An Open Approach to Robotic Prototyping for Architectural Design and Construction

Andrea Rossi[1](✉), Arjen Deetman[2], Alexander Stefas[3], Andreas Göbert[1], Carl Eppinger[1], Julian Ochs[1], Oliver Tessmann[4], and Philipp Eversmann[1]

[1] Universität Kassel, Universitätsplatz 9, 34127 Kassel, Germany
rossi@asl.uni-kassel.de
[2] Eindhoven University of Technology, P.O. Box 513, Eindhoven 5600 MB, Netherlands
[3] Kraenk Visuell GbR, Friedrich-Ebert-Platz 17, 64289 Darmstadt, Germany
[4] Technische Universität Darmstadt, El-Lissitzky-Straße 1, 64287 Darmstadt, Germany

Abstract. Emerging from research in computational design and digital fabrication, the use of robot arms in architecture is now making its way in the practice of construction. However, their increasing diffusion has not yet corresponded to the development of shared approaches covering both digital (programming and simulation) and physical (end-effector design, system integration, IO communication) elements of robotic prototyping suited to the unique needs of architectural research. While parallel research streams defined various approaches to robotic programming and simulation, they all either (A) rely on custom combinations of software packages, or (B) are built on top of advanced robotic programming environments requiring a higher skill level in robotics than conventionally available in an architectural context. This paper proposes an alternative open-source toolkit enabling an intuitive approach to the orchestration of various hardware and software components required for robotic fabrication, including robot programming and simulation, end-effector design and actuation, and communication interfaces. The pipeline relies on three components: Robot Components, a plug-in for intuitive robot programming; Funken, a serial protocol toolkit for interactive prototyping with Arduino; and a flexible approach to end-effector design. The paper describes these components and demonstrates their use in a series of case studies, showing how they can be adapted to a variety of project typologies and user skills, while keeping highly complex and specific functionality available as an option, yielding good practices for a more intuitive translation from design to production.

Keywords: Robotic fabrication · Hardware integration · Open-source

1 Introduction and Background

Emerging from research in computational design and digital fabrication, the use of robot arms in architecture is now making its way in construction (Kohler et al., 2014; Willmann et al. 2018). Due to the complexities of architectural fabrication, highly interactive processes, with integration of advanced hardware components are needed (Stefas et al.

2018), blurring the line between human and machine interfaces. To this date, there are still limited developments of shared approaches covering both digital (programming and simulation) and physical (end-effector design, system integration, IO communication) elements of robotic prototyping suited for architectural research. Most approaches limit themselves to simulation, while methods for development and management of hardware components are mostly lacking.

1.1 Software Platforms for Architectural Robotics

While parallel research streams defined approaches to robotic programming and simulation, they either rely on custom combinations of software, often for specific purposes and not open-source, or are built on top of advanced robotic frameworks, which require skills rarely available in an architectural context. For the first approach, various programming and simulation tools for the Grasshopper environment emerged, such as HAL (Schwartz 2013), KukaPRC (Braumann and Brell-Çokcan 2011), Robots (Soler et al 2017), Scorpion (Elashry and Glynn 2014), and Taco (Frank et al. 2016). With differences, all provide utilities for toolpath creation, code generation, and offline kinematic simulation within Grasshopper. Some, such as HAL, evolved into a standalone framework (Gobin et al. 2021). Machina (Garcia del Castilo Lopez 2019) focuses instead on intuitive interaction with robots through real-time programming. With some exceptions, several of these tools are closed-source, and hence they allow only limited adaptations and extension, when an API is available.

The second approach leverages state-of-the-art robotic frameworks, such as ROS (Robot Operative System) (Stanford Artificial Intelligence Laboratory et al. 2018) or V-REP (Rohmer et al. 2013), and integrate them in architectural interfaces. The most notable case is the CompasFab package (Rust et al. 2018) for the COMPAS framework (Mele et al. 2017), a Python framework for research in computational design. CompasFab enables to integrate the above-mentioned environments in CAD packages such as Rhino or Blender. While this simplifies their usage for architectural researchers, it still presents a significant entry barrier, caused by the need of sound knowledge in Python programming and understanding of complex robotics frameworks.

1.2 Hardware Platforms for Architectural Robotics

If the field of robot programming has seen significant development, hardware integration, another major element of robotic fabrication research, has lagged in the architectural domain. In most cases, the integration of sensors and end-effectors is either performed by specialized system integrators, or it is prototyped using custom approaches, leveraging low-cost computing devices such as Arduino microcontrollers (Braumann and Cokcan 2012), through Grasshopper plug-ins such as Firefly (Payne and Johnson 2013). While the first approach yields reliable fabrication processes at the cost of flexibility, the second often results in non-transferrable processes.

