Online-Journals.org

iJOE International Journal of Online and Biomedical Engineering

iJOE | eISSN: 2626-8493 | Vol. 20 No. 1 (2024) | OPEN ACCESS

https://doi.org/10.3991/ijoe.v20i01.46573

PAPER

A ROS-Based Open Tool for Controlling an Educational Mobile Robot

José Varela-Aldás[1](✉), Guillermo Palacios-Navarro[2]

[1]Universidad Indoamérica, Ambato, Ecuador

[2]University of Zaragoza, Teruel, Spain

josevarela@uti.edu.ec

ABSTRACT

Commercial educational robots provide an accessible entry point into the world of robotics. However, their programming is often limited to specific platforms, which can make it challenging to acquire the skills necessary for industry and research. In this study, we introduce an open-access tool developed using C++ and Arduino IDE that enables us to manage a commercial mobile robot through the Robot Operating System (ROS) middleware. This provides programmers with the ability to work in a powerful programming environment, such as Python. The robot used is the CrowBot BOLT, a kit based on ESP32 that enables wireless communication and includes various peripherals for application development. The mobile robot topics include robot velocities, RGB LEDs, a buzzer, a programmable button, and proximity, light, and line sensors. The proposal is assessed using two controllers: one for proximity and the other for tracking angular light. Both controllers are developed using Visual Studio Code. The experimental results demonstrated the proper functioning of the tool. Additionally, the response time was evaluated, and it was found that optimal performance is achieved at a frequency of 10 Hz. In summary, this proposal provides an accessible option for students and developers seeking to gain skills in robotics using ROS. The project's repository is located at https://github.com/joseVarelaAldas/ROS-Crowbot.

KEYWORDS

Robot Operating System (ROS), open tool, mobile robot, PID control, education tool

1 INTRODUCTION

1.1 Background and motivation

In recent years, there has been a rise in the integration of education and modern technologies, with the introduction of various concepts and applications. For example, elements such as digital educational and serious games [1, 2], as well as mobile applications and e-learning systems, have been incorporated into the educational environment to offer interactive and adaptable learning methods [3–5]. In this

Varela-Aldás, J., Palacios-Navarro, G. (2024). A ROS-Based Open Tool for Controlling an Educational Mobile Robot. *International Journal of Online and Biomedical Engineering (iJOE)*, 20(1), pp. 23–39. https://doi.org/10.3991/ijoe.v20i01.46573

Article submitted 2023-09-17. Revision uploaded 2023-11-04. Final acceptance 2023-11-07.

context, robotics has entered the educational field in various ways, including learning about robotics and the teaching and assistance provided by robots. Specifically, it can be observed that social robots in the educational environment play different roles, such as teachers, tutors, learning partners, or even learners instructed by students [6]. Although there are still challenges to be addressed, including technical issues related to robot intelligence and interaction analysis as well as logistical challenges linked to the integration of social robots into educational practices, there have been numerous studies that have demonstrated positive results [6, 7].

A recent review offers insights into the latest developments in educational social robotics [9]. The scope of social robotics in the field of education has been demonstrated by the wide range of applications and the diverse research conducted in each of these areas. These elements allow us to see the potential of incorporating robots into education to create a positive impact. Although much of the reviewed work has been conducted in the form of studies and not as practical implementations regularly used in educational settings, it is conceivable to consider social robots as active agents in the educational process. With challenges to overcome and advances to be achieved, robots can contribute to improving students' learning outcomes and be accepted and integrated into their contexts of use [10].

The integration of robotics into the field of education can be approached from two distinct perspectives. On one hand, there is the perspective related to the programming of devices or software, while on the other hand, there is the perspective associated with the assembly and operation of devices or hardware [10]. This distinction plays a crucial role when designing our classroom activities, similar to any technology, which must be tailored to the needs of the students [11].

Although most educational applications of robotics focus on programming or topics directly related to technology, they can be extended to a wide range of disciplines, including mathematics, languages, and art [11, 12]. On the other hand, robotics education in higher education primarily relies on simulators. Students can fulfill their educational requirements by working in realistic conditions, utilizing advanced programming languages, and using standardized middleware such as the Robot Operating System (ROS) [14]. However, simulations alone are not sufficient to prepare students for modern industry. In addition, the high costs associated with industrial robots make them difficult to access. For this reason, there is a need for low-cost and open-access alternatives that include professional robot programming features. The study's contributions include the proposal of an accessible robotic tool incorporating ROS, the design of the necessary nodes for controlling a commercial educational robot, the acquisition and control of the mobile robot's velocities from a Linux terminal, and the evaluation of control algorithms related to the mobile robot's actions.

This paper is organized as follows: Section 2 presents the materials and methods, which include the characteristics of the educational robot, the equations of motion, the programming, and the control algorithm used to evaluate the proposal. In Section 3, we present the obtained results, including the evaluation of action control, robot data acquisition, and the control algorithm. This is followed by a discussion in Section 4, followed by a conclusion in Section 5.

1.2 Related works

In related literature, there are few studies that integrate ROS into educational robots or low-cost robots designed to enhance learning using this widely accepted middleware. The found works are presented below.

Robot operating system is a set of tools, libraries, and conventions designed to simplify the process of developing complex and advanced robotic systems. This is how it was used to create educational robotics tools. Robotics Academy offers a collection of exercises that cover recent applications of robots, such as autonomous cars, drones, and vacuum cleaners. It has been successfully utilized in various engineering courses [15]. Uplat, a virtual robotics laboratory managed through the web, offers a user-friendly platform. These tutorials can be started from anywhere without the need to deal with complex installation procedures [16]. Another web-based laboratory that enables remote control of robots is the Robotic Programming Network (RPN) initiative. It is designed to work seamlessly with any ROS-based robot or simulator, providing an excellent alternative for educational and remote training purposes [17].

Clearly, ROS has two main applications: simulating and controlling real robots. In simulation, there are interesting proposals in the literature, with the combination of Gazebo and ROS being widely popular. In reference. [18] They utilize the high-performance physics Open Dynamics Engine (ODE) and the sensor features available in Gazebo for prototyping and creating high-fidelity simulations, encompassing various outdoor environments and conditions. [19] provides a detailed tutorial on creating a new robot model in the Gazebo simulator. The process involves constructing the model, configuring the physics, incorporating sensors, integrating ROS-based control, and developing a ROS project with the model.

