which is a crucial prerequisite for accelerating masses. Kinematics is introduced by requiring drives with the typically inherent parameter of the axis angle to control a parameter like a position by means of a transmission. Friction can be introduced to a project by the overall use case, e.g. requesting the robot to move on slippery surfaces or requiring it to climb slopes.

## 5 Student Projects Derived from CIs

### 5.1 4-DOF Bipedal Robot

The task is to implement a bipedal robot with as few degrees of freedom (DOF) as possible. Aside of the Jansen mechanisms, the artist uses for his mobile installations [8] and of ‘Toothbrush robots’ with only a single degree of freedom and without control of orientation, legged robots require a minimum of 4 drives or degrees of freedom (DOF). Making them move requires understanding of multiple concepts of the robotics, mechanics and other concept inventories.

Figure 4 shows a 4-DOF bipedal robot built by computer science undergraduate students as practical part of a robotics lecture. The robot was assembled from a mechanical kit with conventional scale modeling servo controllers, a raspberry PI mobile computer and a power bank. The main objective was implementing a locomotion scheme. The timeframe was very tight with only few hours allocated to the task. The project thus covered only a few concepts. For example, no sensors were involved in the project such that no data pre-processing, delays, control stability issues were involved. However, there have been some design considerations as can be inferred from the ‘face’ applied to the robots ‘head’.

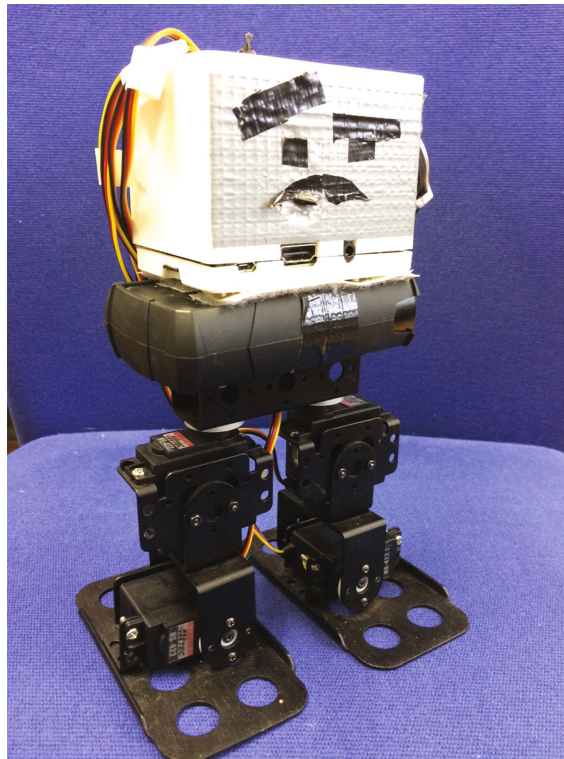


Fig. 4. 4-DOF bipedal robot

Still, students carried out calculations on Newtonian mechanics and mechanical stability. Figure 5a is taken from a student report to estimate the maximum angles of the two joints, which eventually lead to the zero momentum point (ZMP) approach that still is widely used in the robotics domain. Figure 5b, taken from another student report shows the region of stable stance for the possible combinations of the two joint angles.

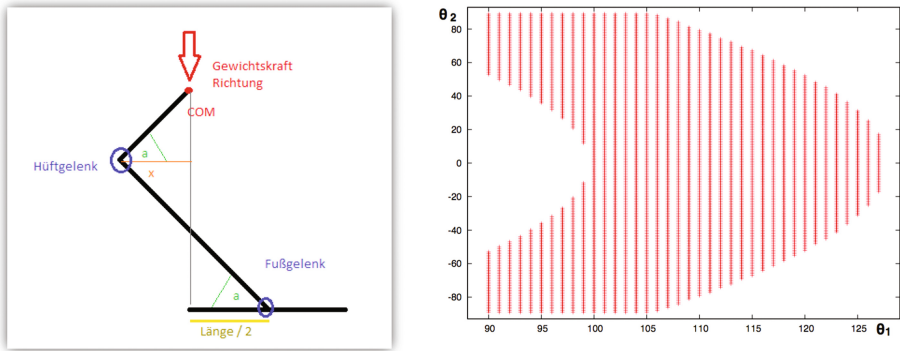


Fig. 5. Student sketches on stability issues related to bipedal robot

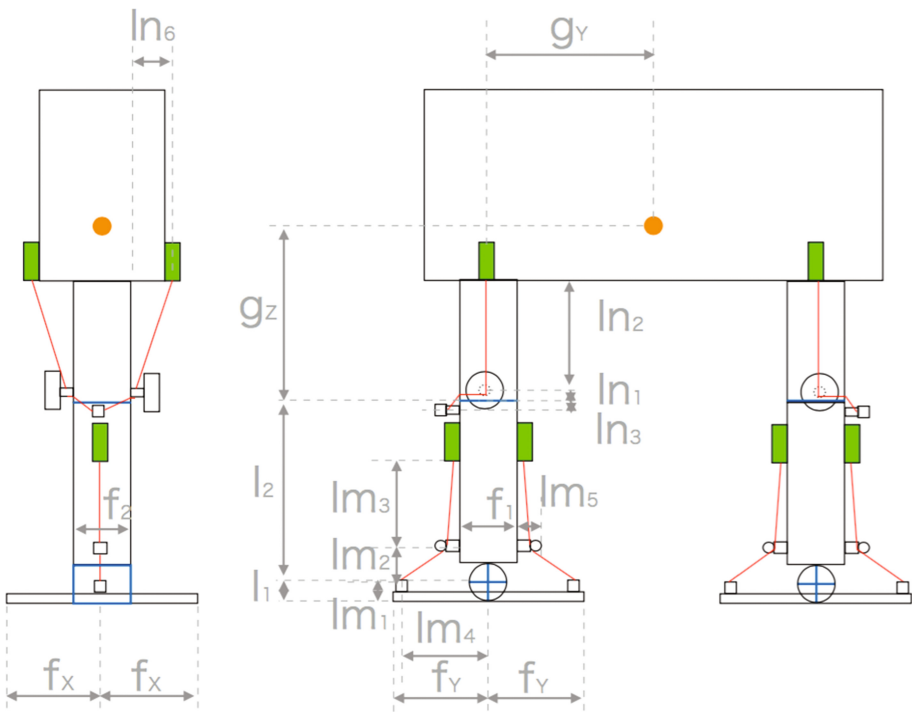


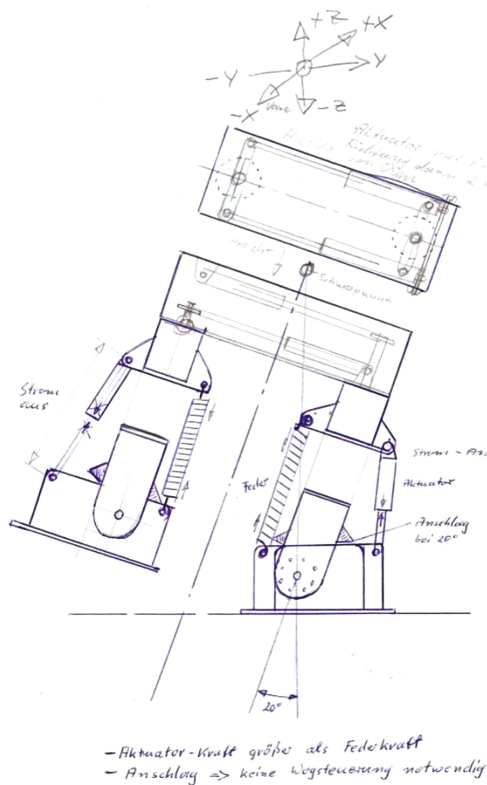
Fig. 6. Engineering drawing of bipedal robot with linear actuators

In a subsequent project, designed to for an even deeper involvement with mechanical concepts, students have been asked to replace the rotational motors by linear actuators and consider the mechanical and kinematic properties, e.g. maximum force required, as opposed to the previous project with all components given.

Figure 6 shows an engineering drawing taken from a student report indicating the linear actuators (green boxes attached to body and legs) and strings with a guiding system to transmit the forces (red liens, attached to the actuators). Furthermore, the dimensions, required for calculating forces and estimating the behavior are given.

Figure 7 shows another design with a rigid coupling of the drives and a spring to replace one of the motors on each leg. In order to address the kinematics the student added the local coordinate system to his sketch. Taken from another report, we see the sketch of the robot's path, used to estimate the step width and speed of locomotion (Fig. 8).

All projects were designed to deepen the understanding of specific aspects of the lecture. The two examples indicate how the focus of otherwise similar setups may be controlled by the design of the project.



**Fig. 7.** Alternative bipedal robot concept

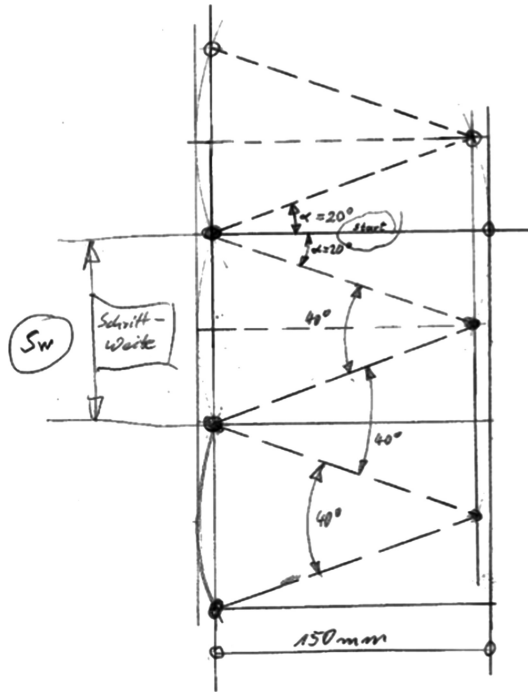


Fig. 8. Concept of locomotion

## 5.2 Mixed-Reality Robots

In this project, we aim to build and work with a number of cheap differential drive robots. These robots are remotely controlled. A central vision system provides the robots with pose and position. Each and every robot is controlled – independently from the other robots – via an infrared interface by a computer.

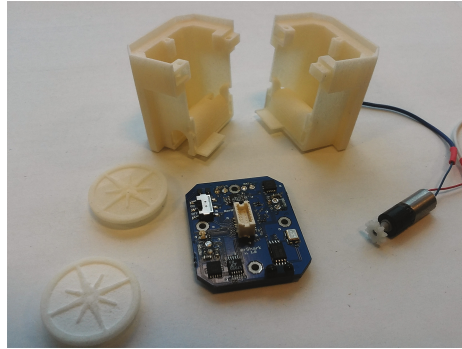
The project was so huge, that we split it up into a number of smaller sub-projects:

- Designing and building the robots
- Controlling the robot including implementing low-level operations
- Playing football with the robots

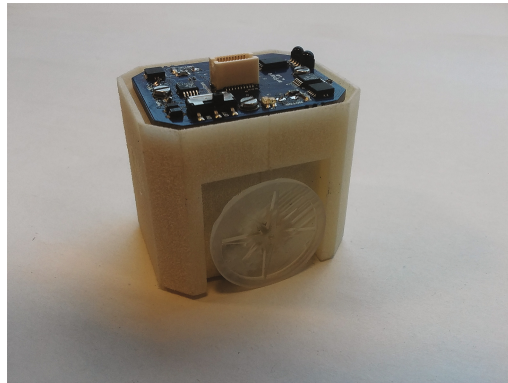
### Sub-project: Designing and building the robots

The goal of this sub-project was to design and to build a cheap differential drive robot using a 3D-printer (see Figs. 9 and 10). Furthermore, the robots should be tested with focus on mechanical tests.





**Fig. 9.** Elements of the differential drive robot



**Fig. 10.** Assembled differential drive robot

### **Sub-project: Controlling the robot including implementing low-level operations**

The first aim of this sub-project was to establish the communication between the robot, the central vision system, and the computer – via a central server. Secondly, the most important low-level operations like moving forward, moving left, etc. have to be established. The functionalities should be tested performing some kind of integration test.

### **Sub-project: Playing football with the robots**

In this sub-project the students should generate autonomous agents based on the results of the two precedent sub-projects. The goal is to build autonomous agents that are able to interact with each other in order to play football on a mixed-reality football field (see Fig. 11).

The sub-projects have been designed in such a way that the foci were on different categories of the concept inventory (Table 3). Test results – the tests were conducted in the first robotics lecture – helped us to build teams and to distribute them on the different sub-projects.



**Fig. 11.** Mixed-reality soccer game

**Table 3.** Categories addressed by the sub-projects

Category	Sub-project 1	Sub-project 2	Sub-project 3
Mathematics	×	×	×
Numerical methods	×	×	×
Mechanics	×		
Control theory		×	
Stability			
Kinematics		×	
Dynamics		×	
Sensing		×	
Perception		×	×
Planning			×
Navigation			×
Decision-making			×
(Dealing with) uncertainty			×
Robot Design	×		
Human-robot-interaction			
Artificial intelligence			×
Project management	×	×	
Electronics	×	×	
Programming		×	
Exploration			×

## 6 Conclusions and Outlook

In this paper we presented an application of concept inventories on project design. Categories of the concept inventory – we have chosen the robotics concept inventory as an example – have been taken as guidelines to design projects. This approach allows us to make a certain kind of internal differentiation in order to help students to eliminate weaknesses.

We have just started with this kind of project design. The first results of the described approach were promising. Nevertheless, we have to collect more data in order to get valid results.

## References

1. El-Howayek, G.: Introducing computer engineering major for first year students using robotic projects. In: 2016 IEEE Frontiers in Education Conference (FiE) (2016)
2. Crepaldi, M., Demarchi, D.: Tackling technical research. *IEEE Potentials* **35**(3), 29–33 (2016)
3. Lindell, R.S., Peak, E., Foster, T.M.: Are they all created equal? A comparison of different concept inventory development methodologies. In: PERC Proceedings, vol. 883, pp. 14–17 (2006)
4. Ogunfunmi, T., Herman, G.L., Rahman, M.: On the use of concept inventories for circuit and systems courses. *IEEE Circ. Syst. Mag.* (2014). Third Quarter
5. Gerndt, R., Lüssem, J.: Towards a robotics concept inventory. In: 6th International Conference on Robotics in Education, Yverdon-les-Bains, Switzerland (2015)
6. Gerndt, R., Lüssem, J.: Concept inventories for quality assurance of study programs in robotics. In: 7th International Conference on Robotics in Education, Vienna, Austria (2016)
7. Wage, K.E., Buck, J.R., Wright, C.H.G., Welch, T.B.: The signal and systems concept inventory. *IEEE Trans. Educ.* **48**(3), 448–461 (2005)
8. Jansen, T.: Strandbeest, in the Internet. <http://www.strandbeest.com>. Last visited 6 Apr 2017

# Teaching Research Methodologies with a Robot in a CS Lab Course

Mathias Landhäuser, Sebastian Weigelt<sup>(✉)</sup>, and Martin Blersch

Institute for Program Structures and Data Organization,  
Karlsruhe Institute of Technology, Karlsruhe, Germany  
{landhaeusser,weigelt,blersch}@kit.edu

**Abstract.** We describe a computer science lab course that teaches a scientific research methodology. Our course is based on a recent research project and augments it with a motivating goal: We program a robot based on the Lego EV3 brick with plain English. Our students developed a multi-module solution; all modules were planned, implemented, and benchmarked following a research life cycle.

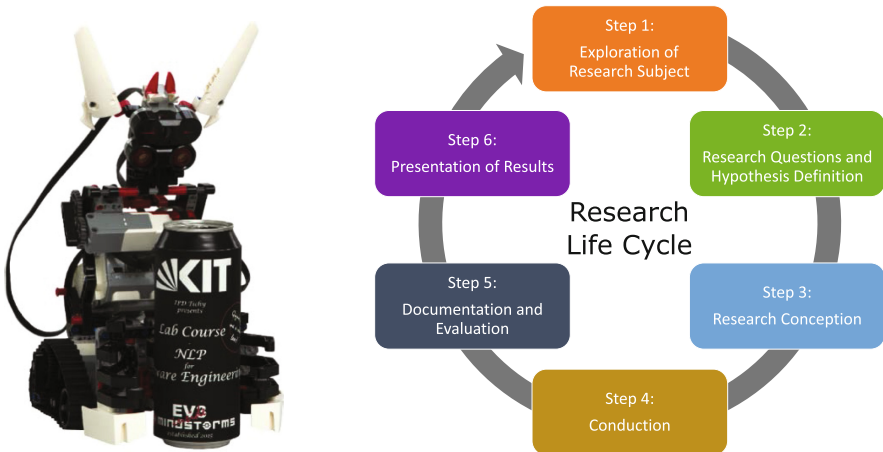
The results of our case study indicate that a lab course can familiarize students with academic research methodologies. Interviews with the participants, conducted half a year after the lab course, confirm our positive impressions: Our participants indeed internalized the research life cycle *and* all told us that they wanted to pursue research in their upcoming curriculum. Three of four stated that the possibility to monitor progress by running solutions on the robot was highly motivating.

## 1 Introduction

In the last decades, teaching computer science (CS) and software engineering (SE) focused on the industrial employability of students. Educators spent much time and effort on industrial processes, agile methodologies, and specific technologies (e.g., references [4, 8, 13]). A different goal, namely academic employability, has re-gained attention in the last few years (e.g., references [3, 10, 12]): While classical CS and SE courses teach problem understanding, decomposition and finally building the solution, the academic research life cycle remains buried under the more “directly applicable” competencies. Many universities in Germany try to shift the education to the academic setting with a “research-based” teaching approach [9]: Teams of (MSc) students are assigned to a research project and plan and conduct actual research [6]. Such courses are an excellent way of exposing students to actual research and show that students can produce significant results. Yet, they are very demanding and labor-intensive, both for the students and for their advisors. (E.g., the department of informatics of the Karlsruhe Institute of Technology (KIT) features a two semester course, *Research Praxis*, that accounts for 24 ECTS credits<sup>1</sup> – a total workload of ca. 720 h. In comparison, the Master’s thesis accounts for 30 ECTS credits.)

---

<sup>1</sup> European Credit Transfer and Accumulation System, see <http://ec.europa.eu/>.



(a) Robot based on the Lego Mindstorms EV3 brick used in the 2015/2016 lab course.

(b) The research life cycle starts with exploring a research subject, covers hypothesis building, the “actual” research, and culminates in documenting and presenting the results.

**Fig. 1.** Gizmo, the robot used in the course, and the research life cycle.

We propose a less intrusive approach to research-based teaching: We layed out a lab course that mimics a research project [11] and can be completed with a workload of less than 10h per week over 16 weeks. It proposes a research topic and decomposes the project in chunks that can be digested by first-year MSc students. Even though the course is pre-structured, it allows the students to experience a research life cycle several times and to observe refinement and improvement of their solutions over time. It also provides a challenging environment that keeps them engaged.

The technical goal of the lab course is to build a compiler that generates Java code for the Lego Mindstorms EV3 robot shown in Fig. 1a from written English<sup>2</sup>. We have picked this project setting because a) the construction of natural language interfaces for robots is a relevant research topic, b) its size and complexity enables students to construct a full-blown system from scratch, and c) we expect students to learn more willingly with a tangible project at hand. The students have to design and implement a natural language processing (NLP) pipeline that analyzes English prose and transforms it to Java code. Because NLP is not a prominent part of the CS education, we introduced the topic and the needed concepts to the students in an introductory session. Students were invited to come to this introduction and decided afterwards whether they want to participate or not. In the introductory session we also described the *modus operandi* of the lab course and explained the rationale behind its design.

<sup>2</sup> See <https://youtu.be/Z.vt1-imBUE> for a short demonstration.

This paper reports the conception of the lab course in Sect. 2. Section 3 describes the 2015/2016 edition of the lab course and the participating students in detail. Section 4 summarizes our findings and the lessons learned and Sect. 5 concludes the paper.

## 2 Conception

To improve academic employability, we want to expose students to scientific methods as early and as intense as possible: The only way to gain experience in research is actually doing research. Yet, conducting an actual research project requires long-term effort and commitment – more than most curricula allow.

### 2.1 Learning Objectives

To make research tempting for MSc students, we have designed a lab course with an attractive setting. We build the lab course around the research life cycle as depicted in Fig. 1b. It describes the life cycle of a research project and starts with exploring the area of interest. At first, hypotheses are formulated, then research is planned and conducted. After evaluation and documentation, the results are presented. Then the cycle starts anew. We expect students to understand and follow this way of working after they finish the lab course. Additionally, we pursue the learning objectives shown in Table 1 to train skills important to conduct research.

MSc students should be equipped with some of the skills, at least partially, e.g. doing a literature review; in these cases, we want to strengthen their abilities and exercise them. Other skills are expected to be totally new for most or all of the students, e.g. building an ontology.

### 2.2 Technical Goal

The overall technical goal of the lab course is to develop a NLP pipeline to translate English prose to runnable Java code.

The application at hand is a robot based on the Lego Mindstorms EV3 brick with a crawler track, three sensors (a color sensor, an infrared and an ultrasonic distance sensor), and one grappler to lift and carry objects. The robot is a modified version of a standard model and was built by the authors for the lab course; it is depicted in Fig. 1a. The Lego Mindstorms EV3 brick is a system on a chip with up to four sensors and actuators respectively. The brick can be booted with Lejos<sup>3</sup>, a firmware that runs Java programs. Lejos comes with an object-oriented (but low-level) Java API that can be used to access all sensors and actuators attached to the brick. We provide the students with a comprehensive and well-documented facade that hides the low-level functions, e.g.,

---

<sup>3</sup> Information about programming the EV3 with Java on Lejos can be found, e.g., in reference [2].

**Table 1.** The learning objectives of the 2015/2016 edition of the lab course and their level on Bloom’s Taxonomy (c.f. reference [1]).

No	Objective (Student completing the course are able to:)	Level
O <sub>1</sub>	Perform a literature review for a given topic	L <sub>2</sub> –L <sub>4</sub>
O <sub>2</sub>	Build up a benchmark for a given problem	L <sub>4</sub> –L <sub>6</sub>
O <sub>3</sub>	Identify, benchmark, choose, and put (scientific) tools and/or approaches to use to solve a specific problem at hand	L <sub>2</sub> –L <sub>6</sub>
O <sub>4</sub>	Incorporate ideas gained in the literature review and from reviewed tools into newly developed approaches	L <sub>1</sub> –L <sub>6</sub>
O <sub>5</sub>	Define, structure, fill, and use an ontology for a specific domain	L <sub>2</sub> –L <sub>4</sub>
O <sub>6</sub>	Build a pipeline of tools to solve a higher-level problem	L <sub>5</sub> –L <sub>6</sub>
O <sub>7</sub>	Run a small-scale project in the area of NLP and software engineering	L <sub>1</sub> –L <sub>6</sub>
O <sub>8</sub>	Implement and benchmark a prototype for a given problem at hand and derive ways to improve the prototype based on the benchmark results	L <sub>3</sub> –L <sub>6</sub> L <sub>4</sub> , L <sub>6</sub>
O <sub>9</sub>	Document their projects, so that they can be reproduced	L <sub>1</sub> –L <sub>3</sub>
O <sub>10</sub>	Present their results and insights gained in a concise and precise manner	L <sub>1</sub> –L <sub>3</sub>

it uses the low-level methods controlling the two electric drives to provide several methods to move the robot and turn it with parameters for the distance and the rotation in degrees, respectively. The facade also provides methods to detect objects (parameterized with the searched-for object’s properties), to follow lines, to detect colors and so on.

We expect the students to build a NLP pipeline that translates an English description into code using the robot facade; an example input and the expected code is shown in Fig. 2. NLP pipelines are widely used and there are text books and standard frameworks, such as Gate, which we use in the lab. Gate is a state of the art toolkit for building NLP applications and provides a workbench as well as an embedded version; there are plenty of documentation and tutorials about how to use Gate [7].

```
// Gizmo turns until it sees a can that is red,
// makes its way towards it and then grabs it.
gizmo.turnUntil(new SeeObjectCondition(redTin));
gizmo.makeTowardsObject(redTin);
gizmo.grab(redTin);
```

**Fig. 2.** Example Java code produced by the NLP pipeline. The input text is given as comment above the generated code. The NLP pipeline must identify objects and actions and must resolve co-references and synonyms.

### 2.3 Assignments and Work Load

Our lab course targets computer science MSc students in their first year and is based on a recent research project. Students earn five ECTS credit points which account for a workload of approximately 150 h. 20 h of the work load are allotted to mandatory sessions that cover introductions to new topics, explanations of assignments, the phase review process, and the final presentation. 40 h of work load are used for optional sessions at our lab that are to be used for guided coding and Q/A. The remaining 90 h are allotted to literature review, design of solutions, and implementation.

Our assignments are based on the pipeline structure we employ. Roughly, each assignment produces one pipeline stage. The main advantage of this segmentation is that assignments are self-contained so that students can work on the tasks without one failed task to jeopardize the entire semester. On the other side, the assignments get interconnected so that later improvements on early stages benefit later stages. This way, we want to encourage the students to continuously improve their solutions. In the final session, the teams are expected to present their full-blown solution, i.e., the complete pipeline, and to show that their approach can translate prose to code. We grade each assignment. The assignments make up 70% of the final grade. The remaining 30% account for the final presentation.

### 2.4 Course Structure

To split the overall goal into smaller chunks, we decompose the translation process into the following steps: First, the input text needs to be read and tokenized; then, sentence boundaries need to be detected. After that, the sentences are fed to a part-of-speech (POS) tagger and lemmas are derived for all tokens. The next stages parse the sentences to produce syntax trees and dependency graphs; the former show the hierarchical structure of the sentences (e.g., what words form a noun phrase), the latter shows grammatical dependencies (such as subjects and objects to a given verb). Co-reference analysis is used to determine chains of words that refer to the same entity in the document. Next, the students have to create an ontology that models the API. The following, most challenging stage – the API ontology search task – determines which tokens represent actions or objects and maps them to elements of the robot facade. The final stage uses the results of its predecessors and produces the code. Compiling the generated code and transferring it to the brick is done manually.

The different assignments are shown in Table 2. All assignments require the students to perform a (limited) literature review to gain an overview of the possible approaches; then we want them to build a benchmark that demonstrates the problems they have to solve in the assignment. The following step depends on the assignment: Some assignments ask the students to implement a solution themselves, e.g., for sentence splitting, tokenization, and co-reference analysis; others require them to benchmark available tools and to choose the one most



**Table 2.** Lab course task decomposition and learning objectives (see Table 1).

Task	Learning Objectives
Tokenizing and sentence split	O <sub>1</sub> –O <sub>2</sub> , O <sub>4</sub> , O <sub>8</sub> –O <sub>10</sub>
POS tagging and lemmatization	O <sub>1</sub> –O <sub>3</sub> , O <sub>10</sub>
Parsing, dependency parsing	O <sub>1</sub> –O <sub>4</sub>
Co-reference analysis	O <sub>8</sub> –O <sub>10</sub>
API Ontology building	O <sub>1</sub> , O <sub>3</sub> , O <sub>5</sub> , O <sub>8</sub> –O <sub>10</sub>
API search using the ontology	O <sub>1</sub> –O <sub>2</sub> , O <sub>4</sub> –O <sub>10</sub>
Code generation	O <sub>1</sub> –O <sub>2</sub> , O <sub>4</sub> –O <sub>10</sub>
Preparation of final presentation	O <sub>1</sub> –O <sub>2</sub> , O <sub>4</sub> –O <sub>10</sub>

suitable for their approach, e.g., the POS tagger with the best performance on benchmark texts.

Ontology building requires the students to provide a converter that reads Java libraries (in their case our robot facade) and stores the API in an ontology. The ontology contains information about the given API (i.e., all classes, methods, data types and so on) and about the environment (i.e., objects of the real world that the robot can interact with). The classes and methods of the facade are not to be hard-coded into the students programs, nor their ontology; we want them to use an ontology as an interchangeable model of the API. Only the real world objects (e.g., cans) and their properties (e.g., their color) must be inserted manually. The students have to design the structure of the ontology by themselves.

The final assignment integrates all previous assignments to build the English to Java code compiler. The final presentation comprises a course and an English program that has to be converted to code by the teams' programs. We do not provide the English program in advance to avoid tuning of the programs towards this text.

## 2.5 Motivation and Competitive Environment

We acknowledge that students are (extrinsically) motivated by grades. To increase the intrinsic motivation, the lab course leverages the following motivating factors.

*Full Project.* We want the students to develop a comprehensive and working system, rather than extending or optimizing an existing one. Implementing own ideas to build a system from the very beginning can increase motivation considerably. The project should not only be an abstract, scientific one, but produce tangible results. Therefore, the project should include real-life objects (or systems) that can be used to demonstrate the overall goal and progress. We expect that students, who are able monitor their progress easily, are more willing to improve results.

*Benchmarks.* Using benchmarks throughout the lab course focuses the teams on competition. For every assignment the teams construct a benchmark each and use it throughout the implementation phase for testing and development. We tell the teams to compile the benchmark during or after the initial research phase and use it during design and implementation to monitor their progress. Benchmarking provides immediate feedback whether solutions suffice or have to be improved. We compile another benchmark that is used for grading the submissions; also the other teams' benchmarks are used to demonstrate the fitness of the respective submissions. All benchmarks are provided to the teams after the submission deadline. In the final presentation the teams compete on various tasks: their own, one given by us and the ones created by the other teams.

### 3 2015/2016 Participants and Schedule

This section describes the 2015/2016 edition of the lab course. The first subsection describes the students, the second subsection describes the course schedule in detail.

#### 3.1 Participants

In 2015, we had eight participants, all of which were in their first year of their MSc studies in CS. Six students attended the introductory session and two joined the lab course in the second session. After we presented the topic of the lab and the *modus operandi*, the students formed two teams. We did not interfere with the team building process because they partly knew each other from other courses. We provided the teams with infrastructure (such as a Git server, conference rooms, and Internet access). Division of labor as well as time management was left up to the teams.

All students that attended the introductory session formally registered for the lab course as well as the latecomers; there were no dropouts. Both teams submitted solutions to all assignments and both teams participated in the final presentation, proving that their solutions were sufficient to solve the overall task.

#### 3.2 Schedule

The lab course started in October 2015 and finished in February 2016, thus the lab course's duration was 18 weeks. In the introductory session, we showed the semester's plan to the students as outlined in Table 3. We allotted one up to six weeks per assignment. Assignments were composed of one to three pipeline stages, depending on the tasks' coherence. The last two weeks of the lab course provided time to improve all prior solutions.

Because the performance of later stages depend on the performance of earlier stages, the students were encouraged to improve their solutions continuously. The TAs provided fall-back code for the individual stages so that a bad solution for an early stage could not jeopardize the overall goal; in the end, both teams used their own solutions only.

**Table 3.** Schedule of the 2015/16 lab course.

Week	Task
1–2	Tokenizing and sentence split
3–4	POS tagging and lemmatization
5–6	Parsing, dependency parsing, co-reference analysis
7	API Ontology building
8–14	API search using the ontology
15–16	Code generation
16–18	Preparation of final presentation

## 4 Findings and Lessons Learned

The teams in the 2015/2016 edition of the lab course performed quite differently: After the first assignment, one team consistently delivered results that impressed us; the other team struggled with time management and separation of concerns. In the end, both teams delivered solutions to all assignments and succeeded in the final presentation.

We had the impression that the students stuck to our research life cycle and used benchmarking to assess their progress. Since there is no formal test after the lab course and the number of participants is too low, we cannot measure quantitatively whether we reached our teaching objectives. In this section, we report qualitative findings that are based on our observations during the sessions and the performance of the teams in the final presentation.

To back up these observations, we conducted semi-structured interviews with the participants half a year after the lab course finished. We did not tell the students in the lab course that there would be interviews and participation was optional. Four of eight students responded on short notice and participated in the interviews. As the interviews were optional, their results are subject to a selection bias, we believe. But note that we filed the grades before we invited the students to the interviews and that students with perfect grades followed our invitation as well as average students. (There were neither fair nor poor graded students.) As the feedback is positive, we conclude that the responses are at least not biased by bad grades: there was nothing to gain nor to lose for the students. We recorded the interviews with the students' consent and transcribed them to assure that we captured all responses. The interview questions considered aspects related to the research life cycle mainly (see Subsect. 4.1). The interviews comprised two parts: First, we asked the participants how they would solve a NLP-related problem, such as implementing a Named Entity Recognizer or benchmarking different solutions to assess whether they are able to apply research methodologies. Second, we asked for general feedback regarding various aspects of the lab course.

## 4.1 Observations and Interviews

The goals of the lab course are getting students in touch with research and teaching research methodology. Therefore we have to ask ourselves, “How well have we taught the research life cycle?” As mentioned before, we can not give a quantitative answer. Instead we will discuss the interviews and give our subjective impressions on this point.

*Observations by the TAs.* We observed that – over time – the students improved in applying research methodologies. The first task was very much solved *ad hoc*: Work was only vaguely coordinated and little benchmarking was done. In a later assignments both teams adhered to the research life cycle. At the end of the lab course most of the participants were familiar with the cycle. Additionally, the reward of obtaining better results further increased the confidence in the research life cycle. Both teams used the respective steps of the cycle to divide the task among the team members. We did not encourage them to do so, as we intended all participants to participate in every step of the life cycle. Retrospectively, the teams concluded that splitting up the work seemed to be a good idea in the beginning but led to considerable coordination efforts.

*Interviews Part A – NLP task.* When asked to solve a NLP task, the interviewees responded quite differently: Two of the four interviewees intuitively laid out an approach that exactly resembled the research life cycle; they even remarked that they would iteratively improve the solution based on the benchmarking results. Furthermore, they emphasized the importance of the different steps. Both perfectly recalled the procedure their teams used for a different task in the lab course and transferred it to the new problem. One of the two mentioned that he got used to the methodology, because they repeated the research life cycle seven times (once for each task). The third interviewee remembered some steps of the research life cycle only. The last one did not use any methodology to solve the task and was unable to describe his imaginary way to the solution. Yet, when asked for specific stages of the research life cycle the interviewee was able to explain the respective step and its importance.

*Interviews Part B – General Feedback.* When we asked them about their engagement and motivation throughout the course, all interviewees stated that they liked the project setting: Three of four emphasized the positive influence of the robot on their motivation. Since the robot illustrated their progress, the students were more willing to improve their solution. Two of them further explained that the creation of a natural language interface for a robot was very interesting. Moreover all interviewees appreciated that they had to develop a complete system during the lab course, not only isolated solutions to various problems. With regards to research methodology, all interviewees confirmed that the research life cycle was very helpful: The sequence of steps to solve a scientific problem helped them to organize their work. Three interviewees also assumed that their results would have been worse without the given methodology. When asked whether

their teams used the research life cycle consequently, again the interviewees responded quite differently: All but one stated that their team consequently used the research life cycle; the other reported they have seen it as a loose guidance only. Two admitted, they adhered to the cycle during the first assignments only. Another one put on record, its team used the research life cycle regularly, but it was most helpful during the API search task. The interviewee assumed that this was due to the challenging and time-consuming task. Coincidentally, all but one think that this was the most interesting sub-task. As there was no full-blown solution to be found in literature, the teams had to create own approaches, which most of all felt like “real research” to them.

We further asked the participants, whether they think that the lab course taught them research methodology. Three of four interviewees echoed positively as the lab course provided the chance to gain an insight into research, they said. Two never got in touch with research before and believe that the lab course was highly valuable. All four interviewees could imagine to further engage in research, either in an extensive course or later in their career (be it academic or industrial research).

## 4.2 Monitoring

KIT monitors the quality of its courses on a regular basis. All courses with more than 5 participating students can and should be evaluated; the evaluation requires the students to anonymously complete a survey. The 2015/2016 edition of the lab course was subject to the official evaluation and we discuss some of the insights we drew from the results.

The results clearly show that all participants felt that they benefited from the lab course in total. The students perceived themselves and their classmates as being hard workers and they liked the lab course. Also, they did not skip any session, even the optional ones right before and after the Christmas break.

Only the effort required to complete the assignments engendered criticism. This was anticipated: Because the lab course was not part of the regular curriculum, we told the students in the beginning that the lab course would account for 3 ECTS points. We noticed during the first two weeks of the course, that we under-estimated the time required to complete the assignments. We asked the department to change the lab course description to 5 ECTS points but the confirmation came after the evaluation. After we told the students that we could award them two additional credit points, they all chose to go for the 5-points version of the lab course. They also told us, that they would have responded far more positively if they had known that before the evaluation.

There is a second interesting observation regarding the effort for the course: Even though the students complained about the demanding assignments, they only did so in the light of the credit points they earned. And all of them were willing to invest the extra work due to the interesting project; it seems that the high-level goal “programming a robot in English” and the pipeline components were very appealing assignments. In other words: The students were willing to go the extra mile because they liked the project.

All but one wrote, that they enjoyed the course and that they profited much from it. The overall rating for the lab course is “good” (i.e. 4 points on a 1–5 point scale) despite the workload. Furthermore, the students indicated that they very much liked the research life cycle. Team work was also seen positively.

## 5 Conclusion and Future Work

A computer science lab course with a workload of ca. 150h can be used to teach research methodologies. Our experience shows that students can successfully apply a scientific research life cycle on given tasks including exploring the research subject, formulating hypotheses, and planing and conducting research. The Lego Mindstorms EV3 robot as target system worked very well. This tangible and interesting project setting as well as the challenging task of creating a natural language interface from scratch motivated the students intrinsically. The students liked working with the robot and they were able to literally “see” their progress. Interviews and personal feedback during the lab course show that the research life cycle was helpful for the students as it helped them organize. At first, work was often uncoordinated but by applying the research life cycle the quality improved.

We state that applying a research life cycle to a tangible project like creating a natural language interface for a robot is a good method to teach research. Our assumption is supported by the participants’ personal feedback. All interviewees understood the meaning and purpose of the research life cycle. They liked the guidance provided by the cycle’s well defined steps and experienced that their results improved when they sticked to it. One interviewee said, without the research life cycle one would not think about research during the lab course because one would focus on the task rather than the methodology. On the other hand, the methodology helps in fulfilling the task.

The interviews revealed that we successfully taught the research life cycle. All participants we have interviewed could (at least partly) remember aspects about the research life cycle like its different steps and their meanings for research half a year after the lab course. Furthermore, two participants intuitively used the research cycle during our interviews to solve a given problem.

The lab course helped waking the participants’ interest for research. In the interviews, three out of four interviewees agreed with that statement and said they would certainly attend another research related lab course and aim at a research related job. The participants liked the project and the process used. The challenging tasks were seen as motivating. They liked the sense of achievement they got when they solved them.

During the lab course, the participants showed interest for research methodologies. Three out of four interviewees said they learned much about research and methodology they did not know before. The application of the research life cycle improved over time as they used the methods more and more properly. Thus, we think we lowered the bar for getting into research.

The proper composition of a team determines whether a project succeeds or fails. For the next lab course, we plan to optimize team composition. Everybody

tends to behave in a particular way when working in a team and has one or more so-called team roles. A mix of people with different team roles improves team performance. To improve the effectiveness of our student teams, we want to use personality tests like the Belbin test [5]. To assess students' programming skills and their willingness to quickly familiarize themselves with new techniques, we will carry out a pretest. In the first session, we will introduce the students to the topic. Between sessions one and two, the students can familiarize themselves with the topic. In the second lesson, they will have to individually solve a given problem. By combining the results of the pretest with the results of the personality tests, we expect better balanced teams.

Furthermore, we plan to use agile methods in our next lab course like retrospectives and sprint reviews. In addition to the final presentation, each team will have to give a ten-minute presentation for each exercise in the sprint reviews. Thus, the teams can present their solutions to the other groups. Sharing insights should help all groups improving their solutions for the final assignment.

## References

1. Anderson, L.W., Krathwohl, D.R., Bloom, B.S.: A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. Longman (2001)
2. Bagnall, B.: Maximum LEGO EV3: Building Robots with Java Brains. Variant Press, September 2014
3. Barnett, R.: Reshaping the University: New Relationships Between Research, Scholarship and Teaching. McGraw-Hill Education, UK (2005)
4. Beckman, K., Coulter, N., Khajenoori, S., Mead, N.R.: Collaborations: Closing the industry-academia gap. *IEEE Softw.* **14**(6), 49–57 (1997)
5. Belbin, R.M.: Team Roles at Work, 2nd edn. Butterworth-Heinemann, Amsterdam; Heidelberg (2010)
6. Budde, M., Grebing, S., Burger, E., Kramer, M., Beckert, B., Beigl, M., Reussner, R.: Praxis der Forschung: Eine Lehrveranstaltung des forschungsnahen Lehrens und Lernens in der Informatik am KIT. In: Neues Handbuch Hochschullehre (NHHL). No. A 3.19, February 2016
7. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M.A., Saggion, H., Petrak, J., Li, Y., Peters, W.: Text Processing with GATE (Version 6) (2011)
8. Daun, M., Salmon, A., Tenbergen, B., Weyer, T., Pohl, K.: Industrial case studies in graduate requirements engineering courses: the impact on student motivation. In: 2014 IEEE 27th Conference on Software Engineering Education and Training, pp. 3–12, April 2014
9. Healey, M.: Linking research and teaching: disciplinary spaces. In: Barnett, R. (ed.) Reshaping the University: New Relationships Between Research, Scholarship and Teaching, pp. 30–42. McGraw-Hill/Open University Press, Maidenhead (2005)
10. Koolmanojwong, S., Boehm, B.: Using software project courses to integrate education and research: an experience report. In: 2009 22nd Conference on Software Engineering Education and Training, pp. 26–33, February 2009

11. Landhäußer, M., Weigelt, S., Tichy, W.F.: NLCI: a natural language command interpreter. *Autom. Softw. Eng.*, August 2016
12. Richardson, I.: Preparing students for software engineering research. In: 20th Conference on Software Engineering Education Training, pp. 367–367, July 2007
13. Wohlin, C., Regnell, B.: Achieving industrial relevance in software engineering education. In: Proceedings of the 12th Conference on Software Engineering Education and Training, pp. 16–25. DUZ Verlags- und Medienhaus GmbH, March 1999



# Teaching Robotics for Computer Science Students

Vesna Kirandziska<sup>(✉)</sup> and Nevena Ackovska

Faculty of Computer Science and Engineering, Skopje, Macedonia  
{vesna.kirandziska, nevena.ackovska}@finki.ukim.mk

**Abstract.** Creating a good curriculum for any course is challenging. This is especially true for courses that cover wide area of interest, courses that are supported by hardware components, or the courses that evolve rapidly, as technology evolves. A course that encompasses these challenges is the course entitled Robotics at the first cycle of studies for computer science students. This paper presents one Robotics course that is customized for computer science students. The effect of the presented course on students' performances is given.