An attempt to address this is Robo.Op, a platform aimed at making robotic prototyping more accessible (Bard et al. 2014; Gannon et al. 2016). The toolkit provides a modular hardware platform for prototyping of end-effectors, and a software interface translating between ABB RAPID code and common creative coding tools. The hardware provides power supplies to mount various standard tools and an Arduino microcontroller connected to the robot IOs. While the project is open-source, at the time of writing development seems to have halted.

1.3 An Open Approach to Computational Tools Development

As mentioned, a key characteristics of architectural research in robotic fabrication is the importance of flexibility and adaptability of workflows, combining both low-cost prototyping components and industrial-standards systems. This is akin to what Eric Raymond described as a "bazaar" (Raymond 1999), where several specialized tools need to be integrated and orchestrated. This highlights the need to maintain openness, both in terms of hardware and software, as well as to allow the creation of shared communication interfaces between software and hardware elements (Stefa et al. 2018). This is why the tools and interfaces described here are provided as open-source packages, aiming at creating a flexible "toolkit" for robotic fabrication rather than a centralized closed "tool" (Mackey and Sadeghipour Roudsari 2018). While the reliance on open source tools might raise concerns in a safety-critical domain such as fabrication, it must be noted that its viability is being increasingly accepted in other domains where software safety has similar, if not higher, requirements than architectural robotics (Gary et al. 2011).

2 Methods

Given this context, this paper proposes an intuitive approach to the orchestration of hardware and software components required for robotic fabrication (Fig. 1). This is based on the definition of communication interfaces between human actors and robotic machines, as well as between various elements of a robotic system.

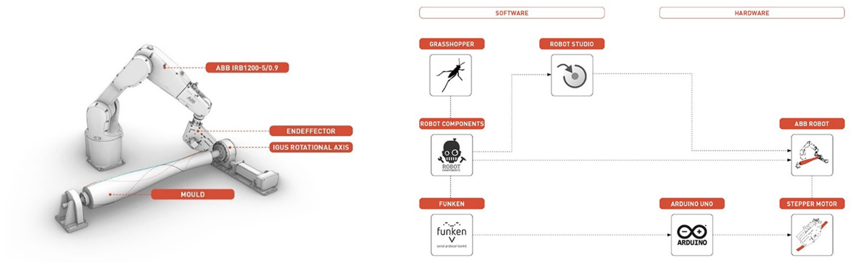


Fig. 1. Example of a process for robotic winding, entirely relying on the proposed pipeline.

Hence, we propose an open-source software and hardware pipeline, relying on three main components:

- Robot Components, an open-source plugin for intuitive robot programming;
- Funken, an open-source toolkit for serial communication;
- an open and modular approach to tool design.

2.1 Robot Components

The first element of the pipeline is Robot Components, an open-source robotic programming and simulation plug-in for Grasshopper (Deetman et al. 2022). This forms the basis to interface CAD geometries in Rhino with ABB robot arms.

Grasshopper Plug-in. The Robot Components plug-in provides an intuitive interface matching the logic of ABB's RAPID code for robotic programming. Given the difficulty of developing generic interfaces for different robot brands, while maintaining access to specific functionality, strictly following the RAPID code means that each RAPID code line is represented by one component. This is beneficial for teaching, since by visually programming with Robot Component students learn the basics of RAPID code without typing code lines. In parallel, to avoid an overwhelmingly complicated interface, Robot Components makes extensive use of casting methods and hidden parameters. This allows beginner users to quickly create robot programs, hiding unnecessary parameters and providing a simple introduction to robot programming, without removing the possibility of adding more advanced functionalities afterwards (Fig. 2).

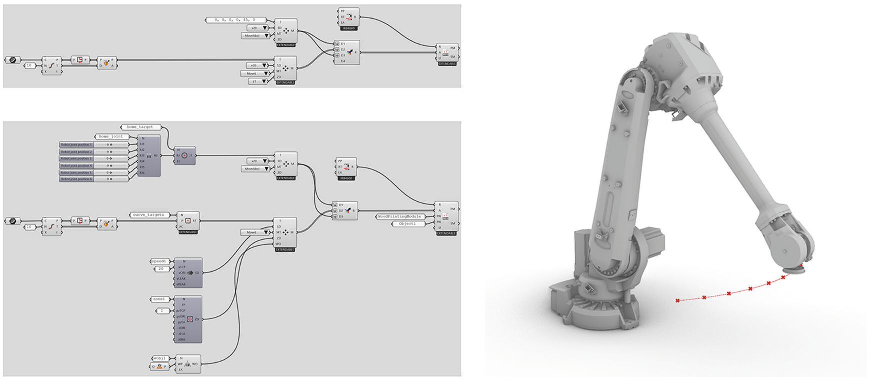


Fig. 2. Comparison between robot programming modes: beginner (top), where most parameters are hidden, and advanced (bottom), where all parameters are declared for more control.