On the other hand, there are proposals that involve actual robots. For example, a system for obstacle avoidance based on ROS was developed using an existing mobile robot with differential wheels controlled by a Beaglebone Black [20]. EUROPA is an educational robot designed for high school students' robotics education. It utilizes ROS and is based on Python [21]. Robotont is an open-source omnidirectional mobile robot platform with physical hardware and a digital twin. This robot has been successfully utilized for university instruction, professional development, and online courses on ROS and robotics [22]. PlatypOU is a mobile robot designed for laboratory practices and demonstrations, capable of being remotely controlled through an electromyography device [23]. MiniRos is a compact, structured mobile robot designed for outdoor educational and research purposes. It is equipped with various types of sensors, including depth cameras and 2D lidar scanners [24]. Finally, during the last pandemic, the COVID-Bot was developed—an open-source robotic platform designed to sanitize single-story environments. The robot is built on a differential robotic base, an RGB-D camera, a tracking camera, three UV-C lamps, and an embedded computer running ROS-based control software [25]. No proposals were found that generally allow the incorporation of ROS into commercial educational robots. Thus, this work proposes a tool based on open-access ROS to control the educational robot CrowBot BOLT.

2 MATERIALS AND METHODS

The aim of this study is to offer students and developers an affordable, open-access tool that enables practical learning with ROS. Figure 1 depicts the general diagram of the proposal, illustrating the need for a system-on-chip (SoC) in the mobile robot to integrate key features of current technologies, such as access to sensors and actuators. This robot communicates with ROS through a local program that creates the Publisher and Subscriber nodes, and the applications are developed in a programming environment on a computer using ROS as middleware.

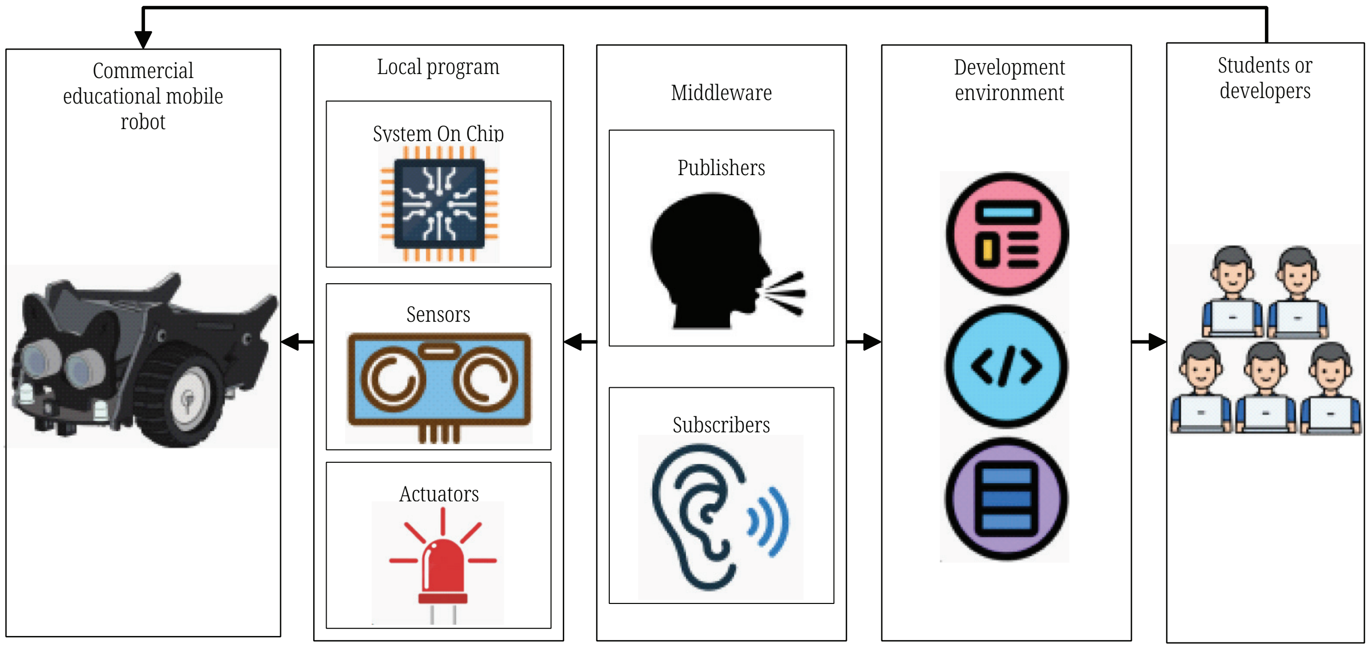


Fig. 1. General diagram of the proposal

2.1 Characteristics of the educational mobile robot

On the market, there are various options for educational mobile robots; however, few are based on the ESP32 microcontroller, a low-cost SoC that includes WiFi and Bluetooth communication. The CrowBot BOLT Programming Kit is an example that fully utilizes the capabilities of the ESP32. The price is approximately \$50 USD, not including the joystick. Figure 2 displays the pertinent components of this kit. Among its inputs, it has a programmable button, an ultrasonic proximity sensor, two light sensors, and two color sensors for line detection. On the other hand, as outputs, it has two DC motors, four RGB LEDs, and a buzzer [26].

2.2 Equations of motion of the mobile robot

The CrowBot BOLT is a differential-wheeled mobile robot equipped with ball-caster wheels and two independent motors, enabling it to produce linear velocity u and angular velocity ω around its own axis [27], as depicted in Figure 3. These velocities are affected by the wheel diameter and the distance between them. The ω_1 and ω_2, the equations of motion (1) and (2) are derived to control the overall movements of the robot based on the angular velocities of the wheels. In this manner, the mobile robot will only receive the necessary speeds as inputs and execute its movements. While it is not possible to directly control the mobile robot in terms of the angular velocity of the wheels (measured in radians per second), it is possible to transform the values obtained by multiplying them by an adaptation constant, K.