**Keywords:** Computer science education · Curriculum design · Educational robots · Robotics

## 1 Introduction

Robotics interleaves the science and the technology that deals with the design, construction, operation, and application of robots. Robotics is also recognized as an excellent way to introduce the students in science and engineering, but it is also used for multitude of other purposes. For example robots are used as educational tools to help students learn the basic concepts for programming, electronics, etc. [1]. Another example are researches that show the essential use of the robots in the education of specific target groups, for example, a group of autistic children [2].

Within the wide scope and use of robotics in education, the present article deals with the problem of teaching the fundamentals of robotics for undergraduate students. Robotics encompasses diversity of topics ranging from mathematics and physics, to mechanics, automatic control, electronics, and computer science. An adequate profile of a robotics course depends on the curricula to which it belongs [3]. The robotics course of interest in this paper is a course for computer science (CS) students and this affects the topics that should be included in the curricula. According to [4] designing and construction of robots are not usual topics of interest for CS students. On the other hand, ACM guidelines suggest that a robotics course for CS students should involve integrating sensors, actuators and software into a robot designed to undertake some task [5].

One more challenge in the Robotics course is that many robotics topics should not be presented to CS students in the same way they are presented to most of the other groups of engineering students. To give CS students the opportunity to acquire the knowledge embodied in the course, the pedagogical basis should be focused on engaging them in an in-depth study of the subject, using programming assignments that

complement the traditional lectures and problem-solving sessions. Programming assignments can be done in a simulation environment [4, 5], or can be implemented as hands-on laboratory work with robots [6, 7]. The programming assignments in the course presented in [8] are simulations done in MATLAB. In [6] where the robot e-Puck is used, all exercises can be done both using the Webots simulation environment and real world environment. The introductory robotics courses described in [4, 7] use the open source Tekkotsu framework that consists of a set of software modules designed specifically to accomplish the goals of the course. An alternative to this framework is ROS (Robot Operating System) which although designed for research, now is also used in education.

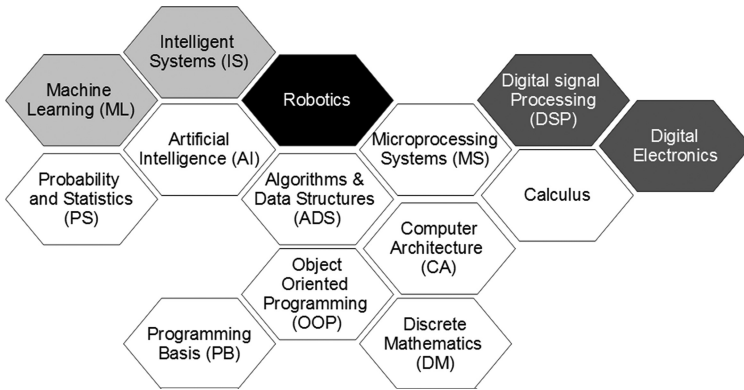
This paper is about the course entitled Robotics at the Faculty of Computer Science and Engineering (FCSE). In the following section the characteristics of robotics courses at FCSE are presented. In Sect. 3 the students' performances on the course is given and in the final section we conclude this paper.

## 2 The Characteristics of the Robotics Course

The Robotics course at FCSE is an elective course for students from the 6<sup>th</sup> semester. It is organized with 2 classes of theoretical lectures, 2 classes of theoretical exercises and 2 classes of laboratory exercises per week. The greatest challenge for us teachers of the Robotics course, was to get CS students at FCSE closer to robotics.

The course curriculum for the Robotics course was tailored in accordance with our students' knowledge and interests. Figure 1 shows some of the courses, taught at FCSE, that introduce knowledge needed for the Robotics course. As shown, prior to taking Robotics, students have taken several core courses in programming. These courses set a solid ground for a student to understand the algorithmic part of the robotics course. Many advanced algorithms important for programming more intelligent robots are introduced in the courses that are popular for our students. On the other hand more hardware oriented courses are available for our students as elective courses (marked in dark gray on Fig. 1), but these courses are not popular for our students. The only obligatory course that is hardware oriented is "Microprocessing systems". Considering the pull of knowledge, our CS students do not have the proper engineering skills and lack the hardware knowledge to be comfortable to work with robots. Our introductory Robotics course gives students the opportunity to learn the bases for using the robots as hardware machines in an understandable and applicable way. This problem was also a challenge for teaching the course "Microprocessing systems". However, it was improved drastically in the last 5 years, according to students needs, introducing gamification [9], and OER approach [10].

The pull of course knowledge given on Fig. 1 served as a guideline to choosing the topics that would be covered in the robotic curriculum in more detail, in contrast to the topics that would be only introduced, or perhaps left out for some more advanced courses. As a result, since 2011 until today several changes were done in the curriculum. First, the details for sensor data and signal processing were omitted from the course, since more was included in other obligatory courses ("Artificial intelligence" and "Microprocessing systems"). Another change in the curriculum was done in 2013



**Fig. 1.** Pull of course knowledge for the Robotics course. (black: Robotics; white: obligatory courses; light gray: popular non-obligatory courses; dark gray: non-popular non-obligatory courses).

when the lectures for robotic perception that included robotic vision and classification were excluded from the course because they had been included in other courses that our students usually take before taking the Robotics course (“Machine learning”, “Intelligent systems”).

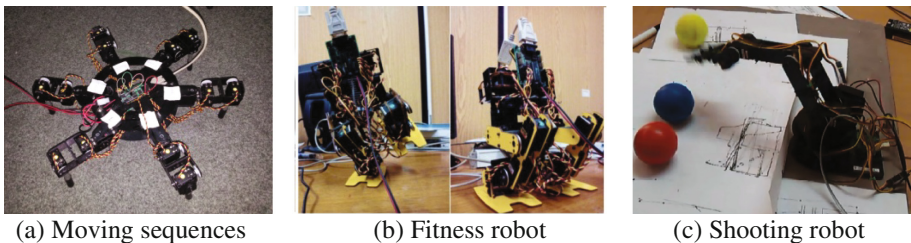
This helped to create a curriculum that was not overloaded with course material, but still the main topics were covered. On the other hand new topics in probabilistic robotics were added to the curriculum that would be new but easy for our students to follow since they have the knowledge of the course “Probability and statistics”. In addition to the standard Robotics topics, some very modern topics were included in the lectures in this Robotics course, such as Cognitive robotics, Emotional robotics etc. These topics introduce less known aspects of robotics that sparkle the students’ interest. The lectures and theoretical exercises were presented to students with traditional presentations.

Although our students are CS students and lack hands-on practice with hardware, the lab exercises seem to be a very important part of the course, that improves the students’ performance, as shown in the next section. Usually they are in a form of smaller programming assignments assigned weekly to students. A great part of the lab exercises are done in a simulation environment using Robotics Toolbox in MATLAB. All things taught on the theoretical exercises about robot manipulation i.e. kinematics, dynamics and path planning for robot manipulators can be exercised using this toolbox. A robot manipulator can be represented and shown visually in a simulation. Using the same toolbox, a vehicle and behavior can be simulated, so students are given vehicle control assignments in a given environment. One such a task is obstacle avoidance. The other labs are done on real robots (Lynx 5, Lynxmotion Biperd Scout, BH3-R Hexapod, Nao and Pioneer P3-DX). Students are given an assignment to program the robot in a real world environment. Although our students are great at programming and modeling, yet when dealing with real world bodies, most of the models have to be adjusted, and tested in order to get the robot to function properly. This was practically experienced by the students on one of these lab exercises.

### 3 Students' Performance

The students on this course are evaluated for their exam result (theoretical and practical exam), laboratory exercises and final projects. The project usually requires creating a more complex robotic application and it is done on actual robots. Projects are usually done in small groups of two or three students, and are graded based on the difficulty of the task, the success of the project and the involvement of each student in the group.

In all years on the Robotics course many great projects were done by our students. Here some examples, that give the general idea on the complexity of the projects, are presented. In several projects students have focused only on programming robotic movement appropriate to given actions. Actions were given by a human via a computer interface. For example in the “Moving sequences” project (Fig. 2(a)) several moving actions for the hexapod robot were programmed that enabled it to move straight ahead and to rotate left or right for a given angle. In more complex programs robots perceive the environment and act based on the information they require or based on their state. The “Fitness robot” was controlled by human speech commands. The sound signals were processed and then a word recognition algorithm was used for the robot to understand the commands. Each command expresses some specific task the Biped Scout robot should complete (Fig. 2(b)). In another project example done with Lynx 5 (Fig. 2(c)), the “Shooting robot” finds where the ball with the specified color is, based on the information from the camera, and then it shoots it.



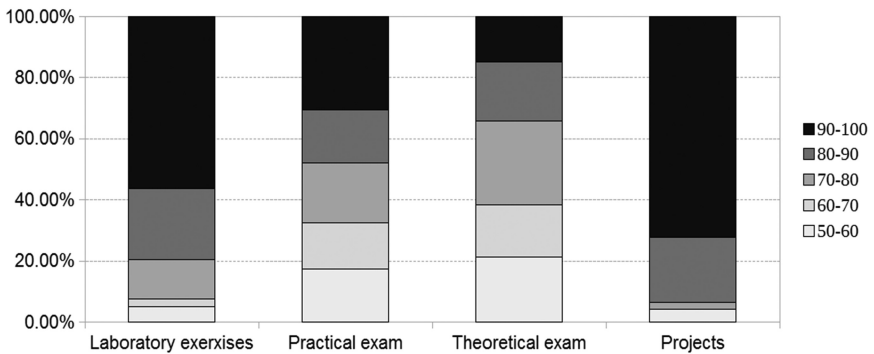
**Fig. 2.** Final projects in Robotics.

In some projects students have actually built their own robots. Examples of robots built by our students include the “Color detection machine robot”, “Trail tracking wheeled robot” and “Two-leg robot”. Most of these robots were controlled using Arduino controller, since it is cheap and easy to be acquired. Some students that have taken the Robotics course have done excellent diploma theses in the area of robotics. One interesting thesis was titled “Robotics as an assistive technology for autistic children” in which special exercises using NAO robot were created for the autistic children. The overall improvement of the children’s knowledge was reported in [2].

The number of students that enroll this Robotics course varies from 5 to 25. The percentage of students that have passed the Robotics course since 2011 is a bit above 80%. Only 5% of the students have not done enough of the activities given as lab

exercises and did not qualify for taking the exam. In order for a student to pass, (s)he has to complete the final project and pass the practical exam (score more than 50%).

From all parts in the course our students have scored the most points on their projects. In fact 72% of all students that have passed the course scored between 90% and 100% of the maximal points (Fig. 3). The students results on their projects is even more valuable considering the task challenge in creating a robotic application that works in a real world environment. The part of the course that was most difficult for our students, according to the scores, is the theoretical exam. Only 15% of all students scored more than 90% (Fig. 3). These results are also an indicator that the students like more the practical work, then the theory itself. The points students achieve in their lab exercises are correlated with the total points students have scored. The first justification of this correlation is that similar problems are given on the practical exam, so by doing the labs students have better chance to achieve greater score on the exam. Even more, the final projects require the knowledge students have gained on these exercises so that a real robot can be programmed.



**Fig. 3.** Total students' performance for each part of the course: laboratory exercises, practical exam, theoretical exam and projects.

In general, students have shown great results on the Robotics course. The average student grade is above 9 and the median grade is again 9. 48.9% of students that have passed have scored the highest score - 10. Of course, one could argue that this might be due to the fact that the course is elective, and only the most motivated students enroll it. This is true, and it can be shown that the students that enroll this course are among the best students in the Faculty. However, the severity of the tasks that the students have to accomplish, the time it takes for the students to finish their projects (compared to other courses in the Faculty) and the difficulty of the exam show that these students not only have to like what they are learning in this course, but should be able to fully understand the complex material that is taught. We believe that these results show that we have accomplished the goal of making a very acceptable syllabus of Robotics for CS students.

## 4 Conclusion

In this paper we elaborate on the characteristics of the Robotics course offered at FCSE. This is one of the few hardware orientated course taught to CS students and that brings some challenges. CS students are usually great in modeling and algorithmic thinking, but they face problems when dealing with embodiment and the challenges that arise from it. These challenges were guideline for creating a unique course syllabus that is customized for our CS students considering their pull of knowledge and interest.

The effects and success of this course is evaluated qualitatively and quantitatively. The best results of this course are the students' projects. We are proud of the final projects of our students, who have never previously been in contact with real hardware as robots. The overall result of the course and the average students grade for all students enrolled on the course is above 9 and this shows the extraordinary results of our students in this course.

## References

1. Balajia, M., Balajib, V., Chandrasekaranc, M., Ahamed khand, M.K.A., Elamvazuthie, I.: Robotic training to bridge school students with engineering. *Proc. Comput. Sci.* **76**, 27–33 (2015). IRIS 2015
2. Tanevska, A., Ackovska, N., Kirandziska, V.: Robot-assisted therapy: considering the social and ethical aspects when working with autistic children. In: *Proceedings of the 9th International Workshop on Human-Friendly Robotics - HFR 2016, Genova*, pp. 57–60 (2016)
3. Cappelleri, J.D., Vitoroulis, N.: The robotic decathlon: project-based learning labs and curriculum design for an introductory robotics course. *IEEE Trans. Educ.* **56**(1), 73–81 (2013)
4. Touretzky, D.S.: Seven big ideas in robotics, and how to teach them. In: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE 2012)*, pp. 39–44. ACM, New York (2012)
5. ACM/IEEE-CS Joint Task Force on Computing Curricula. *Computer Science Curricula 2013*. ACM Press and IEEE CS Press, December 2013
6. Correll, N., Wing, R., Coleman, D.: A one-year introductory robotics curriculum for computer science upperclassmen. *Trans. Educ.* **56**(1), 54–60 (2013)
7. Touretzky, D.S.: Computer Science Education: Robotics for computer scientists: whats the big idea?. *Comput. Sci. Educ.* **23**(4), 349–367 (2013)
8. Nourdine, A.: Teaching fundamentals of robotics to computer scientists. *Comput. Appl. Eng. Educ.* **19**(3), 615–620 (2011)
9. Ristov, S., Ackovska, N., Kirandziska, V.: Positive experience of the project gamification in the microprocessors and microcontrollers course. In: *IEEE Global Engineering Education Conference (EDUCON)*, pp. 511–517 (2015)
10. Ristov, S., Ackovska, N.: OER approach for specific student groups in hardware-based courses. *IEEE Trans. Educ.* **57**(4), 242–247 (2014)

# **Technologies for Educational Robotics**

# TUC-Bot: A Microcontroller Based Robot for Education

Sven Lange<sup>(✉)</sup>, Peter Weissig, Andreas Uhlig, and Peter Protzel

Department of Electrical Engineering and Information Technology,  
Chemnitz University of Technology, 09107 Chemnitz, Germany  
{sven.lange,peter.weissig,andreas.uhlig,  
peter.protzel}@etit.tu-chemnitz.de

**Abstract.** We describe the concept of a mobile robotics course for undergraduate students from an educational point of view in terms of learning goals, experiences, and hardware design. The course as well as the hardware was continuously improved over more than a decade. Hence, we like to describe our motivation and the current structure of the course in order to share our experiences as an inspiration for similar courses.

## 1 Introduction

Autonomous mobile robots are becoming increasingly important in our modern world with autonomous cars being the most prominent but not the only example. Especially, industrial and service robotics are a growing market as the capabilities of the robots increase. Hence, a society without robotics will be unimaginable in the near future.

Thus, robotics should be mandatory in engineering education, particularly for students of electrical engineering, information technology, computer science or similar fields. This has motivated us to continuously improve a course at our university focusing on autonomous mobile robotics which initially started in 2004. Originally, the course was inspired by a previous idea to interest high school students in the field of electrical engineering by organizing a robotics competition called RoboKing — see (Sünderhauf et al. 2006).

For the course at our university, we adopted the competitive character of the event to motivate the students. They have to team up in groups of two or three and solve the given task of programming a mobile robot to navigate through an unknown maze. For this, they have an overall time span of two semesters with an expected work load of 240 h. Most of the time, they have to work independently by self-organizing their team effort to come up with a final solution. Additionally, we guide them by theory lectures and pre-defined milestones, which are described in Sect. 4.1.

In the beginning of the course, suitable commercially available robots for the given task and with an educational focus did not exist, so we decided to design a custom-made mobile robot. Over time, it evolved and we currently use the third generation as described in Sect. 2.



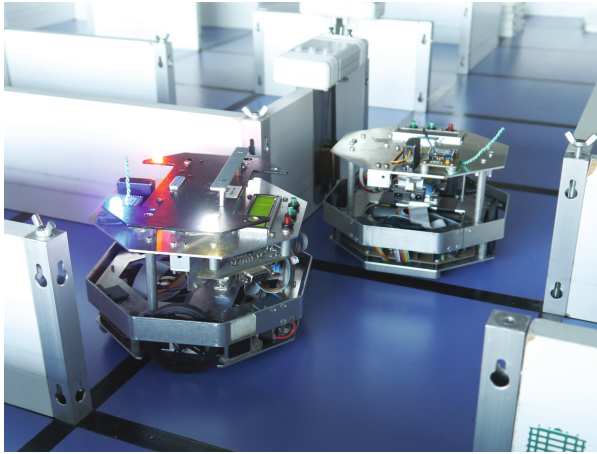
Now that the community has grown over the last years and also because of the big interest of hobbyists, there are many commercially available products, which may be suitable replacements for our custom-made robot. Examples are the E-Puck, Khepera, Thymio, 3pi, or TurtleBot to name a few. However, a problem with using popular commercial products is that solutions for nearly every imaginable task are publicly available and our students might be tempted to use existing solutions to solve the course problems. Hence, we decided to use our own robotics framework and enforce the pedagogical goal of understanding a previously unknown system. A more detailed description of our learning objectives follows in Sect. 4.

The idea to motivate and interest students in robotics by using competitions has been very successful, proven by the existence of various national and international robot competitions which found their way also into education. The authors of (Swenson 2015) for example, are doing a robotics course which follows a slightly different but still comparable approach. They describe similar experiences and name problem-solving as the students' main take-away skill, which is exactly the same impression we have. An interesting modification of the typical one-final-task-per-course style is given by (Cappelleri and Vitoroulis 2013). Their course follows a similar motivation as ours, but breaks down the idea of one final project task into several smaller ones to keep the motivation level up. We follow a similar approach but instead of varying the competition tasks completely, we use interim competitions as milestones toward the final goal which is described in Sect. 4.1. To increase the motivation, we reward the best teams from the interim competitions by giving extra points to be used as an advantage in the final competition.

Another concept, motivated by the thought, that many students feel the urge to experiment with a robot at home, is presented in (Aroca et al. 2013). They achieve a low-cost solution by using a common cellphone as the main processing unit. As an alternative to a conventional course, e-learning in combination with remote access to a real robotic system is especially interesting for a very large number of students — e.g. in (Kulich et al. 2013), the authors present such a concept.

## 1.1 The Task

The general idea of the course is a robot competition with two opposing robots autonomously navigating through a previously unknown maze (Fig. 1). Besides the exploration and navigation, there is a special challenge in locating eight so-called beacons and switching them to a team-specific state. The beacons have three states: neutral (initial state), red and green. The state toggles between red and green by pushing a button at the beacon. The first transition from neutral produces randomly either red or green. The button can be pushed by a robot by slightly colliding with it. The robots can sense the state via an IR transmitting diode, the humans see a red or green LED at the top of the beacon. The two opposing robots belong to the red or green team, respectively. Now the task of the green robot is to explore the maze, find the beacons and switch them to green.



**Fig. 1.** Two of the robots in the maze. The one in front is equipped with LEDs for external pose estimation (tracking) and in the background, aside from the second robot, a beacon can be seen.

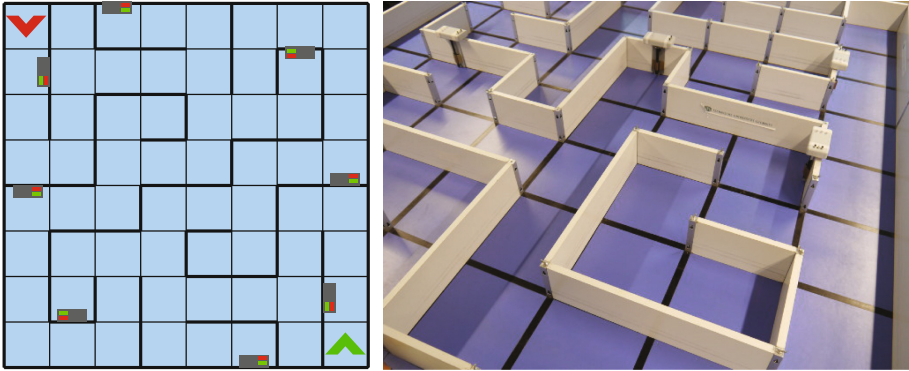
Simultaneously, the red robot tries to find and switch all beacons to red. Thus, each robot needs to revisit all beacons as often as possible and switch them back to their state if necessary. This task can be greatly facilitated, if the robot is able to build a map of the maze with all the beacons and can localize itself reliably within that map. Originally, we used this idea in a national robot competition for high school students called RoboKing, which is described in (Sünderhauf et al. 2005 Sect. III–IV).

As playing field, a flat blue surface with dimensions of  $2.4 \times 2.4$  m is used. It is divided by black lines into 64 quadratic fields of 30 cm length each. The robots can use these lines for orientation and localization. The maze itself is constructed and surrounded by white walls of 15 cm height which are always aligned with the black lines. Therefore, the quadratic fields are only separated, but never subdivided. The walls can be arranged in a very flexible way which enables us to create many different mazes. Since both opposing robots should have the same conditions, they start in opposite fields of a point-symmetrical maze. Figure 2 gives an overview of a possible maze.

A single game lasts up to 7 min. Afterwards the teams get points for each beacon switched to their respective color. If a team had interfered with its robot during the match, maybe because it got stuck, the referees subtract penalty points for each interference. Likewise, penalties up to disqualification may be imposed by the referees, if a robot collides on purpose with its opponent.

## 1.2 Target Audience

The target audience for our course are third year undergraduate students. They have previous knowledge of basic math, physics, computer science,



**Fig. 2.** Two views of different point-symmetrical mazes. *Left:* Schematic top view. The arrows represent the starting positions and directions of the opposing teams. The thick black lines symbolize the fixed walls and the black boxes mark the positions of the beacons. *Right:* Oblique view of our real maze. Easy to recognize are the blue surface, the black lines and the white walls. More difficult to spot are four white beacons in the upper part of the picture.

and microprocessor technology. The course in mobile robotics is the first larger project in their studies, which needs team-oriented self-organization and a large amount of applied thinking. As the courses in basic control theory and sensor technologies start at the same time as our course, we can not expect the needed prior knowledge in these subjects. Hence, we give introductory lectures to enable a smooth start into the interdisciplinary field of mobile robotics.

Besides the undergraduate students in electrical engineering, we also have high school students performing an internship as our second target audience. As they have much less time and less prior knowledge, we designed our system to cope with it and included a simpler API with more basic functions as well as additional sensors to solve a much simpler line-following task.

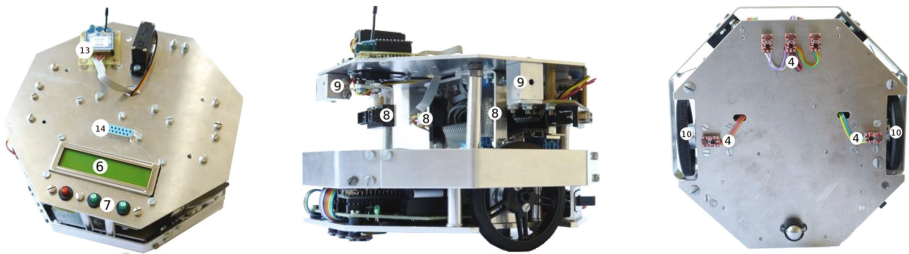
## 2 Hardware

The current version of our robot, as shown in Fig. 1 is called TUC-Bot with a chassis made of aluminum to withstand the students' harsh testing conditions. It is a differential drive robot, hence the sharp turns and dead ends within the maze can naturally be dealt with. Following this kinematics concept, we use two metal gearmotors with 6 V and a maximum current of 2.4 A each. Both motors are equipped with Hall effect sensors reading 48 counts per revolution (CPR) of the motor shaft. Consequently, by using wheels with a diameter of 60 mm, we have a resolution of about 9 counts/mm movement of the robot. Further, the configuration results in a maximum velocity of about  $0.5 \text{ ms}^{-1}$ .

Besides the wheel encoders, the robot is equipped with various sensors that are especially useful for the given task as described below.

Afterwards, a discussion of the user interface, the modular structure, and the maintenance requirements follow.

*Distance Sensors.* For obstacle detection — specifically, the walls and other robots — we equipped the system with three IR-distance sensors (see Fig. 3, no. 8 and Table 1 for the location and exact type). They give a point measurement based on triangulation and return a range-correlated voltage with high resolution starting from about 10 cm and decreasing resolution up to about 80 cm. For ranges shorter than 10 cm, the measurement is ambiguous, hence the two sensors pointing sideways are within the robot’s diameter in such a way that their lines-of-sight are crossing each other. In other words the sensor on the left side is sensing the right wall and vice versa. Additionally, both sensors are tilted slightly forward, which will improve the robot’s capability to follow and adjust its position to the walls. As a special feature the front sensor is mounted on a servo motor and can therefore be turned into different directions.



**Fig. 3.** Our TUC-Bot viewed from different perspectives (left: top-view; middle: left-view; right: bottom-view). 4: line sensors; 6: display; 7: buttons; 8: IR-distance sensors; 9: IR-receivers; 10: wheels; 13: radio transceiver; 14: internal connector for expansions

*Line Sensors.* As already described, the playing field is structured by black lines, which divide it into a grid of squares. If recognized by the robot, the borders may be used for rough position as well as orientation estimation. For this purpose, two line sensors are mounted beside each wheel (see Fig. 3, no. 4). These sensors measure the reflective properties of the ground — consequently, the black lines can be detected. Note, that three additional sensors are mounted in the front to enable the robot to do other tasks like simple line following. This is especially considered for high school students serving an internship.

*Bumpers.* In addition to the IR-distance sensors for collision avoidance, our robot is equipped with a surrounding bumper bar for collision detection. It is slidably mounted and centered with springs. Depending on the point of collision, one or two of the four evenly distributed switches is triggered.

*IR-Receivers and Beacons.* In the task description (Sect. 1.1) we mentioned the beacons. They are attached to the walls and emit a modulated 38 kHz infrared signal. If nearby, the robots are sensing the current state (neutral, red, green) of

the beacons through a modulated infrared signal with different on-off ratios with the help of directed IR-receivers. The directionality of the receivers is realized by putting them into a pin-hole case. Three of these sensors are mounted around the robot, see Fig. 3 no. 9.

*User Interface.* For programming and debugging of the robot, the various ways of user interaction are very important, especially for inexperienced users. This is why the robots are equipped with a display (no. 6), LED illuminated buttons (no. 7) and an IEEE 802.15.4 radio module (no. 13). While the display is intended for showing fast and simple status information, the radio module enables advanced logging and data analysis.

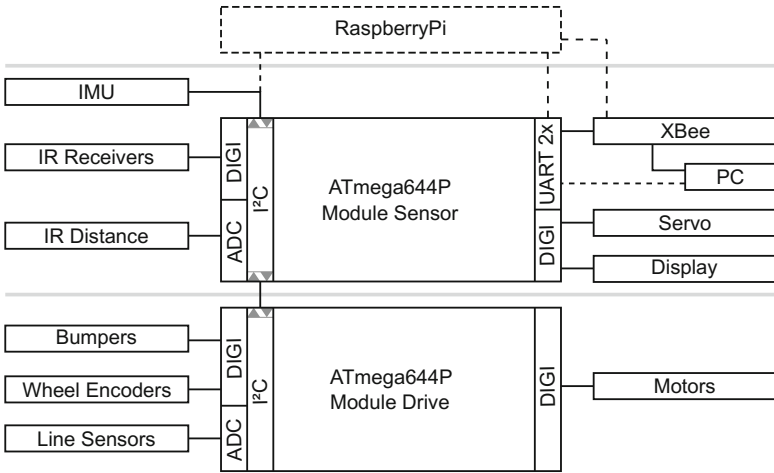
*Modularity.* The third generation of our robots is intended to be used not solely for the course at hand. Hence, we divided the robot's structure into modules, where the base module includes motors, encoders, power supply, etc. as shown in Fig. 4. It has its own preprogrammed microcontroller and provides an I<sup>2</sup>C interface for basic driving functions. This module can be extended by the additional *sensor module* or used in a direct way e.g. in combination with an embedded PC like the *RaspberryPi*. In this version of the course, we use the sensor module as the only extension. It provides additional sensors and a microcontroller to be programmed by the students.

*Maintenance requirements.* Currently we have five robots available for the students and 7 PC workplaces, which need constant maintenance effort. At the beginning of the course, creating new user accounts and other administrative tasks is mostly automated and executed by one of two technicians. This is currently the most time consuming part. Consequently, some time is needed by a second technician to do minor repairs like tightening a loose screw or replacing a broken cable. The otherwise robust hardware is the result of the lessons learned from the previous generations of the robot, which will be described based on examples within the next section. The workload of both technicians together should not be more than 40 h within a year. Of course, this depends a bit on the number of participants which has been fluctuating between 15 and 30 students over the years.

## 2.1 History

The TUC-Bot presented here is the result of a continuous process, which lasted over 10 years. Compared with its predecessor, see (Sünderhauf et al. 2006, Fig. 9), we made several improvements. As the modifications may be of general interest for similar projects, we discuss the most relevant ones following the principle: Old component, problem, solution.

*Servo motors with optical encoders:* Especially the encoders were prone to errors due to scratches. Additionally, the wheels' axis needed a ball bearing as a second support, which sometimes led to unwanted mechanical tensions. Our *solution* was



**Fig. 4.** The robot’s architecture showing the two microcontrollers connected to the various sensor and actor components. The lower part illustrates the components of the preprogrammed base, extended by the second microcontroller with its components in the middle part. Finally, an embedded PC can be placed on top to extend the programmable module or even replace it by using the I<sup>2</sup>C communication.

exchanging those components with robust metal gear motors with internal Hall effect encoders.

*Bumper bar:* the previous construction of the bumper bar was split into a front and a rear bar. Both were directly coupled to their microswitches without further mechanical support. As a result the switches were worn out rapidly and it was possible to rip the bar off during crashes. Now the bar is a closed ring-like construction hold by the chassis.

*IR-distance sensor arrangement:* Previously we used only two sensors mounted on servo motors, in the front instead of three sensors. This directly led to the problem that at least one sensors must be turned around all the time to detect walls either in front or side. This resulted in more complex and prolonged measurements.

*Communication:* The previous ways of communication were limited to a wired serial interface. Especially online debugging and logging of sensor information in this way was not reasonable. Now, with the radio module, advanced debugging features are possible.

### 3 Software-Library

Since we want the students to program an autonomous mobile robot to perform some demanding high-level tasks, it would be counterproductive to let the students program the robot from scratch at the lowest level, especially since

**Table 1.** The robot’s main components result in an overall cost of about 400€.

Component	Description	Costs
2x Microcontroller	Chip45 Crumb644 (ATmega644P)	50€
3x IR-Distance Sensor	Sharp GP2-1080K (10 cm to 80 cm)	20€
3x IR-Receiver	VISHAY TSOP 31238	5€
5x Line Sensor	Pololu QTR-1A Reflectance Sensor	15€
1x Radio Transceiver	XBee (2.4 GHz; Series 1; PCB-Antenna)	30€
1x Display	LCD (2 × 16 symbols; backlight)	15€
2x Motor with Encoder	Pololu 34:1 Metal Gearmotor with 48 CPR Encoder	75€
1x Motor Driver	Pololu Dual MC33926 Motor Driver	30€
1x Battery Pack	6 Cells; NiMH; 7.2 V; 1900 mA h	30€
Miscellaneous	PCBs, Mechanical Components, other	130€

we can not expect previous knowledge in basic microcontroller programming, as stated in Sect. 1.2. Therefore, we developed a software library for our robot, which implements basic functionalities. They are provided as a documented API and include simple functions for getting sensor information or passing commands to the actuators without the need to know on which microcontroller pin they are connected to or how to use e.g. the ADC. Nevertheless, we provide this information and every student has the freedom to implement everything from scratch.

Providing the right amount of such predefined library functions is also controversial. Thus, we first defined the learning objectives and then determined the necessary functionalities to achieve our objectives. For example, implementing a speed control algorithm is part of our learning objective, so we do not have any predefined functions for controlling the speed of the robot’s wheels within our library — only a function for setting the voltage.

In addition to the library and its documentation, we provide sources for relevant datasheets, literature, and other information. Furthermore, every computer workplace is preconfigured with the needed software libraries and an editor including the makefile directive to enable a seamless start for the students in an Arduino-like style.

## 4 Learning Objectives

In general, there are multiple learning objectives present throughout the two semesters of the course, which will be described in the following.

Omnipresent challenges in programming — not only mobile robots, but technical systems in general — are the error susceptibility, imprecision, and individual variations within hardware components like sensors and actuators. One could say that this is one of the main insights the students should take away from this

practical course: Algorithms which work in theory or simulation will not necessarily work with the real hardware. Of course, this is not the only thing to learn. The following general learning objectives describe some of the desired competencies in terms of *The students will be able to*:

1. explain the components' working principles of the previously unknown microcontroller-based mobile robot.
2. analyze a complex task and split it into several sub-tasks like processing sensor data, implementing controllers, mapping, etc. while self-organizing their activities in the time available.
3. implement, modify, analyze, and create microcontroller-based algorithms in C/C++ in consideration of the limited resources of an 8-bit microcontroller.
4. record and analyze sensor data for the purpose of debugging and identification of programming errors or a deeper understanding of the sensors' characteristics and working principles.
5. explain reasons for measurement outliers and select suitable methods to cope with them.
6. explain the concept of a differential drive robot.
7. demonstrate a working system which is solving the given task of navigating through the maze and present their specific implementation details by giving a short oral presentation.
8. use tools like subversion to develop and manage source code in a small team.

#### 4.1 Structure of the Course

We support the above stated learning objectives by additional lectures, guided practical work, software, and other material. As the time for complex topics like digital control theory is not nearly sufficient, most of our lectures are of a more practical type with theory explained on the example of the given mobile robot. A more complete theory is then given in subsequent courses, depending on the student's choice of specialization. In the following, we like to give an overview of the structural design of our course, which is the result of continuous improvement over the last years.

*Workflow and Subversion.* We start the course with team building and proceed with a first introduction into the workflow. This includes an overview of the robot's components and their basic working principle. Additionally, all the tools and given materials are presented and programming the robot is demonstrated, so the students could start right away with programming. Surprisingly, most of the students have no prior experience with any version control system. As this is important for team work, we also give an introduction into version control, especially subversion.

*Control Theory.* An important step in navigating the robot through the maze is the ability to drive straight. Many students without experience in mobile robots do not regard this as a problem, but this is the first issue where theory (all motors



of the same type are identical) and practice (manufacturing tolerances) diverge. Providing the same voltage for each motor will lead to a curved trajectory of the robot because the individual variations within the motors result in different, non-linear characteristic curves. In consequence, wheel encoders have to be used to control the velocity of each wheel to a specific velocity. The needed control theory for implementing a cascaded velocity controller is given within one lecture. Here, cascaded means two individual velocity controllers for each wheel and an additional controller on top to control the error between the two wheel velocities.

*Visualization and Debugging.* In a scientific approach, it is quite natural to analyze a system by recording and plotting the available sensor data. For undergraduate students, this does not seem so natural and has to be learned first. We made the experience, that if the students do not know the exact way to record and plot the data, most of them will not bother and try to circumvent it. Hence, we saw the necessity to do an example-based step-by-step introduction regarding the communication flow and the tools to use for data visualization. We present two toolchains: first an open source solution using a simple python script for serial communication in combination with gnuplot for visualization, and second, a single tool solution by using Matlab for both — communication and visualization. We conclude our lecture with a simple practical task, where the students have to apply the just learned theory on the robot. Simultaneously, we get feedback by assisting the groups. In this way, we often see weaknesses, which optionally could be addressed in further lectures.

*Guided Practical Work.* Parallel to our lectures, the students have to do guided practical work in the first two month of the course, before they can start with their own ideas to solve the competition's task. The guided work is composed of three individual guided task descriptions for getting to know the characteristics of the motors as well as the distance sensors and implementing a controller for the wheels. We introduced this step to familiarize the students in a guided — and therefore faster — way to the robot's properties. Of course, in a project like course, this should not be necessary at all, but our experiences showed that especially groups with low previous knowledge are appreciating this and as a result get more motivated.

*Interim Evaluation.* Unfortunately, students tend to underestimate the work to do for a given task. With regard to this course, it leads to bad results in the final competition and weak learning outcomes because all the work is done within the last two weeks before the final competition. To counteract this behavior, we introduced sub-tasks as milestones with an interim evaluation. They produce no additional workload, because the implementations necessary for solving them are a required stepping stone for the final goal.

In the first evaluation, the robot has to move along an imaginative square of about 2 m in dimension. Mainly odometry and some logic for moving straight has to be implemented. Next, the robots have to navigate through a simplified maze in the fastest way they can. This mainly requires using the distance sensors as

well as the line sensors to implement a navigation strategy. The third evaluation consists of the maze including the beacons but without an opponent. Winner is the team who activated the most beacons within the shortest time.

As well as the final demonstration, these sub-tasks are carried out in a competition like manner. By giving points to the best three groups which can be used as advantage in the final competition, we motivate good performance.

*Oral Presentations.* Even though the course is based on a competition, the students are required to present their findings and experiences after each evaluation task by giving a short presentation. This fosters the mutual information exchange and is an opportunity to improve their presentation skills.

## 5 Continuous Improvement

As mentioned before, we guide our students in their understanding of the basic hardware, especially the sensors. Based on this knowledge, they develop the control algorithms and the overall decision logic and robot control architecture. This architecture is usually either reactive or plan based. The reactive architecture is very simple as it only needs to consider the current state of the robot. Therefore, it is easy to implement and can be tested quickly. The plan based architecture is more sophisticated and complex as it additionally incorporates previous knowledge. For example, the robot creates its own map of the maze with the beacons while exploring. Afterwards, it uses this information for navigation and path planning to drive to the next beacon(s) or for self-localization. As much as we would like the students to implement more sophisticated plan based logic, most of them stop at simple reactive behaviors, because it is already sufficient for the final competition. Admittedly, these simple solutions are often more robust and actually increase the performance in the competition. Nevertheless, there are always some very motivated students who want to explore the limits of their control architecture and try out the advanced methods which we also introduce. Since we do not enforce a particular method, we always see many different approaches in the final competition, often with surprising results.

We are planning various extensions and updates of our current robot hardware and software structure. In the following, we discuss three such ideas.

*Simulation and Reality.* Typically, the students differ in their degree of motivation as well as learning speed. Consequently, some students are overstressed and some are subchallenged. This is partly compensated by the project nature of the task and leads to very different solutions ranging from implementing only rudimentary functionality to very complex algorithms including mapping and path planning. Primarily to support the motivated and subchallenged students, we replicated the maze and the robot within the V-REP simulation. In combination with a Matlab wrapper with the same naming conventions as in our library, it is possible to test algorithms for the robot. This has the charm of circumventing possible hardware flaws and develop algorithms in a rapid-prototyping style.

Additionally, we can exchange V-REP with the real robot to have a hardware-in-the-loop structure, where the robot accepts commands from Matlab and returns its sensor information.

*Extended Sensor Information.* Depending on the quality of the sensor information available, programming the robot is more or less challenging. Hence, if the students have to spend much of their time debugging unreliable sensor information, there will be less time for higher functions like mapping and path planning. On the other hand, if the sensor readings are very accurate and reliable, the learning focus will entirely move away from processing sensor data and relating low-level tasks which are still part of the learning objective. Hence, a balance between both extremes has to be found.

Currently, we have two options for extending sensor information. We recently added an IMU, basically for the gyroscope values around the z-axis. This is a good way of giving the students reliable information regarding the robot's orientation, which is otherwise hard to get — odometry and line sensors are possible ways, but get unreliable after collisions with walls within the maze. Even if the orientation measurements are calculated by integration of the turn rate, tests showed only little drift of about  $5^\circ$  after 5 min movement within the maze. This could be fixed by using the maze's grid lines and the robot's line sensors. In contrast, fusing global information from a magnetic field sensor is unreliable because of the maze's metal foundation.

Besides using an IMU, an external measurement system would be another option for position and orientation estimation. Therefore, we added a camera with fisheye lens above the maze and mounted LEDs with bright illumination on top of the robots. By using the same principle as in (Lange and Protzel 2012), we can determine the robots' poses and orientations with sufficient accuracy. This works well in controlled lighting situations as the system is sensitive to external illumination sources. On the other hand, this system with its global position and orientation information might simplify the task too much which would impair our learning objectives. For this reason, we only used it e.g. for system identification purposes but did not make it available to the students as an additional measurement system.

*Higher Algorithms.* Due to the positive feedback from the students and the good learning outcomes, we plan to offer an advanced course with the same project-like character, but with the focus on advanced algorithms. In consequence, a simple microcontroller as the only processing source is insufficient. So we already experimented with an additional Raspberry Pi embedded PC as visualized in Fig. 4. There are three possible ways for connecting the PC: by using the TWI, or serial interface, where we can address the serial interface either directly on the robot or via the XBee module. As a proof-of-concept, a student developed a simple example of following a red ball with the robot using a camera. For educational use, this was done in three versions with ROS as a base. The actual example solution was realized in Matlab with help of the Robotics System Toolbox, in Python, and in C++ using the OpenCV library.

## 6 Conclusion

We described in detail our motivation and structure in teaching the basics of mobile robotics with a project-based approach. We gave an overview of our hardware and software concept in conjunction with the intended learning goals of an introductory third year undergraduate course in mobile robotics. By following an competition-like approach, we motivate the students to choose additional lectures in this field.