Robot Components provides components for forward and inverse kinematics, allowing a quick visual check on the robot poses without leaving the Grasshopper environment. A controller utility category is available to send and get data from both physical (real robots) and virtual (simulated in ABB RobotStudio software) controllers. It relies on the ABB Controller API, allowing to set and get IO signals in real-time. This, in combination with other proposed tools, allows Grasshopper to become a central communication interface. The robot pose and tool position can also be read inside Grasshopper, simplifying calibration processes. This real-time connection also allows to check toolpaths within Grasshopper by remotely using the advanced kinematic simulation of RobotStudio.

API. For more advanced users and complex processes, an open-source API is available in Grasshopper via IronPython and/or C#. This allows to prototype processes requiring feedback loops that are harder to program relying exclusively on visual programming. These can be used in real-time during fabrication in combination with the controller utilities, but also during design, automatically validating fabrication feasibility through the embedded robot kinematics.

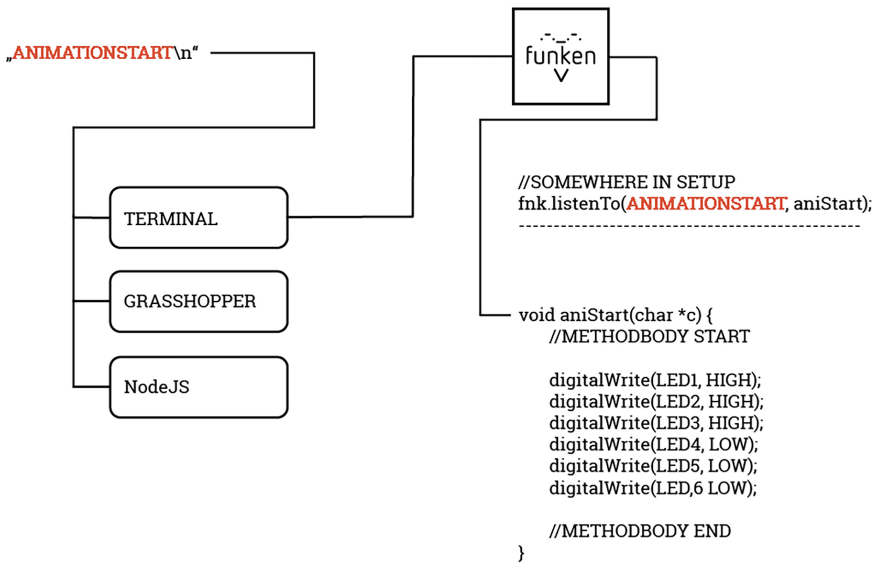
2.2 Funken

As already discussed, robotic programming and simulation is only one component of fabrication processes. In order to tackle the second necessary element, the control of sensors and end-effectors, we propose Funken (Stefas et al. 2018; Stefas 2020), an open-source toolkit for interactive prototyping with the Arduino framework (Mellis et al. 2007). Funken allows the definition of shared interfaces between hardware and software elements of a robotic system, simplifying the execution of complex tasks through a keyword-based callback method. It consists of an Arduino library and interfaces for different frameworks, such as Grasshopper, Python, Processing, and NodeJS.

Arduino Library. The Funken library enables the implementation of even-based programming (Faison 2006) on Arduino-compatible microcontrollers. This links complex functionality defined via Arduino code with simple keywords, that can be called from any serial communication-enabled software or hardware component to execute such functionality, without need to interact directly with the code (Fig. 3). Parameters can be associated with keywords, allowing to further customize the behavior of functions. This allows to create human-readable serial protocols for communication between microcontrollers and other hardware and software elements.

In order to make the usage of Funken accessible without need for programming experience, the library contains basic implementations covering common applications, ranging from reading and writing data, to the control of different motors. For more experienced users, Funken can be easily extended with custom functionality without editing the core library, by creating functions in Arduino code and linking them to keywords, making them accessible to any connected software or hardware.