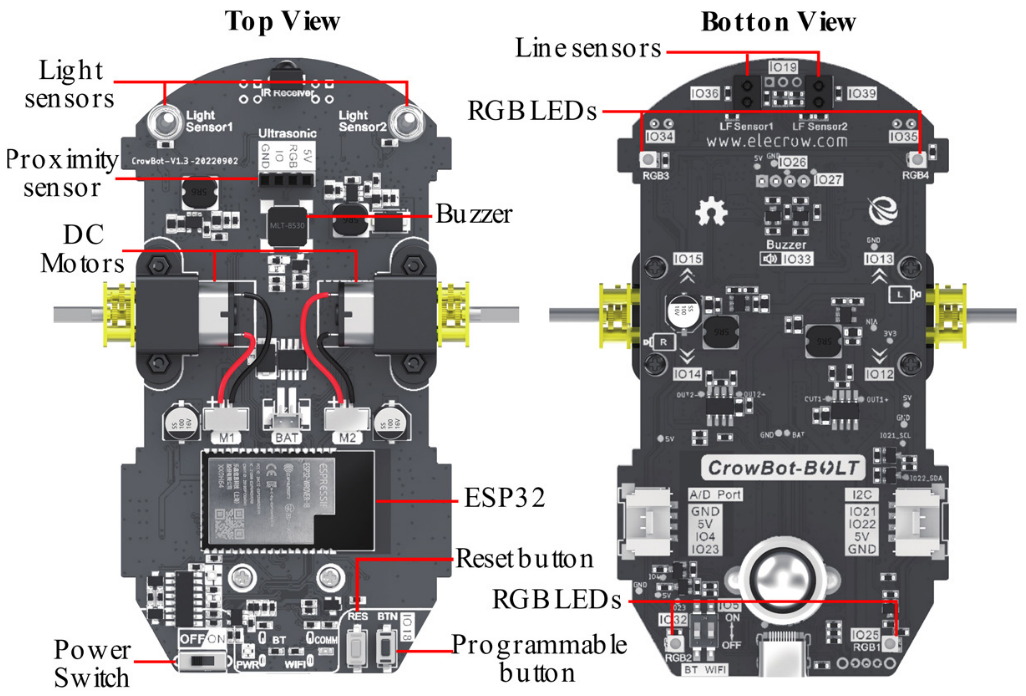


Fig. 2. Relevant components of the CrowBot BOLT educational robot

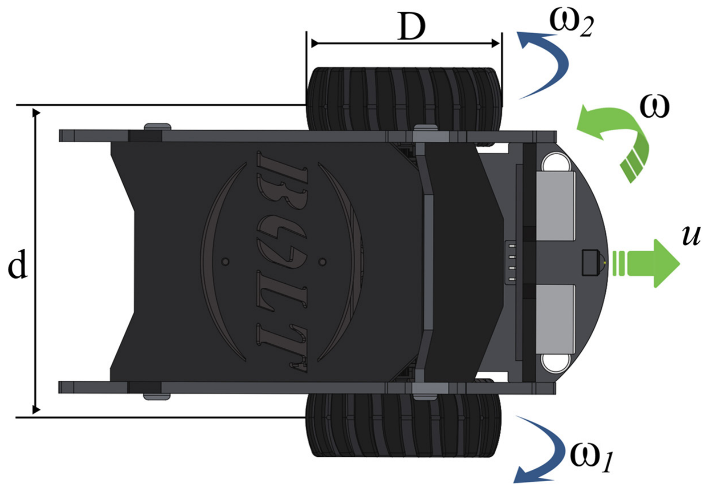


Fig. 3. Movement parameters of the differential wheeled mobile robot

$$\omega_1 = \frac{2u + d\omega}{D} \tag{1}$$

$$\omega_2 = \frac{2u - d\omega}{D} \tag{2}$$

2.3 Mobile robot programming

This study utilizes the Arduino Integrated Development Environment (IDE) along with ROS libraries and C++ code to develop the required algorithms. The code can be accessed in the project repository. The primary program processes are depicted through flowcharts (see Figure 4). When the program starts, all the required variables are configured. The data types for ROS messages are specified in Table 1. The device then establishes a connection to the WiFi network, and the functions for publishing and subscribing are initialized.

On the other hand, the connection to ROS is established only when the 'roscore' node is running and the 'rosserial' bridge over TCP is active in Linux. Once inside the main loop, the interaction between both ends occurs: the subscriber receives the data from the sensors published by the robot, and the publisher sends the data to the node to execute the control functions on the robot, such as speed control, RGB LEDs, and buzzer activation. This interaction is only possible if both nodes are active.

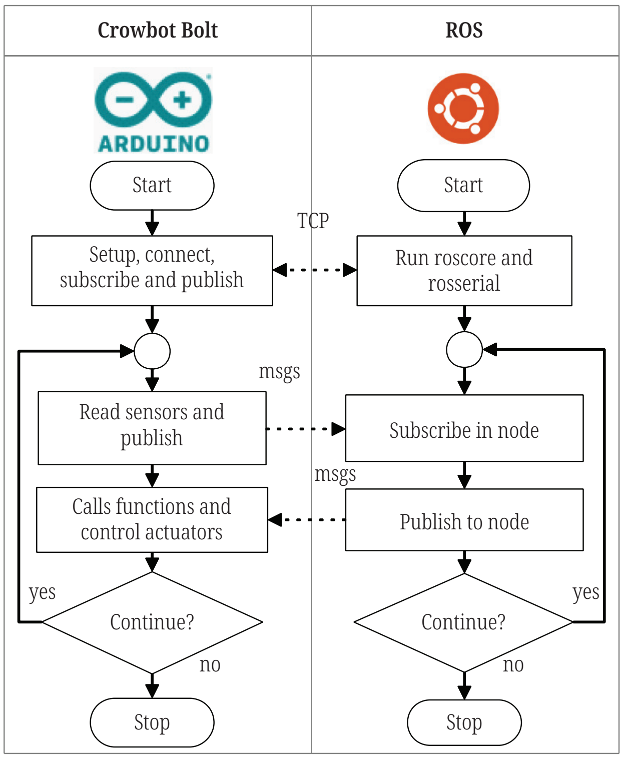


Fig. 4. Flowcharts of the processes in the mobile robot program

Table 1. Data types for ROS messages used in the mobile robot program

Input	Datatype	Output	Datatype
Distance	std_msgs/Int32	Velocities	geometry_msgs/Twist
Light	std_msgs/Int32MultiArray	RGB LEDs	std_msgs/Int32MultiArray
Color	std_msgs/Int32MultiArray	Buzzer	std_msgs/Bool
Button	std_msgs/Bool		

2.4 Architecture of ROS

In this study, Ubuntu 20.04 and ROS Noetic were utilized, with Python as the programming language, using Visual Studio Code. The ROS architecture is based on nodes, and in our case, the mobile robot itself is not considered a node. However, a dummy node is created using the 'rosserial' bridge. In this way, a serial node is established to manage all the topics of the robot, as depicted in Figure 5, which is accessed through the user interface of the 'rqt_graph' plugin. To achieve this, it is necessary to execute the program 'serial_node.py' in TCP mode.

