# Discussion on Problems and Solutions in Hardware Implementation of Algorithms for a Car-type Autonomous Vehicle


**Rahee Walambe, Shreyas Nikte, Vrunda Joshi, Abhishek Ambike, Nimish Pitke and Mukund Ghole**

**Abstract** The self-driving cars will bring fundamental change in the transportation industry. Many hazardous situations like land-mine detection, war zones, nuclear decommissioning highlights the need of autonavigation in open terrain. In the last few decades, due to the increasing interest in mobile robots, a number of autonavigation algorithms have been developed. The autonomous vehicles are used extensively in different domains from passenger car to the hazardous applications. These autonomous cars extensively use mathematical calculations and machine intelligence. In order for the car-type vehicle to manoeuver smoothly in a given workspace, accurate planning (motion and path) algorithms are essential. As part of the research, our group has developed the nonholonomic motion planning algorithms for the car-type vehicle based on differential flatness approach. In the previous work, the hardware realization of these algorithms is presented. This paper discusses the hardware implementation issues that we have faced during this work. Hardware implementation comes with various inherent challenges, such as manufacturing error in the car hardware components, physical limitation of the component, limited processing power of low-power onboard computer, accuracy of data from sensors in diverse conditions.

**Keywords** Nonholonomic motion planning · Car type robot · Hardware implementation · Open loop

## 1 Introduction

Latombe [1] has discussed the motion planning and path planning methods in detail. Although these methods are suitable for specific tasks with holonomic motion plan-

R. Walambe (✉)
Symbiosis Institute of Technology, Pune, India
e-mail: rahee.walambe@sitpune.edu.in

S. Nikte · V. Joshi · A. Ambike · N. Pitke · M. Ghole
PVG's College of Engineering and Technology, Pune, India



A. J. Kulkarni et al. (eds.), *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology*, Advances in Intelligent Systems and Computing 828, https://doi.org/10.1007/978-981-13-1610-4_13

ning problems, in order to manoeuver nonholonomic entities like car-type vehicle, motion planning algorithms developed by Fliess [2] and Agarwal [3] can be effectively used. In [4], differential flatness-based nonholonomic motion planner for a car-type mobile robot is presented. Walambe et al. [5] shows how the spline-based optimization can be incorporated in this motion planner to avoid the singularities and generate the shortest path which also satisfy the nonholonomic and curvature constraints. In [6], implementation of sensor data acquisition system on hardware for feedback and real-time localisation of a car-type vehicle is shown. The hardware implementation of motion planning algorithm and sensor data acquisition was an important step in evaluating our autonavigation algorithm. Hardware implementation, though important, is hindered by various issues such as manufacturing error in the car hardware components, physical limitation of the component, limited processing power of low-power onboard computer, accuracy of data from sensors in diverse conditions. These hardware selection criteria and implementation issues are discussed in the following article. In this paper, the work done towards the development and implementation of the open-loop and closed-loop motion planning of a 1:10 scale down vehicle platform is presented. This includes selection of appropriate car model and interfacing of sensors and other computer systems with this hardware which are essential for implementation of autonavigation algorithm. As main purpose of 1:10 scale down model of car is to reduce cost of the hardware implementation for research, cost-effectiveness and adequate performance of the hardware model are the key requirements while designing the hardware. This paper also touches various hardware implementation issues that our team had to address during this work. The contribution of this paper is mainly in two areas: (1) to list and discuss the various issues that were faced in the hardware implementation and (2) to identify and implement the solution to these issues. Authors hope that this paper will benefit the people working in the hardware realization domain. The kinematic model of hardware model is discussed in Sect. 2. In Sect. 3, description of hardware model is provided. Also selection criteria for the particular car model is discussed. Section 4 covers the selection of onboard computer and issues faced with initial selection of computer. In Sect. 5, issue related to PWM generation technique used in Raspberry Pi boards is discussed. The solution for the issue is stated and results are presented. Section 6 deals with issues related with the default Electronic Speed Controller (ESC) and implemented solution for the problem. Section 7 covers issues related with default NiMH battery and problem related with the use of IC LM7805, a common IC used for 5 V supply. Section 8 discusses the sensor data acquisition process and corresponding challenges.