Grasshopper Plug-in. Funken can connect to a variety of software frameworks, since it relies on serial communication, a basic and widely available protocol for hardware-software interfaces. In the context of robotic fabrication, the most relevant interface is the Grasshopper environment. The open-source GhFunken plug-in allows to remote-control any microcontroller running Funken, by either mirroring basic Arduino functionality, or by connecting to custom-defined Funken functions (Rossi 2020). It also allows to easily connect multiple microcontrollers to the same file, modularizing the control of an end-effector. The plug-in relies on the PySerial library (Liechti 2016), an open-source Python implementation for serial communication.



```
//SOMEWHERE IN SETUP
fnk.listenTo(ANIMATIONSTART, aniStart);
-----------------------------------------------

void aniStart(char *c) {
    //METHODBODY START

    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, HIGH);
    digitalWrite(LED4, LOW);
    digitalWrite(LED5, LOW);
    digitalWrite(LED,6 LOW);

    //METHODBODY END
}
```

Fig. 3. Overview of Funken keyword-based call back method, linking the keyword “ANIMATIONSTART” to the execution of the code in the bottom left.

The plug-in enables users focusing on robotic programming to control custom end-effectors by simply knowing the defined interfaces, without needing complete knowledge of their internal functioning. It also allows to control the level of exposure of Arduino code, providing a platform for education in physical computing, gradually increasing the exposure of students to levels of programming complexity.

Virtual Prototyping of Hardware Interfaces. The combination of Funken and the controller utilities for Robot Components enables to quickly define and test communication interfaces between hardware components without need for hardware wiring. Indeed, by monitoring the robot IOs through Robot Components, and using them to trigger keyword messages through the GhFunken plug-in, it becomes possible to create virtual connections between specific IOs and Funken functions, remotely triggering different end-effectors behaviors (Fig. 4). While such communication determines relatively high levels of latency, potentially incompatible with fully automated production processes, it proved sufficient for prototyping research and testing of hardware systems.

Additionally, the combination of real-time IOs monitoring and Funken functions creates a parallel control system for end-effectors. Using routing functions, provided with Funken, it is possible to control specific end-effectors behaviors both through a wired connection to the robot IO system, as well as through a Grasshopper interface. This parallel control model is particularly valuable when prototyping processes, as it allows fully automated control of the process via wired IOs, but still allows over-writing of such behaviors from the Grasshopper interface.

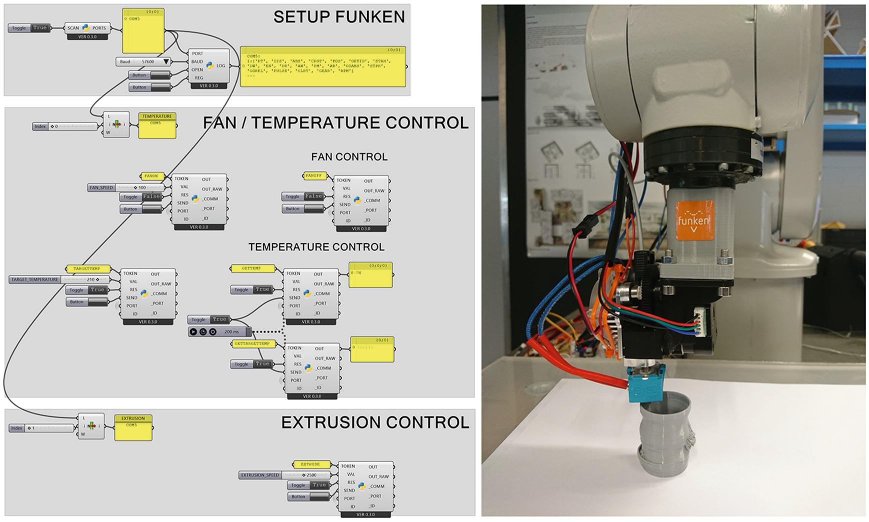


Fig. 4. Control dashboard for a robotic 3d printing process, showing the possibility of overwriting process parameters by changing Grasshopper sliders during execution on the robot.

2.3 Fast Physical Prototyping of End-Effectors

Beyond their control, end-effector prototypes can be easily designed in CAD and rapidly produced through low-cost FDM 3d printing. Using modular principles, such tools can integrate a variety of functions, such as cutting, heating, pumping, and distance measuring for real-time adjustments. Such functions can be controlled via Funken, and in parallel fitted with relay switches to work within the robotic system and outputs. For the bespoken example, low cost and vastly available electronics, such as, stepper motors, temperature sensors, distance sensors etc. can be combined into one prototype tool adapted for a specific application (Fig. 5).