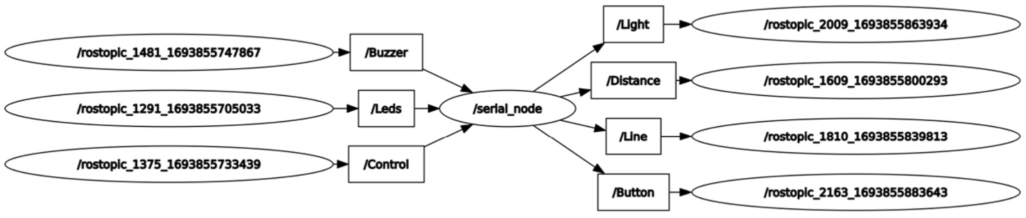
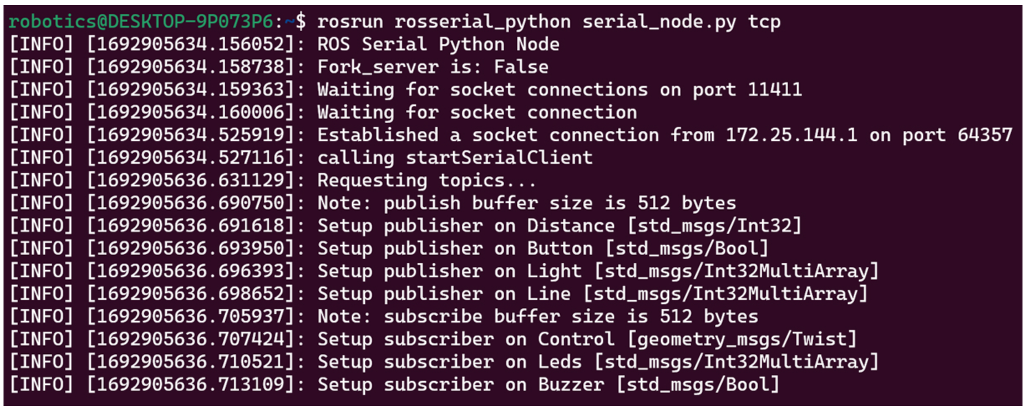

/rostopic_1481_1693855747867
/rostopic_1291_1693855705033
/rostopic_1375_1693855733439
/Buzzer
/Leds
/Control
/serial_node
/Light
/Distance
/Line
/Button
/rostopic_2009_1693855863934
/rostopic_1609_1693855800293
/rostopic_1810_1693855839813
/rostopic_2163_1693855883643

Fig. 5. Node diagram with the topics of the mobile robot

Figure 6 depicts an Ubuntu terminal running 'rosserial,' confirming the presence of four publishers and three subscribers on the mobile robot.



```
robotics@DESKTOP-9P073P6:~$ rosrun rosserial_python serial_node.py tcp
[INFO] [1692905634.156052]: ROS Serial Python Node
[INFO] [1692905634.158738]: Fork_server is: False
[INFO] [1692905634.159363]: Waiting for socket connections on port 11411
[INFO] [1692905634.160006]: Waiting for socket connection
[INFO] [1692905634.525919]: Established a socket connection from 172.25.144.1 on port 64357
[INFO] [1692905634.527116]: calling startSerialClient
[INFO] [1692905636.631129]: Requesting topics...
[INFO] [1692905636.690750]: Note: publish buffer size is 512 bytes
[INFO] [1692905636.691618]: Setup publisher on Distance [std_msgs/Int32]
[INFO] [1692905636.693950]: Setup publisher on Button [std_msgs/Bool]
[INFO] [1692905636.696393]: Setup publisher on Light [std_msgs/Int32MultiArray]
[INFO] [1692905636.698652]: Setup publisher on Line [std_msgs/Int32MultiArray]
[INFO] [1692905636.705937]: Note: subscribe buffer size is 512 bytes
[INFO] [1692905636.707424]: Setup subscriber on Control [geometry_msgs/Twist]
[INFO] [1692905636.710521]: Setup subscriber on Leds [std_msgs/Int32MultiArray]
[INFO] [1692905636.713109]: Setup subscriber on Buzzer [std_msgs/Bool]
```

Fig. 6. Publishers and subscribers topics of the mobile robot

2.5 Control applications

To assess the functionality of this tool, two control applications are suggested: one involving linear speed and the other involving the angular speed of the mobile robot. The first application is to control the distance between the mobile robot and a frontal obstacle, as illustrated in Figure 7a. To achieve this, the position error, denoted as $e(t)$, is calculated using equation (3). In this equation, 'dd' represents the desired distance and $d_r(t)$ is the actual position of the robot. To address this issue, a proportional, integral, and derivative (PID) controller defined by equation (4) is proposed. In this controller, K_p is the proportional constant, K_i is the integration constant, T_i is the integration time, K_d is the derivative constant, T_d is the derivative time, T is the period, and $u_c(t)$ are the linear control actions.

$$e(t) = d_d - d_r(t) \tag{3}$$

$$u_c(t) = K_p e(t) + \frac{K_i}{T_i}\int_0^T e(t)dt + K_d T_d \frac{de(t)}{dt} \tag{4}$$