## 2 System Modelling

Let *A* be a car-type mobile robot capable of both forward and reverse motion. It is modelled as a rigid rectangular body moving in a planar (two-dimensional) workspace (refer Fig. 1). The rear wheels are fixed and aligned with the car while

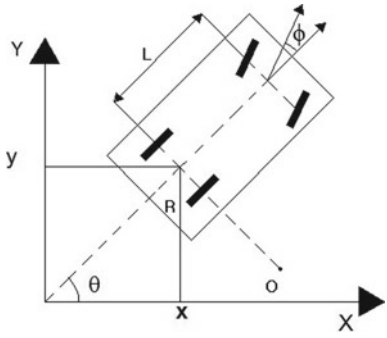


**Fig. 1** Kinematic model of a car

the front wheels are steerable; i.e., they are allowed to spin about the vertical axis with constraints on the maximum angle [7]. The wheelbase of $A$ (distance between the front and rear axles) is denoted by l. The car is assumed to be rear wheel driven, and hence R, the reference point of the car is the midpoint of the rear axle.

A configuration of $A$ is defined by the posture $(x, y, \phi, \theta)$ where $(x, y)$ are the coordinates of R in the Cartesian plane. The orientation of the vehicle is $\theta$, which is the angle between the x-axis and the main axis of the vehicle $(-\pi \leq \theta \leq \pi)$. The steering of angle of the vehicle, which is the angle made by the steerable front wheels with respect to the main axis of the car, is denoted by $\phi$.

The kinematic model of the car where $(x, y, \phi, \theta)$ are the state variables is shown below

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{\tan\phi}{l} \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2 \tag{1}$$

where $u_1$ is driving velocity and $u_2$ is steering velocity also known as control inputs. Development of motion planning algorithm is discussed in [1].The motion planning algorithm was based on this nonholonomic model of a car and employs the differential flatness-based approach for planning the motion and is shown in Algorithm 1.

## 3 Hardware Model of the Car

After development of the algorithms in simulations, the next step was to port these algorithms on the car prototype. The major criteria for selection of the car model was to get scaled down model of a real car which is functionally as similar as possible. Hence, with the above requirement RTR Sprint 2 Drift model manufactured by HPI racing Ltd is selected. The RTR Sprint 2 Drift chassis is an exactly 1:10 scaled down model of the actual sports car Chevrolet Camaro and is a four wheel drive (4WD). The hardware model shown in Fig. 2 is interfaced with Raspberry Pi 3B which is main

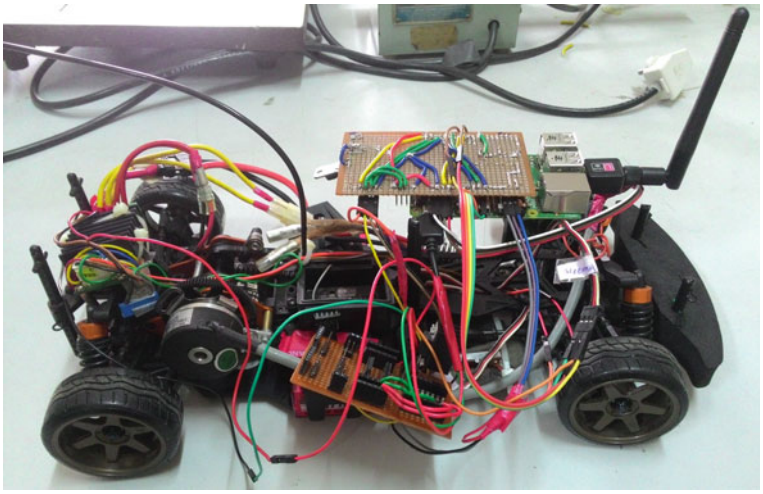

**Fig. 2** Hardware car model

**Algorithm 1** Autonavigation in static obstacle environment

1: Input $\gamma(0)$, $\gamma(1)$ and obstacles
2: Determine obstacle free path $\gamma(p)$ from geometric planner
3: **if** $\gamma(p)$ exists **then**
4: continue
5: **else**
6: exit
7: **end if**
8: $p_0 \leftarrow 0$
9: $p_f \leftarrow 1$
10: **while** $p_0 \neq 1$ **do**
11: **Generate** path between $\gamma(p_0)$ to $\gamma(p_f)$ considering nonholonomic constraints
12: **Optimize** path
13: **if** Path is optimized **then**
14: Check path for collisions with obstacles
15: **if** Collision detected **then**
16: **jump** to line:22
17: **else**
18: $p_0 \leftarrow p_f$
19: $\gamma(p_0) \leftarrow \gamma(p_f)$
20: **end if**
21: **else**
22: $p_f^{new} \leftarrow (p_0 + p_f)/2$
23: $\gamma(p_f) \leftarrow \gamma(p_f^{new})$
24: **end if**
25: **end while**

computer of the system. Arduino Uno is employed for interfacing the motors with the system. Hercules NR-MDR-003 Motor driver is used to control the power given to the driving motor. Optical encoder is used for measuring the distance travelled by wheels, potentiometer is used to measure steering angle, and magnetometer is used for measurement of orientation angle. The system is powered by 5000 mAh 7.2 V Lithium ion battery. A 5 V, 3 A buck converter is used to power devices which require 5 V DC as input.