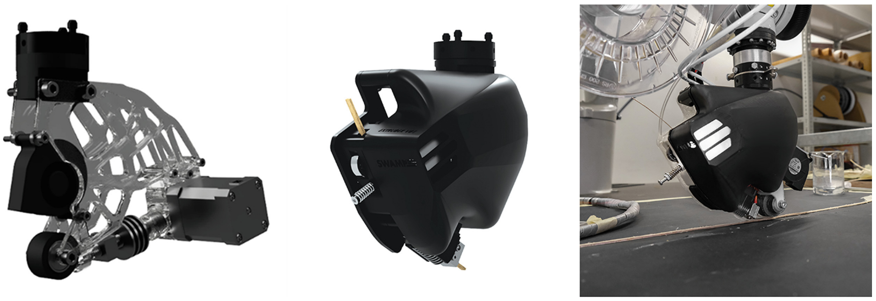


Fig. 5. Example of a 3d printed end effector for extrusion of continuous timber filament, from left to right: internal 3d structure and components, complete model, built extruder.

3 Results: Case Study Processes

The proposed pipeline and its components have been applied for various projects, highlighting their capabilities and application scenarios, as well as the flexibility provided by different soft- and hardware combinations.

3.1 Continuous Timber Filament Winding

As a first case study, the proposed pipeline has been applied for robotic winding of timber veneers (Göbert et al. 2022). The process has several parameters, which need to be synchronized: (a) extrusion speed of the filament, (b) axis rotation velocity and (c) robot Tool Center Point (TCP) speed. In addition, various I/O signals for the veneer extrusion, cutting, and automated adhesive application needed to be controlled during the process (Fig. 6).

The pipeline enabled fully-automated generation of winding processes for different setups. It also allowed fast shift from an initial prototyping setup for small profiles, featuring an Arduino-controlled external axis, to a full-scale production of architectural building components, integrating a robot-controlled external axis (Fig. 7). The entire code in Robot Components required minimal changes in the speed settings and IO-signals. This not only enabled a quick realization of larger demonstrators, but also allowed the use for both research and education, giving students with low programming knowledge the opportunity to generate their own winded elements.

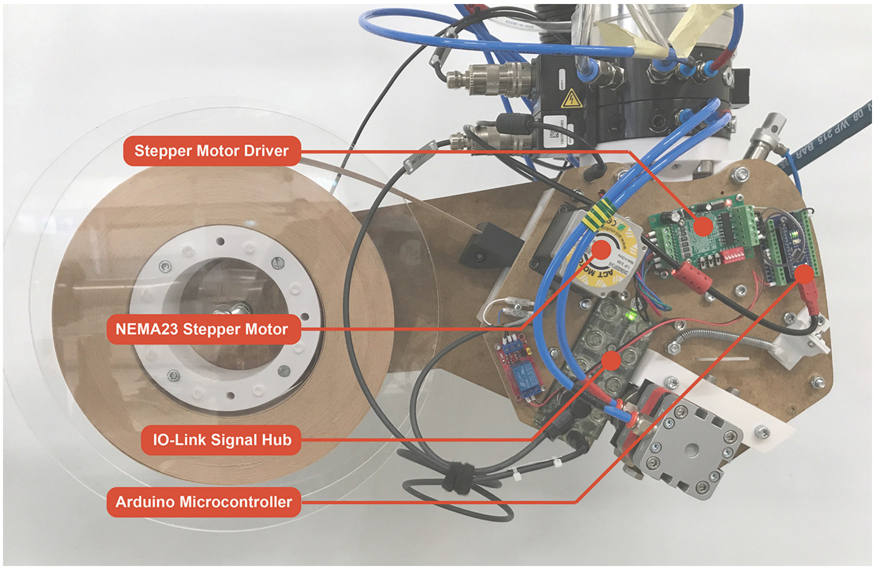


Fig. 6. Extruder for robotic winding with the various hardware components which are orchestrated in the proposed pipeline: a stepper driver and its relative motor, an Arduino microcontroller, and an IO-Link hub interfacing the signals with the robot ProfiNET network.