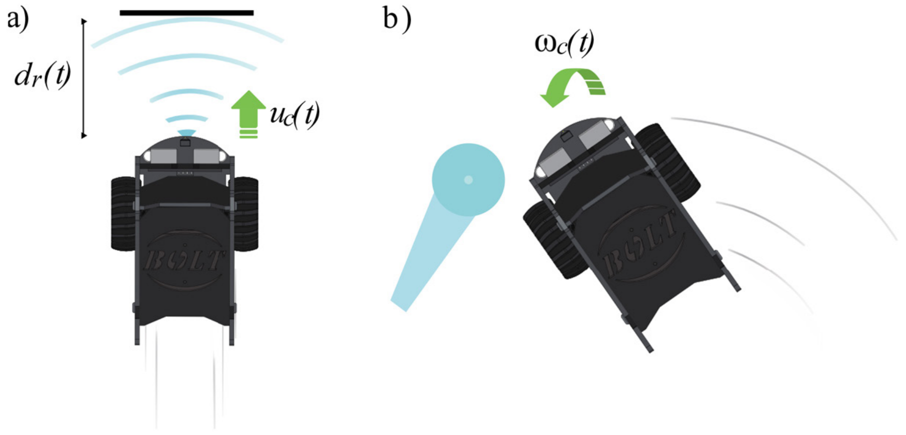


Fig. 7. Illustration graph of the control applications for the mobile robot

The second application involves controlling the angular tracking of light, as illustrated in Figure 7b. In this application, the goal is for the mobile robot to maintain alignment with a lighting source. To achieve this, a proportional controller defined by equation (5) is proposed, in which K_ω is the proportional constant. $L_d(t)$ and $L_i(t)$ represents the variations in illumination measured by the right and left sensors, respectively. Angular control actions are denoted as $w_c(t)$.

$$\omega_c(t) = K_\omega(L_d(t) - L_i(t)) \tag{5}$$

3 RESULTS

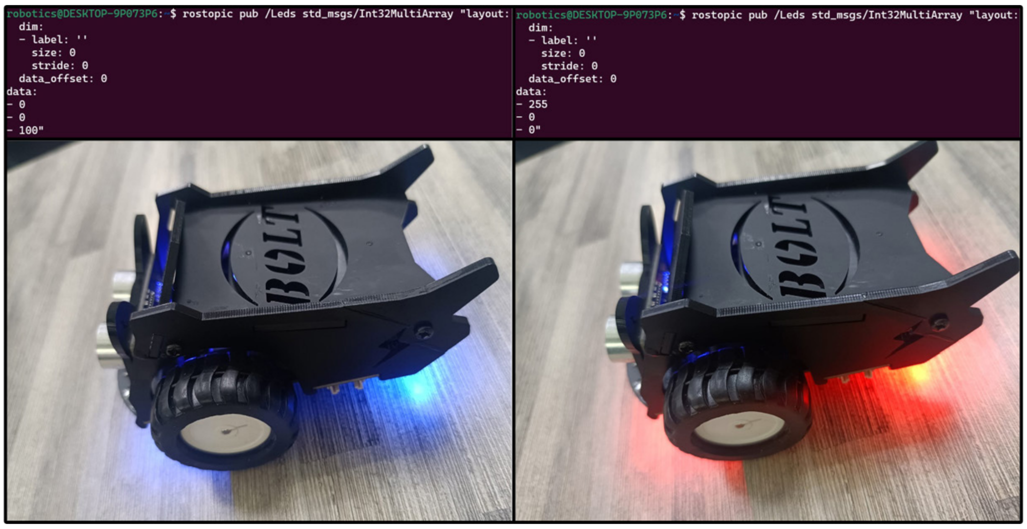
To obtain the results, we begin by loading the main program onto the microcontroller of the mobile robot. Next, preliminary tests are conducted from the Ubuntu terminal. Finally, Python programs are developed for data acquisition and evaluation of the proposed control applications. The following results are described based on the data presented in Table 2. For these performance tests, a communication frequency of 10 Hz was utilized, resulting in a sampling time of 0.1 seconds.

3.1 Action control

During the control tests from the terminal, we evaluate the publication of colors in the RGB LEDs, the robot velocities, and the activation of the buzzer. To achieve this, the 'rostopic pub' function is used, followed by the topic name, message type, and the corresponding data format. Figure 8 displays the control tests of the RGB LEDs, along with the terminal code used and the response of the mobile robot. The color of the LEDs is determined by specifying three values corresponding to the RGB color composition.

Table 2. Control parameters of the mobile robot for performance tests

Parameter	Value	Parameter	Value
K	40	K_i	0.4
d	0.08 m	K_d	0.4
D	0.04 m	d_d	0.2 m
K_p	0.4	K_ω	2



```
robotics@DESKTOP-9P073P6:~$ rostopic pub /Leds std_msgs/Int32MultiArray "layout:
  dim:
  - label: ''
    size: 0
    stride: 0
  data_offset: 0
data:
- 0
- 0
- 100"
```

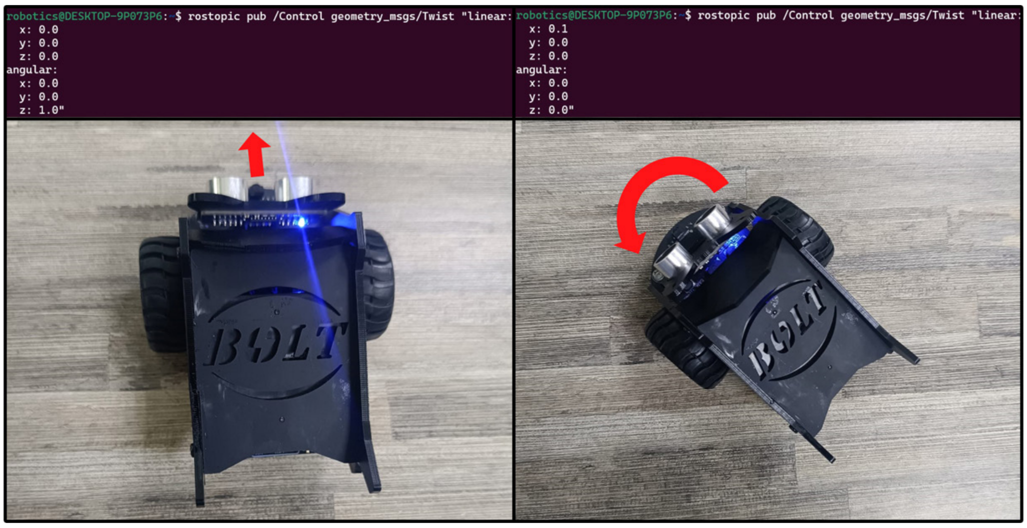
```
robotics@DESKTOP-9P073P6:~$ rostopic pub /Leds std_msgs/Int32MultiArray "layout:
  dim:
  - label: ''
    size: 0
    stride: 0
  data_offset: 0
data:
- 255
- 0
- 0"
```

Fig. 8. RGB LEDs control tests using ROS commands from terminal

Figure 9 displays velocity control tests of the mobile robot, featuring images of linear speed and angular speed, respectively. Each action includes the code used in the terminal for ROS functions. The value of *K*, which is used to adjust the actual response of the speeds, is determined through empirical tests conducted at full battery load, with the aim of achieving the optimal response. The value is presented in Table 2.