## 4 Selection of Onboard Computer

For autonavigation, it is necessary to have a powerful computer platform which can be mounted on the car for implementation of complicated algorithms. At the start of each test case, mathematical calculations are carried out for optimum motion planning using algorithm. Hence, adequate computation power is required. Due to powerful processor and large community support, Raspberry Pi 3B and Beaglebone Black are major SBC platforms. The Raspberry Pi and Beaglebone black are roughly about the size of a credit card and run variety of Linux distributions. One of the biggest downsides to the Beaglebone is that it only has 1 USB port compared to the RPis 4. Hence for Beaglebone Black, creating multiple SPI interface with the USB ports requires additional USB hub. Also, Pi has large number of add-on boards due to the larger community support. Beaglebone Black has more powerful processor and more GPIO pins, although the price of the Beaglebone Black is roughly twice the price of Raspberry Pi 3B, In addition, it is equipped with a Wi-fi port, 40 GPIO pins, I2C, SPI and HDMI interfaces. MCP3008, an external 8-bit ADC chip is required for reading variable voltage of potentiometer. It has a HDMI port, a display interface (DSI) and a camera interface (CSI) for future expansion of the project. Hence, Raspberry Pi 3B was the best choice for the application.

## 5 PWM Generation on Raspberry Pi 3B

Raspberry Pi 3B has one hardware PWM pin and two software PWM pins. The software PWM frequency for raspberry Pi is derived from the pulse time. Essentially, the frequency is a function of the range and this pulse time. The total period will be (range * pulse time in S), so a pulse time of 100 and a range of 100 gives a period of 100 * 100 = 10,000 S which is a frequency of 100 Hz. It is possible to get a higher frequency by lowering the pulse time; however, CPU usage will skyrocket as WiringPi uses a hard loop to time periods under 100 S—this is because the Linux timer calls are not accurate and have an overhead. Another way to increase the frequency is to reduce the range, however that reduces the overall resolution of PWM signal to an unacceptable level (refer Fig. 3).


**Fig. 3** PWM output for 14.1% dutycycle (left) and 14.7% dutycycle (right) on Raspberry Pi 3B

This characteristics of software PWM function of Raspberry Pi creates problem for the driving motor PWM as well as servo PWM signals. As the vehicle is tuned for lower speeds (0–12% duty cycle), resolution of PWM signal should be high enough to get full control of the above range of duty cycle. This will result in lowering of the PWM frequency as discussed above. Also, modern servo position is not defined by PWM, but only by the width of the pulse. Hence for the frequencies above 100 Hz, very low resolution software PWM is generated on Raspberry Pi. In other words, smaller set of on times are available for the servo control. Conversely, if the frequency of the PWM signal is lowered, we can get higher resolution of PWM and hence more number of pulses in the operating range of the servo motor. But it is advised not to run servo motor on such low frequencies. Hence, additional PWM generating device for one of the motors was required.

Arduino Uno development board was chosen for controlling servo motor due to above-mentioned reasons. Raspberry Pi and Arduino Uno are interfaced by SPI using USB ports. The pair acts as master–slave configuration as Raspberry Pi sends the servo angle command to the Arduino. Arduino then maps the angle into the corresponding pulse width and sends it to the servo motor. Hence, PWM issue was resolved by adapting the above method.

## 6 Replacing the Radio Control

The vehicle platform is designed to be a racing car. For the development, implementation and obtaining the proof of concept of the algorithms, the radio control was replaced with the controller board. The onboard Electronic Speed Control (ESC) mechanism was provided for control of the steering motor and the driving motor. In order to avoid interference of receiver and transmitter signals with that of other vehicles, frequency switching technique is employed in electronic speed control. Due to this, the system response of ESC for driving control is time variant, i.e. shifting of dead band. This car is originally designed to run at a speed of about 3300 rpm

(maximum speed). It was difficult for us to implement and observe the behaviour of the algorithms and car at such high speeds. Therefore, a null signal is passed to the driving channel of ESC (primary switching for steering control) and chopper drive is used to eliminate ESC for control of driving velocity at lower speed range of the prototype.