Even though the overview of the course's structure may be obvious to experienced tutors, it should be useful for those creating a new practical course on mobile robots. The course benefits electrical engineering students by challenging them to overcome their software engineering deficiencies as well as computer science students who have never seen a PID control algorithm. In addition to those hard skills, the course is also a good opportunity to sharpen the soft skills in organizing their team work, giving presentations, and learning about new methods like subversion or other suitable agile methods as described e.g. in (Gerndt et al. 2014).

## References

- Aroca, R.V., Gomes, R.B., Tavares, D.M., Souza, A.A.S., Burlamaqui, A.M.F., Caurin, G.A.P., Goncalves, L.M.G.: Increasing Students' Interest With Low-Cost Cell Bots. *IEEE Trans. Educ.* **56**(1), 3–8 (2013). doi:[10.1109/TE.2012.2214782](https://doi.org/10.1109/TE.2012.2214782)
- Cappelleri, D.J., Vitoroulis, N.: Project-based learning labs and curriculum design for an introductory robotics course. *IEEE Trans. Educ.* **56**(1), 73–81 (2013). doi:[10.1109/TE.2012.2215329](https://doi.org/10.1109/TE.2012.2215329)
- Gerndt, R., Schiering, I., Lüsse, J.: Elements of scrum in a students robotics project: a case study. *J. Autom. Mob. Rob. Intell. Syst.* **8**(1), 37–45 (2014)
- Kulich, M., Chudoba, J., Košnar, K., Krajník, T., Faigl, J., Přeučil, L.: SyRoTek—distance teaching of mobile robotics. *IEEE Trans. Educ.* **56**(1), 18–23 (2013). doi:[10.1109/TE.2012.2224867](https://doi.org/10.1109/TE.2012.2224867)
- Lange, S., Protzel, P.: Cost-efficient mono-camera tracking system for a multicopter UAV aimed for hardware-in-the-loop experiments. In: *Proceedings of International Multi-Conference on Systems, Signals and Devices (SSD)* (2012). doi:[10.1109/SSD..6198047](https://doi.org/10.1109/SSD..6198047)
- Sünderhauf, N., Krause, T., Protzel, P.: Bringing robotics closer to students - a three-fold approach. In: *Proceedings of International Conference on Robotics and Automation (ICRA)* (2006)
- Sünderhauf, N., Krause, T., Protzel, P.: RoboKing - bringing robotics closer to pupils. In: *Proceedings of International Conference on Robotics and Automation (ICRA)* (2005). doi:[10.1109/ROBOT.2005.1570774](https://doi.org/10.1109/ROBOT.2005.1570774)
- Swenson, J.: Examining the experiences of upper level college students in 'introduction to robotics'. In: *Proceedings of International Conference on Robotics in Education (RiE)* (2015)

# Open Source Robotics Course at Engineering: Infrastructure and Methodology

Francisco Martín<sup>(✉)</sup>

Rey Juan Carlos University, Fuenlabrada, Spain  
francisco.rico@urjc.es

**Abstract.** This paper describes the characteristics and methodology of a course of programming mobile robots for an Engineering course. We will describe the robot we have built for the classes, as well as the use of gamification and motivation techniques through social networks as teaching techniques. In addition, our group is committed to the open source, so we will use ROS [9] (Robotics Operating System) as the underlying technology. We will show how the motivation of the students and the exploitation of the course have improved. As a final result of this process, several students in this course got involved in the Participation in RoboCup 2016 held in Leipzig.

**Keywords:** Teaching · Robotics · Open source · Gamification · Social networks

## 1 Introduction

Mobile Robotics is an area of knowledge within Robotics present in careers related to Mechanical Engineering, Electronics and Computer Science. This area focuses on the construction and programming of robots that move around to perform tasks. This discipline includes behavior generation, perception, map building, self-Localization and navigation, mainly.

Robotics is one of the technologies that will mark the 21st century. The robots begin to leave industrial environments and begin to populate our homes, cities and roads. Robots that clean our house already exist, that distribute the mail in offices or that circulate by our streets in the form of autonomous vehicles. And everything is yet to come. Many people are worried that this will take away jobs that robots will do from now on. We, on the contrary, believe that it is an opportunity for humanity. Many dangerous or tedious jobs will be done by robots while a great demand will emerge in highly qualified professionals to design and program these robots. Not only do we believe this, but also bodies such as the European Union<sup>1</sup>.

The demand for professionals in Robotics has already begun to grow at a great rate. For this reason, universities are starting to open more robot programming careers, which join the robot-building careers that already exist today.

<sup>1</sup> [https://ec.europa.eu/epsc/publications/strategic-notes/future-work\\_en](https://ec.europa.eu/epsc/publications/strategic-notes/future-work_en).

It is necessary to cover the need for experts in Artificial Intelligence, Machine Learning, Deep Learning, autonomous vehicles, ubiquitous systems or the new man-machine interfaces. There are many degrees that already include in their curricula courses Robotics, being an area so transversal to many degrees in Mechatronics Engineering and Computer Science.

In our case, we taught a course in Mobile Robotics in a Telematics Engineering. At the beginning, our students do not consider this subject as a central part of their career, but experience has shown us that, from this career, many of them orient their professional future to Robotics.

In this article we want to describe how we have managed to make this subject attractive to students, and how we provide them with the necessary knowledge to participate in this revolution in the world of technology. This description is also intended to be a reference for similar courses. We will describe the design of the robotic platform we use, the content and motivation techniques we have used: gamification and use of social networks.

Gamification in teaching consists in using techniques, elements or dynamics of games to enhance motivation and attention in students, in order to improve the acquisition of knowledge of the subject taught. This term [2] has its origin in the digital media industry, around 2008 [4], although it was from 2010 when it was widely used [1,8]. Applied to teaching, this technique can get students to engage, motivate, concentrate and strive to participate in activities that could previously be classified as boring, and with this technique it can be seen as creative and innovative [6]. Gamification is a very important element within b-learning [7], or blended learning, where classroom teaching is combined with teaching online.

The use of *microblogging*, especially Twitter, applied to higher education has proven to be an innovative element that enhances the acquisition of knowledge, mainly promoting debate and discrimination of information [3,5]. There are not many works that show how the use of this tool motivates the students when presenting their results to an interested public in their field. Our thesis is that students can be motivated to perform good practices if they can show them, making them reach the technical public or even potential future employers.

Our group is committed to Open Source. For this reason, all the software used in this subject is based or integrated in ROS. ROS is the current standard in the world of Robotics since its creation in 2006. ROS is a set of libraries and tools to build Robotic Software. In addition, it defines a set of methods and practices to be able to use the software that is already in a standard ROS distribution, and to build new modules that others can use, creating a community around this project. Robot manufacturers now create their drives ROS-compliant, and implementing algorithms with ROS interfaces ensures they can be easily used by thousands of roboticists. In addition, it is an extra element of motivation for our students, as they perceive that they are learning a tool that is a requirement to be incorporated professionally into Robotics.

This article is structured as follows: In the Sect. 2 we will describe the robot design used in the course. In the Sect. 3 we will review the contents of the course

and show how we applied the principles of motivation described above. In the Sect. 4 we will show an analysis of the impact of our approach and, finally, in the Sect. 5, we will provide the final conclusions.

## 2 Low-Cost Open Source Robotic Platform

Our course is not focused on the construction of robots, but on the programming of mobile robots. For this reason, students begin with several robots already assembled and ready to be programmed. These robots have the necessary characteristics to practice all the knowledge that the course contributes. When we designed the platform we established some basic requirements:

- All components of the robot must be fully supported in ROS, both in their real version and simulated in the simulator Gazebo<sup>2</sup>.
- The dynamics of the robot must be stable and simple, and compatible with the navigation software of ROS.
- The robot must have the sensors needed to make 2D maps and navigate its surroundings.
- It must have a camera to perform image processing, both 2D and 3D.
- It must be a platform whose cost allows to have several robots.
- It should be easy to use.

After analyzing several options, we decided to use a mobile platform Kobuki<sup>3</sup>, also known as Turtlebot 2. It is a platform with a differential locomotion. It can mount a structure of platforms that allows to place sensors, actuators and computers, up to 5Kg. It is fully supported in ROS<sup>4</sup>, both the real platform and the simulator. It does not have a computer on board, so it is connected to a conventional computer via USB.

In two of the 4 available robots we have used a low-cost RPLidar<sup>5</sup> laser sensor with a 6 m and 360° radius of perception. In each of the other two robots we have used a Hokuyo laser which we were already available. In addition, we have equipped an Asus XtionRGBD camera<sup>6</sup>. Both sensors are powered by USB.

It is very common to use this mobile platform by placing a laptop on top. In fact, in previous courses we have done so. It has the advantage that each student does the practices in his/her computer and directly executes them connecting it to the robot and the sensors by USB ports. Even so, the disadvantage is that you have to leave the upper platform free to be able to place the computer, and it is not very stable, with risk of computer crash. In addition, It is not very comfortable to debug the programs when the computer is in top of the robot, as shown in Fig. 1.

<sup>2</sup> <http://gazebosim.org/>.

<sup>3</sup> <http://kobuki.yujinrobot.com/>.

<sup>4</sup> <http://wiki.ros.org/kobuki>.

<sup>5</sup> <http://www.slamtec.com/en/Lidar/A1>.

<sup>6</sup> [https://www.asus.com/en/3D-Sensor/Xtion\\_PRO/](https://www.asus.com/en/3D-Sensor/Xtion_PRO/).



**Fig. 1.** Last course robotic platform.

For the current course we equipped the robot with a mini computer Intel Nuc, which acts as a robot controller. This mini computer has an Intel i5 processor, 500GB of solid state hard drive and 8GB of RAM. Another option was to put a Raspberry Pi 2 Model B, but the bandwidth it provided for a 3D image stream from the USB port was not acceptable for any image processing application. With the i5 mini computer we guarantee a good transmission rate and many resources to run software on board. We installed a distribution of GNU/Linux Ubuntu 16.04 LTS and ROS Kinetic. Another decision we have made is that each robot has its own network, through a Wifi router. This allows one or more students to connect via wifi or ethernet to the robot. Finally, we have expanded the robot's batteries to double its standard configuration to guarantee autonomy for several hours.

This configuration allows each student to have multiple options to execute their practices:

1. The student can connect to the on-board computer via ssh. In remote mode you can edit the program code, compile and test it.
2. Another option is to perform a ROS configuration to run some nodes on the robot (Kobuki and sensor drivers) and the rest of the nodes on the student's own computer. In this way, each student has their own programs on their computers without having to copy them at the beginning of the class and delete them from the robot's computer at the end. In addition, during the class the same computer can be shared by several students and has the option of connecting the router to an Internet connection to update its software, or even for students to consult documentation on the Internet.

The connection scheme of the robot elements are shown in Fig. 2. The front interface of the robot is shown in the lower part. The 19V, 2A power connection is only active when the robot is charging its own battery, and is designed to charge the battery of and hypothetical laptop, which we are not going to equip. The power connection of 12V, 5A is enough for the mini computer on board, powering USB to all sensors. The 12V, 1.5A connection feeds the router. The last connection of 5V, 1A is only used in the case of having a Hokuyo laser, which needs a separate power supply. Figure 3 shows the already assembled robots.



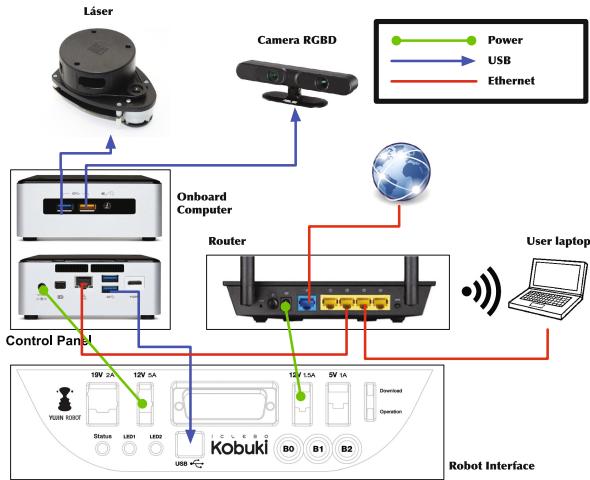


Fig. 2. Connection scheme of the robot.



Fig. 3. The four robots at present.

### 3 Course Content and Teaching Methodology

This course is focused on mobile robot programming. Mobile Robotics is an area within Robotics that deals with the problems that arise when a robot moves around its surroundings. The classic problems of mobile robotics are perception, localization, mapping, navigation and behavior generation. In addition, many authors consider that, among these problems, also the human-machine interface and the manipulation. We are not going to consider them in our course because of the time limits, and because we consider that they can be specific depending on the application of the robot. The content of the theoretical part of the course includes:

1. Sensors and actuators common in robotics.
2. Reactive behaviors formed by simple rules that are generated due to simple sensory stimuli. This theme includes PID controllers and visual control from image processing.

3. Construction of maps using 2D and 3D sensors.
4. Classic navigation techniques: VFF (Virtual Force Field) and Gradient Descent.
5. Probabilistic self-localization algorithms: Monte Carlo, Markov and Kalman.

In the practices of the course we put into practice two teaching methodologies: Gamification and the use of social networks. The practices are organized in a set of competitions in which the students participate. The practice statement is the rules of each competition, which specifies the time limits, what the robot should and should not do, how points and rounds are achieved in the championship. Each competition has objectives and concepts that are described below:

- **Move to the end of a corridor.** The rules of this competition are simple: The robot must use its contact sensors to advance to the end of a corridor delimited by walls. Scoring exclusively depends on the time spent to reach the goal. The competition was structured by an initial phase and a final phase, to which the 4 students with the best points are classified. In the initial phase there were two rounds, with an interval of one hour to allow the participants to improve their software if in the first round they had detected some error or possible improvement. This is common in international robotic competitions. This practice is their first contact with perception and actuation, and the student is taught to coordinate them using finite state machines.
- **Follow lines.** A white line was drawn in the laboratory, as shown in Fig. 4. The line was five centimeters wide and placed above the green carpet in the laboratory. The path had an incremental difficulty. The first phase was a straight line, adding curves in the second phase. The third phase incorporated some obstacles that could hit the robot if it moved away from the line at some point. In the third phase the line had discontinuous sections, and in the final phase again obstacles were incorporated outside the optimal path. If an obstacle was touched, or the robot turned around, the robot was disqualified. The position of the participants in a final ranking depended on the sections that they had completed. In case of reaching the same section, the position was decided according to the time taken to reach it. Each participant had two separate attempts for a one-hour interval. In each round the light conditions



**Fig. 4.** Follow lines arena.



**Fig. 5.** Go to ball competition.

could change (artificial light with different intensities, or natural light), being known 10 min before each round.

In this practice the student must process the image in the HSV color space to detect the line. In addition, students must design an easy way to calculate and setup the colors if the conditions change. Students also have to use a PID feedback control to control the robot.

- **Go to ball avoiding obstacles.** In this competition, the robot starts from an initial position and must go through an arena where there are obstacles, as shown in Fig. 5. At the end there is a red ball, but the position is not fully known. The robot must traverse this area of obstacles and stop 10 cm from the ball without touching it. In case of touching an obstacle or the ball, or leaving the sand, the robot is disqualified.

In this competition the student performs 3D perception and generates actuation reasoning in different 3D coordinate axes. The Virtual Force Field (VFF) algorithm is explained, based on a combination of repulsive vectors (from obstacles) and attractive (towards the ball) vectors.

- **Navigate in a house.** The arena was formed by a set of obstacles that simulated the walls of a house. It had a corridor from where the robot started, an open area that simulated a room with an obstacle in the middle, and two exit corridors of this room, as shown in Fig. 6. Students must build a map of this scenario in the setup phase.

There are three attempts per student. Just when it starts, the student is told which is the exit way the robot must take. From this moment on, the robot can no longer be touched. The robot is disqualified if it hits an obstacle. Each student gets their position in the final ranking based on the minimum time it took to reach the goal.

In this competition, students must build a map of the arena using the laser sensor. In addition, they must implement their own navigation algorithm using GPP (Gradient Path Planning).

As we presented above, microblogging is an increasingly used tool in innovative teaching strategies as a motivating element. All students previously had



**Fig. 6.** Scenario for the Navigate in a House test.

a Twitter account, so it was very easy to apply this technique throughout the course

The URJC Robotics group has a Twitter account with about 250 followers, including student associations from the university, internal school bodies, Robotics companies, groups from different universities, professionals and fans interested in Robotics. Students were encouraged to tweet videos and photos of



**Fig. 7.** Publication of results in social networks.

their practices. When they cite the group, these tweets are re-tweeted from the URJC Robotics group account among their audience (Fig. 7).

Throughout the course they saw that these messages were re-tweeted or marked “like” by other research groups in Robotics, or even by companies in which they could be interested in working, reason why the volume of tweets was growing during the course. For the teacher it was easy to have a quick reference of the progress of the students throughout the course and as a source of the audiovisual material for future courses and conferences.

Three of the most prominent students were interested in the participation of our team in the RoboCup robot world championship, held in Leipzig from June 27 to July 3 (Fig. 8). These students began their End-of-Grade Project in Robotics Works, which was not in their plans at the beginning of the course. Thanks in part to the financing of our university these students were able to participate in this competition, in the only Spanish team that participated.



**Fig. 8.** Students of our university in the RoboCup competition.

## 4 Results

Several months after the completion of the course, students were asked to complete an anonymous survey to measure their perception of the innovations presented in this work. 54% of the students answered.

The Fig. 9 shows the evaluation that the students give to the use of social networks and of *gamificación*. About half do not give any value to the use of microblogging. The others value them positively, not having students who consider it negative. However, the use of *gamification* is highly valued by students.

Figure 10 illustrates how students feel that these two teaching techniques have helped increase their motivation. On social networks one group considers that it did not help them, another that has slightly discouraged them and a third similar group that has slightly motivated them. Few students consider that its use has been very motivating. With regard to the competitions, most of them consider that it has been very positive in its motivation. A student considers it

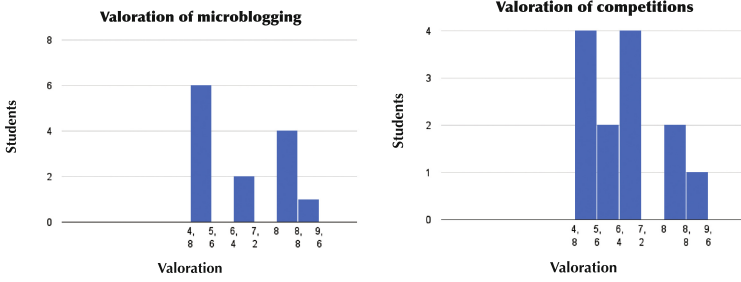


Fig. 9. Evaluation of Gamification and using Microblogging.

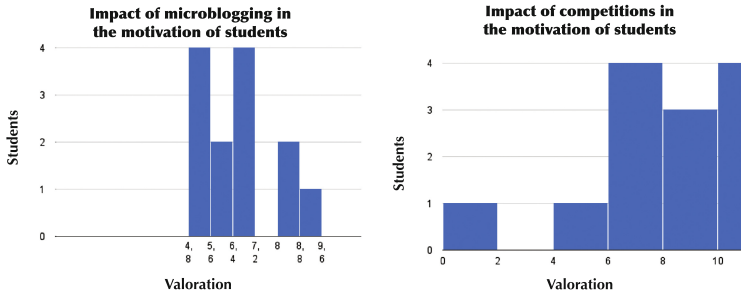


Fig. 10. Motivation produced by gamification and using microblogging

extremely demotivating. Although it is an isolated fact, it will be necessary to analyze if the fact of competing with your partners and not winning is negative in some case, and how to solve it.

Finally, the students were asked their predisposition to develop their career in positions related to robotics. The vast majority consider that their experience in the course has motivated them to work in this field (left graph in Fig. 11).

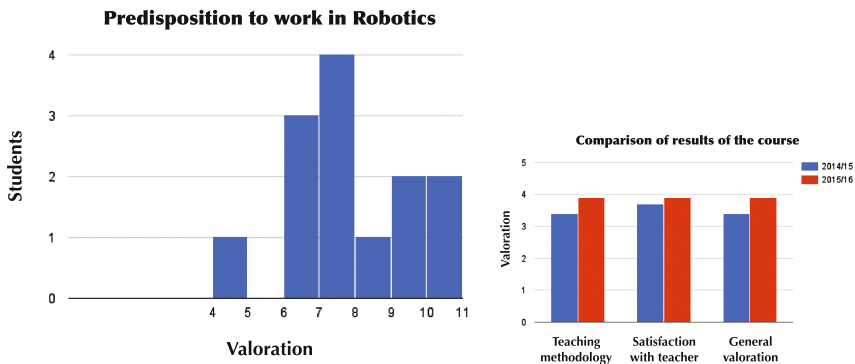


Fig. 11. Motivation to work in Robotics (left). Official valoration of the course (right).

The right graph of Fig. 11 shows a summary of the official college assessments of our university obtained before and after applying these techniques. As can be seen, the valuation in all the sections has improved significantly.

## 5 Conclusions

In this paper we have shown the planning of a course of robot programming in an Engineering. We have presented both the resources and the methodology. Our approach is to use exclusively Open Source elements, so we have used ROS as the basis for all the software used. We have described the design of the robot components, specifying the interconnections used, with the aim of being a guide to replicate this robots in other courses. As an advantage of this robot, we highlight its low cost, for robots equipped with laser sensors and 3D cameras. We have also described the most effective way in which it can be used, including the use of its own processor on board (available for current course).

In addition, we have presented the planning of the course in content. Another contribution of this article is the methodology used in the practices, based on gamification techniques. We think that raising practices such as competitions is a motivating element that encourages students to practice. In addition, it introduces an own factor of the competitions of international robots: the robot can not operate only once, with certain conditions and with an indeterminate term. At the start of the test the robot must be on the start line and must work.

Finally, we have described how we introduce microblogging in teaching methodology. This motivating element is based on the fact that the students can publish their results and have an immediate feedback from a specialized external audience, obtaining a reputation that can be beneficial in their future work.

Already before the end of the course there were several students who decided to focus their future work towards Robotics, being interested in doing their final project of studies in Robotics Mobile. At the end of the course, these students continued working in this area, participating in the RoboCup World Championship in Leipzig in 2016.

In conclusion, we think that the use of Open Source in Robotics allows addressing content in the subject that would otherwise have been complicated. The availability of tools, drivers and specialized libraries allows us to be able to impart mapping, localization and navigation concepts in an accessible and simple way. On the other hand, the use of motivation methodologies has managed to attract students of engineering, not directly related to Robotics, to this area of knowledge.

**Acknowledgment.** The authors would like to thank the School of Engineering in Telecommunications of Rey Juan Carlos University, specially to his director Javier Ramos, for the support to this work, both for providing us with the laboratory equipment and for partially financing student attendance at RoboCup 2017 in Leipzig.

## References

1. Belman, J., Flanagan, M.: Exploring the creative potential of values conscious design: students' experiences with the values at play curriculum. *J. Comput. Game Cult.* **4**(1), 57–67 (2010). Eludamos
2. Deterding, S.K., Nacke, L.R., Dixon, D.: Gamification: toward a definition. In: *CHI 2011 Gamification Workshop Proceedings* (2010)
3. Fernández, M.R., Revuelta, F.I., Sosa, M.J.: Redes sociales y microblogging: innovación didáctica en la formación superior. *Rev. Latinoam. de Tecnología Educativa* **11**(1), 61–74 (2012)
4. Fullerton, T.: *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. Morgan Kaufmann, Amsterdam (2008)
5. Martínez, E., Raya, P.: El microblogging en el proceso de enseñanza-aprendizaje. una experiencia académica con twitter. *Historia y Comunicación Social* **18**, 139–149 (2013)
6. Pérez, C., Carlos, J., et al.: Gamificación y docencia: Lo que la universidad tiene que aprender de los videojuegos. In: *VIII Jornadas Internacionales de Innovación Universitaria Proceedings* (2011)
7. Romero, H., Rojas, E.: La gamificación como participante en el desarrollo del b-learning: Su percepción en la universidad nacional, sede regional brunca. In: *Eleventh LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI 2013) "Innovation in Engineering, Technology and Education for Competitiveness and Prosperity"* Proceedings, pp. 14–16, August 2013
8. Zichermann, G., Linder, J.: *Game-Based Marketing: Inspire Customer Loyalty Through Rewards, Challenges, and Contests*. Wiley, Hoboken (2010)
9. Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., Konolige, K.: The Office Marathon: robust navigation in an indoor office environment. In: *International Conference on Robotics and Automation* (2010)



# The Robobo Project: Bringing Educational Robotics Closer to Real-World Applications

Francisco Bellas<sup>1</sup>(✉), Martin Naya<sup>1</sup>, Gervasio Varela<sup>2</sup>, Luis Llamas<sup>2</sup>,  
Abraham Prieto<sup>1</sup>, Juan Carlos Becerra<sup>3</sup>, Moises Bautista<sup>1</sup>,  
Andres Faiña<sup>4</sup>, and Richard Duro<sup>1</sup>

<sup>1</sup> Integrated Group for Engineering Research,

Universidade da Coruña, A Coruña, Spain

{francisco.bellas, martin.naya, abprieto,  
m.bautista, richard}@udc.es

<sup>2</sup> Mytech, A Coruña, Spain

{gervasio.varela, luis.llamas}@mytechia.com

<sup>3</sup> MINT, A Coruña, Spain

juancarlos@mintforpeople.com

<sup>4</sup> IT University of Copenhagen, Copenhagen, Denmark

anfvr@itu.dk

**Abstract.** The Robobo Project is a STEM-based project that aims to bring educational robotics, in primary and high school, closer to real-world applications. It is based on the use of a smartphone-based robotic platform called Robobo, a very flexible programming environment, and a set of lessons to integrate them. The smartphone provides high-level hardware capabilities in terms of sensors, communications and processing capabilities that allow to create more practical and realistic lessons that exploit human-robot interaction, with a small investment. In this paper, we present the main elements of The Robobo Project in terms of hardware and software, and two illustrative educational projects that can be developed within it.

**Keywords:** Interactive education · Smartphones · Computer vision · Speech recognition · Real-World robotics

## 1 Introduction

Robotics is a key subject in STEM education [1, 2], mainly due to its multidisciplinary and practical perspective. Thus, robotic projects involve mechanical design, electronics and programming skills which, in turn, require a background in mathematics, physics and other technological sciences. In addition, all of these topics promote the students being involved in projects with a practical objective as robots are made to solve problems in the real world. As a consequence, knowledge acquisition is much more effective due to the attention it captures [3].

Table 1 shows the main specifications of a representative set of robotic platforms used in STEM education. As it can be observed, most of them have simple sensors (ultrasonic or infrared), basic communications capabilities (Bluetooth mostly) and low computing power. The main reason behind this is economical, as introducing a set of

robots in a classroom is expensive. But there are other reasons like the possibility of having configurable mechanical kits, made up of simple elements, so the students can change the robot configuration. This feature in a robotic system implies that the hardware elements must also be simple, because they can be easily damaged and repairing them should not be complex.

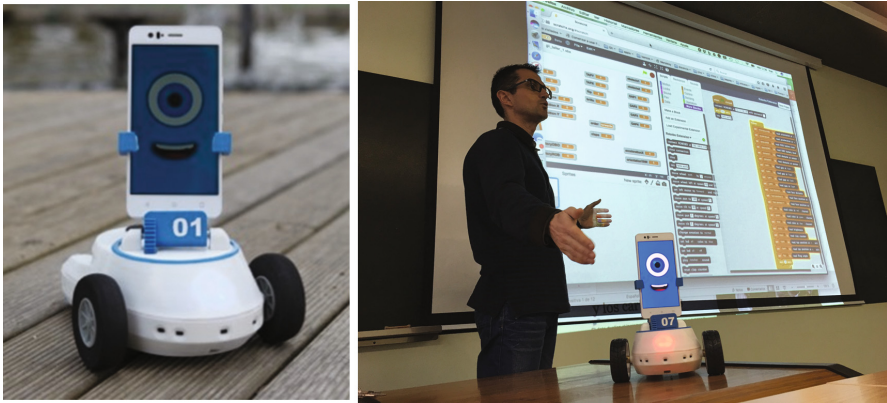
With such basic hardware, the projects that students can carry out are limited to a narrow range of possible applications. This was not a problem a few years ago, but nowadays and in the near future, as students acquire technological skills earlier, these simple robotic platforms will be a valid STEM tool only for primary level courses. Moreover, due to the limitations imposed by the hardware capabilities, most of the robotic lessons in elementary and high schools have traditionally been quite far from reality, that is, from being implemented in a real robot to solve a practical task [4]. For instance, a typical lesson in basic robotic courses implies developing a line-follower [5]. This is a very interesting lesson to learn basic programming skills and classical control, but students perceive robots around them every day without requiring artificial modifications of the environment to operate [6, 7], so this kind of lessons seem artificial to them. And, finally, simple robotic platforms are far from being up-to-date with regards to current hardware advances, so when students are using them, they perceive that the real technology is clearly way ahead. Consequently, educational centers must invest periodically to renew their robotic platforms, defeating the purpose of low cost systems.

**Table 1.** Basic specs of representative robots used in educational robotics

	NAO	LEGO EV3	DASH & DOT	MBOT RANGER	THYMIO II
CPU	Atom Z530 1.6 Ghz Cpu	Arm926ej-S Core@300 MHz	Arm Cortex-M0	Arduino Mega 2560	PIC24 32 MHz
Distance sensor	Sonar	Sonar/IR	IR	Sonar	IR
Sound	Speaker/Microphone	Speaker	Speaker/Microphone	Buzzer/Microphone	Speaker/Microphone
Camera	1280 × 960	No	No	No	No
Speech recognition	Yes	No	No	No	No
Communications	WIFI/USB	Bluetooth/USB	Bluetooth	Bluetooth/USB	WIFI/USB
Average Prize	7.000 €	350 €	230 €	170 €	130 €

In this sense, a few years ago we started the development of an educational robotics project called “The Robobo Project” [8], which aims to bring robotics closer to practical implementations for elementary and high schools. The Robobo Project is based on a hardware platform called Robobo, displayed in Fig. 1, and a set of STEM lessons supported on this platform, which are realized through a programming environment. The Robobo hardware is made up of a wheeled base that transports a smartphone, which provides Robobo with high level sensing, communications and processing capabilities, and that controls the base actuators. The smartphone allows the development of lessons using more complex sensing modalities, like computer vision or speech recognition, together with high processing capabilities to execute them on-board. This permits proposing projects closer to the real applications that robotics demands nowadays, with a high degree of human-robot interaction (HRI) and using more realistic sensors, so students start to deal with them early on. On the other hand,

the cost of Robobo is reduced, as compared to robots with similar hardware capabilities, as it can be observed in Table 1, where the only platform that supports computer vision or speech recognition is the most expensive one. In this sense, two more considerations must be taken into account. First, Robobo is a high-value product with a low-cost for the educational centers, because the main cost is in the smartphone, which can be provided by the student. Second, the lifespan is much longer than typical robotic platforms because it can be updated just by replacing the smartphone with a better one, and, as the phone in use is the student's, there is no cost for the school. Finally, as the student is using her own phone, she can work on the projects any time. As it is well-known, the smartphone market is highly competitive, so the new models are continuously updating sensors and features and they are always at the edge of technology. This implies that, when new sensing modalities appear, the Robobo will be able to handle immediately.



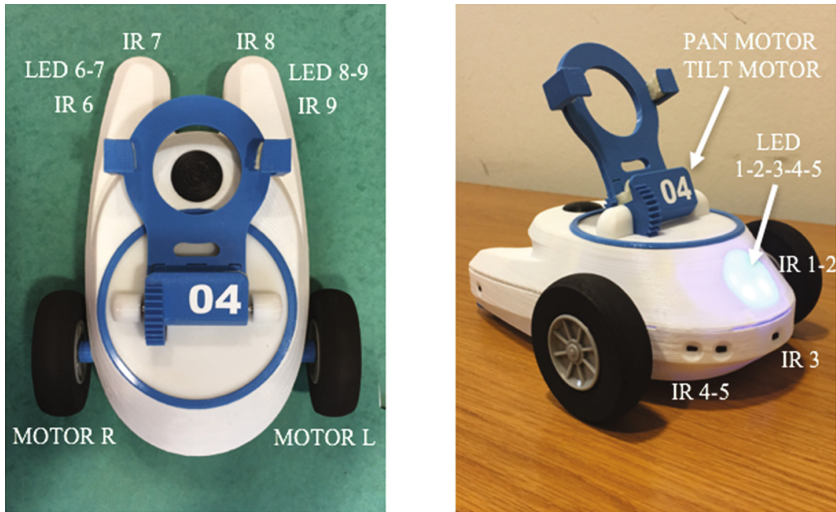
**Fig. 1.** Robobo hardware (left) and its use in a class (right)

In this paper, we present The Robobo Project's main features, that is, a description of the wheeled platform in Sect. 2, the programming environment in Sect. 3 and two illustrative educational projects supported by them in Sect. 4. Finally, Sect. 5 is devoted to some general remarks and conclusions.

## 2 Robobo Hardware

The Robobo robot is made up of two main parts: a mobile platform, called the ROB, that includes a series of low level sensors, the mechanical and electronic components for all the actuators, and a communications channel to connect to a smartphone, and the smartphone itself, the OBO.

Mechanically, the ROB part is divided into two elements that can be observed in Fig. 2: the main platform and the pan-tilt unit that supports the smartphone. The main platform contains the circuit boards, infrared sensors, batteries, docks for add-ons and



**Fig. 2.** The Robobo platform and its main sensors and actuators

the pan-tilt unit. The platform is driven by two wheels which are powered by two geared CC motors that allow the Robobo to achieve a velocity of 1 m/s with 40Ncm of torque. This is quite enough for snappy performance and climbing small ramps. The two wheels can move independently either backwards or forward, allowing the robot to turn.

There is a pan-tilt unit to hold the smartphone. This is a key feature in its HRI capabilities, as it allows the smartphone to perform actions such as nodding or shaking its “head”. When the pan-tilt motions are combined with images displayed on the smartphone screen, the sounds it produces, the displacement of the main platform and the different colored LEDs, the robot can express and transmit emotions, feelings, etc. in a way that is very natural and understandable for humans. Thus, it provides a mechanism for the robot to actively interact with children, influencing their mood, predisposition to learn, or even making them more sociable. Obviously, in addition to the interaction possibilities, the pant-tilt support allows for almost omnidirectional semi-spherical vision all around the robot from the smartphone cameras, providing a lot of information about the environment in which the robot operates.

The ROB platform electronics are structured around a PIC32 microcontroller for real time operation and responsiveness. It is in charge of controlling the ROB’s sensors, actuators and routine programs as well as the communications with the OBO. 9 RGB LEDs, with a color resolution of 12 bits per LED, for a total of  $2^{12} = 4096$  different colors on each RGB LED, are arranged around the ROB platform (see Fig. 2). These LEDs can be used to provide any type of information to the user. For instance, by default they are associated to each one of the infrared sensors, allowing the user to assert what the sensor is sensing in a very simple and intuitive manner. They are in charge of obstacle detection and fall avoidance as well as ambient light sensing. Regarding obstacle detection, 7 of the 9 infrared sensors are placed around the ROB to sense the distance to the different objects, providing obstacle range detection up to

10-15 cm depending on the color of the obstacle, and thus, Robobo is able to carry out collision avoidance tasks when required.

Two frontal sensors are dedicated exclusively to the detection of “holes” (in the sense of lack of floor). These sensors are tilted towards the ground at a  $70^\circ$  angle to optimize hole detection and its range, so that the robot has time to stop before falling. Two of the obstacle detection sensors in the back of the robot share the fall avoidance functionality. It is important to note that, even though the sensors are configured to work as infrared sensors, they can be reconfigured as ambient light sensors if the user so desires. In this configuration, the sensor returns the light intensity value.

The ROB platform communicates to the OBO part of the system by means of a Bluetooth module together with a series of firmware programs whose function is to abstract the low-level control into an API that the OBO can use. That is, the software developed on the smartphone just sees this abstraction of the ROB's functions. The choice of a wireless communications protocol over a wired one, such as USB, had to do with increasing the versatility and reducing the complexity of the platform due to the diversity of connectors and connector positions in different smartphones. Thus, the ROB platform can be used with any smartphone just as long as it has Bluetooth connectivity. As a summary, Table 2 shows the sensors, actuators and communication capabilities of the Robobo robot, both those provided by the ROB and those of the OBO. It must be pointed out that, although the processor of the ROB has low computing capabilities, the one in the smartphone is, in general, much more powerful than any other robot in educational robotics if a mid-range smartphone is considered.

**Table 2.** Robobo main sensing, communication and actuation capabilities organized considering those provided by platform and smartphone separately

	Platform	Smartphone
Sensors	<ul style="list-style-type: none"> <li>✓ 9 IR sensors (front, back and floor)</li> <li>✓ 4 Odometric sensors (motor encoders)</li> </ul>	<ul style="list-style-type: none"> <li>✓ Vision: front and back high-resolution cameras</li> <li>✓ Ambient: proximity, light, temperature</li> <li>✓ IMU: gyroscopes, accelerometers, magnetometers</li> <li>✓ GPS</li> <li>✓ Sound</li> <li>✓ Touch on screen</li> </ul>
Communications	<ul style="list-style-type: none"> <li>✓ Bluetooth</li> </ul>	<ul style="list-style-type: none"> <li>✓ 3G-4G</li> <li>✓ WI-FI</li> <li>✓ USB</li> </ul>
Actuation	<ul style="list-style-type: none"> <li>✓ Two motors on the wheels</li> <li>✓ One motor for PAN</li> <li>✓ One motor for TILT</li> <li>✓ 9 LED lights</li> </ul>	<ul style="list-style-type: none"> <li>✓ High-resolution LCD Screen</li> <li>✓ Speaker</li> </ul>

### 3 Robobo Software

One of the main goals of the Robobo robot is to serve as an educational and entertainment tool for teaching programming, robotics, and STEM disciplines across a wide range of ages. This goal has been the main idea driving the design of the software architecture of the robot, and the result is a completely modular architecture that:

- On the one hand, allows the easy extension of the robot with new features;
- And, on the other hand, allows users to program the robot using very different approaches that suit a variety of educational levels, like block programming, Java language or the Robot Operating System (ROS) [9].

In this section, we provide a detailed description of the Robobo software architecture and the different programming paradigms supported by it.

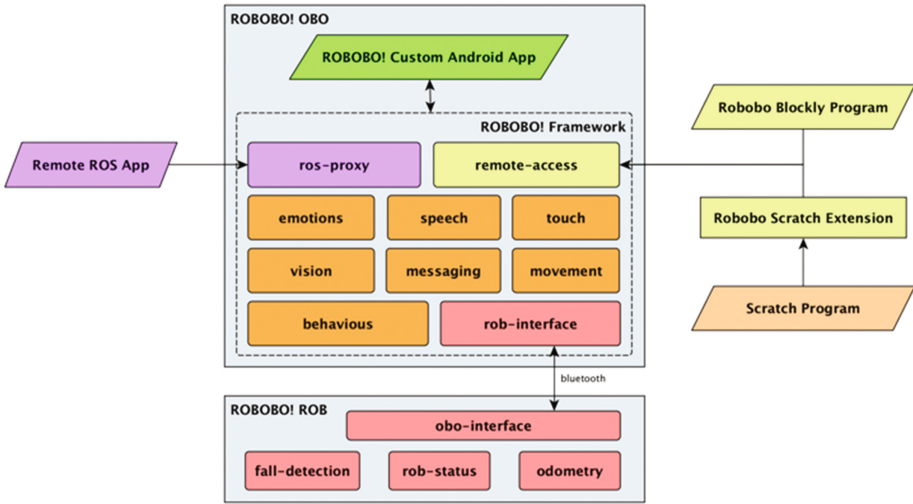
#### 3.1 Software Architecture

As previously introduced, the main requirement of the Robobo software architecture was to support the programming of the robot using very different programming paradigms. The goal was to be able to use the robot for teaching STEM and robotics in very different levels of education, from primary schools to universities, through secondary schools or even for research purposes. With this idea in mind, we have designed the completely modular software architecture shown in Fig. 3. It is based around the concept of Robobo module and a very lightweight library, called the Robobo Framework, which provides the essential mechanisms to manage and run those modules on an Android smartphone (iOS version support is under development). On top of this library, two different kinds of modules can be loaded:

- Functionality modules (the orange ones in Fig. 3): implemented in Java using the Android standard API, they provide different functionalities to the robot like speech recognition, face detection, environment sensing or physical movement. Furthermore, these modules complement the native API of the robot, which can be directly used for programming it using Java for Android, and thus creating Robobo Apps that can be run in any Android smartphone.
- A series of proxy modules (ros-proxy and remote-proxy modules in Fig. 3), also implemented in Java, which provide particular interfaces to other programming environments, like ROS, Scratch or the native Robobo Educational IDE using Blockly. These interfaces provide a translation between an external API or protocol, and the native Robobo API in Java.

Robobo comes by default with a particular set of these modules but, as they are completely decoupled between them and the framework, advanced users can customize the set of modules, and even implement new modules to support new robot functionalities, or even new programming paradigms through proxy modules.

Finally, it is important to note that there exists a module for connecting the Robobo framework and its modules to the Robobo robotic base (ROB). The rob-interface module, shown in pink in Fig. 3, implements the Bluetooth communications protocol of the ROB and provides a control API for other modules to use. This module is not



**Fig. 3.** Block diagram showing the Robobo software architecture and the different programming paradigms supported by it

special at all, it is implemented and loaded as any other module of the framework, but we have decided to present it in a different color because it is an essential module to control the physical part of the robot, the ROB platform.

### 3.2 Robot Programming

As briefly introduced in the previous subsection, the robot can be programmed using many different programming languages or paradigms and, thanks to the modular design of our solution, it can even be expanded to support new paradigms. Currently, Robobo supports four different programming languages or paradigms:

- Java programming for creating Android Apps.
- Scratch programming with ScratchX [10].
- Block programming using our own IDE based on Blockly [11].
- ROS programming for universities and scientific usages.