3.2 Data acquisition

During data acquisition tests, Python programs are executed to re-collect data over a specified period. These programs are available in the public repository of this project. Figure 10 illustrates the states of the mobile robot's programmable button, showing when the button is pressed over a 15-second period.



```
robotics@DESKTOP-9P073P6:~$ rostopic pub /Control geometry_msgs/Twist "linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 1.0"
```

```
robotics@DESKTOP-9P073P6:~$ rostopic pub /Control geometry_msgs/Twist "linear:
  x: 0.1
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0"
```

Fig. 9. Movement control tests using ROS commands from terminal

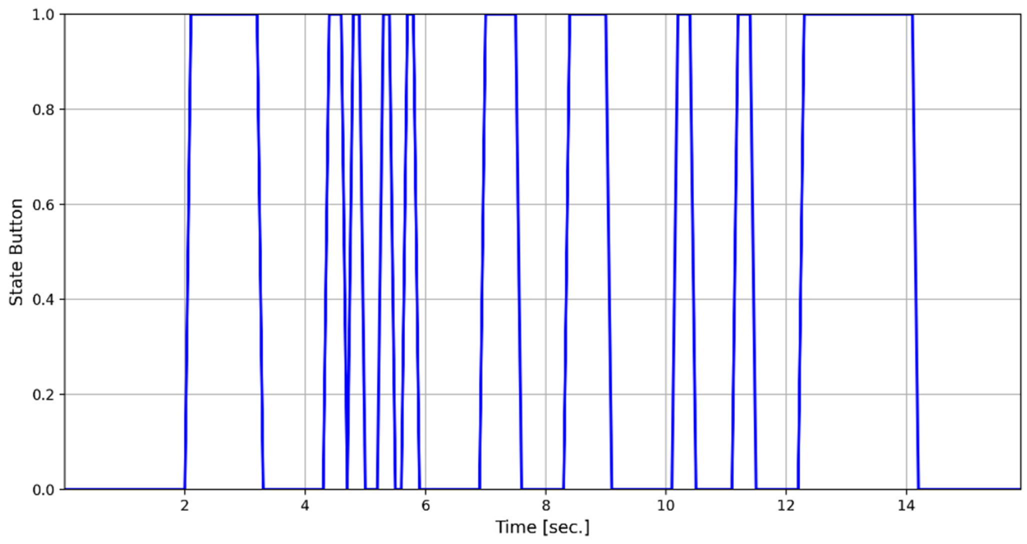


Fig. 10. States of the mobile robot's programmable button

Figure 11 illustrates the collection of proximity data using the ultrasonic sensor, which detects the distance to obstacles over a 30-second period. These proximity variations were generated manually by moving the mobile robot. The measurement range spans from a minimum value of 3 cm to a maximum of 102 cm, although the maximum measurement distance is 200 cm, making it an acceptable range for the development of robotic applications.

Figure 12 depicts the collection of lighting data using two light sensors, which detected variations in light at two points on the robot over a 48-second period. These lighting variations were artificially created using a flashlight. The values range from 0 to 4095 with no specific units, enabling the user to adjust measurements as needed. In the figure, L1 corresponds to the right sensor, and L2 corresponds to the left sensor.