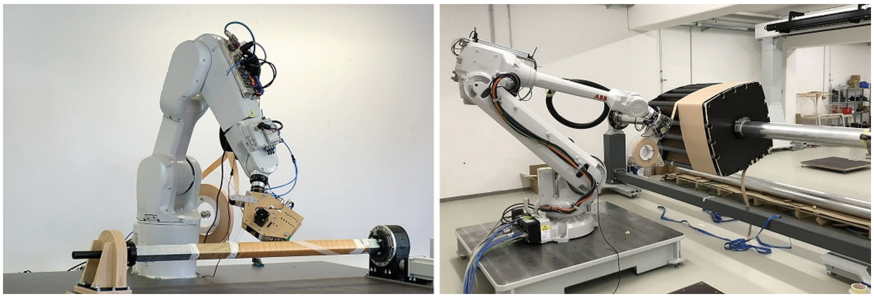


Fig. 7. Process scaling from initial prototyping studies (left) to full-scale production (right) of robotic winded components. As same control and hardware strategies were used, upscaling could be efficiently realized with our toolsets.

3.2 Automated Dowel Placement for Robotic Assembly

In a more focused and specific application, the proposed methods were used to integrate an automated dowel-placement tool for robotic assembly (Fig. 8). The tool recreates the manual placement process of wooden dowels, by using a pneumatic cylinder as hammer. Through rapid prototyping, we implemented an 80-dowels magazine, which required only limited refills during placement. A series of motors were synced to guide the dowels to the hammering position, and check for consistency in all steps. To guarantee a fluid fabrication process, the tool was equipped with several sensors, which, using Funken and Robot Components, were made accessible to the operator. Hence, the tool could sense the need for refilling, send a signal to the robot to move to a safe position and wait for the operator's confirmation that the feed has been refilled, to continue its original path. This diminished the time necessary to insert dowels by a factor of five.

3.3 Human-Robot Interfaces for Interactive Installations

Shifting the context to the development of more intuitive human-machine interfaces, the proposed tools could also be applied for flexible prototyping explorations. This was demonstrated in a series of installations, partly described in (Betti et al. 2018), where users were allowed to interact with a small ABB IRB120 robot arm cutting foam bricks. Visitors could control the shape of the cuts with their hands, tracked through a Leap Motion tracker. The resulting positions were converted into cutting motion using Robot Components. The hardware system included a custom-build rotating table, controlled via Funken, allowing to select the brick face to cut (Fig. 9). This enabled to quickly prototype the workflow, manufacture the custom table, and integrate its electronics in the fabrication process, without need of wired connections between the table and the robot controller. A similar approach was used in another installation, linking room lighting, controlled via DMX standards, with robot motion through Funken (Belousov et al. 2022; Wibranek 2021).

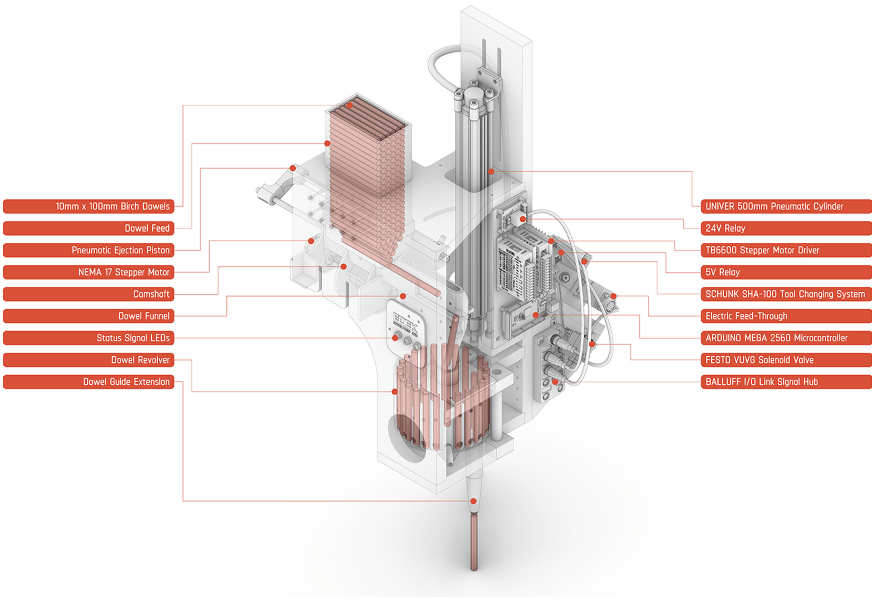

10mm x 100mm Birch Dowels
Dowel Feed
Pneumatic Ejection Piston
NEMA 17 Stepper Motor
Camshaft
Dowel Funnel
Status Signal LEDs
Dowel Revolver
Dowel Guide Extension
UNIVER 500mm Pneumatic Cylinder
24V Relay
TB6600 Stepper Motor Driver
5V Relay
SCHUNK SHA-100 Tool Changing System
Electric Feed-Through
ARDUINO MEGA 2560 Microcontroller
FESTO VUVG Solenoid Valve
BALLUFF I/O Link Signal Hub

Fig. 8. Robotic doweling tool with the different components allowing for full automation of dowels loading, alignment and hammering through a pneumatic piston. A series of motors is used to guide the dowels from the magazine to the insertion position, and sensors track the process ensuring consistency.