Java programming is directly supported by the native API provided by the different modules. Using the Robobo framework users can create custom Android Apps that control the behavior of the robot. These apps use the programming interfaces of the Robobo modules to access the different functionalities of the robot and build new robot behaviors or solve new tasks.

For block programming, we currently support two different approaches. Scratch, and our own block-based IDE that uses Blockly. As can be seen in Fig. 3, both approaches are connected to the framework using the same proxy, the Robobo Remote Access Protocol (RRAP). This is a simple JSON based protocol that allows remote access to the Robobo modules' APIs.

Scratch is supported by a Scratch extension, implemented in Javascript, that translates between Scratch blocks and operations of the Robobo modules exported through the RRAP. Our Blockly IDE follows a similar approach, where blocks are directly translated to Javascript code that uses the RRAP to control the robot remotely from the user's browser.

Finally, Robobo can also be programmed using the Robot Operating System (ROS), commonly used by the scientific community and for teaching robotics at many universities. ROS is supported by a ros-proxy module that translates between the native APIs of the modules and ROS messages and topics.

## 4 Robobo Interactive Lessons

To illustrate the type of interactive lessons that can be developed within The Robobo Project and the potentiality of using more advanced hardware in educational robotics, in this section we are going to present some examples of learning projects that have been designed, and that use the Robobo hardware and software commented in the previous two sections. These lessons are adequate for high-school students that have some basic notions of block-based programming, and they can be implemented with the ScratchX blocks already implemented in the Robobo software.

All the classical robotic lessons that can be developed using the typical sensors present in many other robotic platforms like infrared, IMU, simple sounds, ambient light or ground color, are directly applicable to Robobo. Consequently, we are going to focus our attention in the sensorial modalities that are rarely used in STEM robotics: computer vision, advanced sound processing, speech recognition and touch interaction.

All of these lessons start from the following didactical premises:

1. The students use their *own smartphone* in classes: this is a motivational aspect because they exploit a familiar element that, in addition, can also be used at home.
2. The Robobo can be *programmed* using a *tablet or a PC*: depending on their programming skills, preferences or educational center policy
3. The Robobo software simply requires that the *Robobo* and the *smartphone* are on the *same WI-FI* network (a dedicated one can be created in the classroom)
4. All the Robobo lessons can be *applied in the real world*

### 4.1 Simon Says Musical

In this first example, the project that students must solve is creating an interactive game consisting on a musical version of the classical Simon Says game using ScratchX blocks. Two elements are required in this case, a Robobo and a piano (see Fig. 4a for an example of configuration). They must be close and in a low noise environment. Specifically, the students must develop a game with the following steps:

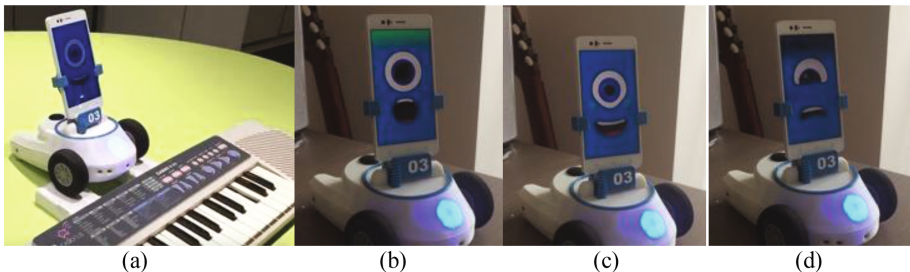
1. The robot operation starts with a clap that “wakes up” Robobo, which says “Ready”
2. With a second clap the game starts, and the robot tries to challenge the user saying something like “follow me if you can”
3. The robot produces a single musical note



4. The facial expression in the LCD screen changes to one that seems to pay attention to the user (Fig. 4b shows a possible expression) and the Robobo starts recording the musical notes the user creates
5. The original note produced by Robobo must be compared with the one sensed by it:
  - 5.1. If they are the same, the robot changes its expression to a happy one (Fig. 4c shows an example), the pan-tilt unit moves up and down (like saying YES with the head) and it emits a congratulating sound. The program returns to step 3 but now playing one more note.
  - 5.2. If they are different, the expression changes to a sad one (Fig. 4d), the pan-tilt unit moves from side to side (like saying NO with the head) and a failure sound is emitted. The program returns to step 1.

To solve the lesson, students must apply the following ScratchX blocks: clapDetector, speechProducer, noteProducer, noteDetector, soundProducer, emotionProducer and motorCommand, as well as other ScratchX native blocks to create the algorithm. The configuration of the blocks to achieve a fluid and reliable interactive game is part of the students work because, as highlighted in the introduction, the goal is to solve problems that can be used in the real-world.

As it can be observed, this lesson does not require any movement of the robotic platform, so one could think that a robot is not necessary in this case. But the approach behind The Robobo Project is that the human-robot interaction is more than just movement, and developing this type of static interaction is also very important. Moreover, the pan-tilt unit must be moved in this lesson so, from a didactical point of view, the students must use the Robobo motor commands.



**Fig. 4.** (a) The Robobo and the piano (b) Waiting for the user notes (c) User success (d) User failure

## 4.2 Robobo Pet

The second example project is focused on programming the Robobo to act as a pet. That is, the robot will ask the user for food, attention and “affection”. The modules required to solve this project use all the interactive modalities of Robobo, that is, computer vision, speech recognition, sound production and touch. In this case, the specifications provided to the students start with the premise that only a Robobo and a user are required in a controlled environment, in terms of other elements that can

interfere between them. Moreover, the program that controls the robot does not follow a sequential scheme in this case, but an event-based execution that is fired by the user with its interaction. The pet behavior must work as follows:

- *Basic instincts*: Robobo must store two variables that represent thirst and hunger levels. They start at a predefined value and they decrease proportionally to the robot activity (basically, motor movements). If these levels are lower than a predefined threshold, the robot must say “I am hungry” or “I am thirsty” until they are refilled as explained below.
- *Movement*: Robobo must move in order to capture user attention if no event has been executed for a predefined time-lapse. For instance, it can spin or emit a sound.
- *Touch*: two different behaviors must be implemented depending on the type of screen touch modality:
  - *Tap*: if the user touches the screen with a single tap, Robobo must react differently depending on the part of the “face” that is touched. If it is in the “eye” or in the “mouth”, it will move the tilt motor backwards and show an angry expression (see Fig. 5a and b for examples). If it is in any other point, it will show a laughing face and emit a sound saying “tickles” (see Fig. 5c and d for examples)
  - *Flip*: if the user slides a finger over the screen, the pan-tilt unit moves accordingly. The possible slide directions must be discretized into four pan-tilt movements: tilt backwards or forwards and tilt rightwards or leftwards.
- *Voice*: Robobo must react to the following predefined phrases:
  - *Here comes the food*: the robot prepares to receive food
  - *Here comes the drink*: the robot prepares to receive drink
  - *Hello*: the robot answers by saying “Hello”
  - *How are you?*: the robot will respond by saying its thirst and hunger levels
- *Vision*:
  - *Color*: Robobo must detect 2 different colors, green for food and blue for drink, but only after the corresponding voice command has been detected. In both cases, it must say whether the color is correct or not. For instance, if the user says *Here comes the drink*, the robot must look for a blue color area of a predefined size in its field of view (the left image of Fig. 6 shows example), and after a time-lapse, it must say “Thank you for the drink” or “I don’t see the drink”.
  - *Face*: Robobo must detect the user face. If it is below a threshold, the robot will move backwards (Fig. 6 right)

To solve this second project, students must apply, and configure, the following ScratchX blocks: speechProducer, speechRecognition, faceRecognition, colorDetection, touchDetection, emotionProducer and motorCommand, as well as other ScratchX native blocks to create the algorithm. The main feature of this project is that it can become as complex as the student wants, because achieving a fluid response of the pet is not simple. Furthermore, its upgrading possibilities are huge, and this is very important to promote student creativity. Thus, the teacher could propose several improvements. For instance, the robot could dance following the rhythm of a musical piece, it could follow the user voice or face while he/she is moving, or it could serve as a movable alarm clock.

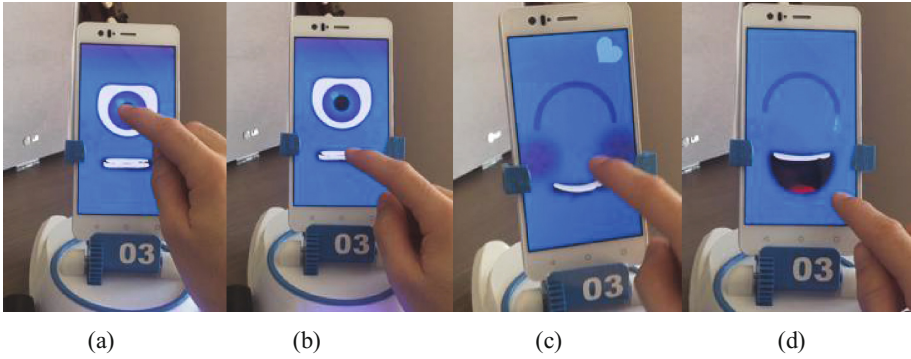


Fig. 5. Possible expressions in the touching behavior



Fig. 6. The user showing a green ball that represents “food” (left) and his face, which makes Robobo move backwards (right)

## 5 Conclusions

The Robobo Project aims to be a starting point in a new didactical methodology for educational robotics, which is based on more realistic projects to be solved by students. It is supported by a smartphone-based robotic platform, which exploits the high-end technology that is included in current mobile phones without implying a large investment for the educational centers. Moreover, it uses a very flexible programming environment that allows students to program in different languages depending on their skills, so the Robobo lifespan is large. In addition to presenting the main features of hardware and software architectures, two illustrative lessons that can be developed with Robobo have been described. As it can be observed in both examples, a more natural and realistic interaction can be obtained with the high-level sensing capabilities

provided by the robot, allowing for more realistic applications to be developed by students. Currently, Robobo is under testing in selected high schools in Spain, with the objective of being introduced to general public education from September 2017. As a consequence, some improvements, mainly in software and lessons, are under development right now.

**Acknowledgements.** This work has been partially funded by the EU's H2020 research and innovation programme under grant agreement No 640891 (DREAM project) and by the Xunta de Galicia and redTEIC network (ED341D R2016/012).

## References

1. Alimisis, D., Moro, M.: Special issue on educational robotics. *Robot. Auton. Syst.* **77**, 74–75 (2016)
2. Barker, B.S., Nugent, G., Grandgenett, N., Adamchuk, V.I.: *Robots in K-12 Education: A New Technology for Learning*, pp. 1–402. IGI Global (2012)
3. Kandlhofer, M., Steinbauer, G.: Evaluating the impact of educational robotics on pupils' technical-and social-skills and science related attitudes. *Robot. Auton. Syst.* **75**(Part B), 679–685 (2016)
4. Danahy, E., Wang, E., Brockman, J., Carberry, A., Shapiro, B., Rogers, C.B.: Lego-based robotics in higher education: 15 years of student creativity. *Int. J. Adv. Rob. Syst.* **11**(2), 27 (2014)
5. Khalid, A., Zahir, E.: Optimizing the turning velocity in a line follower robot. *Int. J. Comput. Appl.* **117**(3) (2015)
6. Jones, J.L.: Robots at the tipping point: the road to iRobot Roomba. *IEEE Robot. Autom. Mag.* **13**(1), 76–78 (2006)
7. Pepper Robot web page. Aldebaran Robotics: <https://www.aldebaranrobotics.com/en/cool-robots/pepper>
8. The Robobo Project web page: <http://theroboboproject.com>
9. ROS web page: <http://www.ros.org>
10. ScratchX web page: <http://scratchx.org>
11. Blockly web page: <https://developers.google.com/blockly/>

# Architectural Overview and Hedgehog in Use

Clemens Koza<sup>(✉)</sup>, Martin Wolff, Daniel Frank, Wilfried Lepuschitz,  
and Gottfried Koppensteiner

Practical Robotics Institute Austria (PRIA), Vienna, Austria  
{koza,wolff,frank,lepuschitz,koppensteiner}@pria.at  
<https://pria.at/team/>

**Abstract.** Robotics is a versatile tool for teaching STEM topics, as it supports various disciplines, skill sets and target audiences. However, controllers used in Educational Robotics are often limited in their use cases. In this regard, Hedgehog tries to be flexible by design. This paper introduces Hedgehog’s architecture, currently implemented and future use cases, and experiences from our first Hedgehog workshops.

**Keywords:** Raspberry Pi · Controller · Visual programming · Blockly · Python

## 1 Introduction

As a joint research and educational organization, the Practical Robotics Institute Austria (PRIA) offers students various ways to participate in educational robotics activities. From elementary school, where PRIA conducts LEGO Mindstorms courses, to technical high school, where students participate in international robotics competitions and research projects, there are a lot of different approaches and requirements to our offers.

As such, PRIA uses a variety of robotics controllers depending on the use case. However, investing in and supporting different systems results in more complex maintenance and higher costs. A device that offers a wide variety of applications for audiences of different ages, experience levels and interests is therefore desirable. With Hedgehog, we have striven to create such a system.

## 2 Related Works

There are many different robot controllers on the market. Here, we show three controllers in use at the PRIA lab: LEGO EV3, “Link,” and “Wallaby”, the latter two by KIPR, an American organization that runs the Botball Program.

The LEGO Mindstorms EV3 [1] is a robust, commercial robot controller. It uses a visual programming environment for coding, programs are transferred to the controller over a Mini USB cable. While the commercial support and the thought-out accompanying robotics kit are a big benefit, it is also a closed system. Up to four motors and sensors each are connected to the EV3 using

proprietary cables. Servos are not supported, and the focus is clearly on children, with no official way to do textual programming on the EV3.

The KIPR Link and Wallaby controllers [2] were developed for the Botball [3] robotics competition. They are both programmed in C, and thus not suitable for elementary school children. The Link uses native desktop IDEs (Integrated Development Environments) for Windows and OS X, and WiFi or USB for transmission. The USB driver is known to make problems in some situations.

The Wallaby is connected to a development machine via WiFi or USB, where it is detected as a network device. In some Windows configurations, driver installation has problems, but as long as the driver is installed properly, it works reliably. By default, each Wallaby opens a wireless hotspot with a unique SSID. In situations where many Wallabies are used at the same time, wireless connections don't work any more. Consequently, a software update was released that gives the user the option to deactivate the WiFi, or connect to another network.

A great obstacle for using one of these controllers for different audiences is their programming languages. The EV3 was not developed to get an update for textual programming, and Link and Wallaby were not developed for visual programming. Other than that, none of these controllers support the user in combining the robot API (Application Programming Interface) and other libraries.

### 3 Design and Architecture

Hedgehog's versatility stems to a large extent from its architecture. Furthermore, Hedgehog embraces open source technologies, enabling and inviting users to tinker with almost all parts of the controller.

The controller consists of two main parts: the Software Controller (SWC) is a Raspberry Pi 3 Model B that provides connectivity via WiFi, Ethernet, and USB. It is master to the Hardware Controller (HWC), which handles external components. The HWC is implemented as a printed circuit board (PCB) with an STM32F4 microcontroller at its core [4].

A controller is accessed over the network using WiFi or Ethernet, or from the device itself. Programs can be written using a web-based IDE.

The following subsections provide a bottom-up view of Hedgehog's hardware and software architecture. While many users will mostly use the Hedgehog IDE to write programs, all of these parts are open to examination.

#### 3.1 Hardware Controller (HWC)

One of the Hardware Controller's main purposes is providing the physical connection to peripheral hardware. Table 1 gives an overview of the connectors, some of which are routed through from the Raspberry Pi. The pins are compatible to the Link and Wallaby controllers.

Hedgehog uses an external 7.4 V power source, such as a 2-cell Lithium Polymer battery. For power management, the HWC board contains two switching regulators: one for the Raspberry Pi, and one for the servo ports. The motor drivers

**Table 1.** HWC connectors overview

Connector	Count	Description
Motor	4	2 pin connector, motor driver
Servo	4	Standard RC servo connectors
Digital I/O	8	3 pin connector, configurable as input & output; optional pullup or pulldown resistors for input
Analog input	8	3 pin connector, 12-bit analog/digital converter inputs with optional pullup or pulldown resistors
I <sup>2</sup> C Bus	1	Inter-integrated circuit from Raspberry Pi
SPI	1	Serial Peripheral Interface from Raspberry Pi
SPI	2	Serial Peripheral Interface from Microcontroller

are connected directly to the battery. The HWC contains a buzzer and four programmable LEDs, and monitors the battery voltage. Currently, the buzzer is used to issue a low-voltage warning.

HWC and SWC interface via Universal Asynchronous Receiver/Transmitter (UART) [4]. In addition, the microcontroller’s reset and boot pins are connected to the Raspberry Pi, allowing replacement of the firmware using the STM32F4’s UART bootloader.

The default firmware services requests by the SWC, but future versions might use HWC-initiated communication as well, e.g. for interrupt-based sensor input.

### 3.2 Software Controller (SWC)

The Software Controller contains most of the Hedgehog’s logic. It uses Raspbian and runs the Hedgehog’s server processes.

Hedgehog’s main process is a python program called the “Hedgehog Server”. It services commands that are received over the network, communicates with the HWC, and provides a discovery service so that Hedgehog controllers can be found on the network. The server is modular to simplify updates and customization, e.g. addition of new message types. The required software can be installed using the Python package manager, pip, so that software updates are easily possible.

The SWC also runs the “Hedgehog IDE” web application, which will be available via the node package manager, npm. Furthermore, an STM32F4 tool-chain [4] and the Protobuf compiler are installed on the Raspberry Pi, meaning all Hedgehog software can be built and deployed using the controller.

The installation process is currently best documented for Unix-like operating systems. It is a clear and easy procedure starting with an empty Micro SD card. Easy installation encourages development and testing of modified software setups. For example, the default text-only Raspbian can be replaced with one providing a graphical shell, otherwise following the same installation instructions.

As Hedgehog does not have a touch screen or similar, a network connection is required for using it. This leads to the problem of a sensible default network

configuration. Hedgehog solves the problem by using a configuration file provided on a USB flash drive. At startup, the controller looks for this file and saves the settings, including networks and the controller's device name.

By default, Hedgehog does not open a WiFi hotspot, it instead connects to an existing network. The rationale is that in a classroom setting, having one wireless network per controller results in terrible performance. Ad-hoc networks are not universally supported and were therefore not considered further.

### 3.3 User Programs and Hedgehog Protocol

By accessing the controller over the network, programs can be run on any machine. In the usual case, a local process connects to the Hedgehog Server, but that is not required. For example, one program could connect to multiple controllers at the same time, as shown in Listing 1.

```

1 from time import sleep
  from hedgehog.client import connect

  # connect to two Hedgehog controllers. Default is to connect
  # to the local Hedgehog Server on 127.0.0.1:10789.
6 # Specifying 'endpoint=None' overrides this to use the
  # discovery mechanism: only a controller that is configured
  # as a 'robot-arm' is considered for connection.
  with connect() as hedgehog1, \
    connect(endpoint=None, service="robot-arm") as hedgehog2:
11     # move local motor for one second
    hedgehog1.move(0, 1000)
    sleep(1)
    hedgehog1.move(0, 0)

16     # enable and move remote servo
    hedgehog2.set_servo(0, True, 1500)
    sleep(0.5)

  # upon termination, all motors and servos are disabled

```

Listing 1: Python sample program connecting to multiple controllers

The Hedgehog Server listens to a well-known port, so local programs can use that port and the device's loopback address to connect to it. For remote applications, users can either specify the IP address or device name manually, or use the discovery mechanism to find controllers that advertise a particular service. For example, a program might connect only to a controller that offers the custom `robot-arm` service.

Connections between client processes and the Hedgehog Server use ZeroMQ [5], a messaging library. Protocol messages are encoded and decoded with Protobuf [6]. Both of these libraries are available in many programming



languages, allowing for client programs in various languages [4]. Client libraries are intended to follow the philosophy of the language in question so that using Hedgehog, that language can be taught properly.

The Hedgehog protocol and client libraries support accessing sensors, motors, and servos. In addition the protocol defines messages to start and manage processes on the controller, e.g. to run user programs from an IDE. Protobuf allows it to handle unknown and changed message types, enabling updates to the protocol while maintaining compatibility on the wire.

An important design goal of using a language-agnostic, extensible network protocol for all hardware access is the ease of implementing new kinds of clients. For example, a first prototype of the Hedgehog IDE was a standalone Python desktop application. The new web-based version is a very different kind of software, but it can access the controller in the same way.

An example program using the Python client is shown in Listing 1. There is also a node.js client library, and a Scala/Java library prototype for Android apps, which was used to create a mobile visual programming environment [4].

### 3.4 Hedgehog IDE

The Hedgehog IDE is a web application that uses node.js for the backend and Angular 2 for the frontend. Both are written in TypeScript and share code where possible. From a Hedgehog architecture point of view, it is a rather complex, but otherwise ordinary client program: it runs locally to the controller and connects to the Hedgehog Server using the node.js client library to run user programs.

The Hedgehog IDE allows users to create, edit and execute applications for the controller. Internally, all projects are managed as git repositories. This is normally hidden from users, but it creates a universal way for advanced users and third party tools to access Hedgehog projects.

The IDE supports both textual and visual programming, shown in Figs. 1 and 2, respectively. The latter is based on Blockly, a library for building visual programming editors. The visual editor provides two “toolboxes” of command blocks, one targeted at beginners, and a more comprehensive toolbox for advanced users. Blockly programs translate directly to Python code that is then executed. Users can inspect the code that is created from their visual program and thus use the visual editor to transition to textual coding.

As a web application, the IDE doesn’t need to be installed on the programming device, and supporting different platforms does not require separate applications. Access via WiFi or Ethernet does not require any driver installation, and no firewall rules are necessary to access the IDE. Although development is currently focused on traditional desktop computers and browsers, the IDE can later be optimized for mobile devices.

When using a Raspbian version with a graphical shell, Hedgehog IDE can be used directly on the controller by connecting peripherals to the Raspberry Pi’s HDMI and USB ports.

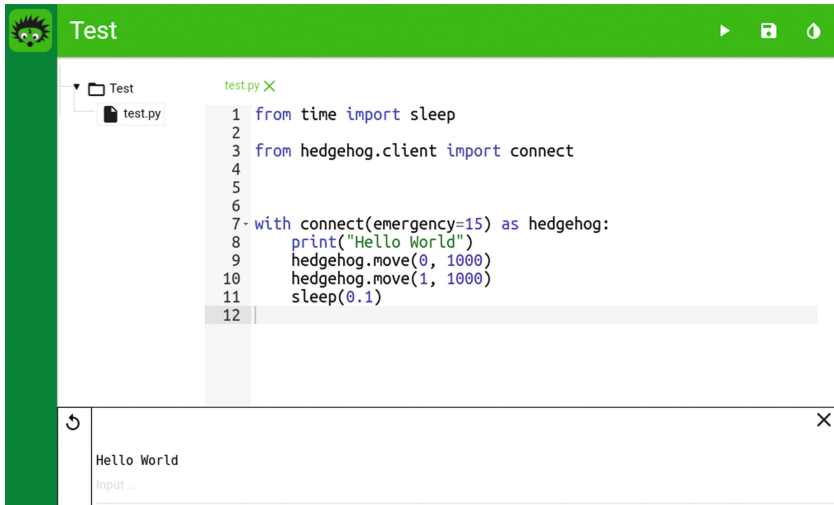


Fig. 1. Hedgehog IDE Python editor.

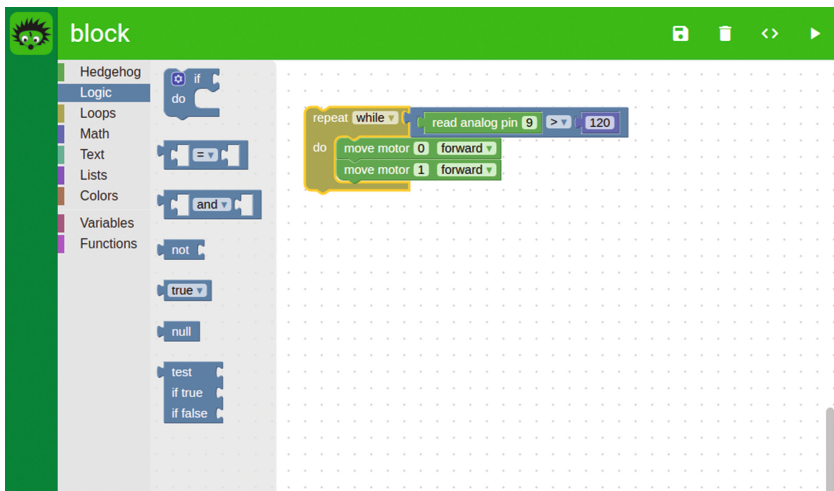


Fig. 2. Hedgehog IDE Blockly editor.

### 3.5 Maker Aspects and Open Source

Most parts of Hedgehog can be assembled using equipment typically present in fab labs. In particular, a laser cutter for the acrylic case of the controller and a reflow oven for SMD (surface mounted device) assembly are required [4].

For space reasons, the HWC circuit board has four layers, making it one part that can not easily be manufactured in fab labs. Other hardware components

that need to be purchased include a battery, Raspberry Pi, SMD parts and pin headers, and acrylic sheets and screws for the case [4].

Hedgehog runs mostly free, open source software: Raspbian is a variant of the open source Debian Linux distribution. All parts of the Hedgehog Server, IDE, and firmware are licensed under AGPLv3 and available from GitHub. Required third party software, such as ZeroMQ, are open source and installed from Raspbian, PyPI (Python Package Index), and npm repositories. The microcontroller firmware also relies solely on free software libraries [4].

The HWC circuit board layout and acrylic case design are being prepared for open sourcing as well. When this is done, all aspects of Hedgehog development can be done on the controller itself.

## 4 Use Cases

As seen in the previous section, Hedgehog is very open about almost all parts of its hardware and software. Some parts are more suitable for advanced users, but many others are accessible to beginners as well.

As Hedgehog is only the controller, we emphasize topics here that focus on that device. For example, combined with an omnidrive chassis, Hedgehog can be used to introduce that mode of motion, along with supporting trigonometric concepts. However, that is mostly due to the chassis, not Hedgehog. We will of course assume that basic robotics components are available.

The following subsections highlight some ways to use Hedgehog in the classroom and other educational settings, ordered by approximate age appropriateness.

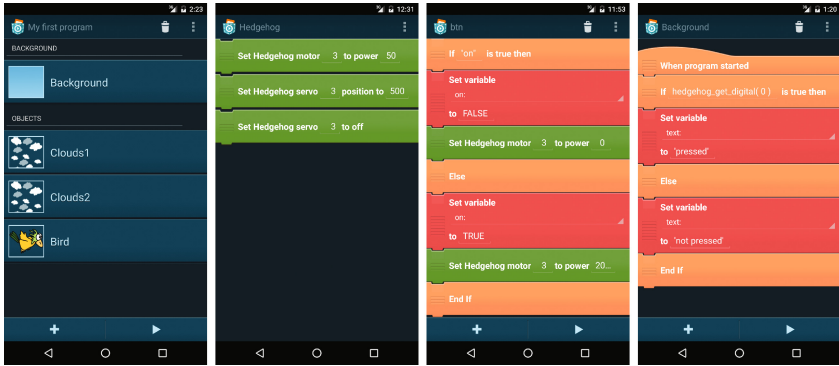
### 4.1 Visual Programming Using Pocket Bot and Blockly

As mentioned above, Hedgehog features a visual programming environment in its IDE, and also in the Pocket Bot app. Graphical environments can be used to teach the basics of imperative programming at an early age, and the robot provides a graspable target platform for commands.

Pocket Bot [4] is an extension of the Pocket Code [7] Android app. A Pocket Code program has a stage that is composed of background and objects. Each of these can have scripts that trigger on different conditions, such as the start of the program, or tapping an object.

After the trigger block, command blocks specify actions to execute, such as moving an object on screen. There is a distinction between blocks, which execute actions and have no result, and functions, which have results and generally no side effects. Pocket Bot adds blocks for controlling Hedgehog's motors and servos, and functions for reading sensor values. Figure 3 shows examples of both [4].

Figure 2 in the previous section shows the Blockly editor of the Hedgehog IDE. Control structures are shown in a way that supports the notion that they group commands, and puzzle-like knobs indicate how blocks can be combined: commands flow vertically, while expressions are combined horizontally.



**Fig. 3.** Pocket Bot visual programming environment for Android. From left to right: background and objects in the Pocket Code demo application; Hedgehog command blocks; part of an object’s script that toggles a motor; and usage of a Pocket Bot function for sensor readout [4].

The “Code”-button switches the view to a read-only Python version of the program. Once students are comfortable with formulating algorithms in the visual environment, they can use this feature to understand how their code translates to Python, or they can use the visual environment in self-study to remind themselves on how to write specific pieces of code.

Both visual coding environments provide a slightly limited feature set compared to the Python API: they only support connecting to one controller, and process management or multithreading features are missing. The assumption here is that visual coding is mainly used for beginners, where such advanced features are not needed.

## 4.2 Textual Programming in Python

Textual programming is less appropriate for young children, but has other benefits. We focus here on Python, but these benefits apply to most other programming languages as well.

Python is a multi-paradigm, general-purpose language. It has a large standard library and ecosystem of third-party packages for various kinds of problems [4]. It is further regarded as easy to learn [8], with little syntactic overhead.

Listing 2 shows a very short Hedgehog program. Compared to a program in C or a related language, there is no main function and all code is executed from top to bottom. Although explaining the exact workings of the first four lines would be too much for a beginner, the `connect()` call at least motivates that the code up to that point sets up a connection for the rest of the program.

Although it makes Hedgehog programs look busier at first, we decided against executing setup code before running the user’s actual script. For experienced users, this means that there are no surprises in running one’s code. When a

```

from time import sleep
from hedgehog.client import connect

with connect() as hedgehog:
5     hedgehog.move(0, 1000)
       sleep(1)

```

Listing 2: Smallest viable Hedgehog program

beginner grows out of the core Hedgehog environment, they don't have to first learn how to imitate the setup that was "magically" done for them before.

### 4.3 Distributed and Concurrent Applications

The Python Hedgehog library is designed to be threadsafe, and all of Hedgehog is ready to be used in a distributed setup. A simple example of a distributed application would be a remote control, which children love, but is often hard to implement because of the necessary communication.

Listing 1 already showed that accessing two controllers at the same time is very easy. Creating a remote control becomes a matter of reading the sensors of one controller, and effecting the actuators of a second one.

Having different programs to communicate with each other requires more work on the user's part, but many necessary features are already in place: the discovery system is not limited to advertising the Hedgehog Server, it can be used to find any service on the network. A server also needs to listen for clients' messages, usually on a new thread; the Hedgehog library's thread safety is helpful in that regard. Users still need to take care of the usual pitfalls of concurrency, but they don't need to search library code while looking for problems.

### 4.4 Microcontroller Programming

As explained earlier, the SWC comes with a toolchain for the HWC's STM32F4 preinstalled. In fact, the main way of updating the HWC firmware is to pull the newest source code from GitHub, building it on the Raspberry Pi, and then flashing the HWC.

Compiling code for a microcontroller requires cross-compilation [4], which can be tricky to set up. Having a device that combines development environment and executing device is therefore very convenient, especially when students might bring their own devices with different operating systems, etc.

Users are of course bound to the hardware layout of the HWC [4], but there still is a lot of freedom. The microcontroller's UART, reset and boot pins are wired to the SWC, and others are connected to the motor drivers or to the servos' power supply. The microcontroller's SPI and analog/digital IO pins, for example, can be used freely as with a discovery board.

Users can modify the firmware to do new things, or ignore the HWC and SWC software entirely and create their own microcontroller projects. To restore the original state of the controller, re-flashing the official firmware is sufficient.

#### 4.5 Project Participation by Students

For PRIA as a joint research and educational organization, Hedgehog is also a valuable tool to include students in our projects. For example, the Hedgehog IDE described in this paper is being developed by a student team from TGM, a technical high school in Vienna, Austria, as part of their graduation project.

Other projects in which high school student teams used or improved Hedgehog technology include “00SIRIS”, where students designed a 3D-printed robot arm for classroom use (shown in Fig. 4), and “AndriX”, which became the predecessor [9] of Hedgehog.

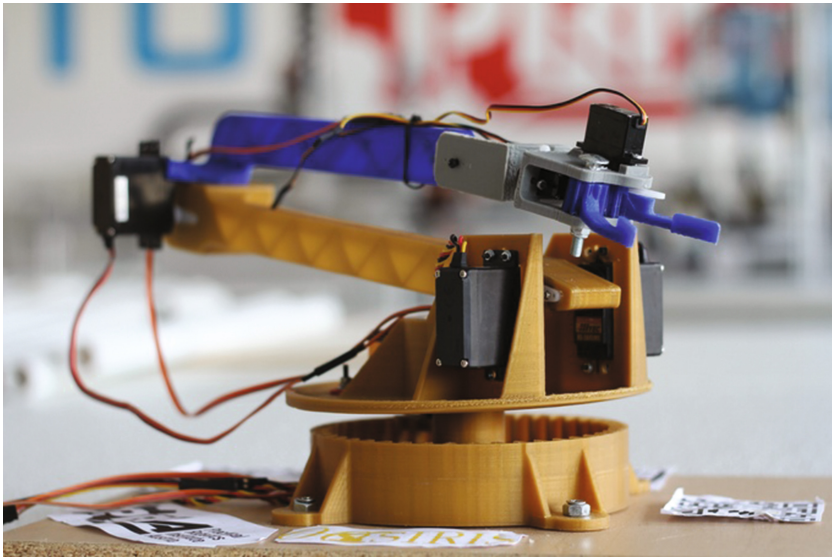
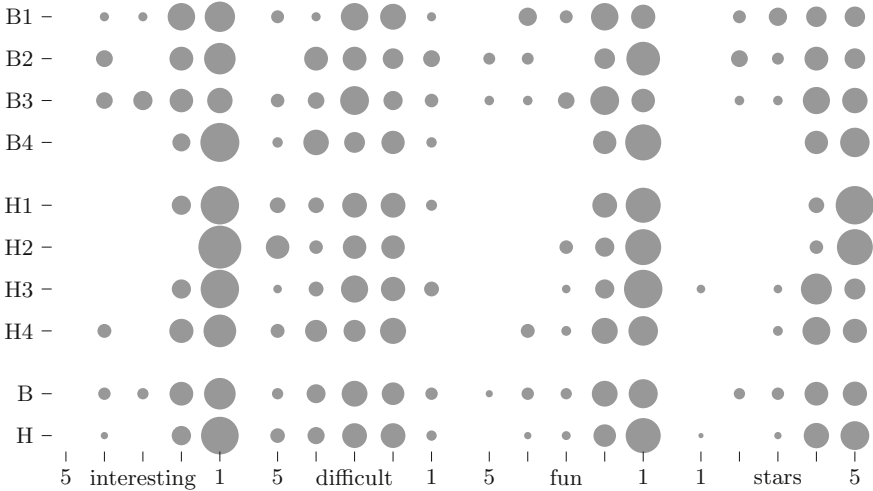


Fig. 4. 00SIRIS 3D-printed robotic arm

## 5 Workshop Experiences

As part of the ER4STEM project, first Robotics workshops using the Hedgehog controller were held. Students were taught the same topics as in previous workshops with controllers from the Botball program, except for the programming language: C in case of Botball controllers, Python in case of Hedgehog.

As of this writing, four Hedgehog workshops with a total of 70 students aged 11 to 16 have taken place. Figure 5 shows metrics for these workshops.



**Fig. 5.** Response counts from four workshops each, for statements “working with robots was interesting/difficult/fun” (5: don’t agree, 1: agree) and “overall, I would rank this course with ... stars.” Bottom: Totals for Botball and Hedgehog.

For comparison, four Botball workshops with a total of 72 students in the same age range are presented as well. The chart shows response counts relative to group size as the area of the bubbles.

For the difficulty question, there is no significant difference between the kinds of workshops. For the other questions, it can be seen that grades of 5 and 4, or 1 and 2 star ratings, are almost absent from Hedgehog workshops. Of those that gave a grade of 1 or 2, students in Hedgehog workshops more often gave a 1 than those in Botball workshops. For the star rating, this is less clear: two of the four workshops had only 4 and 5 star ratings and a very high proportion of 5 star ratings, while the other two workshops had mostly 4 star ratings.

The absence of bad grades and ratings shows that less students did not engage with the workshops, and the gravitation towards top grades indicates that enthusiasm in those already engaged was greater as well. It has to be noted, however, that other factors were not corrected for in the data. For example, different people held these workshops, and students from different schools participated in each one.

Oral responses from tutors indicated an overall satisfaction with Hedgehog. Compared to Botball controllers, they say that the system runs more stably, the wireless connection is more convenient than USB, and that the use of Python accelerates the workshops because there is no compilation step.

## 6 Conclusion and Future Work

First results suggest that Hedgehog in its current state is already a viable teaching tool. A further essential step is the completion of the Hedgehog IDE.

Some teams have registered to use Hedgehog in competitions at the European Conference on Educational Robotics (ECER) 2017 in Sofia, and Hedgehog will be used in workshops throughout the school year and during the summer holidays. All this will give us further feedback and experience reports to improve the system.

Plans for improvements and new features include a change to the HWC circuit board layout to make the controller smaller, APIs for computer vision and controlling iRobot Create robots available at the PRIA lab, and a dataflow programming environment to allow for an alternative to imperative programming.

**Acknowledgements.** The authors acknowledge the financial support from the European Union's Horizon 2020 research and innovation program under grant agreement no. 665972. Furthermore, the authors want to thank the high school students who contributed to Hedgehog, in particular Markus Klein, Thomas Fellner, and Hassan Mohamad for their work on the Hedgehog IDE.

## References

1. LEGO Mindstorms EV3. <https://www.lego.com/en-us/mindstorms/about-ev3>
2. KISS Hardware/Software. <http://www.kipr.org/hardware-software>
3. Koppensteiner, G., Merdan, M., Miller, D.P.: Teaching botball and researching robotics. *Robot. Educ.* (2011)
4. Koza, C., Lepuschitz, W., Wolff, M., Frank, D., Koppensteiner, G.: Hedgehog light – a versatile, white box educational robotics controller. In: *Edurobotics 2016: Educational Robotics in the Makers Era*, pp. 229–232 (2016). doi:[10.1007/978-3-319-55553-9\\_19](https://doi.org/10.1007/978-3-319-55553-9_19)
5. Hintjens, P.: *ZeroMQ: Messaging for Many Applications*. O'Reilly Media, Inc., Sebastopol (2013)
6. Protocol Buffers. <https://developers.google.com/protocol-buffers/docs/overview>
7. Catrobat Pocket Code. <https://www.catrobat.org>
8. Donaldson, T.: Python as a first programming language for everyone. In: *Western Canadian Conference on Computing Education* (2003)
9. Koza, C., Krofitsch, C., Lepuschitz, W., Koppensteiner, G.: Hedgehog: a smart phone driven controller for educational robotics. In: *6th International Conference on Robotics in Education*, pp. 50–52 (2015)



# Open-Source Robotic Manipulator and Sensory Platform

Luka Čehovin Zajc<sup>(✉)</sup>, Anže Rezelj, and Danijel Skočaj

Faculty of Computer and Information Science, University of Ljubljana,  
Večna pot 113, Ljubljana, Slovenia  
{luka.cehovin,anze.rezelj,danijel.skocaj}@fri.uni-lj.si

**Abstract.** We present an open-source robotic platform for educational use that integrates multiple levels of interaction through the use of additional vision sensor. The environment can be used in virtual, augmented-reality and real-robot modes, enabling smooth transition from a virtual robot manipulator to a real one. We describe the main aspects of our platform that ensure low production costs and encourage openness of both its hardware and software. The main goal of our work was to create a viable low-cost robotic manipulator platform alternative for the university level courses in intelligent robotics, however, the application domain is very broad.

**Keywords:** Robotic manipulator · Education · Open-source · Open-hardware · Computer vision · Augmented reality

## 1 Introduction

Robotics is a very attractive and increasingly important research and engineering discipline. With the growing robotics market and very bright outlooks for the future, it is to be expected that the demand for educated robotics will even grow. Moreover, it is very advisable that also other engineers as well as young people in general become more familiarized with the robotics technology to cope with the challenges of the future. In addition, due to its attractiveness, robotics plays an important role in STEM education in general. Therefore, it is very important to develop teaching equipment and methodology that would enable efficient and generally accessible educational process in the field of robotics.

Robotics is also very engaging due to its hands-on experience; the students operate with real robots acting in a real world causing real effects in the environment. The robots are usually equipped with sensors to enable them to perceive the environment in order to act accordingly, therefore to close the perception-action loop. It is crucial that the students are exposed to real robots, yet industry-grade like robotic manipulators remain expensive equipment that only a few educational institutions can afford. Many educators are instead resorting to simulation-based solutions [8] which are cheaper and easier to set up and maintain, but do not offer the real experience and are more difficult to interact

with. The high price of robot manipulators has resulted in multiple attempts to build medium [9] and low-cost [1, 3, 5–7] alternatives with varying level of reliability. Aforementioned works have focused primarily on the engineering side of the challenge such as type and cost of the building material (mostly alloys) and low-level control. The price of such systems spans from 4000\$ to 50\$ where the low-end systems have severely limited functionality, required for university-level courses on robotics.

In our work we embrace the recent trend in manufacture of physical objects using the 3D printing technology. Our primary focus is not in the assembly of the physical components, what sets our system apart from the rest in the educational aspect is the extension of our manipulator in a robotics sensory platform. Using these extensions we enable smooth transition between the simulated environment which is much more convenient in the early learning and development stage and the real world which is important for later stages because of the information richness and realism. We believe that a smooth transition between the simulated environment and the real world is crucial for teaching high-level robotics concepts. Using the combination of hardware and software, we have developed a system that can be used at three different levels:

1. *Simulated environment*: The robot manipulator operates in a simulated software environment, however the students can control it using the same interface as the real one.
2. *Augmented environment*: The students use the real camera to perceive the environment and to detect objects in the scene; however, they still operate the simulated robot manipulator. The robot manipulator is virtually positioned in the real world image stream using augmented reality technique.
3. *Real-world environment*: The system consists of the real camera as well as of the real robot manipulator. The robot performs both, perception and action in the real world environment.