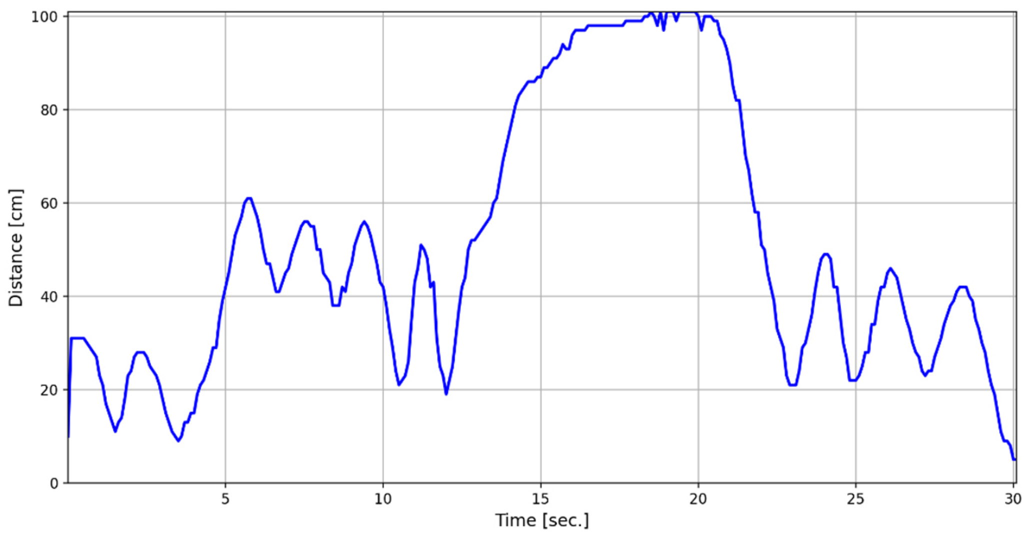


Fig. 11. Proximity data from the mobile robot's ultrasonic sensor

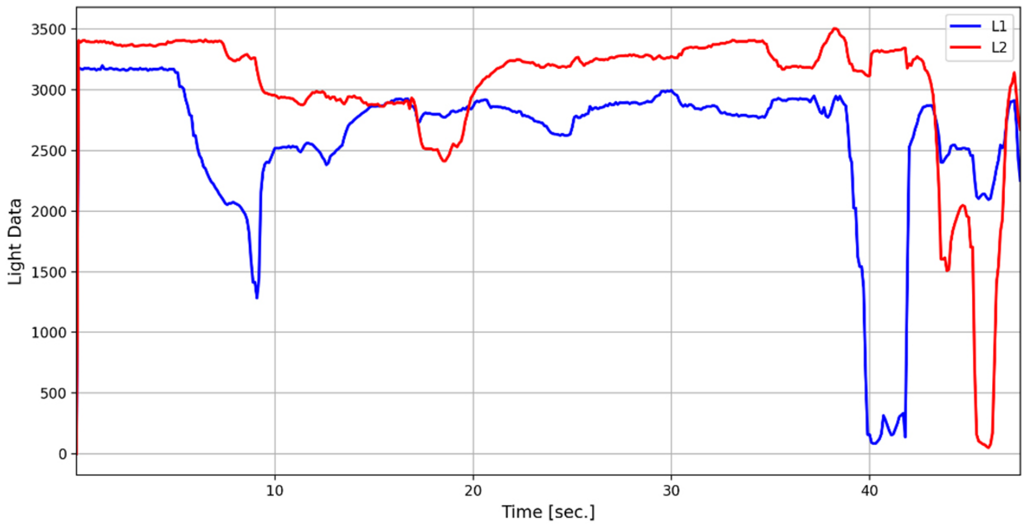


Fig. 12. Illumination data from the light sensors on the mobile robot

Figure 13 depicts the collection of color data using line sensors, which detect color variations at both ends of the robot over a 40-second period. Values range from 0 to 4095 with no specific units, allowing the user to adjust measurements as needed. In the figure, 'S1' corresponds to the right sensor and 'S2' corresponds to the left sensor.

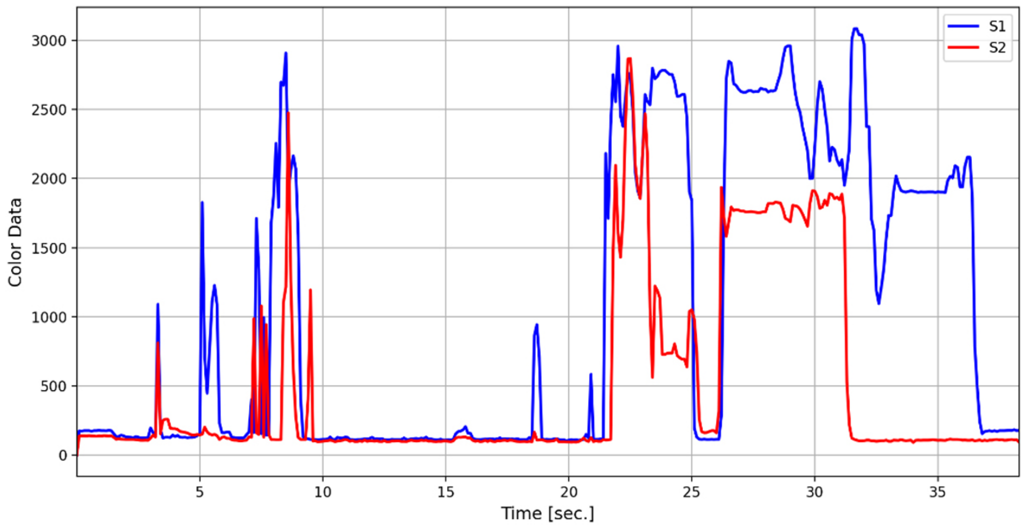


Fig. 13. Color data from line sensors on the mobile robot

3.3 Distance control

The control applications were evaluated using the parameters listed in Table 2, and the codes for these programs are accessible in the public repository of this project. The PID constants were adjusted through empirical methods until satisfactory controller performance was achieved. Figure 14 illustrates the proximity errors, while Figure 15 displays the linear velocities as control actions during the execution of the PID controller. These results demonstrate effective proximity control, correcting errors each time they occur. The reader can observe the operation of this controller in the following public video: https://www.youtube.com/watch?v=5j4wBaW1kaM.