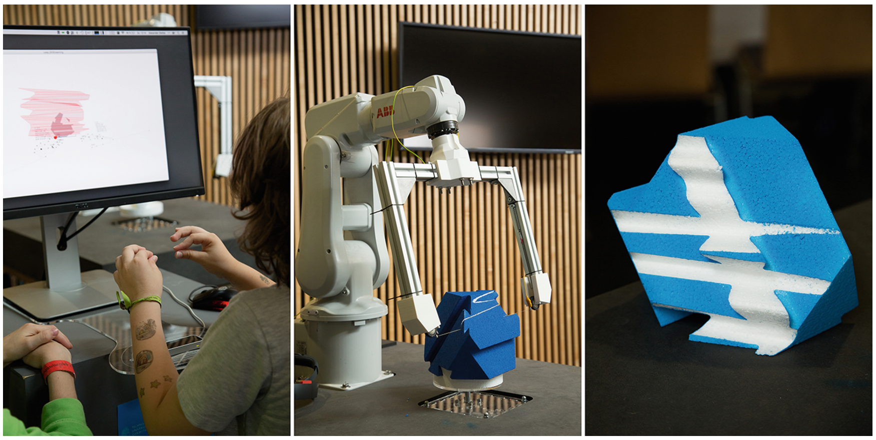


Fig. 9. Human-machine interface for translation of hand movement into robotic cutting instructions, from left to right: user hand tracking interface, robot cutting process, finished brick.

4 Conclusions

Results of case studies show that the workflow is adaptable to various projects and levels of user skills, while keeping complex and specific functionality available as an option.

Our framework can be applied in prototyping studies and interactive installations, as well as for structured fabrication workflows with large robots and a variety of actuators, sensors, and industrial protocols. The scale and variety of the applications demonstrates its suitability for research and prototypical production. The pipeline provides a framework for the definition of interfaces within complex human-machine fabrication processes, where roles are often redefined during research. As the structure of the different software elements relies on a layered structure, where only relevant information is made accessible to users with different levels of expertise, it allows to increase usability in safety-critical domains (Gary et al. 2011). Moreover, as all software components are built on top of common computational design and prototyping frameworks, relying on simple interfacing models, they have been designed maintaining the possibility of extension to other software environments.

At the current stage, the community revolving around the proposed tools is still small, and many users rely only on certain elements and not on the whole pipeline. As having an active community is key for the sustainability of an open-source project, future efforts will be directed towards the expansion of documentation and example files, covering various workflows, aiming at increasing the user-base and adoption of the proposed methods.

In conclusion, our research creates interfaces between the elements of fabrication systems through open-source tools and human-readable interfaces, rather than imposing specific and ultimately rigid workflows,. This yields good practices for a more intuitive translation from design to production and a flexible communication model between human and machinic actors. Overall, our research aims at providing shared models of human-machine interfaces, fostering open research and collaboration in robotic fabrication.

Acknowledgments. Case studies were partly funded by DFG, grant #436451184, and BBSR, grant #10.08.18.7-20.24.

References

Schwartz, T.: HAL. In: Brell-Çokcan, S., Braumann, J. (eds.) Rob | Arch 2012, pp. 92–101. Springer, Vienna (2013). https://doi.org/10.1007/978-3-7091-1465-0_8

Braumann, J., Brell-Çokcan, S.: Parametric robot control: integrated CAD/CAM for architectural design. In: Proceedings of the 31st Annual Conference of the Association for Computer Aided Design in Architecture, pp. 242–251. Banff, Alberta (2011)

Soler, V., Retsin, G., Jimenez Garcia, M.: A generalized approach to non-layered fused filament fabrication. In: Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture, pp. 562–571. Cambridge, MA (2017)

Elashry, K., Glynn, R.: An approach to automated construction using adaptive programing. In: McGee, W., Leon, M.P. (eds.) Robotic Fabrication in Architecture, Art and Design 2014, pp. 51–66. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04663-1_4

Frank, F., Wang, S.-Y., Sheng, Y.-T.: Taco: ABB robot programming for Grasshopper (2016)