The students are first trained using the simulated robot manipulator that enables convenient learning and experimentation. Since they do not need the physical robot manipulator, they can also use the simulated system at home. Moreover, by adding a (widely accessible) web-camera, the students can learn computer vision techniques using real images and observe the operation of the robot manipulator in an augmented environment.

In this paper we briefly describe the hardware and software components of our platform (Sect. 2), describe where it has been used and tested so far (Sect. 3) and conclude with our vision for the future (Sect. 4).

## 2 Platform

In order to make the system even accessible to the as broadest interested public as possible, we have designed the platform to be open from the hardware up. We plan to open-source the blueprints, build instructions and software so that in principle, the platform could be produced and/or assembled by the students

(or hobbyist), which increases their involvement even more and broadens the range of potential users.

## 2.1 Hardware

The main challenge of developing the robotics platform was to design a robot manipulator, which is reliable, but also low-cost to produce. Our manipulator is shown in Fig. 1. It has 5 DOF and is quite robust but also lightweight and easy to use. It is safe enough to be operated by the kids, while on the other hand realistic enough to be useful for the students of robotics. The key factor in openness is that most of the structural components can be printed by a 3D printer. The first iteration of our manipulator is based on the work from [4]<sup>1</sup> that was adapted to our needs, while other components can mostly be bought in a hobby store. The only notable exception that is not readily available are the motor controllers which are based on the OpenServo designs [2] that we have further improved. Those controllers are replacing original controller in servo motors. For servo motors we used Hitec HS-311, HS-645MG, HS-485HB and three Emax ES08A II which are used in base, shoulder, elbow, wrist and gripper respectively. Maximum payload is about 100g. Power for motors is supplied using 5 V 10A power supply for better voltage stability in high current cases.

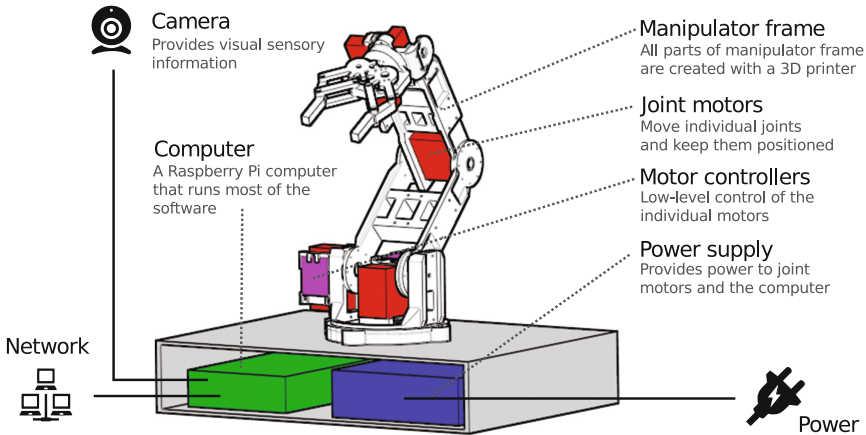


Fig. 1. Robotic manipulator and other components of the platform

While the robot manipulator can be connected to an arbitrary computer, our platform is designed to be used with low-cost single-board computers, like Raspberry Pi. The Raspberry Pi computer and the power supply unit are enclosed in

<sup>1</sup> The model is available at Thingiverse (<http://www.thingiverse.com/thing:30163>) under a permissive Creative Commons license.

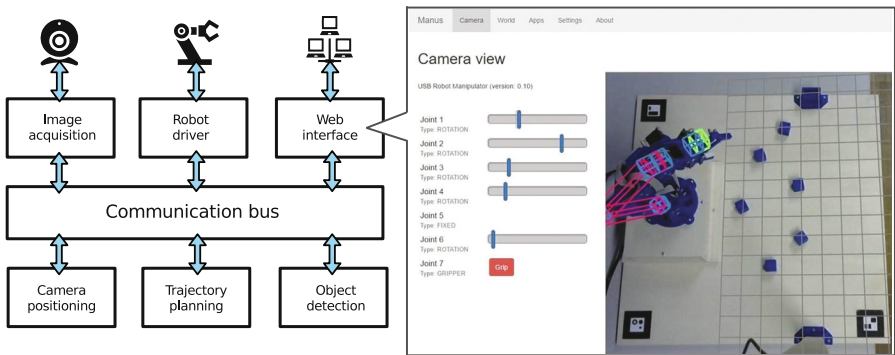
the base of the platform. The camera can be positioned above the manipulator and is registered with the working surface using special markers.

We estimate the cost of the entire system (including 3D printer material, a Raspberry Pi computer and a camera) to be around 250€ which makes it really affordable. We would also like to point out that this is the small quantity price which may be further reduced by bulk orders of certain components.

## 2.2 Software

The software stack that was developed for our platform follows several design goals: (1) it has to be lightweight so that it can be deployed on low-cost embedded devices, like Raspberry Pi, (2) it has to be flexible and modular so that it can be adjusted to different educational scenarios, (3) it has to be robust to withstand long-term usage, (4) it has to be accessible from without complicated installations, and (5) it has to rely only on open-source technologies to reduce cost of the system and to retain its openness.

To address these constraints we have developed a multi-process architecture, similar to the one of more complex robotic middlewares, e.g. ROS [10], but more optimized in terms of memory and processing requirements. The components run on Linux-based system and communicate between each other using a message bus based on system sockets as shown in Fig. 2. Individual components take care of image acquisition, image processing, low-level robot control, trajectory planning and interaction with the user.



**Fig. 2.** Software architecture of the system and a screenshot of the web-interface

Since the platform is designed to be used at different levels of educational process, the interaction also varies a lot. At the higher levels the interaction is done using web interface which alleviates operating system and compatibility constraints. We have created a web front-end (Fig. 2) that enables monitoring of the current status and we plan to extend it with rudimentary programming facilities aimed at younger audience. More advanced usage scenarios involve remote

control using web API (e.g. from Matlab) or direct integration of code into the local ecosystem by interacting with components via the message bus.

### 3 Use Cases

The developed platform was so far used in two different educational scenarios. We have introduced it into curriculum of a course on robotics and computer vision at the university level and used it in educational events that promote the field of computer science, the target audience in this case being potential future students and other tech-savvy youth.

**University-level course:** During the course on robotics and machine perception it is important that the students acquire a suitable level of knowledge about both, perception and action part of the robot manipulation. Using the developed robotic platform, we introduce the students to the different topics in a gradual manner. The teaching process involves the stages presented in Table 1:

**Table 1.** Gradual teaching stages.

	Perception	Action	Activity
1	Simulated	None	Learning basic computer vision algorithms by processing stored images
2	Real	None	Learning more advanced computer vision algorithms by capturing and processing live images
3	None	Simulated	Learning the basics of robot manipulation in simulated environment
4	None	Real	Learning to operate the robot manipulator in the real world
5	Real	Simulated	Detecting the objects in the scene and pointing at them with the virtual robot manipulator
6	Real	Real	Detecting and grasping the objects in the scene with the physical robot manipulator

The separation to these different stages is not strict, they are also intertwined with the goal to achieve a maximal motivation of the students and speed up learning. We have began with the introduction of the proposed platform into a robotics and machine perception course in year 2017 to replace a simple simulation solution. We have discovered that our multi-level system scales extremely well as students were able to accomplish most of the tasks already at home using the provided simulated and augmented environment and finalize the given assignments in time on a limited number of real platforms available at the university. Moreover, due to the open-source design and low-cost components, some

students have also decided to build a robot manipulator by themselves to even deepen their knowledge and to use it for other projects.

**Educational events:** During the educational events organized to promote computer science the contact of the audience with the platform is usually very brief, therefore the message has to be more focused. We have designed a demonstration in which a robot is sorting boxes of two colors. In some cases the robot may fail to grasp the box correctly and the audience members are invited to help the robot by guiding it using simple forward-kinematic control to demonstrate that the task is more difficult than it seems. The demo attracted great interest of the participants (mostly primary school pupils) and it was very well accepted.

## 4 Conclusion

In this paper we have presented a low-cost open-source robotic manipulator extended with the vision sensor. We have described the hardware and software aspects of the platform as well as our current use cases in educational activities. At the moment we have produced an initial batch of six robot manipulators and have tested the system at the university level course on robotics and machine vision as well as in several educational activities. The platform is robust and stable. In the future we would like to upgrade the manipulator to 6 DOF for additional flexibility. We are also preparing to organize a summer school for kids in primary school where we will be able to evaluate the system in different conditions. After improving the production process we plan to release all the plans and the software for the platform so that it may be used as well as contributed to by the community.

## References

1. Adebola, S.O., Odejobi, O.A., Koya, O.A.: Design and implementation of a locally-sourced robotic arm. In: AFRICON (2013)
2. Carter, B.: OpenServo (2008). <http://www.openservo.com/>
3. Cocota, J.A.N., Fujita, H.S., da Silva, I.J.: A low-cost robot manipulator for education. In: 2012 Technologies Applied to Electronics Teaching (TAAE) (2012)
4. Gómez, A.C.: Diseño y puesta en funcionamiento de un brazo robótico imprimible (2012)
5. Karmoker, S., Polash, M.M.H., Hossan, K.M.Z.: Design of a low cost PC interface Six DOF robotic arm utilizing recycled materials. In: 2014 International Conference on Electrical Engineering and Information and Communication Technology (2014)
6. Pol, M.R.S., Giri, M.S., Ravishankar, M.A., Ghode, M.V.: LabVIEW based four DoF robotic ARM. In: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1791–1798 (2016)
7. Pronadeep, B., Vishwajit, N.: Low cost shadow function based articulated robotic arm. In: 2015 International Conference on Energy, Power and Environment: Towards Sustainable Growth (ICEPE) (2015)

8. Qassem, M.A., Abuhadrous, I., Elaydi, H.: Modeling and simulation of 5 DOF educational robot arm. In: 2010 2nd International Conference on Advanced Computer Control (ICACC), vol. 5, pp. 569–574. IEEE (2010)
9. Quigley, M., Asbeck, A., Ng, A.: A low-cost compliant 7-DOF robotic manipulator. In: 2011 IEEE International Conference on Robotics and Automation, pp. 6051–6058 (2011)
10. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software (2009)

# OTO – A DIY Platform for Mobile Social Robots in Education

Thomas Vervisch<sup>1</sup>, Natan Doms<sup>1</sup>, Sander Descamps<sup>2</sup>, Cesar Vandeveldel<sup>1(✉)</sup>, Francis wyffels<sup>2</sup>, Steven Verstockt<sup>2</sup>, and Jelle Saldien<sup>1</sup>

<sup>1</sup> Department of Industrial Systems and Product Design,  
Ghent University, Ghent, Belgium  
{tbvervis.vervisch,natan.doms,cesar.vandevelde,  
francis.wyffels,jelle.saldien}@ugent.be

<sup>2</sup> Department of Electronics and Information Systems,  
Ghent University, Ghent, Belgium  
{sander.descamps,steven.verstockt}@ugent.be

**Abstract.** This paper describes the design of OTO, a do-it-yourself expansion kit for OPSORO (Open Platform for Social Robots), that enables and facilitates the creation of mobile social robots. The expansion kit consists of modular, adaptable building blocks combined with a software toolkit, and is aimed at applications within the maker community, STEM education, and the market for creative inventor kits. Keeping reproducibility and adaptability in mind, the expansion kit can be produced entirely using digital manufacturing technology and low-cost, off-the-shelf components. Using the building blocks offered by this system, users can easily design, build and customize mobile social robots. The software is designed to address a wide range of users by offering different programming options depending on the user's skill and experience. Inexperienced users are offered a graphical programming environment based on Blockly, whereas more advanced users can program their robot using Lua or Python. The OTO toolkit offers a fun and playful context in which a wide range of STEM-related skills are addressed.

**Keywords:** Social robotics · Human-robot interaction · Emotions · Facial expressions · Mechatronics · DIY · Modularity

## 1 Introduction

Recent years have been marked by a rise of digital rapid prototyping techniques [6], such as lasercutting and 3D printing. Access to these technologies has also steadily been improved through FabLabs [8], the huge amount of web resources, open source projects [4, 7], and the support of the “maker movement” [2]. This has resulted in increased possibilities for makers to build their own custom robots [11]. Despite this evolution, there was a lack of open-source and easily adaptable social robotic platforms for human-robot interaction [3].



This has led to the development of OPSORO – Open Platform for Social Robotics [10] – a modular platform that enables human-robot interaction with do-it-yourself (DIY) social robots. This platform uses an innovative modular system design that enables the creation of different embodiments to represent anthropomorphic robots, emphasizing face-to-face communication. However, this platform is limited to stationary robots with facial expressions. This paper describes the process of adapting and extending this platform to enable mobile social robots. The expansion kit, OTO, is complementary with the existing OPSORO platform and has a similar design architecture. The mobile platform consists of modular units that transform a stationary social robot into a vehicle-like robot with facial expressions, similar to cartoon characters (e.g. “Cars” and “Thomas the Train”). Furthermore, OTO has been designed with modular subunits built from off-the-shelf and easy-to-produce components. In this way, users can build and customize different embodiments of social robots. For programming the OPSORO offers multiple options to address a wide range of users with varying programming skills.

## 2 Application in Educational Settings

The topic of robotics is frequently chosen by teachers as the subject of STEM-focused problem-based learning. The reason for this is obvious: as Benitti [1] and Johnson [5] show, teaching robotics is a very effective way to motivate students and integrates many different knowledge domains of the curriculum. Furthermore, it also stimulates students’ social and teamwork skills. As a secondary aspect, robots are something that captures the imagination of many children, which serves as a motivator [5].

Research shows that the design of the robot can improve social HRI in the correct context [9]. Furthermore, the link between the development of a meaningful social interaction between robots and people and the degree of anthropomorphism in the physical characteristics of a robot has been investigated and shows us that this anthropomorphism has a positive impact if used correctly [3].

## 3 Design of the OPSORO Hardware

A rolling platform was chosen to make the social robot mobile. This solution offers a higher efficiency than a walking robot, allowing longer run times with batteries [13]. In addition, the robot does not have to be connected through a tether, allowing it to move freely. A rolling platform also minimizes the complexity of a mobile robot, facilitating the construction process for less experienced users. Differential steering was chosen as the steering method. This method minimizes the mechanical complexity of the robot, thus limiting the number of modules needed. Differential steering makes the robot more maneuverable by enabling it to rotate around its own axis.

As base plates to mount the emotion and driving modules, the same grid as the existing OPSORO kit is used, though with different dimensions. In order

to be able to build up modules lengthwise, such as the driving modules, and in the height, such as the emotion modules, grids mounted perpendicular to each other are used instead of stacking the modules on one another. This enables a larger degree of variation in the positioning of the modules, both in height and in depth. The modules are designed in such a way that they can be fabricated from 3mm ABS sheets using a lasercutter.

The modules are designed in such a way that they can be mounted on every face of the module. This increases the creative possibilities, which is an important aspect of the open source platform. Due to the differential steering, only one type of wheel module is required. The wheel-modules use continuous rotation servos. These servos do not have a limited travel angle and can therefore rotate continuously forward or backwards. They can be controlled in the same manner as regular servos, using the same hardware and software, which makes them ideal for this purpose. The mounted wheels consist of laser-cut slices onto which a K’Nex tire is attached. The slices are in turn attached to the servo horn. For aesthetic purposes, 3D printed or even cardboard wheel caps can also be attached.

The camera module is used as the extra input source for the social robot. It allows the robot to detect and react to events that occur in its environment. Face-detection and tracking algorithms were also implemented within the OPSORO platform, opening new modes of interaction. For example, the robot can say ‘hello’ when a face is detected, or the robot can make eye contact when it detects someone. A Raspberry Pi provides the necessary computational power to complete these tasks.

The power source is an important aspect in order to make the robot completely mobile. The power supply issue of a mobile social robot is solved by using rechargeable NiMH batteries. These batteries provide sufficient power and are resistant to peak currents caused by initializing the servos. These off-the-shelf batteries are safe to use by children and other inexperienced users.



(a) Papercraft Volkswagen van



(b) RC car body



(c) Thermoform

**Fig. 1.** Possible car shells

We propose two options for the bodywork of vehicle-like social robots: thermoformed plastic shells and papercraft bodies. A papercraft body is a quick, easy and cheap way to create a one-of-a-kind social robot body. It can be made to any desired size, shape or sort of vehicle. This kind of body is ideal for workshops or for users who want to create a lot of different characters. A Volkswagen

van prototype, shown in Fig. 1(a), was made to demonstrate the possibilities of this technique.

The papercraft technique can also be used to create a quick mock-up version of a proposed thermoformed body. Keeping with the open source paradigm, templates of these papercraft bodies can also be easily shared online.

For the thermoformed bodies, there are again two possibilities: either a hobby-grade RC car body (Fig. 1(b)) can be purchased, or a thermoformed body (Fig. 1(c)) can be crafted using a DIY CNC-milled mold. The thermoformed bodies can be decorated according to personal taste. Erasable whiteboard markers can even be used for temporary decoration.

## 4 Software Architecture

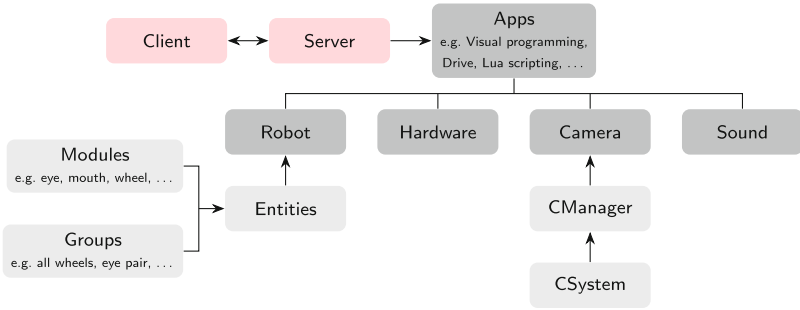
The existing platform code is implemented as a web service running on a Raspberry Pi. The platform provides a simple web interface for the users. The web service offers a variety of apps, each of them having their own purpose. These apps have direct access to the back-end of the platform, which is a virtual module structure that corresponds to the physical building blocks of the robot. An schematic overview of the architecture is given in Fig. 2. In order to implement the new functionalities in the existing OPSORO-platform, a number of changes were made in the software code.

To make the OTO ride, wheel modules were added into the structure. In the OPSORO platform, each physical building block corresponds to a module in code, and each module has to be configured to make the robot work. Because of the modularity of the building blocks, it is important to mirror that modularity in software. After the wheel modules were added to the software, the challenge arose to find a clear and intuitive way to configure and steer the robot.

By adding Entities and Groups, this issue was resolved. The underlying data structure was reorganized as a tree instead of a list of modules. Each Entity represents a node in this tree of modules. Groups are virtual sets of modules, they allow a single instruction to be executed on each module of that group. To further refine which entity in the data tree to control, a module tag system was introduced. These tags are based on the position and type of the entity (e.g.: [eye, right], [eye, left]), and offer a simple way to filter through all modules.

A final architecture change is the action system. Normally, the position of each servo is set individually through a function call. However, this system posed difficulties for children as it can be hard to know which servo needs to be set at which value to perform a certain movement. The action system was designed to provide a simpler way to execute certain movements. Users specify the tags of the components they wish to drive and the action they wish to execute, and the code does the rest. The software selects the right modules and searches for the correct method to call, making it easier for children to program their robot.

By adding a camera, extra possibilities for interaction between the robot and humans were introduced. A face detection and tracking algorithm makes it possible to detect human faces and follow them. The Viola-Jones algorithm [12]



**Fig. 2.** Schematic overview of the software architecture



**Fig. 3.** Two DIY mobile social robots created using the OTO platform.

is used for face detection. In order to guarantee the necessary speed and stability of the platform, the image processing algorithm is executed in a separate thread. This way, the computer vision tasks will not slow down the rest of the robot.

## 5 Conclusion and Future Work

In this paper, the design of a new social robot platform, OTO, is described. This social robot distinguishes itself from other social robots by providing an open source and modular building kit, allowing the user to fully customize their robot. OTO builds upon the existing OPSORO platform, but distinguishes itself by providing a mobility aspect and by linking emotion expressiveness with mobility. The software platform is easily accessible through a web interface and can be programmed using visual building blocks, opening new opportunities for children to build and interact with their robots.

The added dimension of mobility introduces many new possibilities for the OPSORO platform. It can be used to add new emotional behavior or to enable new functionality. The sensing capabilities of stationary robots are limited because they can only perceive what is nearby. By adding a mobile aspect to the platform, sensing capabilities are greatly expanded. For instance, the robot can follow a person or object, and can continuously gather new input from its environment. This input can be used to generate new behavioral or emotional outputs, as specified by the users’ scripts.

The platform emphasizes a user-friendly interface that allows children and inexperienced programmers to use it. Still, the system is sufficiently flexible so that system developers and researchers can adapt it to specific tests and

experiments. Because of the modular design, the platform can be easily expanded with new sensors and actuator modules. For these modules, new software blocks can also be developed to make them effortless to operate. In addition, due to the flexible and unspecified structure of the robot, new aesthetics (papercraft body, plastic shells, LEGO blocks, ...) can be designed and shared. Finally, the research was implemented in two working prototypes, as shown in Fig. 3.

## References

1. Benitti, F.B.V.: Exploring the educational potential of robotics in schools: a systematic review. *Comput. Educ.* **58**(3), 978–988 (2012)
2. Chorianoopoulos, K., Jaccheri, L., Nossum, A.S.: Creative and open software engineering practices and tools in maker community projects. In: Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS 2012, p. 333, January 2012
3. Duffy, B.R.: Anthropomorphism and the social robot. In: *Robotics and Autonomous Systems*. vol. 42, pp. 177–190 (2003)
4. Hars, A., Ou, S.: Working for free? motivations of participating in open source projects. *Int. J. Electron. Commer.* **6**(3), 25–39 (2002)
5. Johnson, J.: Children, robotics, and education. *Artif. Life Robot.* **7**(1–2), 16–21 (2003)
6. Kolarevic, B.: Digital fabrication: manufacturing architecture in the information age. In: Proceedings of the Twenty First Annual Conference of the Association for Computer-Aided Design in Architecture, pp. 268–278 (2001), ISBN 1-880250-10-1
7. Lerner, J., Tirole, J.: The open source movement: key research questions. *Eur. Econ. Rev.* **45**(4–6), 819–826 (2001)
8. Mostert-Van Der Sar, M., Mulder, I., Remijn, L., Troxler, P.: FabLabs in design education. In: Proceedings of the 15th International Conference on Engineering and Product Design Education: Design Education - Growing Our Future, EPDE 2013, pp. 629–634, September 2013
9. Sabanovic, S., Michalowski, M.P., Simmons, R.: Robots in the wild: observing human-robot social interaction outside the lab. In: International Workshop on Advanced Motion Control, AMC 2006, pp. 576–581 (2006)
10. Vandavelde, C., Saldien, J.: An open platform for the design of social robot embodiments for face-to-face communication. In: Proceedings of the 11th International Conference on Human-Robot Interaction, Christchurch, New Zealand, pp. 287–294 (2016)
11. Vandavelde, C., Vanhoucke, M., Saldien, J.: Prototyping Social Interactions with DIY animatronic creatures. In: Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction, TEI 2015, pp. 509–512. ACM, New York (2015)
12. Viola, P., Jones, M.: Robust real-time face detection. *Int. J. Comput. Vis.* **57**(2), 137–154 (2004), <http://link.springer.com/article/10.1023/B:VISI.0000013087.49260.fb>
13. de Wit, C.: Trends in mobile robot and vehicle control. In: Siciliano, B., Valavanis, K.P. (eds.) *Control Problems in Robotics and Automation*, pp. 151–175. Springer, Heidelberg (1998)

# An Elementary Science Class with a Robot Teacher

Alex Polishuk<sup>1</sup> and Igor Verner<sup>2</sup>(✉)

<sup>1</sup> Israel National Museum of Science,  
Technology and Space (MadaTech), Haifa, Israel  
alex.polishuk@madatech.org.il

<sup>2</sup> Faculty of Education in Science and Technology,  
Technion – Israel Institute of Technology, Haifa, Israel  
ttrigor@technion.ac.il

**Abstract.** This paper considers a science lesson given through mediation of the humanoid RoboThespian to groups of school students (grades 5–7) at the science museum MadaTech. The lesson included theoretical explanations, hands-on experiments, and a knowledge quiz, all instructed and managed by means of the robot-teacher through programmed behaviors and remote teleoperation. We present the lesson design and implementation in two settings with different characteristics of teacher immediacy, discuss students’ outcomes and perceptions. The study shows the feasibility of using robotic assistants in science classes and uncovers the factors that influence learning in such settings.

**Keywords:** Student-robot interaction · Robot-teacher · Science lesson · RoboThespian · Elementary school · Classroom setting

## 1 Introduction

Recent progress in intelligent robotics has opened new opportunities for learning through interaction with robots. Student-robot interaction (SRI) is a new, rapidly growing direction of research which considers how interaction with a robot can facilitate and enhance human learning [1, 2].

Interaction with robots that imitate human appearance and behavior has attracted particular interest of researchers [3, 4]. The majority of the studies of learning interactions with different human-like robots explore their use by individual students to learn a second language, elementary mathematics, and additional subjects [5–7]. Less investigated are settings in which a science lesson, mediated by such robot, is delivered to a group of students.

The HRI research group in Tokyo University of Science (TUS) initiated experiments in which science classes on different topics were given to elementary school children through teleoperation of the android SAYA in the role of the teacher [6]. In 2011 the TUS group invited our Technion group to conduct a collaborative study.

In this study our group implemented a pilot version of the lesson “The function and law of the lever” with RoboThespian at the MadaTech museum [8]. For the lesson we translated into Hebrew and adapted the instructional materials developed by the TUS

group, designed and programmed teaching behaviors based on the rich functionality of the RoboThespian, and implemented the lesson with a small group in an informal studio.

MadaTech has a Department of Education which provides school classes that visit the museum with different outreach activities. Our group closely collaborates with the MadaTech Gelfand Center for Model Building, Robotics & Communication in the development of new approaches to learning through interaction with robots [9]. Motivated by the positive results of the pilot lessons with RoboThespian, the MadaTech Department of Education asked our Technion group to further develop the lesson with the aim to use it on a regular basis for class visits. Thus, a new study was initiated in which we redesigned the lesson to improve the learning environment and the instructional strategy implemented in the lesson.

The design of the lesson and its learning outcomes are presented in our recent article [10]. In the current paper we further elaborate these issues based on the constructive reviews of our work. We describe in more detail the research method, expand the analysis of learning outcomes and perceptions, and discuss a broader view of the factors influencing robot-teacher immediacy.

Our research motivation evolved with the development of the educational project with RoboThespian. Initially it was to prove the feasibility of using the robot to play the role of a teacher and get data about students' perceptions of the robot's preprogrammed and teleoperated behaviors. Then, when the lesson was implemented in two different classrooms, our desire has become to understand how the classroom organization can influence the outcomes of learning through interaction with the robot.

The study presented in this paper is the first step in this direction. In the following sections we describe RoboThespian, the approach used for programming teaching behaviors, the two classroom settings and the lesson. Then we report results of the evaluation study and make conclusions.

## 2 RoboThespian

RoboThespian is a 175 cm tall, 33 kg, 24 DOF humanoid robot intended for interaction and communication with people in public environments. The robot is visibly presented at the manufacturer's website <https://www.engineeredarts.co.uk/>. The robot is made almost entirely of white aluminum with pneumatic artificial muscles (McKibben muscles), DC motors, and passive spring elements to simulate human body motion. The robot can be controlled in two modes: execution of preprogrammed scripts in an open loop via the operation kiosk, and teleoperation using a special user interface. In the teleoperation mode, the operator can command the robot to use simple movements like turning, basic gestures, and live-interaction. There is a small bank of preprogrammed responses that the operator can use to interact with the public. In the preprogrammed mode, the robot performs scenes autonomously, incorporating all of its features.

RoboThespian has modules and functions that can be utilized to mimic human locomotion and behavior. DC motors and artificial muscles are used to create human-like full arm and abdominal movements. Using head movements, the robot observes its surroundings through a high-resolution RGB camera mounted on its forehead. The robot also has communication capabilities, including: gaze expressions

through blinking and squinting of the LCD eyes, speech based on pre-recorded audio files, and real time lip synchronization. The head is equipped with multi-colored LEDs which can color the face and are utilized to convey mood and emotional expressions.

### 3 Programming Teaching Behaviors

Designing a robot-teacher behavior is a challenging task of imitating the real teacher's functions of conveying knowledge, engaging students in learning and managing the class. To implement this role, it requires developing robot interactive behaviors composed of verbal and non-verbal communication cues such as tone of voice and intonation, facial expressions, gaze, gesture, and bodily movements [7, 11].

Kennedy [7] and Verner et al. [10] offered to develop robot-teacher's behaviors, taking into account the pedagogical concept of teacher immediacy to characterize student's perception of the psychological distance between the teacher and the student [12, p. 65]. Educational literature suggests concrete recommendations for teachers on how to enact immediacy behaviors that positively affect learning in class [13]. We used them when programming RoboThespian.

The autonomous behaviors were programmed using a graphical user interface (GUI) based on the 3D animation software Blender with the soundtrack imposed on the animation timeline in order to synchronize speech and motion. In order to create lively scenarios, we defined sequences of essential key-frame postures through a trade-off between the number of key-frames and the length of the scene.

Our strategy to design a suitable scene was to place meaningful gestures at information peak key-frames, and to add only small motion increments before changing to the next key-frame. Another important parameter that had to be considered was the movement speed between key-frames. Although the robot could move quickly between different gestures, we chose to limit its speed for safety reasons and in order to feature a more human character.

We programmed robot-teacher's gestures with reference to the nonverbal behaviors recommended for real teachers [12]:

- Moderate gestures integrated temporally with the speech they accompany.
- Occasional turning of the robot torso and head, and gaze shifting from side to side, in order to raise awareness that the robot is communicating with the entire classroom.
- Turning the torso and head, and directing the gaze toward the slides projected on the screen, in order to emphasize the importance of their content.
- Hand gestures like finger counting, pointing, opening arms in invitation, etc., in order to add subtle dramatization to the speech.

While programming gestures of RoboThespian, we strove for maximal resemblance to a human teacher's behavior; in programming facial expressions, we followed a different approach and used the facial capabilities of the robot to display expressions that are characteristically non-human, in order to avoid the Uncanny Valley effect [14].

Based on widely shared perception of emotional responses to different colors [15], we changed the color of the robot's face to express different emotions. That is, orange



face represented warm response, red stated for anger, and blue for sadness. Another use of facial expressions was to draw the students' attention to the colors of objects presented by the robot, by staining its face with the same colors. A display of signs was programmed into the robot's "eyes", such as "×" to express puzzlement, "stop" to call to order, or "♥" to show pleasure from a correct answer given by a student.

## 4 Classroom Setting

As known, the classroom setting strongly influences the educational process [16]. The effect of the classroom on the student-robot interaction has not been discussed in literature. In this study we implemented the lesson with the same robot-teacher in two different classrooms described below.

The first classroom was a studio originally intended for informal meetings. To adapt the studio for the lesson, we placed RoboThespian in front of three tables, with four seats at each (Fig. 1). One microphone was used to receive students' questions and transmit them to the operator (an experienced museum mentor). Figure 2 shows the operator controlling the lesson from the control room.



**Fig. 1.** The lesson in the studio

The limited capacity of the studio (until 12 students) caused difficulties in providing the lesson to school classes which included more students and required to split them. To answer the problem, the MadaTech Department of Education provided us with a larger classroom which is described below.



**Fig. 2.** Operator in the control room

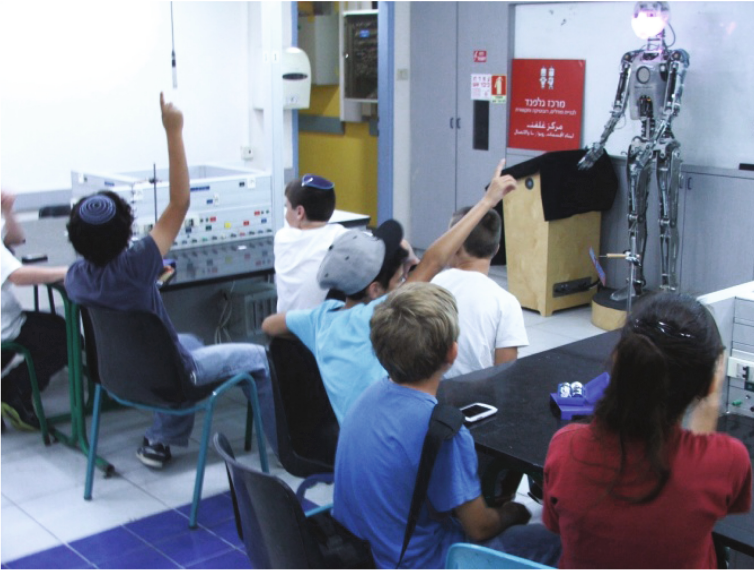
The second classroom was the interactive lab – one of the departmental science laboratories adapted for lessons assisted by RoboThespian (Fig. 3A).

The interactive lab had four tables with six workplaces at each table, as shown in the scheme (Fig. 3B). Microphones and video cameras were attached to each of the tables, providing real-time data. The classroom response system received students' answers to questions asked by the robot-teacher and operatively transmitted them to the operator. The operator's workplace was equipped with the robot control PC to launch preprogrammed behaviors and communicate using the lip-sync option, a PC to run the PPT slides, a PC to receive and analyze students' responses, a monitor to display video streams from the cameras, a headset to listen students' utterances from the microphones, and a microphone to speak with the class directly.

Using the advanced communication tools of the interactive lab, we extended the repertoire of robot-teacher behaviors by those related to asking multiple-choice questions and responding to students' answers. To summarize the aforesaid, the characteristics of the two classrooms are compared in Table 1.

One can see that the load on the operator for managing the lesson through robot teleoperation in the interactive lab is much higher, as the data stream from the students' workspace is more than four times larger than that in the studio.

A.



B.

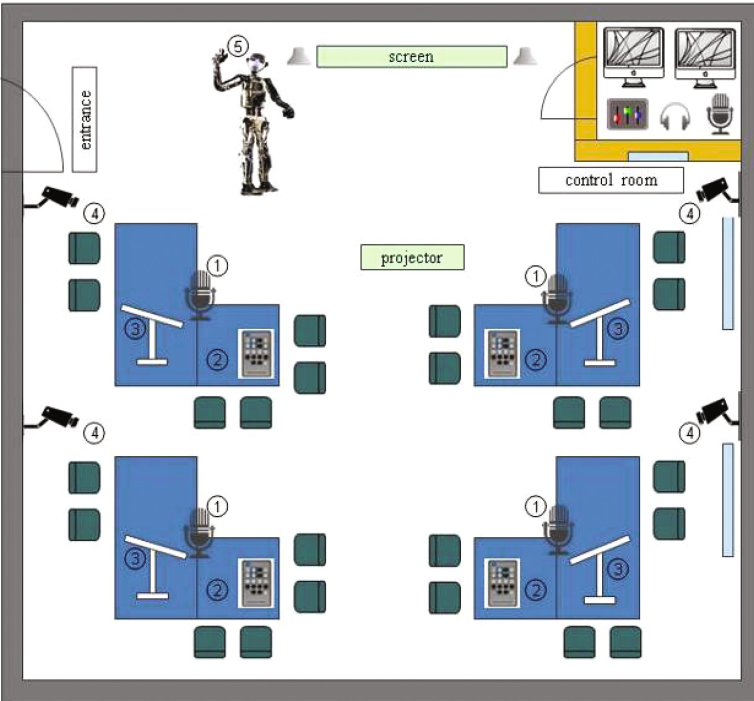


Fig. 3. A. The lesson in interactive lab; B. Layout of the lab

**Table 1.** Classroom characteristics

Setting	Capacity	Response system	Mic	Cam	A/V interface
Studio	≤ 12	No	1	1	Single
Interactive lab	≤ 24	Yes	4	4	Multiple

## 5 Evaluation Study

When evaluating the lesson, we focused on the three specific questions:

1. Are there indications that elementary school students participating in the science lesson mediated by a humanoid robot acquire and understand the concepts taught?
2. What are students' perceptions of the lesson and the robot- teacher?
3. Are there differences in learning results and perceptions of the lesson given in the two classrooms?

We hypothesized that the students will acquire and understand the lever concepts taught in the lesson. We also conjectured that learning outcomes and students' perceptions of the robot-teacher and the lesson will be higher in the interactive lab.

The study sample included 189 students (15 groups) from grades 5–7 (11–13 years old) who took the lesson at MadaTech in the period of 2013–2014. Among them 118 students learned it in the studio and 71 in the interactive lab.

The study was organized as a quasi-experiment. At this stage we did not intend to compare the lesson given through mediation of the robot with the traditional lesson and, therefore, did not have a control group. Also, we did not pretest students' knowledge to measure the learned gain from the lesson, but only looked for indications of the acquisition and understanding of the studied concepts. On the conditions of this museum experiment, we could not get more information about the students and, therefore, did not compare learning results in different school groups. The instruments used and results related to the three research questions are presented below.

### 5.1 Acquired Concepts

The acquisition and understanding of the concepts studied in the lesson was evaluated by means of the quiz which included nine questions. The first three questions asked to identify leverage points. Questions 4 and 5 checked understanding of the dependency between the effort and the lever arm distance. In the scheme of the lever supplemented to the questions, the names of its three main points (effort, fulcrum, and load) were not given but marked by A, B, and C. The questions asked to find the missing word in the following two sentences:

*If the distance AB is \_\_\_\_\_ then less effort is needed to lift up the load.*

*If the distance BC is \_\_\_\_\_ then less effort is needed to lift up the load.*

Question 6 asked how to counterbalance a given weight put on one side of the lever. Question 7 tested understanding of the law of the lever. The last two questions were on the use of the lever in two every-day life situations: wallet holding and seesaw swinging.

The average score on the quiz for all the students was 75.2% (SD = 16.8). This shows that the majority of the students successfully passed the quiz and demonstrated knowledge and understanding of the lever concepts.

To get an indication of the students’ progress in understanding the lever concepts, we asked two control questions similar to #4 and #5 from the quiz before the practice with the balances. The progress was examined for students from two classes of 28 and 17 learned in the interactive lab. The mean scores are given in Table 2.

**Table 2.** Mean scores of repeated questions

Time	Class 1 (N = 28)		Class 2 (N = 17)	
	Mean (S.D.)			
	Q4	Q5	Q4	Q5
Time-1	89 (31)	68 (47)	94 (24)	56 (51)
Time-2	90 (31)	86 (35)	88 (33)	88 (34)

We used the repeated measures ANOVA with one within-subjects factor (time) and one between-subjects factor (school class). The test revealed a significant main effect of time related to question 5  $[F(1, 42) = 7.6, p < 0.01, \eta_p^2 = 0.153]$ . There was not a significant main effect of time for question 4 and for the interaction between the time and school class factors. The statistical analysis indicates the significant progress of the students in both classes in question 5 and about the same high scores in question 4.

**5.2 Students’ Perceptions**

To answer the second and third research questions, we looked at students perceptions of the robot-teacher and the lesson. For this purpose we used a questionnaire that included seven multiple choice questions presented in Table 3 and an open question discussed later on. The seven questions were selected from the questionnaire that we developed in the collaborative study with the TUS group [8], based on the Godspeed Human-Robot Interaction Questionnaire [17]. As the language of the Godspeed instrument is not appropriate for children [7], we rephrased the questions. Results of the questionnaire from the studio and from the interactive lab are given in Table 3.

We note that the absolute majority of the students (83%) liked the robot-teacher and perceived it as friendly (81%). The majority of 73% characterized the robot as responsive. Less, but more than half of the students pointed that RoboThespian was energetic (63%) and behaved like a real teacher (65%). With regard to the lesson, for most of the students it was pleasant (87%), and the absolute majority found it interesting (78%).

In addition to the seven questions, an open question asked the students about limitations of the robot-teacher. 63 studio students and 60 interactive lab students answered the question. The more often mentioned limitations related to: eye contact (20%) and delayed response (21%).

**Table 3.** Students' perceptions: percentage of positive answers

Questions	All students	Studio	Interactive Lab
Was the robot responsive to the class?	73	77	64
Was the robot friendly during the lesson?	81	86	70
Did the robot behave like a real teacher?	65	73	49
Was the robot energetic during the lesson?	63	78	34
Do you like the robot- teacher?	83	82	85
Was the lesson with the robot- teacher pleasant?	87	84	94
Was the lesson with the robot- teacher interesting?	78	76	81

### 5.3 Differences in Lesson Outcomes

The average score on the quiz in the interactive lab was 79.6% (SD = 12.9) and in the studio 72.6% (SD = 18.3). The two-tailed T-test indicated that grades in the lessons interactive lab were significantly higher than that in the studio:  $t(182) = -3.09$ ,  $p < 0.01$ . We attribute the better quiz results in the interactive lab to the use of the response system to operatively test students' understanding during the lesson and provide additional guidance when needed. To help students comprehend the quiz problems we reformulated the quiz problems as multiple choice questions. As mentioned in Sect. 4, we extended the RoboThespian's behavior in the interactive lab so that the robot-teacher presented the multiple-choice questions of the quiz verbally and emphasized their meaning by intonations and gestures.

More students in the studio than in the interactive lab perceived it as responsive, friendly, energetic, and behaving as a real teacher. We explain this by the different conditions in the two classes in relation to the number of students, their proximity to RoboThespian, and eye contact with the robot-teacher. In education, classes with up to 20 students, sitting at a distance of 1.2–3.7 m from the teacher and within eye contact are considered most favorable for teaching communication [18]. The lessons in the studio were given to up to 12 students who sat near three tables at distances from 1.8 m to 2.4 m in full eye contact with the robot-teacher. The lessons in the lab were given to up to 24 students, 16 of them sat at a distance exceeding 3.7 m and the majority of whom had limited eye contact with the robot.

Unlike the perceptions of the robot, those of the lesson were higher in the interactive lab than in the studio. Our explanation is that the upgraded audio-visual system the classroom response system used in the lab provided better communication between the students and the operator. Namely, the operator was able to observe the behavior of every student and react to questions, answers and comments when operating robot interactions.

Differences were also in responses about the limitations of the robot-teacher: limited eye contact was noted by 8% in the studio versus 36% in the interactive lab, while robot's delayed response was mentioned by 23% in the studio and 15% in the interactive lab. The differences give additional evidence of a shorter psychological distance between the students and the robot in the studio.