The peak current of driving motor is 11 A for full loading and voltage supplied by power supply (Li-Po battery) is 7.4–8.4 V and minimum safe operation is 6 V, and therefore, voltage requirement is greater than 6 V.

Logic levels of PWM from Raspberry Pi is 3.3 V. The driver with these specifications is provided by NEX robotics named as Hercules-NR-MDR-003. This driver is rated as 6–36 V, 15 A with a peak current of 30A and can be operated up to 10 KHz PWM. It can be interfaced with 3.3–5 V logic levels. The driver has terminal block as power connector and 7-pin 2510 type connector for logic connections.

Hence, Hercules-NR-MDR-003 is selected as a driver by bypassing the default ESC of the driving motor.

## 7 Power Supply for the Hardware

These issues are twofold, i.e. issues with the default NiMH battery and issues with using LM7805 IC for 5 V power supply.

The RTR Sprint 2 Drift model came with 7.2 V 2000 mAH NiMH battery. The capacity of battery was not enough for daily testing of 1–2 h. Also, NiMH battery has higher self-discharging rate. The number is around 5% on the first week after the charge and about 50% on the first month. Charging time battery is also long. Hence, it was not suitable for the purpose. On the other hand, Li ion battery is more reliable, smaller and can be recharged faster. Hence, 7.4 V, 5000 mAH Lithium ion battery is chosen to power circuitry.

Although most of the electronic circuitry requires 5 V power supply, this voltage is required to supply all the sensors like magnetometer, encoder, and potentiometer, steering servo motor, Raspberry Pi and Arduino Uno. So, initially in order to provide required 5 V supply, LM 7805 IC was used. But due to higher power demands from the new added circuitry, 7805 IC was not reliable power solution for long term. It was observed that, the voltage across 7805 IC was dropping below acceptable value due to higher current requirements. Hence, above-mentioned equipments got affected at each voltage sag, viz. Raspberry Pi often used to reset, also inaccurate sensor data was acquired, especially from the sensors which are the function of input voltage, such as potentiometer.

Hence after recalculating the power requirement for the electronic components, LM7805 circuit was replaced by 5 V 3 A DC-DC buck converter with over-temperature and short-circuit protection. These power circuit boards are cheap and readily available in the market.

## 8 Sensor Data Acquisition

According to the open loop results in [5] it is seen that, there are deviations from the open loop simulated trajectory and its hardware implementation. This is primarily due to the fact that the kinematic model does not take into consideration dynamic state variables. Hence, we are currently working on the development of feedback algorithm based on the real-time sensor data. The sensor data acquisition posed a challenging problem in selection, interfacing and placement of the sensors. This has been discussed in details in [6]. Magnetometer, optical encoder and potentiometer are used for sensor data acquisition.

## 9 Conclusion

This paper presented a viable hardware solution for testing various nonholonomic motion planning algorithms. By addressing the above-mentioned issues and providing the solution for these problems, we were able to create robust and reliable vehicle platform with required sensors and onboard computers needed for autonavigation purpose. In future, advanced sensors like camera and LIDAR can be included on this platform for creating real-time mapping system for AI-based or mathematical algorithm development.

**Acknowledgements** This project is funded under the WOS-A scheme (SR/ET/WOS-34/2013-14) of Department of Science and Technology, Government of India.

## References

1. Latombe JC (1991) Robot motion planning. Kluwer Academic Publishers, Boston
2. Rouchon P, Fliess M, Levine J, Martin P (2011) Flatness and motion planning: the car with n trailers. Automatica
3. Ramirez HS, Agarwal S (2004) Differentially flat systems. CRC Press, Boca Raton
4. Agarwal N, Walambe R, Joshi V, Rao A (2015) Integration of grid based path planning with a differential flatness based motion planner in a non-holonomic car-type mobile robot. Inst Eng Annu Tech J 39 (November)
5. Walambe R, Agarwal N, Kale S, Joshi V (2016) Optimal trajectory generation for car-type mobile robot using spline interpolation. In: Proceedings of 4th IFAC conference on advances in control and optimization of dynamical systems ACODS 2016, Tiruchirappalli, India, 1–5 Feb 2016
6. Walambe R, Dhotre A, Joshi V, Deshpande S (2016) Data acquisition and hardware setup development for implementation of feedback control and obstacle detection for car type robot. In: International conference on advancements on robotics and automations(ICAARS), Coimbatore, India, June 2016
7. Laumond J-P, Jacobs PE, Taix M, Murray RM (1994) A motion planner for nonholonomic mobile robots. IEEE Trans Robot Autom 10:577–592