Gobin, T., Andraos, S., Schwartz, T., Vriet, R.: HAL robotics framework. In: Proceedings of the International Symposium on Automation and Robotics in Construction, vol. 38, pp. 733–740. IAARC Publications (2021)

Garcia del Castillo Lopez, J.L.: Enactive Robotics: An Action-State Model for Concurrent Machine Control. Ph.D. Dissertation, Harvard University (2019)

Stanford Artificial Intelligence Laboratory et al.: Robotic Operating System. https://www.ros.org. Last accessed 10 June 2022 (2018)

Rohmer, E., Singh, S.P.N., Freese, M.: V-REP: A versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1321–1326 (2013). https://doi.org/10.1109/IROS.2013.6696520

Rust, R. et al.: COMPAS FAB: Robotic fabrication package for the COMPAS Framework (2018). https://doi.org/10.5281/zenodo.3469478

Mele, T., et al.: COMPAS: A framework for computational research in architecture and structures (2017). https://doi.org/10.5281/zenodo.2594510

Braumann, J., Cokcan, S.-B.: Digital and physical tools for industrial robots in architecture: robotic interaction and interfaces. Int. J. Archit. Comput. **10**, 541–554 (2012)

Payne, A.O., Johnson, J.K.: Firefly: interactive prototypes for architectural design. Archit. Des. **83**, 144–147 (2013)

Bard, J. et al.: Seeing is doing: synthetic tools for robotically augmented fabrication in high-skill domains. In: Proceedings of the 34th Annual Conference of the Association for Computer Aided Design in Architecture, pp. 409–416. Los Angeles, California (2014)

Gannon, M., Jacobson-Weaver, Z., Contreras, M.: Robo.Op. https://github.com/peopleplusrobots/robo-op (2016). Last accessed 10 June 2022

Raymond, E.: The cathedral and the bazaar. Knowl. Technol. Policy **12**, 23–49 (1999)

Stefas, A., Rossi, A., Tessmann, O.: Funken - Serial Protocol Toolkit for Interactive Prototyping. In: Computing for a better tomorrow - Proceedings of the 36th eCAADe Conference, vol. 2, pp. 177–186. Lodz, Poland (2018)

Mackey, C., Roudsari, M.S.: The tool(s) versus the toolkit. In: De Rycke, K., Gengnagel, C., Baverel, O., Burry, J., Mueller, C., Nguyen, M.M., Rahm, P., Thomsen, M.R. (eds.) Humanizing Digital Reality, pp. 93–101. Springer Singapore, Singapore (2018). https://doi.org/10.1007/978-981-10-6611-5_9

Gary, K., et al.: Agile methods for open source safety-critical software. Softw. Pract. Exp. **41**, 945–962 (2011)

Deetman, A. et al.: Robot Components: Intuitive Robot Programming for ABB Robots inside of Rhinoceros Grasshopper (2022). https://doi.org/10.5281/zenodo.5773814

Stefas, A.: Funken - Serial Protocol Toolkit. https://github.com/astefas/Funken (2020). Last accessed 10 June 2022

Mellis, D., Banzi, M., Cuartielles, D., Igoe, T.: Arduino: an open electronic prototyping platform. In Proc. Chi **2007**, 1–11 (2007)

Faison, T.: Event-Based Programming. Springer (2006)

Rossi, A.: GhFunken. https://github.com/ar0551/GhFunken (2020). Last accessed 10 June 2022

Liechti, C.: PySerial documentation. https://pyserial.readthedocs.io/en/latest/ (2016). Last accessed 10 June 2022

Göbert, A., Deetman, A., Rossi, A., Weyhe, O., Eversmann, P.: 3DWoodWind: robotic winding processes for material-efficient lightweight veneer components. Const. Robot. **6**(1), 39–55 (2022). https://doi.org/10.1007/s41693-022-00067-2

Betti, G., Aziz, S., Rossi, A., Tessmann, O.: Communication landscapes. In: Willmann, J., Block, P., Hutter, M., Byrne, K., Schork, T. (eds.) ROBARCH 2018, pp. 74–84. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-92294-2_6

Belousov, B., et al.: Robotic architectural assembly with tactile skills: Simulation and optimization. Autom. Constr. **133**, 104006 (2022)

Wibranek, B.: Robotic Digital Reassembly: Towards physical editing of dry joined architectural aggregations. Ph.D. Dissertation, Technische Universität Darmstadt (2021). https://doi.org/10.26083/tuprints-00018578