## 6 Discussion and Conclusion

The study shows that an elementary school science class can be mediated by a humanoid robot such as RoboThespian, so that explanations, examples, assignments and correct solutions are given in the autonomous open loop mode, while other parts of the lesson, requiring robot-teacher responses, are given through the teleoperation mode. When programming the robot-teacher, we tended to implement teacher immediacy behaviors, both verbal and nonverbal.

As indicated by the study, the students acquired and understood the concepts taught and had positive perceptions of the lesson and of the robot-teacher.

We conducted the lessons on the same topic mediated by the same robot in two settings with different characteristics of teacher immediacy and found certain differences in learning outcomes and perceptions. In this study we did not intend to choose the best setting, but just uncover the factors affecting the teacher immediacy in each of them, as preliminary work to develop criteria for evaluation robot-teacher immediacy in different learning settings.

The study highlights that besides the communication cues demonstrated by the robot-teacher, its psychological distance to the students depends on the classroom setting characteristics such as the number of students and their proximity to the robot. Rational choice of these features and characteristics can make lessons mediated by robots more effective and enhance students' engagement.

We consider the recent study of Kennedy [19] who proposed non-verbal and verbal immediacy scales and used them for evaluation of a face-to-face learning interaction with robot Nao, as a work in the same direction. Further development of the scales is needed to make them applicable for our case. Educational researchers point to the difficulties in evaluation of teacher immediacy based on students' feedback. Anderson [13] notes that students typically perceive nonverbal immediacy without being aware of all its components. Richmond et al. [20] related to unsuccessful attempts to develop on the basis of student opinions teaching immediacy scales that answer the criteria of validity and reliability. She pointed that such scales were developed based on self and expert evaluation.

The repeated questions that people ask us and possibly other colleagues: Do you intend to replace teachers by robots? Is there any special value in learning with robot-teachers? To the first question we answer that the robots are intended to assist teachers and extend their possibilities for teaching classes. To the second question we tell that on one of the recent lessons a schoolboy jumped on his feet and shouted: "Robot, you're cute! I love you!" We are teachers with many years of experience, but no one ever jumped up to express such feelings to us.

Besides helping teachers, the foremost human-like robots, as mediators of educational processes in science museums and other public spaces, can contribute to the development of public understanding of robotics and the culture of human-robot communication.

**Acknowledgments.** The study was supported by the Technion Autonomous Systems Program and by the MadaTech Gelfand Center for Model Building, Robotics and Communication.

## References

1. Fong, T., Nourbakhsh, I., Dautenhahn, K.: A survey of socially interactive robots. *Rob. Autonom. Syst.* **42**(3–4), 143–166 (2003)
2. Slagen, L., Keulen, H., Gravemeijer, K.: What pupils can learn from working with robotic direct manipulation environments. *Int. J. Technol. Des. Educ.* **21**, 449–469 (2011)
3. Kanda, T., Hirano, T., Eaton, D., Ishiguro, H.: Interactive robots as social partners and peer tutors for children: a field trial. *Hum.-Comput. Inter.* **19**(1), 61–84 (2004)
4. Polishuk, A., Verner, I.: Interaction with animated robots in science museum programs: how children learn? In: 7th ACM/IEEE International Conference on Human-Robot Interaction, Boston, pp. 265–266 (2012)
5. Mubin, O., Stevens, C., Shahid, S., Mahmud, A., Dong, J.: A review of the applicability of robots in education. *Technol. Educ. Learn.* **1**, 1–7 (2013)
6. Hashimoto, T., Kato, N., Kobayashi, H.: Development of educational system with the android robot SAYA and evaluation. *Int. J. Adv. Rob. Syst.* **8**(3), 51–61 (2011)
7. Kennedy, J., Baxter, P., Senft, E., Belpaeme, T.: Higher nonverbal immediacy leads to greater learning gains in child-robot tutoring interactions. In: 7th International Conference on Social Robotics, Paris, pp. 327–336 (2015)
8. Hashimoto, T., Kobayashi, H., Polishuk, A., Verner, I.: Elementary science lesson delivered by robot. In: 8th ACM/IEEE international conference on Human-Robot Interaction, Tokyo, pp. 133–134 (2013)
9. Verner, I., Polishuk, A., Klein, Y., Cuperman, D., Mir, R.: A learning excellence program in a science museum as a pathway into robotics. *Int. J. Eng. Educ.* **28**(3), 523–533 (2012)
10. Verner, I., Polishuk, A., Krayner, N.: Science class with RoboThespian: using a robot teacher to make science fun and engage students. *IEEE Robot. Autom. Mag.* **23**(2), 74–80 (2016)
11. Liu, P., Glas, D., Kanda, T., Ishiguro, H., Hagita, N.: How to train your robot: teaching service robots to reproduce human social behavior. In: 23rd IEEE International Symposium on Robot and Human Interactive Communication (ROMAN), Edinburgh, pp. 961–968 (2014)
12. Richmond, V.P.: *Teacher Nonverbal Immediacy, Uses and Outcomes. Communication for Teachers*, chap. 6, Allyn, Bacon, Boston (2002)
13. Anderson, P.: Immediacy, *Encyclopedia of communication theory*. In: Littlejohn, S., Foss, K. (eds.) SAGE Publications Inc, pp. 501–503 (2009)
14. Walters, M., Syrdal, D., Dautenhahn, K., Boekhorst, R., Koay, K.: Avoiding the uncanny valley: robot appearance, personality and consistency of behavior in an attention seeking home scenario for a robot companion. *Autonom. Rob.* **24**, 159–178 (2008)
15. Zammito, V.: The expressions of colours. In: *Changing Views: Worlds in Play*. Digital Game Research Association (2005)
16. Fraser, B.: Classroom learning environments. In: Abell, S., Lederman, N. (eds.) *Handbook of Research on Science Education*, chap. 5, pp. 103–124. Lawrence Erlbaum, Mahwah (2010)
17. Bartneck, C., Croft, E., Kulic, D.: Measuring the anthropomorphism, animacy, likeability, perceived intelligence and perceived safety of robots. *Int. J. Soc. Rob.* **1**(1), 71–81 (2009)
18. Hall, E.T.: *The Hidden Dimension*, pp. 121–124. Doubleday, Garden City (1966)
19. Kennedy, J., Baxter, P., Senft, E., Belpaeme, T.: Using immediacy to characterise robot social behaviour in child-robot interactions. In: 1st Workshop on Evaluating Child-Robot Interaction, 7th International Conference on Social Robotics, Paris (2015)
20. Richmond, V.P., McCroskey, J.C., Johnson, A.D.: Development of the nonverbal immediacy scale (NIS): measures of self-and other-perceived nonverbal immediacy. *Commun. Q.* **51**(4), 504–517 (2003)



# Design of Robot Teaching Assistants Through Multi-modal Human-Robot Interactions

Paola Ferrarelli<sup>(✉)</sup>, María T. Lázaro, and Luca Iocchi

DIAG, Sapienza University of Rome, Rome, Italy  
{ferrarelli,mtlazaro,iocchi}@diag.uniroma1.it

**Abstract.** The interest on introducing robots in schools has increased significantly in recent years. Robots in these environments are managed by educators who design teaching activities where the students can consolidate the knowledge acquired in the classroom by interacting with the robot.

In this context, the use of multiple modalities of communication can become a determining factor to achieve the success of the interaction and a better learning experience. However, the design of such multi-modal interactions can be a complex and time-consuming process, specially for teachers lacking of technical expertise.

In this paper, we propose a formalism for the description of multi-modal interactions based on the use of *interaction templates* which facilitates the design and management of the multi-modal behaviour by non-expert users (i.e., teachers). We provide an example of application of our approach on the educational context, using an autonomous robot as a teaching assistant, showing the usability and extendability of our system.

**Keywords:** Human-robot interaction · Multi-modal interaction · Interaction design · Learning with robots · Teaching with robots

## 1 Introduction

In the last years, the interest on introducing robotics in schools has increased significantly due to the possibilities it offers from the educational perspective. Robots in school become an interactive and more visually appealing educational tool attracting the interest of students who become active subjects during the learning process, instead of listening passively to a lesson. Through the use of robots in the classroom, the students can reinforce certain contents explained during the lessons in different ways, for example, by developing themselves real applications that can be executed on the robot or through interactive sessions with the robot. The latter is our application domain, for which we propose a framework for the generation of multi-modal interfaces for human-robot interaction.

In this context, the key actors of the educational process, (i.e., the teachers and the students) are usually non expert in the usage of the robotic tools available. As a consequence, the robotic tools to be used at school must have certain

desirable features, such as easy to be used by the teachers in order to design and prepare the lesson with the robot, and intuitive and easy to understand by the children. This usability can be increased by using multi-modal human-robot interactions (by combining for example, speech, graphical user interfaces, robot motion, etc.). However, when a multi-modal interaction is desired, the difficulty in designing and developing the interactions significantly increases.

This paper contributes to an easy definition and development of many short-term multi-modal interactions through the definition of a language for defining *interaction templates*. More specifically, a high-level formalism is defined to describe interactions in terms of the flow and of the content and the modalities of the basic communication actions. The interactions are described at two levels: (i) a high-level description that provides an abstraction with respect to particular subjects and to the actual modalities, phrases, language, etc. used during the interaction (thus called *interaction templates*), (ii) a formal declarative description of the specific actions that are included in the interaction templates. The main advantages in using a formal language for specifying interaction behaviors are: *compactness*, notwithstanding the ability of representing multiple and complex interactions, *easy-of-use*, because it does not require knowledge about the internal implementations, *portability*, because of the abstraction with respect to the specific device, and *maintainability*, since such compact representation with a clear semantics allows for an easy management, updating, debugging, etc. The described method has been fully implemented and in this paper we propose a use case of using our framework to design and implement a school lesson where an autonomous robot supports the teacher during the explanation of a subject.

## 2 Related Work

Recently, the use of robots in classrooms, supporting teaching and education, spans from kindergarten to elementary and middle schools. A systematic review undertaken on published literature in seven international online bibliographic databases over a period of 10 years was done by Benitti [1]. She found, among others, the following interesting conclusions: the robot is used to teach technical subjects, such as robot assembling and programming; few experiments were done with home-made robots, while most were using Lego robotic kits; finally, most of the robotics activities were done during summer camp and/or after-school programs. The last data was confirmed also by Mubin et al. [9], that made a review of the applicability of robots in education. They remark that *efforts must be devoted not only to the development of robotic hardware and software for education but also to the design of learning material and appropriate curriculum and to the role of the teacher, that is directly linked to the role the robot plays in the learning activity*. Moreover, in the conclusion, they emphasize that the role of the robot is not to replace human teachers but being a stimulating, engaging and instructive tool, through which the interest of the students versus the curricula subject can be increased [8]. As experimented by Walker and Bursleson [12] students seek activities that provide them with an

appropriate level of challenge, feelings of discovery, opportunity for physicality, and a sense of responsibility for the robot. About student motivation, Kanda [5] studied the effects of the use of social behaviors in an experiment with children taught uniquely by a Robovie robot using a learner-centered approach. Although children showed better social acceptance and motivation in the first two lessons, this effect disappeared after the novelty passed. This suggests the importance of the role of the teacher inside the classroom, in particular influencing how students accept new tools, as observed by Hussain [4] and Lindh [7]. However, as reported by the Teaching Profession in Europe [3], there still exists a technology gap in teaching. This report states that European teachers express moderate and high professional development need levels, especially in relation to 'Information and Communication Technology (ICT) skills for teaching' (57%) and 'new technologies in the workplace' (53%) topics (2013 data). These needs seem to be experienced almost uniformly, regardless of the subject they teach. This lack of technology skills could represent an obstacle to the implementation of the lessons using new technologies tools, like robots are, transmitting, as a consequence, negative attitude to the students towards them [11]. High-level formalisms have been used in contexts such as Human-Computer Interaction (HCI) [2, 10] but not in the educational robotics context. The use of a high-level formalism can reduce the efforts required to describe the human-robot interactions where the use of low-level formalisms could be hard to learn by teachers. We propose our methodology as a bridge between robots in education and high-level formalisms for multi-modal Human-Robot Interactions (HRI).

### 3 Methodology

We propose the following methodology to design school lessons in which an autonomous social robot acts as teaching assistant. In general, we propose to divide the lesson in  $n$  time slots in which we specify the actions that are realized by teacher, robot and students involved in the lesson. These actions can be arranged in a table as shown in Table 1.

We think the teacher should have an active role in supporting and managing the student's learning experience. Typical teacher actions  $A_{T_i}$  include: managing the time schedule, asking the robot to introduce itself and starting the test sessions, managing the student-student and student-robot interaction. The robot, like a real teacher assistant, does actions  $A_{R_i}$  that include: speaking to the classroom, calling the students, moving toward them, asking questions, displaying images, videos or statistical data. Finally, students actions  $A_{S_i}$  include: listening to the robot, answering to the questions, discussing between them and with the teacher, asking the robot about more explanations.

While it is sufficient to describe the actions performed by the teacher and the students in natural language, the robot actions  $A_{R_i}$  must be described in a formal way, since they must be executed by the robot software. The use of such a formalism provides a level of abstraction that hides the technical details to non-expert users and facilitates its posterior implementation and execution.

**Table 1.** Organization of a lesson with a robot teaching assistant.

Time slot	Teacher	Robot	Student
1	$A_{T_1}$	$A_{R_1}$	$A_{S_1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n$	$A_{T_n}$	$A_{R_n}$	$A_{S_n}$

## 4 Formalism

The formalism proposed in this paper to describe HRIs focus on interactions in which the robot takes the initiative of formulating questions and the answers are given by the users. These types of interactions are applicable to a variety of short-term interactions to provide information based on user choices or quizzes. Other forms of interaction are under investigation and will be considered in a future work.

Our approach for the generation of HRIs is based on the definition of *interaction templates*. As in the case of the templates existing in programming languages, the interaction templates allow to describe generic interactions that can be later *instantiated* according to a concrete situation.

Let us formally define an interaction template  $\mathcal{T} = \langle \sigma_1, \dots, \sigma_n \rangle$  as a sequence of actions  $\sigma_i, i \in 1 \dots n$  with

$$\sigma_i = \begin{cases} \emptyset: \text{no action} \\ a_i: \text{robot action} \mid \text{interaction} (state, ask, answer) \\ \phi ? \mathcal{T}_1 : \mathcal{T}_2 \\ \text{choice}(\mathcal{T}_1, \dots, \mathcal{T}_n) \\ \text{LABEL} \\ \text{GOTO LABEL} \end{cases}$$

The conditional operation  $(\phi ? \mathcal{T}_1 : \mathcal{T}_2)$  allows to enable different interaction templates during the course of the interaction. The boolean expression  $\phi$  is formulated over a set of variables whose meaning may comprise the result of both robotic or interaction routines. The syntax of this operator is the following: if  $\phi$  evaluates as TRUE, the interaction template  $\mathcal{T}_1$  is activated, otherwise  $\mathcal{T}_2$  is executed. Similarly, the choice operation allows to select one template among a set  $(\mathcal{T}_1, \dots, \mathcal{T}_n)$  based on a random, predefined or learnt policy. Finally, LABEL and GOTO LABEL allows to annotate particular actions and introduce loops in the interaction. Notice these operators produce an interaction graph which makes the formalism suitable for generating an equivalent Petri-Net Plan (PNP) [14] to manage the interaction, as explained in the next section.

In contrast to HCIs, the flow of an HRI includes both *robotic* (e.g., motion and perception) and basic communication actions. Our template-based interaction design considers three main interaction actions - *state, ask, answer* - whose syntax is defined below:

- *State* actions are used by the robot to communicate some fact to the user.

$$state < \text{Type}, P_1, \dots, P_k >$$

where **Type** is the type of the state action and  $P_1, \dots, P_k$  are the (possibly empty) instantiated parameters (i.e., ground values needed to instantiate the statement).

- *Ask* actions are used to denote questions from the robot to the user. They have a similar structure to the *state* action and, in addition, require to specify a set of possible answers  $S$ .

$$ask < \text{Type}, P_1, \dots, P_k, S >$$

- *Answer* actions represent answers given by the user and always refer to the last *ask* action in the template.

$$answer < X >$$

where  $X$  is the name of a variable that will be instantiated with a value corresponding to the choice of the user. After the user interaction, the variable  $X$  will be assigned to one of the values specified in the set  $S$  related to the last *ask* action.

The set of possible answers  $S$  to a given question is predefined. This design choice allows for increasing robustness of recognition and correctness of interaction, since allowing for a completely open set of answers is still not manageable by current technologies. This set  $S$  can be specified by the user as a constant set (e.g.,  $\{yes, no\}$ ) or obtained by an external procedure accessing a database, a knowledge base, on-line resources, etc.

So far, this formalism provides a high-level description of the interaction. In general, the actual execution of an action will depend on the specific task and modality of interaction chosen. Our approach allows for a multi-modal definition of each action. Modalities can vary depending on the application, the available interaction devices on the robot, etc. For example, we can consider the case of a robot equipped with a touch-screen in which text, images and videos can be displayed and with a Text-to-Speech (TTS) system to transmit messages to the user by voice. Input from the user can be received through the touch-screen or an Automatic Speech Recognition (ASR) system.

In order to achieve a multi-modal interaction, we implement for each action **Type** the modalities  $\gamma$  that we want to make available (e.g.,  $\gamma = \{text, image, TTS, video, \dots\}$ ). Let us define  $\varphi_\gamma = \text{Type}_\gamma(P_1, \dots, P_k)$  a function that given an action **Type**, its parameters  $P_1, \dots, P_k$  and the selected modalities  $\gamma$  returns the actual interaction  $\varphi_\gamma$  to be executed. Let us also denote  $\varphi = \bigcup_\gamma \varphi_\gamma$  the set of actual interactions over all modalities available (e.g.,  $\varphi = \{T, I\}$  contains a text and image when modalities  $\gamma = \{text, image\}$  are available). The set of interaction actions are specified as:

$state < \mathbf{Type}, P_1, \dots, P_k >$	$ask < \mathbf{Type}, P_1, \dots, P_k, S >$	$answer < X >$
$\varphi_\gamma = \mathbf{Type}_\gamma(P_1, \dots, P_k)$ $comm\_output(\varphi)$	$\varphi_\gamma = \mathbf{Type}_\gamma(P_1, \dots, P_k)$ $comm\_output(\varphi)$ $comm\_prepare(S)$	$X := comm\_input()$

Here,  $comm\_output(\varphi)$  represents the system-specific procedures to be carried out to finally communicate the interaction to the user using the chosen modalities, for example, the display of an image  $I$  on a GUI or the transmission of a selected text  $TTS$  to the speech component.  $comm\_prepare(S)$  represents the process in which the robot prepares the interaction devices for an answer from the user among the set of possible answers  $S$ . Finally,  $comm\_input$  assigns the result of the interaction to the given variable.

The presented approach is easy to manage and requires little effort from the designer. In practice, s/he has only to provide: (1) the interaction template  $\mathcal{T}$  and (2) the description of the interaction actions  $\mathbf{Type}_\gamma(\cdot)$ . Section 6 will provide examples of implementation and execution of this formalism.

## 5 On-Line Execution of the Interaction Template

On-line execution of the interactions is based on three main components. The first one is the Petri Net Plan (PNP) engine that executes the interaction plan according to its semantics. The PNP execution algorithm is described in details in [14] and implemented in the PNP library. The second component is the PNP-ROS bridge (also available in the PNP library) that allows the execution of PNP's on the robot. The third component is the implementation of the set of robotic and interaction actions described in the PNP. In this paper we focus on the interaction actions which are implemented in a client-server fashion, where the robot actions are clients that exchange commands and results of the interactions with an interaction server. The client-server architecture also allows for distributed computation and portability. In our implementation, we use a Linux laptop for the control of the robot and the execution of the PNP and a Microsoft Windows tablet for user interaction.

While most of these components are already available, the contribution described in this paper is related to the instantiation and the execution of the interaction actions described in the previous section. This is obtained with the implementation of a Multi-Modal User Interface (MMUI) that manages a Python GUI and a C# speech server using the multi-language Microsoft Speech Recognition and Synthesis engine.

More specifically, the MMUI component acts as a server, executes the interaction actions when enabled by the PNP engine, and returns the results of the interactions (i.e., input from humans to the robot) as conditions that are evaluated by the PNP engine to enable the proper transitions. The MMUI component implements the actions  $state$ ,  $ask$ , and  $answer$  by first collecting all the modalities for producing an interaction. In particular,  $state$  implements the  $comm\_output$  function (i.e., communication from the robot to the human)

by showing a statement with one or more of the output modalities available on the robot (text, images or videos shown on the tablet, spoken sentences by the robot). *ask* implements the *comm\_output* function in the same way as in *state* and then it implements the *comm\_prepare* function for receiving the input from the user (e.g., by displaying buttons on the GUI and loading specific speech recognition grammars). Finally, *answer* implements the *comm\_input* function that assigns the result of the interaction (either through GUI buttons or speech) to the given variable.

During the execution of the functions  $\text{Type}_\gamma(\cdot)$  defined in the interaction actions, variables  $P_1, \dots, P_n$  are always instantiated and the action execution algorithm guarantees the storage of the values of the variables and the use of the corresponding values when needed.

## 6 Implementation and Experiment

As already mentioned, the proposed framework has been fully implemented and tested by using the social robot Diago interacting with students in a classroom helping the teacher to do a Physics lesson<sup>1</sup>. Diago is a social mobile robot used for experiments in social human-robot interaction, knowledge representation and reasoning, and cognitive robotics. Diago stands 170 cm tall, is built on top of a Segway mobile base, equipped with a RoboTorso that contains laser sensors for motion, RGBD camera, microphone, and audio speakers for HRI, a laptop for controlling the mobile platform and the sensors, and a tablet for HRI and speech recognition and synthesis. We propose the design of a lesson of 50 min about Gravity, organized like in Table 2.

The students and the teacher are standing up in a classroom to facilitate Diago movements and its interaction with students. The robot moves around approaching the students with a welcoming message. The interaction template and the definition of the interaction actions are defined below:

$\langle \textit{approach}, \textit{state} < \textit{Welcome}, \textit{“Physics”} >, \textit{state} < \textit{InfoArgument}, \textit{“Gravity”} > \rangle$

`Welcome_TTS(P1):`

Hello everybody. It’s time for a @P1 lesson.

`InfoArgument_TTS(P1):`

`choice:`

Today, we will learn about @P1.

[P1=Gravity] Together we will discover why objects are heavy.

[P1=Force] Together we will discover why objects move.

`Welcome_text(P1):`

`choice:`

---

<sup>1</sup> More information about the robot used and the results of the experiments are provided in the web site <https://sites.google.com/a/dis.uniroma1.it/robot-at-school/>.

**Table 2.** Lesson plan with a robot assistant.

Time slot (min)	Teacher actions	Robot actions	Student actions
Introduction, 5	Ask the robot to introduce the argument of the lesson	Move inside the circle	Arrangement in a circle
Pre-test, 10	Ask the robot to start the pre-test session. Manage the student-robot interaction	Call students. Move toward him/her. Tell the first question. Get the answer. Tell the next one. Repeat for each student	Answer to all the questions
Discussion, 10	Start and Stop the discussion activity	Display statistic of answers, videos and images	Free discussion
Ask to Diago, 5	Manage the students	Answer to students	Ask to Diago for deepening
Post-test, 10	Ask the robot to start the post-test session. Manage the student-robot interaction	Call students. Move toward him/her. Tell the first question. Evaluate it: if correct, tell the next one; if wrong, repeat the question. Repeat for each student	Answer to all the questions
Conclusion, 10	Ask the robot to summarize the lesson	Summarize the lesson. Thank the class. Make congratulations. Tell the next argument. Ask to arrange at the desk. Say goodbye message	Greet and thank to teacher and Diago. Arrangement at the desk

Welcome to the @P1 lesson.  
Good morning.

InfoArgument\_text(P1):  
choice:

@P1 is the topic we are going to study today.

Argument: @P1. Lesson Schedule: introduction, pre-test, discussion, Q&A, post-test, summary

In order to describe the interaction actions we use the following syntax: (1) a choice operator is presented to select one of the possible actions (it can be omitted if there is only one option), (2) @P1 is replaced by the exact content of the variable  $P_1$ . Thus, for `Welcome_text("Physics")` action, sentences "Welcome to the *Physics* lesson?" or "Good morning" would be generated, (3) it is possible to particularize a concrete action given a specific parameter value using the syntax `[P1=ParameterValue]`. For example, in `InfoArgument_TTS("Gravity")`, both "Together we will discover why objects are heavy." or "Today, we will learn about Gravity" could be selected.

An example of execution of this template, together with the modalities activated is shown below:



**R** : [TTS] *Hello everybody. It's time for a Physics lesson.*

**R** : [text] *Good morning.*

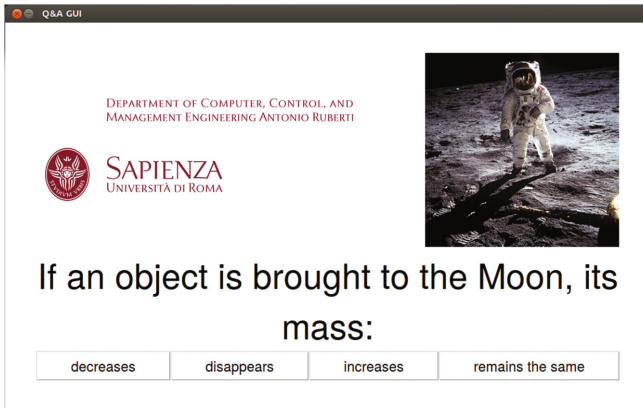
**R** : [TTS] *Today, we will learn about Gravity.*

**R** : [text] *Argument: Gravity. Lesson Schedule: introduction, pre-test, discussion, Q&A, post-test, summary*

Then, the teacher proposes a pre-test of 4–5 questions to students. So, the robot calls each student by name, goes forward him/her and ask him/her for questions like:

*“If an object is brought to the Moon, its mass:  
(a) decreases (b) disappears (c) increases (d) remains the same”*

These type of questions follow a template of the form  $\langle ask < Quiz, GetChoices() \rangle, answer < Result \rangle$ , where the questions can be stored in a database containing also multimedia data. `GetChoices()` is a generic function that queries the designed database to retrieve the set of possible answers for each of the proposed questions of this example, and `Result` is a variable that will store the selection made by the student, which will contain the semantic meaning of the answer, i.e., if the answer was correct or wrong. Figure 1 shows an example of a question formulated to a student using our Graphical User Interface.



**Fig. 1.** Example of GUI interaction with a student in a lesson about Gravity.

Here, the student can answer either by using the touch-screen on the robot or by voice. Once the selected answer is registered or recognized by the robot, the next question is displayed. During the pre-test session the correct answer is not displayed because the aim is to catch students misconceptions about the argument of the lesson (Gravity in this example). When all the students have answered the questions, the teacher asks the robot to display the statistics of correct/wrong answers, starting the discussion about the wrong ones. The robot

assists the teacher by showing videos or images (e.g., the moon landing image) in order to create cognitive conflict in the group, stimulating in this way the discussion and the recognition of the correct answer by the group. Following, a question and answer (Q&A) activity will give to students the opportunity to interact with Diago, asking for deepening about Gravity, using images and videos. The post-test session is similar to the pre-test one, because the robot calls each student by name and goes to her/him asking for the same questions of the pre-test, collecting the answers but, unlike before, if the answer is wrong a failure message, an invitation to try again and the same image/video displayed during the discussion session are shown. A feasible interaction template for a post-test session and an example of interaction is shown below:

```
(LABEL1, ask < Quiz1, GetChoices() >, answer < Result1 >,
  Result1 = wrong?state < Answer, "wrong" >, GOTO LABEL1
  : state < Answer, "right" >, LABEL2, ask < Quiz2, GetChoices() >, ...)
```

Answer\_text(P1):

choice:

The answer is @P1.

[P1=right] Congratulations!

[P1=wrong] I am sorry, this is not the right answer. Try it again.

Answer\_image(P1):

choice:

[P1=right] img\_happy\_smile.png

[P1=wrong] img\_sad\_smile.png

**R**: [Both TTS and text] *If an object is brought to the Moon, what happens to its mass?*

**R**: [Image] (See Fig. 1.)

**H**: [Speech] *disappears*

**R**: [Both TTS and text] *I am sorry, this is not the right answer. Try it again.*

**R**: [Image] (a smile or sad emoticon)

Due to the GOTO instruction, question is reformulated until the correct answer is obtained.

Then, the interaction continues with other quizzes.

Finally, Diago summarizes what they learn during the lesson and greets the classroom. The role of the teacher, during the lesson, is managing the student-robot interaction, for example checking that each student start and stop his/her own interaction, without interrupting it; managing the time in order to maintain the schedule; managing the student-student interaction, a social activity that high school students needs and appreciate but rarely they are able to control it. Finally, the robot asks the students to sit down at their own desk and greets them with a goodbye message.

**R**: [TTS and text] *Goodbye! Have a nice day!*

## 6.1 Experimental Validation

A reduced instance of the lesson described above has been used for a user study. During an Open event in our Department, we hosted 50 high school students of around 18 years old (10–12th grade).

**Table 3.** Godspeed questionnaire data about perceived intelligence and safety. The full results are in the web site mentioned at the beginning of this section.

Interaction with the robot during the event	Yes average-variance	No average-variance	P value
Perceived Intelligence			
Incompetent versus Competent	4.30-0.47	3.91-0.54	0.40
Ignorant versus Knowledgeable	4.41-0.63	4.22-0.56	0.38
Irresponsible versus Responsible	4.00-0.75	4.09-0.67	0.38
Unintelligent versus Intelligent	4.35-0.62	4.09-0.93	0.20
Foolish versus Sensible	3.23-1.07	3.50-0.58	0.07
<b>Perceived Safety</b>			
Anxious versus Relaxed	4.06-1.68	4.56-0.45	<b>0.0008</b>
Calm versus Agitated	1.64-0.99	1.65-1.07	0.45
Quiescent versus Surprised	3.71-0.85	3.78-1.14	0.26

We proposed them short Physics lessons (15 min), organized as described previously. After the lesson we distribute a questionnaire Godspeed Questionnaire Series (GQS) [13] that is frequently used for HRI evaluation. We used GQS for assessing the success of the robot, evaluating if the emotional state and the impression of the robot was influenced by the fact that the students were interacting or not with it. A significant sample of data analysis of perceived intelligence and safety is showed in Table 3.

The general scores were all positive, showing a general acceptance of the experience. The relatively high P-values calculated for almost all the GQS parameters show that, during the curricula lesson with a robot, students' emotional state and their impression of the robot was not influenced by the fact that they interacted with it. A notable exception was found for the anxiety state, for which the data analysis shows a significant higher variance in the students that interacted with the robot. This fact indicates that interaction with a robot during a teaching experience can generate anxiety that must be taken into account by the teachers. Nonetheless, the overall result confirms that the role of the robot as teacher assistant is well accepted without the need that each student has to interact with it.

## 7 Conclusions

In this paper, we have presented a formalism to describe templates of multi-modal interactions. Once the interaction template has been designed, it is instantiated and executed on the robot by using the PNP formalism.

The use of the templates is intuitive and facilitates the work of the designer, who must not necessarily be an expert from the technological point of view, and, as shown in the example, the templates are easily extendable and re-usable in different contexts. A user-friendly mechanism to populate the robot database represents a good opportunity to facilitate the use of such tools by the teacher, being a chance to enhance his/her technology skills while trying to catch and hold the student interest for the subject.

Work in progress includes the application of this methodology in a more structured teaching experience and a deeper evaluation of the results. Another interesting direction is personalization of the interaction that allows to maintain the interest over time of the students, as demonstrated by Lee [6].

**Acknowledgements.** This work has been partially developed within the COACHES project. COACHES is funded within the CHIST-ERA 4<sup>th</sup> Call for Research projects, 2013, Adaptive Machines in Complex Environments (AMCE) Section. Sapienza University is funded by MIUR (Italy).

## References

1. Barreto, F., Benitti, V.: Exploring the educational potential of robotics in schools: a systematic review. *Comput. Educ.* **58**(3), 978–988 (2012)
2. Bottoni, P., Costabile, M.F., Mussio, P.: Specification and dialogue control of visual interaction through visual rewriting systems. *ACM Trans. Program. Lang. Syst.* **21**(6), 1077–1136 (1999)
3. European Commission/EACEA/Eurydice. The teaching profession in Europe: Practises, perceptions, and policies (eurydice report). Technical report, Luxembourg: Publications Office of the European Union (2015)
4. Hussain, S., Lindh, J., Shukur, G.: The effect of lego training on pupils' school performance in mathematics, problem solving ability and attitude: Swedish data. *J. Educ. Technol. Soc.* **9**(3), 182–194 (2006)
5. Kanda, T., Shimada, M., Koizumi, S.: Children learning with a social robot. In: *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction, HRI 2012*, New York, NY, USA, pp. 351–358 (2012)
6. Lee, M.K., Forlizzi, J., Kiesler, S., Rybski, P., Antanitis, J., Savetsila, S.: Personalization in HRI: a longitudinal field experiment. In: *7th ACM/IEEE International Conference on Human-Robot Interaction*, March 2012
7. Lindh, J., Holgersson, T.: Does lego training stimulate pupils' ability to solve logical problems? *Comput. Educ.* **49**(4), 1097–1111 (2007)
8. Mitnik, R., Nussbaum, M., Soto, A.: An autonomous educational mobile robot mediator. *Autonom. Rob.* **25**(4), 367–382 (2008)
9. Mubin, O., Stevens, C., Shahid, S., Al Mahmud, A., Dong, J.-J.: A review of the applicability of robots in education. *Technol. Educ. Learn.* (2013)
10. Schaefer, R., Bleul, S., Mueller, W.: *Dialog modeling for multiple devices and multiple interaction modalities*, Heidelberg, Berlin, pp. 39–53 (2007)
11. Vollstedt, A.M., Robinson, M., Wang, E.: Using robotics to enhance science, technology, engineering, and mathematics curricula. In: *American Society for Engineering Education Pacific Southwest Annual Conference* (2007)

12. Walker, E., Burleson, W.: User-centered design of a teachable robot. *Lecture Notes in Computer Science*, vol. 7315, pp. 243–249 (2012)
13. Weiss, A., Bartneck, C.: Meta analysis of the usage of the godspeed questionnaire series. In: 2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pp. 381–388, August 2015
14. Ziparo, V.A., Iocchi, L., Lima, P.U., Nardi, D., Palamara, P.F.: Petri net plans - a framework for collaboration and coordination in multi-robot systems. *Auton. Agent. Multi-Agent Syst.* **23**(3), 344–383 (2011)

# **Virtual Environments, Cloud Tools and Artificial Intelligence**

# eduMorse: An Open-Source Framework for Mobile Robotics Education

Daniele De Martini<sup>(✉)</sup>, Andrea Bonandin, and Tullio Facchinetti

Department of Electrical, Computer and Biomedical Engineering,  
Università degli Studi di Pavia, Pavia, Italy  
{daniele.demartini01, andrea.bonandin01}@universitadipavia.it,  
tullio.facchinetti@unipv.it

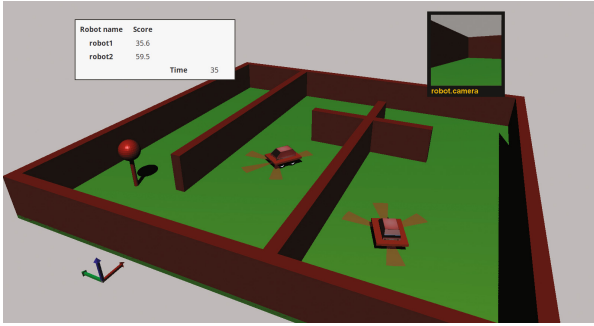
**Abstract.** The increasing spreading of robotics applications requires the formation of more and more experts with knowledge in core aspects of robotics systems. This paper introduces eduMorse, a novel framework for the education in the scope of mobile robotics. The framework addresses the accurate simulation of single- and multi-robot systems, with special focus on the possibility to implement path planning, navigation and control strategies, to handle sensors and actuators, and the communication among robots, thus allowing for the simulation of multi-robot coordination strategies. eduMorse leverages open-source tools to build a modular client-server framework for the simulation of mobile robots, with the aim of a simple setup of the simulation as a primary goal. The paper describes the components of eduMorse and its architecture. An example of application is also presented to show the effectiveness of the robotics simulation and the usage workflow of the system.

**Keywords:** Robotics · Education · Simulation · Multi-robot systems · Path planning · Navigation · Coordination · Free/Open source software

## 1 Introduction

The learning of basic and advanced robotics concepts requires accurate simulation of the physical environment and the interaction between such environment and one or more robots. While real hardware experimentation allows to face challenges related to mechanical and electronics issues, the study of control algorithms such as path planning and navigation, collaborative and coordination strategies, can hugely benefit from an accurate simulation of the system.

eduMorse is a novel framework for mobile robotics education that leverages open-source tools to build a modular client-server framework for the simulation of mobile robots. The framework allows the accurate simulation of single- and multi-robot systems (MRS), focusing on aspects such as path planning, navigation and control strategies, realistic sensors and actuators management, Simultaneous Localization and Mapping (SLAM). In MRS the robots can communicate to implement effective coordination strategies targeted at solving a specific task.



**Fig. 1.** Example of view provided by eduMORSE graphical window, including the MORSE simulation window and the scoring window, where 2 robots move in the scene, and one robot is equipped with onboard camera.

Typical objectives of the simulation are search tasks in unknown environments. The accuracy in the simulation of the physics and robot components is achieved by leveraging the MORSE (Modular OpenRobots Simulation Engine) simulator [7, 8]. The client-server organization is functional to the modularization of the simulator, so that each component communicates through standard TCP sockets and a dedicated Application Programming Interface (API). This approach allows the execution of different modules on different computers, even leveraging virtual machines, completely decoupling the execution of the modules, improving reliability, safety and security of the simulation. Multiple clients can be connected to the same server, allowing the simulation of multi-robot systems. Thanks to the communication using standard sockets and the dedicated API, a client can be implemented using the preferred programming language, provided that correctly handles the communication with the simulator.

The framework aims at a simple setup of the simulation. For this purpose, the configuration of the simulation, including the map and the scoring options, and the setup of each involved robot, are concentrated in few configuration files with standard syntax. The installation of the simulator is done under the Linux operating system by leveraging the existing packets available for standard distributions. A script is provided that installs all the dependencies and downloads the necessary code from public repositories.

The paper is organized as follows. Section 2 presents some related works. Section 3 provides a general overview of the framework, while Sect. 4 describes in details its architecture and components. The configuration possibilities are described in Sect. 5. An example of application is provided in Sect. 6, and Sect. 7 reports some considerations on the use of the framework in the last two year of a course on Robotics. Finally, Sect. 8 concludes the paper.



## 2 Related Works

The growing interest in mobile robotics education motivated the development of several learning platforms, based on both hardware and simulated environments. A popular hardware platform is represented by the Lego MindStorm NXT [13]. The historical RoboCup event is a challenging robotics soccer competition where both hardware/software solutions must be developed to implement a competitive robot [2, 4]. Robot competitions, both simulated and for real robots, are a common test bed for development of new solutions and for teaching purposes [1, 3, 19]. Recently, the Robotarium [18] cloud platform was proposed to allow the remote upload of control firmware on a laboratory platform that can be monitored from the web during the operations.

While hardware solutions focus more on physical assembling and tuning of robots, simulated approaches mostly deal with motion and control logic. Since this paper focuses on a simulated framework, this section will concentrate on similar approaches. A popular simulator is RoboCode [12]. It allows the simulation of elaborated strategies to prevail in a multi-robot fight. This environment, however, is limited to the development of strategies, since dynamics and interaction with the physical environment is almost absent. The authors of [17] leverage simple robotics simulations as a mean of teaching a programming language. The mentioned work, however, focuses on the teaching aspects related to the programming language, while it does not provide any detail regarding the specific programming environment used in the course.

Alternative solutions include the use of full-featured robotics simulators as core components of the education framework, such as Gazebo [14], ROS (the Robot Operating System), Blender and OpenCV [15].

## 3 Framework Overview

The aim of eduMorse is to make the students overcome the gap between theoretical concepts and practical implementation, to make them face problems like path planning and triangulation without the hassle of relying on a real hardware platform, but still in a realistic environment.

To address students with heterogeneous backgrounds and experiences, one of the most important features the tool was its independence from a specific programming language, both for configuration and for the programming exercise itself. The key goals of eduMorse are:

1. being based on FLOSS (Free/Libre and Open Source Software) components;
2. to deliver an accurate simulation of the environment, the physics, and robot devices (sensors and actuators);
3. to provide a pleasant graphical environment where the simulation is run (Fig. 1 shows an example of such an environment);
4. to allow a distributed execution on different machines in a client-server organization;

5. the support multi-robot applications;
6. no constraints on the programming language to implement the robot path planning, navigation and control logic;
7. the support of flexible scoring rules for competitions and programming tournaments;
8. simple creation and setting up of a simulation session;
9. ease of installation.

Several robotic simulation environments have been evaluated in the early stage of the project. RMTool [11] is a simulation tool that focuses on path planning and navigation algorithms in a planar environment. However, it did not meet the requirements of an accurate simulation of environment and sensors, and it is strictly related to the Matlab framework. Stage [10] is an open source robotic simulator born under the umbrella of the Player project. It simulates robots and their components; it is designed for fast simulation of multi-agent systems and exposes the interface to sensors and actuators with APIs in various programming languages, thanks to the Player project. However, it is limited to 2D environments. Webots [16] is a commercial simulator that offers 3D realistic simulations, API for the major languages and inter-robot communication, but has a non-open license. Finally, Gazebo [14] is a fully-featured open-source robotic simulator, capable of simulation of robots and components in a realistic 3D environment and that gives access to a library of standard robots, sensors and actuators. It provides APIs for various programming languages thanks to the Player support, and interfaces to the ROS framework.

Nevertheless, the framework described in this paper leverages the open-source MORSE robotics simulator [7]. MORSE provides accurate simulation of robot dynamics and its interaction with realistic 3D environments. This is made possible since MORSE is based on Blender, a popular open-source 3D modeling software, and the rendering is based on the Blender Game Engine [9], which supports shaders, lighting options and multi-texturing. Moreover, the Bullet library is used for real-time physics simulation [5]. An important feature that led the choice of MORSE is that simulations are generated by Python scripts with a simple syntax. MORSE provides models and simulation of standard sensors and actuators, e.g. cameras, laser scanner, and velocity controllers and all the data streams can be accessed using standard sockets, framework such ROS or Yarp or using the built-it Python library. Popular models for robotic platforms, such as quadrotors, ATRV, Pioneer3DX, PR2, are also already available in MORSE.

eduMorse adds functionalities to the yet feature-rich MORSE simulator without changing the MORSE simulation core itself. Indeed, the most relevant are:

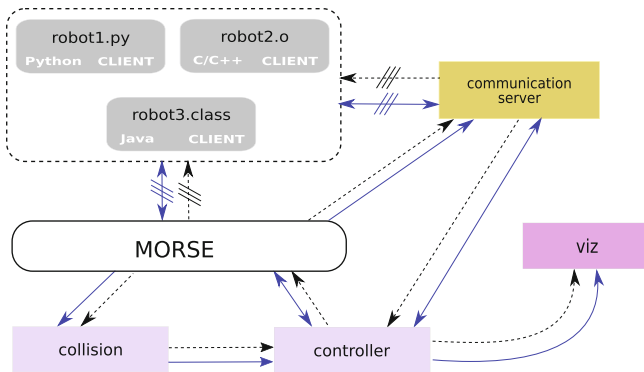
- A communication server that enables the communication between robots;
- A scoring system to evaluate the performance of the robots during contests or tournaments;
- A flexible configuration system;
- Libraries to communicate and to access sensors and actuators.

Besides, the framework includes an install procedure to simplify the installing of the required software and libraries and some pre-configured maps and simulations are provided, for getting started with the platform.

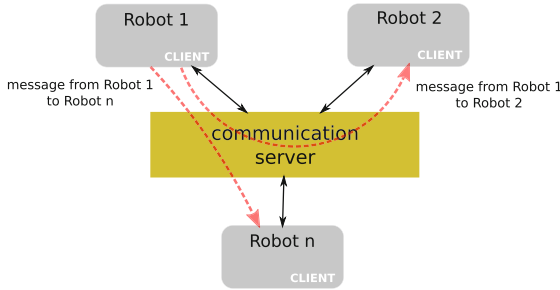
### 4 The System Architecture

The system architecture is composed by the core MORSE simulator, a server process that handles the communication between robots, and a scoring system. The processes communicate over the network or locally on the same computer by means of standard TCP sockets. The overall system is based on a multi-process instead of a monolithic architecture. Although the latter may have lower runtime overhead due to the absence of message passing, the former is more flexible, favors the system modularity, simplifies the debug phase and allows for distributing the processes over different computers if needed. On one hand, all the processes can run on the same machine during the development of the robot logic. On the other hand, during the verification of the robot behavior by the teacher, or during a contest with multiple robots involved, all the components dedicated to the simulation (i.e., MORSE, the communication server and the scoring processes) can run on one machine, while robot processes can be executed on different computers. Therefore, each robot process does not compete for computational power with others; it may even run on an embedded computer to perform Hardware In the Loop (HIL) simulations. Safety and security are guaranteed since possible malicious code contained in the robot process is confined to run within a dedicated machine. Moreover, during the simulation, the robots can not access any information that is not allowed by the simulation itself: all the relevant data are provided by exchanging messages with the simulator.

Figure 2 depicts the components of the architecture. Each component is implemented as a separate process. The figure shows the relationships among



**Fig. 2.** Overall system architecture: solid blue arrows represent the interprocess communication direction, while dashed black arrows indicate the ownership of the communication stream.



**Fig. 3.** Robots are connected to the communication server, which forwards the messages between them.

processes. In particular, arrows denote the direction of the communication between processes (solid blue arrow) and the ownership of the communication stream (dashed black arrow). The ownership of the stream identifies the process that initiates the communication with another process.

This section provides more details regarding these components.

### 4.1 Communication Server

The communication server process enables and controls the communication between the processes implementing the robot logic during the simulation. Such processes can run on the same machine as the server or on different physical machines. After the connection with all robots is opened, the server continuously checks the presence of new messages, which are forwarded to the destination robots. Every message must be sent to the server, to be properly forwarded to the destination robot (see Fig. 3). In this sense, the server acts as the simulator of a wireless communication channel. In this way, the server can enforce an optional bandwidth limitation to artificially restrict the communication possibilities during the simulation, making the coordination task more challenging. In fact, one parameter for the wireless communication channel is the maximum number of bytes that can be exchanged within the time unit.

It is worth to note that the simulation of the wireless communication channel does not model low level physical effects such as interference, collisions, packet losses, etc. As a consequence, the transmission of a message can be considered always correct. The only applied limitation regards the maximum data rate.

### 4.2 Scoring System

The score of a simulation is affected by two factors: the time required to complete the task and collisions with objects. The final score of a simulation is calculated according to Eq. (1):

$$s = s_0 + k \cdot t + \sum_i n_i \cdot p_i \tag{1}$$

where  $s_0$  is an initial score,  $k \leq 0$  is a constant negative decay factor and  $p_i$  is the weight related to a collision with the  $i$ -th object; its contribution to the final score is multiplied by  $n_i$ , i.e., the total number of collisions with the corresponding object. The weight  $p_i$  can be either positive or negative. In this way, an object can play the role of obstacle or a target.

To track the score during the simulation, the scoring system is based on 3 processes: a collision detector (`collision`), a collision and control handler (`controller`) and an optional visualization process that exposes a user interface (`viz`). In particular, as shown in Fig. 2, the data flows from `collision` to `controller` to `viz` (blue arrows).

The `collision` process detects collisions between robots and objects. By default, a collision sensor is added to each robot. The collision sensor belongs to the standard sensor library of MORSE. The sensor surrounds the body of the robot and will detect every collision with an object. The size and the position of the sensor must be selected to fit the specific robot geometry. These parameters can be fully configured by means of configuration files, as explained in Sect. 5.

When a collision occurs, a counter structure is updated and the process checks if it is a new contact or the robot is leaning against an object. This control is made possible by storing the timestamp of a collision. The difference between timestamp of the current event and timestamp of the previous collision is considered to figure out whether a new collision event shall be generate. Whenever a collision event is detected, the information is notified to the `controller` process. The information passed to the process includes the name of the robot which collided and a list of objects that were hit. The `controller` process determines the contribution of the collision to the robot score, according to the values of the parameters in Eq. (1). These parameters are configured as discussed in Sect. 5.

The `viz` process presents a simple user interface to display the score associated to all the robots during the simulation. The information displayed by the `viz` are requested to the `controller` process. For this purpose, the `viz` process periodically polls the `controller` process by sending information requests and receiving the corresponding update.

## 5 Configuration of the Simulation

One of the goals the project is the possibility to easily configure the relevant aspects of the simulation by means of configuration files. Configuration files use the Tom's Own Minimal Language (TOML) to assign parameters and values.

The different components of a simulation can be configured. In particular, the following components can be distinguished: the simulation itself, rules, robots and obstacles. Each of them will be described in this section. The configuration stack is layered on top of the MORSE simulator. This means that the various components will be handled by the simulation as standard. While some parameters are mutated from MORSE, some parameters are introduced by eduMorse, in particular those referring to the scoring and the communication system.

*Configuration of the simulation.* The most important parameters of the simulation include the name of the configuration file that defines the rules of the simulation. A flag that defines the rendering method used by MORSE is also configurable: if true, MORSE switches to a simpler rendering method (called *wireframe*) that leads to faster graphical performance but turns down the possibility to use the onboard robot camera. Moreover, it can be specified the name of a robot that carries an onboard camera for first person view; only one robot can carry a camera in a simulation, which is reasonable for single-robot simulations; position and pose can be configured w.r.t. the robot reference frame.

For each robot involved in the simulation an unique name associated to the robot and the name of the configuration file that specifies the robot parameters must be indicated. The name will be then used to access its sensors and actuators using the socket interface.

*Configuration of the rules.* The rules define the behavior of a simulation run. The most important parameters that can be configured are:

- the number of robots allowed in the simulation;
- the number of obstacles inserted in the simulation;
- the name of a Blender file containing the map;
- the types of actuators and sensors allowed in the simulation;
- the position and the pose of the external camera that provides the view of the whole scene, expressed w.r.t. the simulation reference.

For each robot involved in the run, its initial position and orientation can be configured. Moreover, every object in the scene is characterized by the following parameters: the name of the blend file containing the object, an unique name given to it as well as the 3D position and orientation in the map.

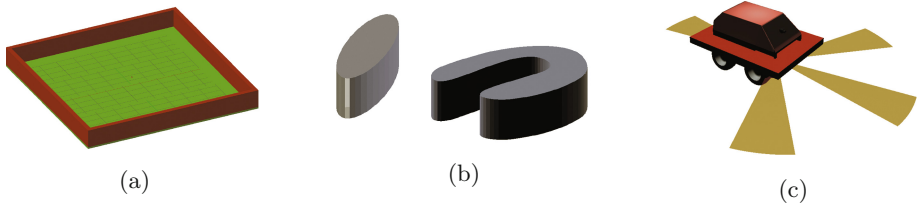
Finally, there are a number of options related to the score and the time, i.e. the total simulation time and the parameters used in Eq. (1) (the initial score and the  $k$  constant) can be specified.

*Configuration of robots.* The type of robot can be chosen among the available types supported by MORSE. A list of sensors and actuators can be specified. Each item must include an unique identifier in the form of a string, the type of the actuator (currently supported are `MotionVW` and `Keyboard`), and the interface (e.g., ‘`socket`’ specifies the standard TCP socket); The mandatory collision sensor is configured in terms of its relative displacement w.r.t. the robot frame, which allows for adapting its arrangement considering the selected type of robot.

*Configuration of the objects in the map.* For each object, the interesting parameters are the name of the object and the score  $p_i$  assigned to the object.

## 6 Example of Application

The typical workflow in the use of eduMorse requires the definition of a simulation, i.e., the characteristics of robots and the score, and the selection of one or



**Fig. 4.** The components of the example simulation: Fig. (a) depicts the empty map, Fig. (b) shows the obstacles placed in the scene, while Fig. (c) shows the ATRV differential mobile robot used in the example.

more maps to test the robot's logic. The example of application provided in this section considers a 4 wheel differential ATRV robot equipped with a Pose sensor, proximity sensors and the collision sensor. The Pose sensor allows to obtain the location and the orientation of the robot, while 3 proximity sensors are installed in front of the robot, with orientation of 30 degree from each other. The robot is also equipped with a differential drive. The ATRV model used in the simulation is shown in Fig. 4(c).

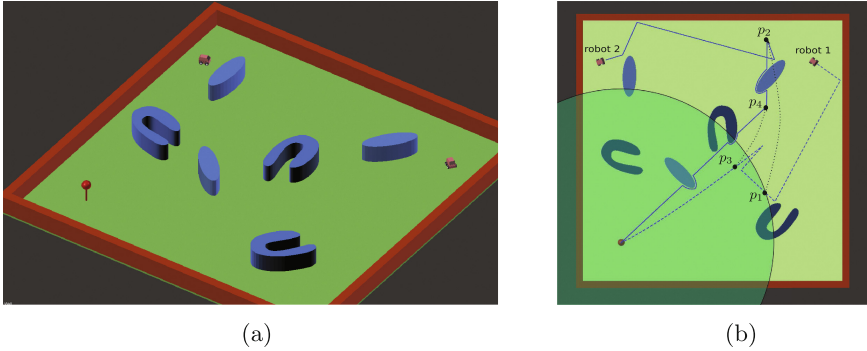
## 6.1 Definition of the Map

The map is created from a base empty arena, where obstacles are placed. eduMorse follows the peculiar approach of MORSE to configure these elements: each element can be contained in a separate graphical file, and through configuration files one object can be inserted in the scene with given coordinates and orientation. In this way, one object can also be placed more than once, in different locations, provided that different elements in the scene are labelled with different names. The name represents the reference of the object, which is used to associate features such as the scoring options.

Figure 4 shows the elements that compose the example simulation. The actual simulation is shown in Fig. 5, where all the components are combined within the MORSE simulation window. Figure 5(b) provides an example of path generated by two ATRV robots: the goal is to locate and reach a target, whose distance can be sensed by a robot within the range shown in figure. The two robots generate random way points and move to the next way point until robot 1 detects the target and sends its own current co-ordinates to robot 2 (point  $p_1$ ). At this point, while in point  $p_2$ , robot 2 move towards to the received co-ordinates while robot 1 triangulates the target location. Once obtained the target co-ordinates in point  $p_3$ , robot 1 sends this information to robot 2, when it is in point  $p_4$ . Both robots use the Bug 2 algorithm [6] for collision avoidance during trajectory tracking.

## 6.2 Interface Between Robot and Simulator

To enable the communication between a robot and the simulator for getting sensor values or controlling the actuators, the MORSE simulator allows three



**Fig. 5.** Example of multirobot navigation and target localization problem. Figure (a) shows the 3D representation of the scene, while Fig. (b) depicts the paths generated and the messages sent during the navigation, although the lines are not actually present in the output provided by eduMorse.

```

while (1) {
    Pose pose; // Get the pose sensor measure
    flag = getPose(f, parent, "pose", &pose);
    irMeas irmeas; // Get the ir sensor measurement
    flag = getIR(f, parent, "IR", &irmeas);

    if ( getDist(&irmeas) < THRESHOLD ) {
        flag = setSpeed(f, parent, "motion", 0, 0);
        break;
    } else {
        flag = getControlSpeed(&pose, &v, &w);
        flag = setSpeed(f, parent, "motion", v, w);
    }
    usleep(DELAY)
}

```

**Fig. 6.** Example of usage of the API from C language.

options: through frameworks such as ROS or Yarp, sockets, or leveraging the built-in Python library. eduMorse takes advantage of the socket communication interface and exposes a dedicated API implemented in various programming languages, while keeping the possibility to use ROS or the Python library.

Figures 6 and 7 show the usage of the API in a C and a Python program – respectively – to achieve a simple task: the robot moves accordingly to his pose using the `setSpeed` function until it finds an obstacle and then it stops.

## 7 Usage in Undergraduate Course

The eduMorse framework has been used in the last two years to assign practical school works in the context of the robotics course at the University of Pavia. The goal is to allow a fruitful programming experience of path planning and navigation in a simulated but realistic environment. Since the background of the



```

while True:
    posemeas = pose.get() # Get the pose sensor measure
    irmeas = ir.get() # Get the ir sensor measurement

    if getDist(irmeas) < THRESHOLD:
        motion.publish({"v": 0, "w": 0})
        break
    else:
        v, w = getControlSpeed(posemeas)
        motion.publish({"v": v, "w": w})
        time.sleep(DELAY)

```

**Fig. 7.** Example of usage of the API in Python language.

students attending the course is heterogeneous, the framework is been continuously developed to permit every student to meet the goal of the assignment.

The task assigned to the robot is to reach a unknown target location in a 2D maze. The robot is equipped with a position sensor, providing the robot's position w.r.t. the inertial frame, 4 infrared sensors, as shown in Fig. 5, and a sensor that returns the distance of the robot from the target, but not the heading towards it. This practical project was assigned to more than 30 students to date. Most of the students solved the problem using standard bugs algorithm for path planning; some few solutions were based on Voronoi graphs or potential field planners [6]. The identification of the target position was performed by applying triangulation techniques, based on the target distance measured in different locations, and the robot position sensors. Trajectory tracking and obstacle boundary following was performed using classical proportional controllers, where the error is calculated on the desired orientation and the proximity value, respectively. Most of the solutions have been implemented in C language, while Python was chosen by students with less programming experience. The control algorithm has been always implemented as a single process and no one has leveraged the ROS interface, which is not currently part of the robotics course.

## 8 Conclusions

This paper presented eduMorse, an educational simulation platform for mobile robotics applications. eduMorse goals, characteristics and organization have been discussed. The focus was on the accuracy of physical environments simulation, modularity of the system (i.e. its client-server organization), simplified configurability and installing, and definition of simulation rules to address common collaborative and competitive tasks. This latter point is addressed by the availability of a scoring system allowing to evaluate the robot performances. eduMorse has been used in academic courses for facing robotics programming tasks.

## References

1. Almeida, L., Fonseca, P., Azevedo, J.L.: The micro-rato contest: a popular approach to improve self-study in electronics and computer science. In: IEEE International Conference on Systems, Man and Cybernetics, October 2000
2. Amy Eguchi, L.A.: RoboCupJunior: promoting stem education with robotics competition. In: 4th International Conference on Robotics in Education (RIE) (2013)
3. Azevedo, J., Oliveira, M., Pacheco, P., Reis, L.P.: A Cooperative Cyber-Mouse@RTSS08 Team, pp. 251–262. Springer, Heidelberg (2009)
4. Birk, A.: The true spirit of RoboCup [Education]. *IEEE Robot. Autom. Mag.* **17**, 108–108 (2010)
5. Boeing, A., Bräunl, T.: Evaluation of real-time physics simulation systems. In: Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia, pp. 281–288. ACM, New York (2007)
6. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Boston (2005)
7. Echeverria, G., Lassabe, N., Degroote, A., Lemaignan, S.: Modular open robots simulation engine: MORSE. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 46–51, May 2011
8. Echeverria, G., Lemaignan, S., Degroote, A., Lacroix, S., Karg, M., Koch, P., Lesire, C., Stinckwich, S.: Simulating complex robotic scenarios with MORSE. In: Proceedings of the 3rd International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SiMPAR). pp. 197–208. Springer (2012)
9. Felinto, D., Pan, M.: Game Development with Blender. Cengage Learning PTR (2013)
10. Gerkey, B.P., Vaughan, R.T., Howard, A.: The player/stage project: tools for multi-robot and distributed sensor systems. In: Proceedings of the 11th International Conference on Advanced Robotics, pp. 317–323 (2003)
11. Gonzalez, R., Mahulea, C., Kloetzer, M.: A matlab-based interactive simulator for mobile robotics. In: 2015 IEEE International Conference on Automation Science and Engineering (CASE), pp. 310–315, August 2015
12. Hartness, K.: Robocode: using games to teach artificial intelligence. *J. Comput. Sci. Coll.* **19**(4), 287–291 (2004)
13. Klassner, F., Anderson, S.D.: Lego mindstorms: not just for k-12 anymore. *IEEE Robot. Autom. Mag.* **10**(2), 12–18 (2003)
14. Koenig, N., Howard, A.: Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: International Conference on Intelligent Robots and Systems, Sendai, Japan, pp. 2149–2154, September 2004
15. Lentin, J.: Learning Robotics Using Python. PACKT (2015)
16. Michel, O.: Cyberbotics Ltd Webots TM: professional mobile robot simulation. *Int. J. Adv. Robot. Syst.* **1**, 39–42 (2004)
17. Ortiz, O.O., Franco, J.P., Garau, P.M.A., Martn, R.H.: Innovative mobile robot method: improving the learning of programming languages in engineering degrees. *IEEE Trans. Educ.* **PP**(99), 1–6 (2016)
18. Pickem, D., Wang, L., Glotfelter, P., Diaz-Mercado, Y., Mote, M., Ames, A.D., Feron, E., Egerstedt, M.: Safe, remote-access swarm robotics research on the Robatarium. ACM Computing Research Repository (CoRR) abs/1604.00640 (2016)
19. Simões, D., Brás, R., Lau, N., Pereira, A.: A Coordinated Team of Agents to Solve Mazes, pp. 381–392. Springer (2016)

# Teaching Robotics with Cloud Tools

Igor Zubrycki<sup>(✉)</sup> and Grzegorz Granosik

Lodz University of Technology, Łódź, Poland

[igor.zubrycki@dokt.p.lodz.pl](mailto:igor.zubrycki@dokt.p.lodz.pl)

<http://robotyka.p.lodz.pl>

**Abstract.** With the cloud computing taking root in the general computing ecosystem we can use many cloud computing tools to support teaching robotics to different groups. In this paper, we provide a general overview of using cloud tools, provide a review of some of them as well as give some recommendations based on our experience.

## 1 Introduction

The task of teaching robotics is a complex one. Students need to gather theoretical knowledge from various disciplines and preferably understand diverse methods and their limitations, as well as be able to design and produce multiple practical projects. All of this has to be done in a limited time, usually in couple semesters, with only a few subjects directly connected to the robotics.

Several categories of computer programs can help students in their knowledge and skills acquisition. Concretely, programs for modeling, simulation, calculation (both symbolic and numerical), program development environments (for robots with operating systems and microcontrollers) and interface development tools. However, while many of these programs would be useful throughout student's education in many cases student's experience is limited to a few hours of laboratory work and is not continued in their professional life. This is because most of those programs require lengthy setup as well as licenses from the producents.

For example, installation of the Autodesk Inventor, a program for designing and virtual assembling of complex devices requires a 40 GB for the installation files and recommended memory of 20 GB. Other useful programs, such as MATLAB (program for calculations and graphing), Keil Vision (for microcontroller programming) have similar requirements. Also, these programs are non-free and non-open source and are generally expensive, so students, after graduating and losing academic license rights, would also need to uninstall them.

All this might be demotivating for students and teachers who want to use only some, limited functionalities of the programs while studying some specific subject. For example, when the teacher introduces some particular aspect of robotics (e.g. numerical inverse kinematics), it would make sense for students to play with the code themselves to understand better the particular parts and limitations. When faced, however, with the need to install MATLAB, many students will only watch teacher program and lose the chance to learn by experience.

What we and other robotics groups from around the world do, is using cloud tools in such scenarios. Cloud tools are programs that run directly from internet browser and where most of the data is stored and calculated on some servers, to which users have access via a browser app. This enables us to introduce students to different types of applications through a much-simplified process, where students usually need only to create an internet account to access the app functions.

Using cloud tools works particularly well when classes are aimed at online audiences in MOOCs (Massive Open Online Courses). Schools and organizations offer their coursework online both for their students and for online communities. Such courses can be offered to thousands or even hundreds of thousands of participants and technical support for installing some specialized software is, in such setting, very difficult. Therefore, many such courses offer online tools for the participants. For example, several Udacity [12] and Coursera courses have online exercises given along python code interpreter [6], while EdX courses offer an online version of MATLAB environment (for example, Discrete Time Signals and Systems [3]). Circuits and Electronics course by MIT uses online simulation tool for electronics [1].

There are several examples of using cloud or available online tools that could be used in normal curriculum. Such tools give wider access to laboratory equipment (via remote access) or give more varied exercises. Notably a Robotnicka robots can be accessed and programmed remotely by Slovak Students in the “remotely accessible robotics laboratory” [13]. A Robot Programming Network is a initiative to give several robotics labs tools for remote access and remote exercises both on real robots and on simulators, based on ROS (Robot Operating System) [4].

In this paper, we want to give some overview of what is possible with cloud applications, but also to explain the limitations of this approach. We give the example of our cloud app that we use when teaching basics of robotics – mymodel robot. We conclude with giving some recommendations based on our experience.

## 2 Various Programs Useful for Robotics

There is a multitude of cloud programs that could be useful in teaching robotics. As robotics is a wide field and various skills are needed, programs could range from 3D CADs, various programming environments and programs illustrating or guiding some particular robotics ideas.

As online tools generally require no setup, teachers can test and use a larger number of programs than with tools requiring installation, therefore, can give students the opportunity to compare different approaches or solutions.

We listed programs that we have used personally in our teaching endeavors, particularly in: basics of robotics course (bachelor/engineer level), medical and social robotics (project based master’s course), mobile robotics (bachelor/engineer level) and when tutoring students from robotics association.

## 2.1 3D Programs

Tinkercad [2] ([www.tinkercad.com](http://www.tinkercad.com)) is Autodesk's online CAD, designed primarily for building 3D printable shapes. The shapes are designed as a combination of basic shapes (cylinders, boxes) into desired objects or extruded from an SVG vector file. The typical use case is for students to design a robot part or some adapter.

The main advantage of the software is its simplicity, but it can also be a disadvantage as the software has only basic functions. It is a good tool to introduce students to 3D design, and have them printing their own parts rapidly. Parts can be also presented in Tinkercad gallery.

Onshape [7] ([www.onshape.com](http://www.onshape.com)) is a 3D parametric CAD program that is much more advanced than Tinkercad. Aimed at professionals, Onshape uses typical 3D design workflow. Parts are designed as 2D drawings, that can be then swept, extruded and combined.

When working with a parametric program, students can use variables when defining parts, and this allows them to compare different designs or iteratively improve parts. The program is free for open designs, that is for those that are available for other people to download and modify.

The program can be used to design nearly very complex devices, using assemblies of parts. Students use it to design complex parts for 3D printing or to design a multi-part assemblies.

## 2.2 Programming Tools

Mbed Platform ([www.mbed.com](http://www.mbed.com)): Mbed is a set of services for ARM developers, including the ARM Integrated Development Environment with the ability to program selected ARM boards in the cloud [9]. Similar in spirit to Arduino, the development platform has easy to use functions and a large set of libraries.

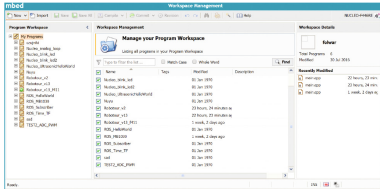
Mbed is fully compatible with Robot Operating System (ROS) which significantly helped our students to develop low-level control software for Nucleo platform being one of the ROS nodes [5] (see a photograph of mobile robot with Mbed board in Fig. 1).

Students can start programming by opening examples or some other people's projects and modify to their needs. The program is then compiled in the cloud and can be uploaded to the controller through putting the compiled file into a folder (the Mbed device is visible from the operating system as a memory device). The IDE has built-in version control and sharing ability.

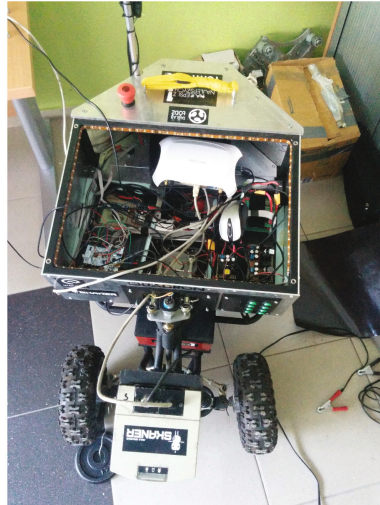
The Mbed software is mainly used by our robotics club students as they can introduce new members to the code base swiftly. Also, as an ARM-centric project, it gives the students an opportunity to work with modern, 32-bit hardware architectures of the similar price of 8-bit Arduino boards.

Repl.it

Repl.it ([repl.it](http://repl.it)) is one of the several online coding environments, giving the opportunity to present some working code (in various programming languages) and modify it [10]. Programming languages (in our case mainly Python) are



(a) Screen from students Mbed project for ROS-based mobile robot



(b) Photograph of robot with Mbed board ( Nucleo Board) used as a low-level controller

**Fig. 1.** Example of Mbed IDE and boards used in mobile robotics

available with popular libraries, such as Numpy or Scipy, for numerical or symbolic computing.

We use the environment to show example programs for typical robotics problems (inverse kinematics, trajectory generation) and students can modify the examples from their own computers. This can be used in class or as a part of a homework.

5p.js (JavaScript Processing)

5p.js (5p.js) is a JavaScript library for simplifying the process of building programs with a visual output in the browser [11]. The library is based on a popular and well-documented Processing programming environment for Java but it directly aims at interactive, browser-based programs. A p5.js-widget is an additional library that helps embed a p5.js, modifiable code on a website, with the ability to change the code, run and debug [14].

We use it when giving students written materials on a website, with the runnable code embedded in. Students can analyze the program along the explanation, and multiple versions of the program can be added to the post, to explain how the program is developed. The idea is similar to Knuth’s Literate Programming where a program is explained in natural language mixed with code snippets [8]. This allows students to see theory and experience the process of creating implementation in one coherent way.

As an example a blog post written for students (gentle.pl website), explaining numerical inverse kinematics, with a 5pjs-widget included for code exploration is presented in Fig. 2. Students can build through modifying the example code

We now need to "just" calculate the inverse of Jacobian. The Jacobian itself is:

$$J = \frac{\partial \vec{x}}{\partial \vec{q}} = \begin{bmatrix} -l_1 \sin(t_1) & -l_2 \sin(t_1 + t_2) \\ l_1 \cos(t_1) & l_2 \cos(t_1 + t_2) \end{bmatrix}$$

and its inverse:

$$J^{-1} = \frac{1}{l_1 l_2 \sin(t_2)} \begin{bmatrix} l_2 \cos(t_1 + t_2) & l_2 \sin(t_1 + t_2) \\ -l_1 \cos(t_1) & -l_1 \sin(t_1) \end{bmatrix}$$

We can see that for  $\sin(t_2)=0$  we cannot calculate the inverse of the Jacobian.

The robot's pose for which this happens is called a singularity and our algorithm will not be able to calculate the inverse kinematics.



```

p5.js ▶ Play ■ Stop
1 var l1=50;
2 var l2=50;
3 var q;
4 function setup() {
5   createCanvas(200, 200);
6
7
8   q=createVector(0,1); //początkowe wartosci q
9
10 }
11
12 function kinematyka_prosta(q)
13 /
14 /
  
```

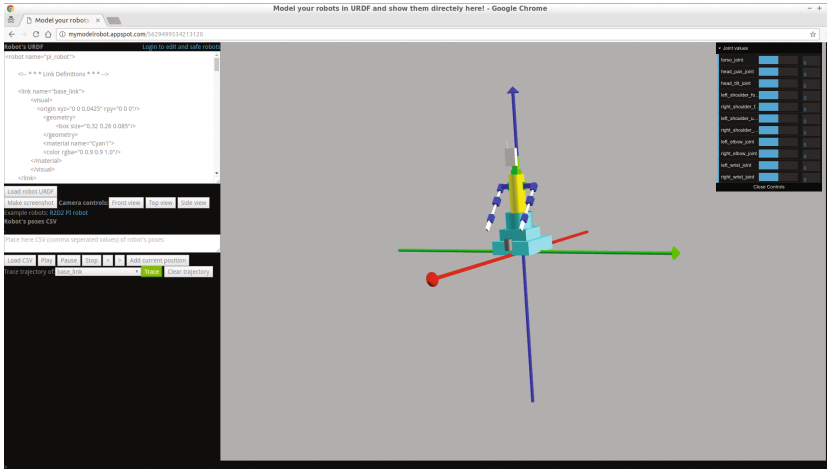
**Fig. 2.** A blog post written for students, explaining numerical inverse kinematics, with a 5pjs-widget included for code exploration.

inverse kinematics algorithm for two degree of freedom manipulator to make it follow the mouse movement. The program visualisation is upgraded real-time, so the students can see results of their changes.

### 2.3 MyModelRobot Development

Mymodelrobot ([mymodelrobot.appspot.com](http://mymodelrobot.appspot.com)) is our browser-based program to view and animate URDF based robots. We want to give our application as a successful development of cloud software for robotics education and to explain some technologies that are behind most of such apps.

The main enabling force is the development of javascript frameworks and its capabilities. In our app, we use WebGL, which is JavaScript API for rendering 3D models to give students the ability to view and animate robots. Logging, saving the work is done on Google Cloud through Google Accounts. The rendering and animation of robots are done using Three.js JavaScript framework. Additional JavaScript libraries (backbone.js, jquery, dat.gui) were used to realise Model-View-Controller scheme.



**Fig. 3.** Default screen from MyModelRobot application

Robots to be viewed correctly in MyModelRobot have to be described as a URDF (Universal Robot Description Format) model that students need to write themselves and doing that they learn about different coordinate frames and transformations between these frames. An image of a robot with 11 degrees of freedom viewed in our app can be seen in Fig. 3.

Universal Robot Description Format requires the definition of coordinate frames relative positions and orientations, types of joint and adequate axes of rotation/ translation and definition of elements that make a link (assembled from basic elements such as cylinders, boxes).

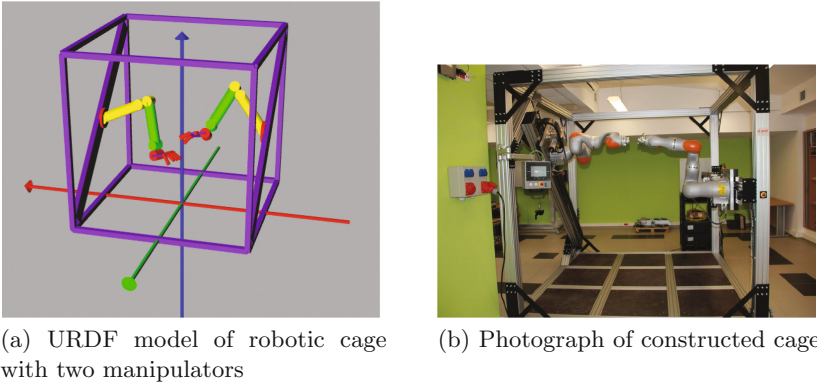
Recently, we have also included a Denavit-Hartenberg (D-H) table to URDF converter, so students can convert their D-H description to skeleton URDF model, and add visual details to that.

We use our program in different parts of the curricula. When doing lectures or exercises for “basics of robotics”, the lecturer can animate the coordinate systems attached to the robot, based on D-H notation and students can do the same on their own computers or mobile phones too. The experience of moving frames gives students a better understanding of key subjects of forward kinematics and robot description.

In “basics of robotics” laboratory, students build their own models of industrial robots using the MyModelRobot and proceed to use the tool as a way to view the results of Inverse Kinematics calculations. This was described in our previous paper on the RiE conference.

The tool is used also in student’s project work and extracurricular activities. Students modeled their mobile robots in MyModelRobot before proceeding to simulate them in Gazebo (a physics simulator that is connected to ROS). Our faculty used MyModelRobot to rapidly produce a simple model of a robotic cage as an illustration for a grant proposal (see Fig. 4).





**Fig. 4.** Example of MyModelRobot cloud tool for visualizing robots

Most of the use cases described here derive from the ease of use and availability of the tool. Students and faculty work from different computers, not necessarily their own, and a simple but always available tool is in many situations more preferred to the one that is powerful but requires a difficult setup.

### 3 Cloud Tools Pros and Cons

The cloud tools that could be used in robotics education are various, but there are some general pros and cons of using non-installable, cloud tools for the purpose of robotics education. Some contexts are more fitting than the others and we describe our approach in Recommendations subsection of this Section.

#### 3.1 The Pros of Cloud Tools

Facilitate communication, team working. As the app itself and the data are stored online, it is usually very easy for students to share their designs (through sharing functionality of the app) and work with other people when using cloud tools. This is specifically valuable because the teamwork ability and effective communication skills are a must for any kind of engineer.

No setup. The setup of installable tools is usually a major hurdle in their application. To install software the students need to have computers passing some requirements, space, etc. In the case of many tools, the students need to pass some licensing issues, which makes some of them use pirated versions – bad from the ethical point as well as the virus-prone. In the case of cloud applications, they usually require no setup in case of a browser app, other than having an up-to-date browser or a small installation on the mobile device.

No maintenance. When cloud software is used in a classroom setting, it requires no maintenance from the teacher as there is the newest version always available.

Computing power available. As some programs require the projects to be rendered/ compiled this could be too much for students machine.

Flexible paid options. Most of the cloud programs have paid options that are in form of monthly fee.

An easy way to compare apps, and use a multitude of them. As the apps generally have some free options, students and teachers have an easy way to check their functionality and use the most fitting versions – as their availability is the same. In the case of desktop programs, one tends to use the application already installed which can be not the best for the problem at hand.

### 3.2 The Cons of Cloud Tools

Using cloud programs have several cons, some of which can disqualify this option in some settings.

The biggest hurdle with cloud programs is that they give the user much less control over the program and the data. While this is the reason that there is no install and maintenance needed, users depend on the developers on proper maintenance and access to their app.

The risk of the company closing the app. While even for installable programs there is a problem when the program developer folds, as there would be no further versions of the program, in the case of cloud apps it could mean that there is no access to app whatsoever. The recent example of Codebender – programming interface for Arduino, shows that even large communities can close quite rapidly – much faster than it takes to change the curriculum.

The risk of app unavailability due to server upgrade. In case that the app servers are unavailable, the app will not work. In the case of desktop setup, the program is usually broken only on a single computer.

Also, because the apps usually auto-update, there is a risk that the new version will lack necessary features or be broken. Also, the teacher does not have time to prepare for the new version and it can be a surprise to the teacher and his/her students.

### 3.3 Recommendations

When teaching robotics with using cloud apps, there are some issues to consider, to provide good education experience and minimize the risk of failure.

Have examples and students data backup in interchangeable formats. As explained in the cons subsection, cloud services can close, which leads to students' and teachers' files being inaccessible. Having a backup data in some standard format (STL files for 3d shapes, step files for CAD, text files for c++, javascript, python source files), gives the chance for a transfer to other service or offline program. We also recommend checking whether the downloaded backup file is, in fact, working in other programs.

If possible, set up mirror copies of the websites. In the case of open source programs, the best option would be to run an own server with the cloud service.

This is possible with MyModelRobot and p5js. However, the need for server setup diminishes some of the advantages of cloud applications.

Cloud applications are best used in particular scenarios. One is the use in short courses or demonstrations. As installation of CAD programs or programming IDE's can take hours, it can leave considerable less time for the educational part of, for example, 10h course. Similarly, a demonstration of some functionality on a cloud program can lead to students experimenting on their own, as less setup is required. However, in long course, limited functionality, internet dependence and the possibility of lack of service can demotivate students.

Cloud applications are also good for homework and self-work, due to no setup as well as the ease of sharing the results or working in groups.

## 4 Conclusion

In the paper we presented a number of cloud applications with their use cases and described how our cloud application – MyModelRobot is designed and how it can be used in teaching robotics. We discussed the advantages and disadvantages of using cloud tools and provided some recommendations towards their classroom application.

In conclusion, we think that cloud app have a place in teaching robotics. In many settings using cloud apps gives students the ability to do their work faster, in other, they would not use an app at all and have a worse learning experience. However, it is certainly not the solution to all the problems, and teachers need to be aware of the inherent risks and decide accordingly.

We hope that this paper will inspire other educators in using cloud tools but also give them insight in what could be possibly wrong, and plan accordingly.

## References

1. Agarwal, A.: Circuits and electronics 1: Basic circuit analysis (2017). <https://www.edx.org/course/circuits-electronics-1-basic-circuit-mitx-6-002-1x-0>
2. Backman, K., Mononen, M.: About us – onshape (2017). <https://www.tinkercad.com/about/features>
3. Baraniuk, R.G.: (2017). <https://www.edx.org/course/discrete-time-signals-systems-part-1-ricex-elec301-1x>
4. Casa, G.A., Cervera, E., Moughlbay, A.A., Alemany, J., Martinet, P.: Ros-based online robot programming for remote education and training. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 6101–6106, May 2015
5. Folwarczyk, L.: Control of mobile robot based on Nucleo and ROS. Ph.D. thesis, Lodz University of Technology (2017)
6. Greiner, J., Wong, S.: An introduction to interactive programming in python (2017). <https://www.coursera.org/learn/interactive-python-1>
7. Hirschtick, J., McEleney, J., Li, T.: About us – onshape (2017). <https://www.onshape.com/about-us>
8. Knuth, D.E.: Literate programming. *Comput. J.* **27**(2), 97–111 (1984)

9. Ltd., A.: What is mbed? – mbed (2017). <https://www.mbed.com/en/about-mbed/what-mbed/>
10. Masad, A., Odeh, H.: About repl.it (2017). <https://repl.it/site/about>
11. McCarthy, L.: p5.js website (2017). <https://5p.js>
12. Norvig, P.: Design of computer programs. <https://www.udacity.com/course/design-of-computer-programs-cs212>
13. Petrovi, P., Balogh, R.: Deployment of remotely-accessible robotics laboratory. Int. J. Online Eng. (iJOE) **8**(S2), 31–35 (2012)
14. Varma, A.: About p5.js-widget (2017). <https://toolness.github.io/p.5.js-widget/>

# Needs, Opportunities and Constraints on the Way to the Wide Introduction of Robotics to Teaching at Secondary Vocational Schools

Mikulas Hajduk<sup>1</sup>, Zbigniew Pilat<sup>2</sup>, Adrian D. Olaru<sup>3</sup>,  
and Marek Vagaš<sup>1(✉)</sup>

<sup>1</sup> Technical University of Kosice, Kosice, Slovakia  
{mikulas.hajduk, marek.vagas}@tuke.sk

<sup>2</sup> Industrial Research Institute for Automation and Measurements PIAP,  
Warsaw, Poland  
zpilat@piap.pl

<sup>3</sup> Machine and Manufacturing Systems,  
University Politehnica, Bucharest, Romania  
aolaru\_5l@ymail.com

**Abstract.** The current situation of increasing European and global competition and the resulting downward pressure on prices and forcing at companies use modern automated and robotic manufacturing systems. After the state of the art analysis evaluation there were certain conclusions gained, showing lack of information from the field of robotics among graduates. Low level of education in the field of robotics is caused by poor material and technical facilities at secondary schools as well as by poor level of theoretical and practical experiences of teaching staff. Therefore, the staffs are not able to pass on important and satisfactory level of education in the field of robotics. This clearly demonstrates the need and necessity of education in the field of robotics that can be adequately insured by professionally trained teachers. Teachers will benefit in terms of gaining the latest information and knowledge from the field of industrial and service robotics, which is now highly popular and attractive giving them the benefit of professionalism. The paper presents the works within the RUSOS project, which is focused on educating of teachers of secondary vocational schools in the field of robotics.

**Keywords:** Robotics in education · Vocational education and training · Secondary vocational schools

## 1 Introduction

The current state in industry can be characterized by rapid development of robotics in almost all industrial areas. It is mainly focused on new generation of intelligent dual arm robots [1]. In terms of public life, here are service robots used for professional applications (e.g. health care, rescue operations) and personal assistance applications for humans, particularly for handicapped and older people. The number of robots deployed

in industry since 2010 is continually increasing. The main reason for the sharp increase of robot deployments in 2010 was the rapid development of automation with the aim to increase competitiveness, and thus achieve higher production and quality [2].

In 2016, the number of robots was exceeding over 200,000 [3]. In 2018, it would attack worldwide sale of threshold 400,000 industrial robots per year [4]. The focus on selected project priorities is designed based on development of robotics in all areas of industry with regard to project partnerships. The need for professional services is especially apparent in the automotive industry, where the most required employees are those with secondary vocational education working in sphere of service of automated and robotized equipment or at production lines. However, existing graduates are prepared insufficiently for these activities. Therefore, some employers provide the short-term courses in the supplier robotic companies or at universities due to profiling their robotic positions. They sometimes provide this type of courses themselves. The lack of these courses relates to their narrow specialization and superficial expressing the idea of robotic system [5].

One of the ways how to improve this situation can be seen in the introduction of such robotics subjects at secondary vocational schools that are missing today. Robotics is only part of certain subjects, without a coherent view. The need to resolve this problem should begin at substance of case, for example, by teachers training at secondary vocational schools in the field of robotics with the aim to have the relevant, innovative, high-quality and timely information from robotics. Primarily, RUSOS project focuses on teachers who will be able to transfer reached knowledge of robotics to their students at the professional level [6].

## 2 Background and Aims of the RUSOS Project

The RUSOS project focuses on the education of teachers at the secondary vocational schools in the field of robotics and on innovative, high-quality obtaining information from robotics. The most innovative part of the project includes the creation of the study materials for teachers of technical subjects that are created based on the essential as well as the latest knowledge from robotics [7]. Subsequently, the teachers transform obtained materials and knowledge into the contents of educational subjects, dual education or the optional courses. Planned innovative educational materials, ICT and a virtual laboratory platform are conceived as an interactive source of knowledge to enable interaction between the students and teachers.

Project outputs and activities are as follows:

1. An analysis of the requirements of engineering companies for recruiting the new employees, especially graduates of secondary vocational schools - finished output. The aim of intellectual output was to conduct an in-depth analysis in order to identify the needs and requirements of target groups, subsequently, implement in all countries of partnership. The representatives of target groups were directly contacted by the project partners, both via the personal meetings and the new form of data collection as online questionnaire survey. The evaluation of analysis was done for each country, but also globally.

2. A set of training materials for the secondary vocational school teachers in the field of robotics - ongoing output. Purpose of this output is the creation of training materials based on the requirements that are suggested by the production companies and the secondary vocational schools. Requirements have been obtained under output 1. The study materials will be compiled on the basis of underlying as well as latest knowledge of robotics. Their structure will be conceived as an interactive learning resource enabling interaction between students and teachers. Subsequently, teachers of secondary vocational schools will transform the obtained materials and knowledge into the curriculums of subjects, or include them as a part of related subjects.
3. An educational-training ICT platform for the secondary vocational school teachers in the field of robotics - ongoing output. Educational-training ICT platform is a major intellectual project output. Currently, it is elaborated into the stage of installation, administration and structure of e-learning portal Moodle. Moodle as a learning platform was created by a general concept of training course, defining respective roles and competences, as well as template of a standard lesson with the aim to test and comment by the project partners [8]. Moodle portal contains one sample lesson from industrial robotics.

### 3 Analyses of Needs and Requirements of Target Groups

Many secondary schools are not adequately equipped and hence cannot provide adequate training in robotics for students. Similarly, they do not have adequate and timely information from this field [9]. Therefore, it was prepared a depth analysis for identification of needs and requirements of target groups. Basic axioms for realisation of this analysis were characterized in the following points:

- Must be realized based on needs from automotive sector,
- Insufficient preparation of operators graduates in the field of automated and robotized equipment and production lines,
- Missing profiling subjects from robotics at secondary vocational schools with a coherent perspective,
- The need for training teachers at secondary vocational schools to obtain relevant information from the field of robotics,
- Short-term courses of firms and companies are specifically addressed only to a narrow range and low depth of robotics conception.

The analyses consist of two parts. First, an analysis of the requirements of engineering companies for recruiting of new employees - graduates of secondary vocational schools. Second, secondary school teacher's needs analysis for training in robotics. Analyses were carried out in all countries of partnership (Slovakia, Poland, and Romania). All partners were involved in the project and evaluations were carried out both for each country separately and for all partner countries globally [10].

It was aimed to obtain information about real needs of participants. To achieve required quality indicators, it was necessary to reach a minimum of 20 companies and

20 secondary vocational schools in each country of partnership. To sum up, it fills about 240 questionnaires, which can be regarded as sufficient sample to obtain relevant information. The results of these analyses were used directly for creation of intellectual output 2.

## 4 RUSOS Training System

RUSOS training system is an education system designed for training at automated and robotized systems for the teaching staff of secondary vocational schools, companies and students. It is focused on information from the field of industrial and service robotics that will be processed and available for the learners of ICT platform in the form of e-learning lessons [11].

### 4.1 Educational-Training ICT Platform

Educational training ICT platform is main intellectual output of RUSOS project. It is based on e-learning, whose main advantage is its availability strictly according to time and space that a user needs. This educational platform allows controlled and continuous access to study materials as well as the opportunity to exchange their knowledge and experience with other course participants through the implementation of various communication modules such as forum, chat and other [12].

### 4.2 E-Learning Course

Concerning to the development of e-learning education, e-learning course was created with a focus on industrial and service robotics [13]. It is implemented into the Moodle portal and divided into lessons. Prepared e-learning course is mainly focused on providing the essential as well as the latest knowledge from robotics concerning to three target groups:

- teachers,
- students,
- employees of companies.

These target groups were chosen because in case of the teachers, they do not have sufficient information from robotics. In case of students, it would be useful, if they obtained at least basic information from robotics. Finally, in case of employees, they will receive benefits in the form of access to the training materials, coaching and unlimited advantages of e-learning course.

Each lesson contains a theoretical part which explains relevant topic. It also includes appropriate images, presentations and also video files. At the end of each lesson, control questions for users of course can be found. Only after successful mastering of control questions, one can be passed to the next lesson. To complete the lesson, it is necessary to obtain minimum number of points [14].



### 4.3 Virtual Lab with Industrial and Service Robots

Special part of ICT platform will include a virtual laboratory with robotic technique that will be used for evaluation of practical knowledge resulted from the study. Virtual laboratory will be created of several virtual 3D models of various robots and other automated equipment. Individual virtual models of robots and other devices will be compiled into different configurations and thus, it is possible to create various robotic workplaces in virtual reality [15]. Virtual lab allows the users to work from their homes and to verify obtained knowledge of industrial and service robots simulating real workplace conditions. So, they can use virtual lab equipment which is not usually available to all persons for testing. Authorized users can access into laboratory at anytime from anywhere.

## 5 Future Development

To ensure a correct entrance into the virtual laboratory, it is necessary to allow entrance by only single user at predefined time. Therefore, this activity requires a schedule of individual users concerning to access of virtual laboratory. In order to avoid unauthorized entry, each access into the virtual laboratory has to be monitored and archived by software in form of log files, respectively by storing data into the database. We expect that it will need to store following information:

- login that will be associated by administrator of virtual laboratory and has to be unique for each user,
- user password with sufficient length,
- time of user access into virtual laboratory,
- time of activities in the laboratory,
- type of user activity during operation in virtual laboratory.

## 6 Summary

Project activities are intended to improve cooperation in the field of robotics education for teachers at secondary vocational schools as well as students and employers. Another aim of project is to improve cooperation and preparedness of successful graduates of secondary vocational schools and their integration into the labor force of companies. Secondary school teachers will benefit from a project by acquiring latest information and knowledge in the field of the industrial and service robotics, in order to amend themselves education in this highly attractive, innovative and demanded field. This will enhance their job and allow them the possibilities for career progression and better enforcement in the labor market.

**Acknowledgements.** The paper presents results of researches supported by EU within the project The paper presents results of researches supported by EU within the project RUSOS “Robotics for teachers of secondary vocational schools”, 2015-1-SK1-KA202 - 008970, under the ERASMUS+ Programme. This publication represents only author’s opinion and neither the European Commission nor the National Agency is not responsible for any of the information contained in it.

## References

1. Smith, C., Karayiannidis, Y., Nalpantidis, L., Gratal, X., Qi, P., Dimarogonas, D.V., Kragic, D.: Dual arm manipulation: a survey. *Rob. Auton. Syst.* **60**, 1340–1353 (2012). doi:[10.1016/j.robot.2012.07.005](https://doi.org/10.1016/j.robot.2012.07.005)
2. Tolnay, M.: Automated Production Systems and Industrial Robot Effectors (In Slovak). FX s.r.o, Bratislava (2008)
3. IFR International Federation of Robotics. [https://ifr.org/img/uploads/Executive\\_Summary\\_WR\\_Industrial\\_Robots\\_20161.pdf](https://ifr.org/img/uploads/Executive_Summary_WR_Industrial_Robots_20161.pdf)
4. Hajduk, M., Baláž, V., Koukolová, L., Zubrzycki, J.: Proposal of multi robotic cells for production lines. *Appl. Mech. Math.* **613**, 60–65 (2014). doi:[10.4028/www.scientific.net/AMM.613.60](https://doi.org/10.4028/www.scientific.net/AMM.613.60)
5. Baláž, V., Vagaš, M., Pachniková, L.: Presentation of robot activity using web technologies (In Slovak). In: Innovation Process in e-Learning: the Forth Scientific International Conference, Bratislava, pp. 1–5 (2010)
6. Vince, T., Kováč, D., Molnár, J.: VMLab in the education. In: *Sistemas y Tecnologías de Información: Actas de la 7ª Conferencia Ibérica de Sistemas y Tecnologías de Información*. AISTI, Madrid, pp. 334–338 (2012)
7. Hricko, J., Havlik, S.: Design of compact compliant devices – mathematical models vs. experiments. *Am. J. Mech. Eng.* **3**, 201–206 (2015). doi:[10.12691/ajme-3-6-9](https://doi.org/10.12691/ajme-3-6-9)
8. Świć, A., Wołos, D., Zubrzycki, J., Opielak, M., Gola, A., Taranenko, V.: Accuracy control in the machining of low rigidity shafts. *Appl. Mech. Mater.* **613**, 357–367 (2014). doi:[10.4028/www.scientific.net/AMM.613.357](https://doi.org/10.4028/www.scientific.net/AMM.613.357)
9. Duchoň, F., Huňady, D., Dekan, M., Babinec, A.: Optimal navigation for mobile robot in known environment. *Appl. Mech. Mater.* **282**, 33–38 (2013). doi:[10.4028/www.scientific.net/AMM.282.33](https://doi.org/10.4028/www.scientific.net/AMM.282.33)
10. Online questionnaire survey. [https://docs.google.com/forms/d/e/1FAIpQLSdEftVd2cZPFXfa\\_wW7zCSBMcjZERa4wKOL0H-OCihlfAgCRw/viewform?c=0&w=1](https://docs.google.com/forms/d/e/1FAIpQLSdEftVd2cZPFXfa_wW7zCSBMcjZERa4wKOL0H-OCihlfAgCRw/viewform?c=0&w=1)
11. RUSOS Moodle portal. <http://rusos.sjf.tuke.sk/moodle/>
12. Thorsteinsson, G.: Using ICT for training teachers in design and technology education (TTDTE). *I-Manager's J. Educ. Technol.* **9**, 9–13 (2012)
13. Novák, P.: *Mobile Robots - Drives, Sensors, Control* (In Czech). Nakladatelství BEN - Technická literatura, Prague (2005)
14. Technical University of Kosice. <http://roboreha.sjf.tuke.sk/moodle/course/category.php?id=2&lang=en>
15. Vagaš, M., Sukop, M., Varga, J.: Design and implementation of remote lab with industrial robot accessible through the web. *Appl. Mech. Mater.* **859**, 67–73 (2016)
16. Vagaš, M., Sukop, M., Varga, J.: Design and implementation of remote lab with industrial robot accessible through the web. In: *Applied Mechanics and Materials: ICMERA 2016*, Switzerland, TTP, 2016, vol. 859, pp. 67–73 (2016). ISBN 978-3-03835-667-7

# An Open Robotics Environment Motivates Students to Learn the Key Concepts of Artificial Neural Networks and Reinforcement Learning

Tapani Toivonen<sup>(✉)</sup>, Ilkka Jormanainen, and Markku Tukiainen

University of Eastern Finland, Joensuu, Finland  
{tapani.toivonen,ilkka.jormanainen,markku.tukiainen}@uef.fi

**Abstract.** Educational robotics is a widely recognized tool to motivate students and concretize abstract and complex topics, such as artificial intelligence in computing education curricula. Lego Mindstorms series is one of the most popular robotics platform due to its flexibility and relatively cheap price. We used Lego Mindstorms EV3 robots with a novel Open Learning Environment for Artificial Intelligence (OLE-AI) to teach concepts of reinforcement learning and artificial neural networks (ANNs) to computer science students. OLE-AI uses a white box approach to expose internal structures of an ANN to students. Results from the pilot study with OLE-AI indicate that the participating students were able to deepen their knowledge about AI topics through a practical and open exercise that involved them in controlling EV3 robots by manipulating the ANN and Q-Learning algorithm.

**Keywords:** Educational robots · Q-Learning · Reinforcement learning · Artificial neural networks · EV3 robots

## 1 Introduction

Educational robotics is a widely used tool in computing education especially at K-12 level [2], but also institutes of higher education have increasingly adopted these tools [13] as a part of their curricula. Flexibility of educational robots allow them to be used in various learning contexts and levels. Nowadays it is not unusual to integrate educational robotics to courses that are connected to fundamentals of programming, algorithms, programming languages, operating systems, or artificial intelligence (AI) [13]. Selection of available educational robotics kits is wide, and institutions can easily find robotics solutions to match their needs and budget. Some of the robotics kits are too expensive or complex to be purchased and used by the universities as teaching tools for general computing education, but reasonably priced alternatives such as Lego Mindstorm series exists and they are widely adopted in different teaching contexts [24].

The recent developments and raising interest in AI creates a demand for computing specialists to possess an understanding in the subject. AI as a wide discipline and its sub-domain machine learning are taught all over the world as

a part of computing degrees. Both AI and machine learning are considered as hard subjects to teach, and various tools for teaching these topics have been developed [14]. However, existing tools are usually driven by black box thinking where only the inputs and the outputs of artificial intelligent algorithms are visible to learners [4]. In the field of machine learning, the artificial neural networks are the most notable example of black boxes [4].

In our previous work [20], we have developed a prototype of a learning tool framework that aims to bring internal functionality of an ANN to a white box instead of presenting it with a traditional black box approach. The framework allows usage of screen-based visualizations or physical computing devices, such as educational robotics, to concretize the learning experience. In this paper, we present results from a pilot study to examine perception of computer science major students towards using *Open Learning Environment for Artificial Intelligence* (OLE-AI) that uses the previously developed framework and Lego Mindstorms EV3 robots. In the study, the participants were using OLE-AI to teach an EV3 robot controlled by an artificial neural network to drive around and avoid obstacles in an open and unpredictable environment. The study presented in this paper aims to answer to the following research questions:

**RQ1:** Are there indicators of benefits that students gain when using a white-box approach in OLE-AI with educational robots to learn fundamentals of ANNs and reinforcement learning?

**RQ2:** Is it technically possible to use OLE-AI to introduce the concepts of artificial neural networks to computer science major students?

The paper is organized as follows. First, we will discuss about the previous work related to introducing machine learning concepts to the students with the help of educational robotics. We also present our earlier work with the framework development. Second, we will introduce the research setting for the experiment, methodology, and main results obtained from the study. We will conclude the paper with a discussion about the implications of the results and directions for the future work to expand the current research context.

## 2 Background

Reinforcement learning is sub-domain of machine learning and it is especially well-suited to control robots [18]. The basis of the reinforcement learning is in rewarding and punishing. The reinforcement learning agent interacts with the environment and receives feedback from the actions it has chosen. There are many reinforcement learning algorithms available for different use cases. For our tools, we have applied the *Q-learning algorithm* [23] to be used together with EV3 robots. In this section, we will discuss about the earlier work related to reinforcement learning and educational robotics. We also cover the JS-Eden environment with the OLE-AI that we have developed for machine learning education purposes.

## 2.1 Related Work

Lego Mindstorms is an educational robotics series that has evolved through three different generations. EV3 is the newest and the most powerful model in terms of processing power [9] and connectivity. Earlier versions of Lego Mindstorms series are RCX and NXT series. Lego Mindstorms robots can be programmed with a specific visual programming environment that is bundled with robot sets [11]. It is also possible to use other programming tools, such as Java to develop programs for Lego robots [17]. Besides explicitly programming a Mindstorms robot to achieve a certain goal, machine learning techniques have been applied to Mindstorms robots for both research and educational purposes [25]. For example, Van Der Vlist et al. [22] arranged a course for design students where Q-learning and supervised learning methods were applied to NXT robots with Java programming language. The objective of the arranged course was to introduce the concepts of ANNs to the students.

Vamplew [21] suggested using Q-learning method and Mindstorms RCX series to introduce the concepts of the reinforcement learning but instead of ANNs, they implemented Q-learning with a look-up table. The look-up table approach was chosen because of the limited processing power of the used Lego intelligent bricks. Look-up table is easier to implement than an ANN, but the major drawback is that look-up tables are suitable only for simple environments [3]. Processing power required to manipulate even a simple ANN is usually so big that using Lego Mindstorm intelligent bricks together with ANNs is not feasible.

Irigen-Gioro [9] used Q-learning method together with Lego NXT series and C language to introduce the concepts of the reinforcement learning to computer science graduate students. The goal was to develop a robot that would obey to its owners commands by using an ultrasonic and a light sensor. Ultimately, the robot would learn the optimal policy for avoiding obstacles.

Besides the reinforcement learning paradigm, Klassner [12], and Parsons and Sklar [16] integrated Lego RCX robots to an AI course where the students were offered large amount of tasks related to intelligent agents. In the course organized by Klassner [12], the students in general had strong background in programming languages and instead of using the visual programming environment provided by Lego, the students used NQC (Not Quite C) to program the robots. Similarly Parsons and Sklar [16] used NQC during their AI course.

The feedback given by the participating students in the abovementioned courses was mainly positive. Using Mindstorms robots to introduce the AI concepts enhances the creativity of the students [9], and using the Mindstorm robots is an economical choice [9]. Furthermore, combining AI with Mindstorms robots let students to deal with real life complexities [9]. Using physical robots in AI education supports the active role of students by letting them to manipulate concrete objects and assimilate new knowledge through observation. As Klassner [12] addresses, the visual programming environment provided by Lego is not very flexible, but it is possible to use other programming languages to program Mindstorm robots. Also one target of the criticism against using Lego

Mindstorms robotics is the limited processing power of the intelligent bricks. EV3 series robots are by far the most advanced bricks but also they do fall in the category of limited power embedded systems with only a 300 MHz ARM processor [8].

Besides the limitations that the previous researches encountered, all of the described interventions follow the traditional write - compile - execute cycle where the learning process of a machine learning algorithm can be observed only by looking the behaviour of the robot. In this model, sensor values serve as inputs to the machine learning algorithm and the behaviour of the robot represents the output of the algorithm. Thus the state and internal functionality of the machine learning algorithm or ANN is explicitly hidden from the students in a black box. We have overcome the limitations of the Lego Mindstorms intelligent bricks by keeping the neural network and processor intensive calculations in a host computer that controls remotely a Lego EV3 robot.

In our approach, the robot serves as a real-time concretization object that brings the status and quality of the ANN explicitly visible to the learner. Furthermore, the robot sends its sensor readings to the host computer for input to the ANN. The host computer controls driving of the robot based on the state of the ANN. This solution allows visualizing the state and structure of the ANN to the learning in the OLE-AI environment running in the host computer, hence bringing the ANN from black box to a white box. Advantage of our approach is also that use of ANN is not limited by the computing capacity of EV3 intelligent brick or Arduino microcontroller. This allows usage of more complex scenarios and rich visualizations of the ANN, comparing to environments where the machine learning algorithm is implemented in the robot. Also, all changes to the construals controlling the robot or Q-learning agent are applied on the fly according to the EM principles [20] without need to re-compile or re-interpret the script after changes. All changes to ANN structure or parameters are reflected immediately in the robot's behaviour. We argue that this helps learners to understand internal functionality of the ANN and machine learning algorithm better than a tool following the traditional black box approach. Next in this paper we present Empirical Modelling approach and JS-Eden tool that are technologies behind the OLE-AI environment.

## 2.2 Empirical Modelling and Making Construals

OLE-AI, as well as our previous work in the field, are developed on the top of JS-Eden modelling environment, which is a part of Empirical Modelling (EM) toolset. EM is a computer-based modelling doctrine that aims to develop models (computer-based artefacts known as construals) to represent subjective experience of the modeller about a subject. Empirical Modelling tools are not limited to computing education only, but tools and EM approach in general are applicable in a wide range of teaching contexts. The current state of EM tools development is focused on building interactive open educational resources (OERs) that enable collaboration between developers, teachers, and students in an unified web-based environment where both development and use of a construal

take place [6] without traditional software development cycle. The current EM environment, JS-Eden, allows interconnection between screen-based construals and physical computing devices such as Arduinos<sup>1</sup> and Lego Minstroms EV3 robots<sup>2</sup>. Furthermore, the extensions developed to JS-Eden allow integration of Q-learning agents into the construal for example to control a robot's behaviour through an artificial neural network (ANN).

One of the goals of Empirical Modelling is to drive an blended development and use process of open educational resources (OERs). This process is referred *making construals* (MC). Currently, MC takes place in a browser-based JS-Eden environment that uses Node.js runtime environment [1] in the backend. Through three key EM concepts, *observables, dependencies and agents* [5], a user can create interactive OERs that capture the subjective experience of a phenomenon into a sense-making model [7].

Observable is a feature of a referent that the user is modelling while dependencies are relationships between observables that describe how the observables are indivisibly linked in change. Unlike constraints that express persistent relationships in a closed world, dependencies represent the expectations of a user about how the change of one feature will have an influence on the related features [7]. An agent is a set of notations in the domain being modelled that is being perceived as a capable of initiating a state-change of the observables [7].

One of the key values of making construals activity is the openness, and ANNs learning tools usually operate in a black box. ANNs and open making construals activity were previously blended together to predict disability severities from road traffic accidents in Thailand [19]. Furthermore, making construals activity played a crucial role to open data mining process through the decision tree classifier in the context of educational robotics in [10]. In both cases, the openness of making construals and associated tools led to deeper understanding of a prediction process itself.

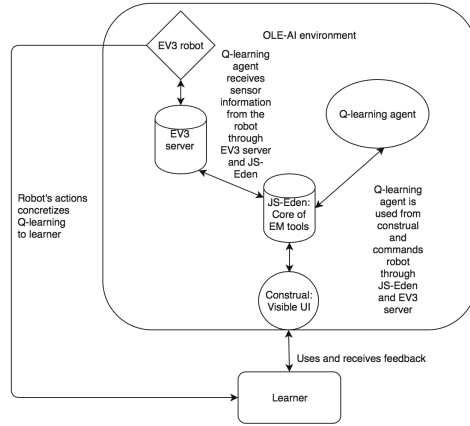
### 2.3 JS-Eden and OLE-AI

In our previous research we introduced a JS-Eden extension to physical computing and machine learning that links JS-Eden to EV3 robots or Arduino micro-controllers through either a USB cable or Bluetooth connection. In addition to establish a connection between JS-Eden and physical devices, the framework allows users to integrate a Q-learning agent to making construals activity [20]. The current OLE-AI tool has been built on top of this framework. The Q-learning algorithm in the OLE-AI uses a recurrent neural network [15] as a function approximator.

Connection between EV3 or Arduino and JS-Eden is established through Node.js servers that handle the communication over web sockets and serial ports. Controlling the robots and Q-learning agent is enabled by standard JS-Eden syntax and OLE-AI supports the use of observables, dependencies and agents.

<sup>1</sup> <https://www.arduino.cc>.

<sup>2</sup> <https://www.lego.com/en-us/mindstorms/about-ev3>.



**Fig. 1.** Architecture of OLE-AI environment.

On runtime, the communication between robots and JS-Eden is managed by Node.js servers, and Q-learning implementation is loaded to users' web browser enabling Q-learning agent to be used as a standalone add-on with screen-based construals. The Q-learning implementation of the OLE-AI follows the EM principles of openness. All layers of the ANN are exposed real-time to the user and they can be viewed from *observable list* of JS-Eden. Both activation values of the neurons and the weights of the ANN can be viewed simultaneously. The architecture of OLE-AI and its relations to JS-Eden platform is depicted in Fig. 1.

### 3 Pilot Study

In November 2016, we arranged a pilot study to research how computer science major students perceive OLE-AI for learning the basic concepts of ANNs and reinforcement learning. The purpose of the study was two-fold. First, our goal was to study how use of OLE-AI with Lego EV3 robots change students' knowledge and interpretation about the internal functionality of ANNs. Our second goal was to study technical aspects of using JS-Eden, OLE-AI, and Lego Mindstorms EV3 robots.

#### 3.1 Research Setting

As this study was for piloting both the OLE-AI environment and research setting for further, more extensive studies, number of participants was intentionally small ( $n = 4$ , three males and one female). The students were computer science majors from School of Computing, University of Eastern Finland. During the two-hour session students were asked to use a pre-made construal in the OLE-AI environment to control Q-learning agent to drive a Lego Mindstorms EV3 robot. The predefined goal for the students was to teach Q-learning agent to control

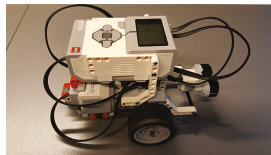


the EV3 robots so that the robot would avoid collision with the obstacles in the classroom. All students were familiar with Empirical Modelling, JS-Eden, and making construals approach as they had participated to a workshop in these topics in the previous semester. The choice of having participants with previous knowledge on making construals was driven by the piloting nature of the study, as previous knowledge about the tools would help them to focus on the actual topics of the study, instead of familiarizing themselves first with the JS-Eden environment. Three of the students had some theoretical knowledge about ANNs, but none of them had any experience in the reinforcement learning. At the beginning of the study, the participants were asked to answer the following five questions mapping their understanding about ANNs and reinforcement learning. The same five questions were asked after the experiment.

1. What is an artificial neural network?
2. What different parts artificial neural networks do have?
3. How do rewarding and punishing relate to artificial neural networks?
4. In the context of reinforcement learning, what are the inputs and the outputs of the artificial neural network?
5. Describe what happens in the learning phase of artificial neural network?

The questions 1, 2 and 4 tested students' general knowledge about ANNs whereas questions 3 and 5 tested their knowledge about reinforcement learning. After filling the pre-questionnaire, students were given a short introduction about ANNs. During this introductory session, participants were exposed to a graph-like ANN visualization and a description how inputs of the ANN are treated during the reinforcement learning process to form ANN outputs was also given to them.

After the introductory session, the participants selected a pair to work with. Both pairs had a laptop with OLE-AI environment running on the top of locally installed Node.js server. Participants were provided a pre-assembled Lego EV3 robot (Fig. 2), which connected to OLE-AI environment over Bluetooth connection. The robots were equipped with an ultrasonic sensor to measure distances to obstacles. When connected to the OLE-AI environment, the ultrasonic sensor value acted as a state of the Q-learning agent. The possible actions of the Q-learning agent were "forward", "backward", "left" and "right". The students were able to make changes to the number of hidden neurons of the Q-learning agent, learning rate, epsilon greediness, the size of the memory that the Q-learning agent used, gamma value (future reward discount factor), rewarding



**Fig. 2.** Model of the EV3 robot used in the study

and punishing, and iterations that were used to fold the memory. All these parameters were provided in OLE-AI environment, and all changes to the parameters were applied on the fly in the ANN, allowing participants to observe effect of the modifications real-time.

### 3.2 Materials and Methods

During the pilot study, the data from the students was collected with questionnaires that the students filled in before and after the experiment. The questionnaires were similar except that in the post-experiment questionnaire, we asked feedback about how students felt participating the experiment. Besides the questionnaires, participants' laptop screens were recorded and additional field notes were taken. Also log files from JS-Eden containing changes that the participants made to the construals, were saved.

The research material collected was analyzed qualitatively. The main analysis was based on a comparison of the pre-questionnaire and post-questionnaire answers. Analysis was supported by the collected field notes and the recordings. Field notes, for instance, pointed out that the class room environment that was used during the pilot study was too challenging for robots' navigation capabilities because of the amount of obstacles, such as chairs and tables.

### 3.3 Results

Results of the study indicate two major findings. First, participants' answers indicate that OLE-AI helped those without prior knowledge about ANNs to comprehend the basic fundamentals. The questions 1, 2, and 5 tested the participants' knowledge about ANNs, and pre-experiment questionnaire answers indicate that participants, except of Student A, had some level of theoretical understanding about ANNs.

Student A, who had no prior knowledge about the ANNs, answered to the first question (*What is an artificial neural network?*) as follows. The answer indicates that Student A gained during the experiment new knowledge about how ANNs are implemented.

*Algorithm which sorts data?* (Student A, pre-questionnaire)

vs.

*Multidimensional array that has values.* (Student A, post-questionnaire)

Answers from other students with more prior knowledge about ANNs did not evolve significantly, but the answers took more precise form. For example, Student B answered to the first question:

*ANNs are used in machine learning. A network structure which is used to predict the most probable event* (Student B, pre-questionnaire)

vs.

*A data structure which is used in machine learning and can be used to teach a computer* (Student B, post-questionnaire)

The Question 2 (*What different parts artificial neural networks do have?*) tested the participants' knowledge in the architecture of ANNs. Again, the answer from Student A during the pre-questionnaire showed the lack of knowledge in ANNs. However, during the experiment Student A gained new and precise information about the architecture of ANNs:

*Nodes and leaf nodes.* (Student A, pre-questionnaire)

vs.

*Input, hidden and output layer which all have neurons.* (Student A, post-questionnaire)

Also the answers from the participants with more experience in ANNs evolved again into more precise form.

*Input, layers, output* (Student B, pre-questionnaire)

vs.

*Input layer, hidden layer, output layer* (Student B, post-questionnaire)

*Usually ANNs are layered so that they have an input layer, and an output layer, and between them one or (usually) more hidden layers.*" (Student D, pre-questionnaire)

vs.

*Input / hidden / output layer, and connections with weights between them.* (Student D, post-questionnaire)

As the second major finding, results indicate that students were able to transform their existing, theory-based knowledge about ANNs into more practical form: the experiment had influence in the answers of the post-experiment questionnaires. Both Question 3 and Question 4 were related to reinforcement learning instead of ANNs. The post-experiment questionnaires indicate that the students achieved some level of practical understanding in the reinforcement learning by using the OLE-AI with EV3 robots. For example, Student C answered to Question 3 (*In the context of reinforcement learning, what are the inputs and the outputs of the artificial neural network?*) as follows:

*Inputs are feature vectors and outputs are classifications. A reverse 'input' results from the output; reinforcing punishments or rewards*" (Student C, pre-questionnaire)

vs.

*The inputs are the current state and outputs a chosen action* (Student C, post-questionnaire)

<Empty answer > (Student B, pre-questionnaire)

vs.

*Inputs are numbers, for instance sensor values and the outputs are actions such as go forward, go left etc.* (Student B, post-questionnaire)

Overall, it seems that using OLE-AI environment together with EV3 robots could lead to more pragmatical understanding about ANNs and the reinforcement learning. The questions related to reinforcement learning were the most difficult for students to answer during the pre-questionnaire, but according to the

post-experiment answers, the participants' knowledge level about reinforcement learning increased. It is crucial to bear in mind that the number of subjects in this pilot study was low, and generalization of the results is not possible to large extent. However, the results form a good basis for the further experiments with larger number of the students.

## 4 Conclusions

Besides K-12 education, Lego Mindstorms robots have been used to teach the basics of AI at the university levels [13]. It seems that using physical robots in the AI related courses to increases the motivation of the students [22], while enabling the students to use their creativity in a practical way. Our approach with the OLE-AI tool opens the machine learning process by exposing the whole ANN to the students.

The comparison between the pre-questionnaires and the post-questionnaires indicates that the participating students gained a level of practical understanding of the concepts related to the reinforcement learning and ANNs. While the participating students deepened and practicalized their knowledge in ANNs during the pilot study, they were able to understand the underlying mechanisms of rewarding and punishing in the reinforcement learning.

In this paper we introduced two research questions. Based on the research result, we present answers to these questions as follows.

**RQ1:** *Are there indicators of benefits that students gain when using a white-box approach in OLE-AI with educational robots to learn fundamentals of ANNs and reinforcement learning?*

The results of the pilot study indicate that the use of the OLE-AI helps students to get familiar with the concepts of the ANNs and the reinforcement learning. Use of OLE-AI also helps to turn their existing theory-based knowledge to more pragmatic and concrete.

**RQ2:** *Is it technically possible to use OLE-AI to introduce the concepts of artificial neural networks to computer science major students?*

During the pilot study no technical challenges related to OLE-AI occurred, and the platform built with JS-Eden and the EV3 robots is stable for more extensive studies.

## 5 Discussion and Future Work

The results from this pilot study with OLE-AI environment were encouraging, and opening the ANNs internal functionality from a black box to a white box seems to be beneficial for students' learning. In particular, their concepts of ANNs turned to more concrete and vocabulary they used in answers during the post-experiment questionnaire got more pragmatic forms.

Besides answering the questions about the ANNs and the reinforcement learning, all students gave us comments and suggestions related to the experiment.

Feedback was mostly positive with some issues that were arised. Student A felt that working in pairs was problematic because there appeared to be some disagreements about how to train the Q-learning agent during the experiment. In the feedback, the Student A mentioned that “*I would have got more out from the experiment if I worked alone, I always disagreed with my pair...*”. Although working with a peer student seemed challenging, Student A admits that “*...I gained new information.*”. In the future experiments we may have to take into account the possibility to let participants to work on their own to avoid challenges regarding teamwork.

The other challenge appearing in the feedback was that the experiment environment was not the best one. The room was filled with chairs, tables, and other obstacles and it was challenging for the EV3 robots to observe surrounding with only one ultrasonic sensor. Thin legs of chairs and tables were not visible to the sensors, leading to unrealistic input to the ANN. This easily leads to difficulties to comprehend how the ANN is working in a particular moment when sensor readings are not meeting the real-world situation. Student C wrote: “*Set up the area for training the robot in a controlled way*”. For the future experiments we have to clear the environment from all of extra obstacles that may interfere unnecessarily sensor readings and in this way, the purpose of the experiment.

Despite the promising results presented in this paper, further studies with larger number of subjects are needed to get a full picture about the benefits of OLE-AI as a tool for familiarizing students with concepts of reinforcement learning. In this study, we focused on the reinforcement learning, but OLE-AI could be used also to introduce concepts of unsupervised and supervised machine learning by opening the undergoing processes in a similar way as presented in this paper.

## References

1. Js-eden source code. <https://github.com/EMGroup/js-eden>. Accessed 12 May 2016
2. Altin, H., Pedaste, M.: Learning approaches to applying robotics in science education. *J. Baltic Sci. Educ.* **12**(3), 365–377 (2013)
3. Baird, L., et al.: Residual algorithms: reinforcement learning with function approximation. In: *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 30–37 (1995)
4. Benítez, J.M., Castro, J.L., Requena, I.: Are artificial neural networks black boxes? *IEEE Trans. Neural Netw.* **8**(5), 1156–1164 (1997)
5. Beynon, M.: Radical empiricism, empirical modelling and the nature of knowing. *Pragmatics Cogn.* **13**(3), 615–646 (2005)
6. Beynon, M.: Computing applications from an empirical modeling perspective: progress, projects and prospects. In: *2016 8th International Conference on Knowledge and Smart Technology (KST)*, p. 318. IEEE (2016)
7. Beynon, M., Roe, C.: Computer support for constructionism in context. In: *Proceedings of the IEEE International Conference on Advanced Learning Technologies*, pp. 216–220. IEEE (2004)
8. Danahy, E., Wang, E., Brockman, J., Carberry, A., Shapiro, B., Rogers, C.B.: Lego-based robotics in higher education: 15 years of student creativity. *Int. J. Adv. Robot. Syst.* **11**, 1–15 (2014)

9. Irgen-Gioro, J.J.Z.: Teaching artificial intelligence using lego. In: Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS), p. 209. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp) (2016)
10. Jormanainen, I., Sutinen, E.: Using data mining to support teacher's intervention in a robotics class. In: 2012 IEEE Fourth International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), pp. 39–46. IEEE (2012)
11. Kim, S.H., Jeon, J.W.: Programming lego mindstorms nxt with visual programming. In: International Conference on Control, Automation and Systems, ICCAS 2007, pp. 2468–2472. IEEE (2007)
12. Klassner, F.: A case study of lego mindstorms<sup>TM</sup> suitability for artificial intelligence and robotics courses at the college level. *ACM SIGCSE Bull.* **34**, 8–12 (2002). ACM
13. Klassner, F., Anderson, S.D.: Lego mindstorms: not just for k-12 anymore. *IEEE Robot. Autom. Mag.* **10**(2), 12–18 (2003)
14. Michal, D.S., Etzkorn, L.: A comparison of player/stage/gazebo and microsoft robotics developer studio. In: Proceedings of the 49th Annual Southeast Regional Conference, pp. 60–66. ACM (2011)
15. Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., Khudanpur, S.: Recurrent neural network based language model. In: Interspeech, vol. 2, p. 3 (2010)
16. Parsons, S., Sklar, E.: Teaching ai using lego mindstorms. In: AAI Spring Symposium (2004)
17. Pedersen, R.U., Nørbjerg, J., Scholz, M.P.: Embedded programming education with lego mindstorms nxt using java (lejos), eclipse (xpairtise), and python (pymite). In: Proceedings of the 2009 Workshop on Embedded Systems Education, pp. 50–55. ACM (2009)
18. Peters, J., Vijayakumar, S., Schaal, S.: Reinforcement learning for humanoid robotics. In: Proceedings of the Third IEEE-RAS International Conference on Humanoid Robots, pp. 1–20 (2003)
19. Rungrattanaubol, J., Na-udom, A., Harfield, A.: An exploratory neural network model for predicting disability severity from road traffic accidents in Thailand. In: Proceedings of the Third International Conference on Knowledge and Smart Technologies, vol. 2011, p. 23 (2011)
20. Toivonen, T.: Using js-eden to introduce the concepts of reinforcement learning and artificial neural networks. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, pp. 165–169. ACM (2016)
21. Vamplew, P.: Lego mindstorms robots as a platform for teaching reinforcement learning (2004)
22. Van Der Vlist, B., Van De Westelaken, R., Bartneck, C., Hu, J., Ahn, R., Barakova, E., Delbressine, F., Feijs, L.: Teaching machine learning to design students. In: International Conference on Technologies for E-Learning and Digital Entertainment, pp. 206–217. Springer, Heidelberg (2008)
23. Watkins, C.J., Dayan, P.: Q-learning. *Mach. Learn.* **8**(3–4), 279–292 (1992)
24. Weinberg, J.B., Yu, X.: Robotics in education: low-cost platforms for teaching integrated systems. *IEEE Robot. Autom. Mag.* **10**(2), 4–6 (2003)
25. Xu, K., Wu, F., Zhao, J.: Simplified online q-learning for lego ev3 robot. In: 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), pp. 77–80. IEEE (2015)

# Author Index

## A

Ackovska, Nevena, 193  
Albó-Canals, Jordi, 100  
Avsar, Cem, 26

## B

Bağcı, B. Baransel, 88  
Bautista, Moises, 226  
Becerra, Juan Carlos, 226  
Bellas, Francisco, 226  
Ben-Ari, Mordechai, 77, 132  
Ben-Bassat Levy, Ronit, 132  
Blersch, Martin, 180  
Bonandin, Andrea, 289  
Brief, Klaus, 26

## C

Canaleta, Xavier, 100  
Čehovin Zajc, Luka, 250  
Cvetkovic-Lay, Jasna, 126

## D

De Martini, Daniele, 289  
Descamps, Sander, 257  
Doms, Natan, 257  
Duro, Richard, 226

## F

Facchinetti, Tullio, 289  
Faiña, Andres, 226  
Ferrarelli, Paola, 274  
Filippov, Sergey, 38  
Fradkov, Alexander, 38  
Frank, Daniel, 238  
Friebronn Yesharim, Mor, 77

## G

Gerndt, Reinhard, 141, 166  
Girvan, Carina, 113

Granosik, Grzegorz, 301  
Greenberg, Ronald I., 65  
Grizioti, Marianthi, 113  
Gueorguiev, Ivaylo, 113

## H

Hajduk, Mikulas, 311  
Heuer, Tanja, 141  
Hofmann, Alexander, 15

## I

Ince, Gökhan, 88  
Iocchi, Luca, 274

## J

Jagust, Tomislav, 126  
Jambor, Thomas N., 3  
Jormanainen, Ilkka, 317

## K

Kamaşak, Mustafa, 88  
Kirandziska, Vesna, 193  
Koppensteiner, Gottfried, 238  
Koza, Clemens, 238  
Kryza, Lennart, 26  
Krzic, Ana Sovic, 126

## L

Landhäuser, Mathias, 180  
Lange, Sven, 201  
Lázaro, María T., 274  
Lepuschitz, Wilfried, 238  
Llamas, Luis, 226  
Lüssem, Jens, 166

## M

Mahmoudian, Nina, 154  
Martín, Francisco, 214  
Mayerová, Karolína, 53

**N**

Naya, Martin, 226

**O**

Olaru, Adrian D., 311

**P**

Pilat, Zbigniew, 311

Polishuk, Alex, 263

Prieto, Abraham, 226

Protzel, Peter, 201

**R**

Rezelj, Anže, 250

Richter, Georg, 15

Rubenzer, Sabrina, 15

**S**

Saldien, Jelle, 257

Schiering, Ina, 141

Sersic, Damir, 126

Sharkov, George, 113

Sharkov, Petar, 113

Shirokolobov, Ilya, 38

Skočaj, Danijel, 250

**T**

Ten, Natalia, 38

Todorova, Christina, 113

Toivonen, Tapani, 317

Tukiainen, Markku, 317

**U**

Uhlig, Andreas, 201

**V**

Vagaš, Marek, 311

Valls, Albert, 100

Vandavelde, Cesar, 257

Varbanov, Pavel, 113

Varela, Gervasio, 226

Verner, Igor, 263

Verstockt, Steven, 257

Vervisch, Thomas, 257

Veselovská, Michaela, 53

**W**

Weigelt, Sebastian, 180

Weissig, Peter, 201

Wolff, Martin, 238

wyffels, Francis, 257

**Y**

Yiannoutsou, Nikoleta, 113

**Z**

Ziaeeefard, Saeedeh, 154

Zubrycki, Igor, 301