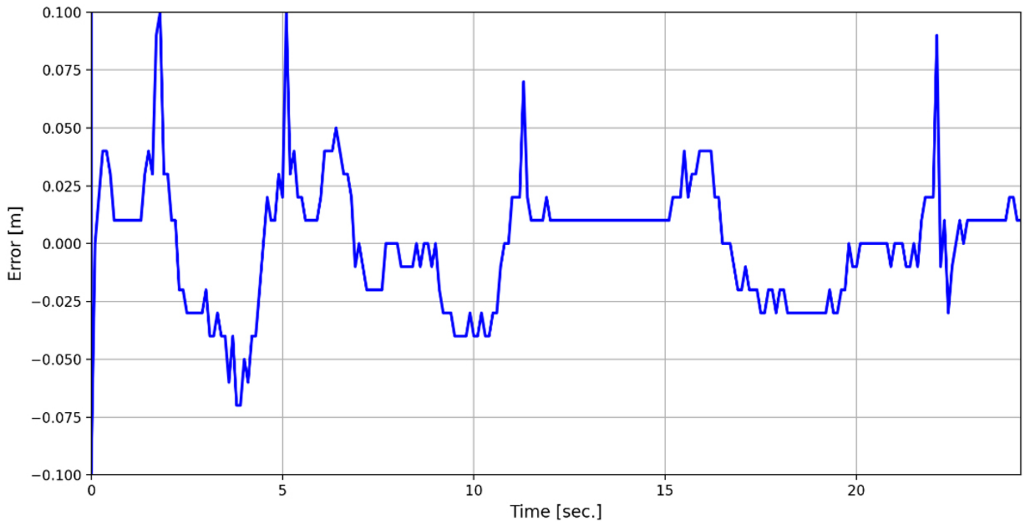


Fig. 14. Proximity errors in PID controller execution

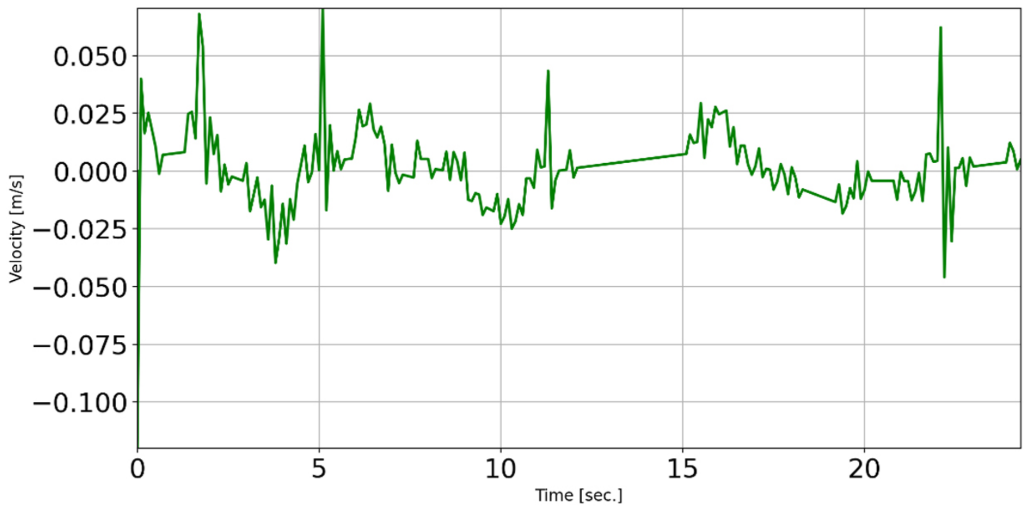


Fig. 15. Control actions in the execution of proximity PID controller

3.4 Angular light tracking

The proportional constant of the light angular tracking controller was adjusted empirically until satisfactory tracking was achieved, and this value is specified in Table 2. Figure 16 displays the tracking errors, while Figure 17 illustrates the angular velocities as control actions during the execution of the controller. The results demonstrate sufficient monitoring, with errors being corrected promptly each time they occur. The reader can observe the operation of the angular light tracker in the following public video: https://www.youtube.com/watch?v=xRhDtNmLW4Y.

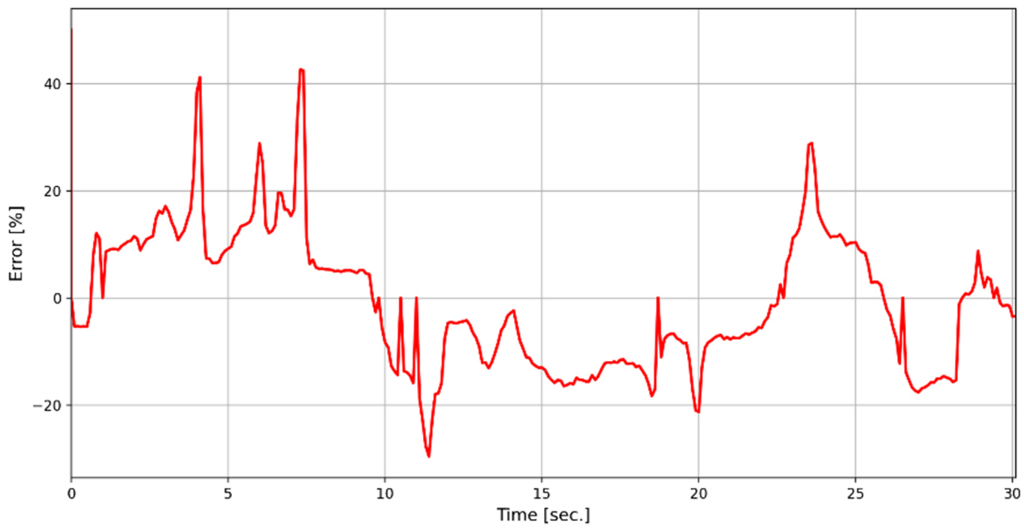


Fig. 16. Light tracking errors in proportional controller execution

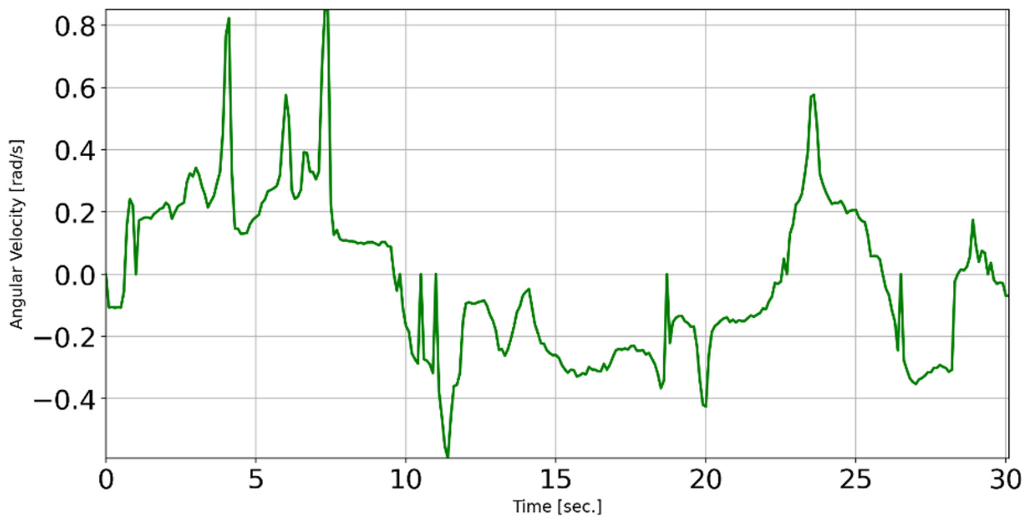


Fig. 17. Control actions in light angular tracking controller execution

3.5 Response times

In addition to the performance tests, the execution times at various frequencies have been analyzed. Figure 18 depicts the response times at a frequency of 10 Hz, with a consistent sampling time of 0.1 seconds. On the other hand, Figure 19 illustrates the response times at a frequency of 20 Hz, showing more significant variations compared to the previous frequency, which results in a sampling time of 0.05 seconds. This is because the robot starts to encounter difficulties in responding to a communication speed of 20 Hz. However, a frequency of 10 Hz is considered sufficient for the development of educational robotic applications.

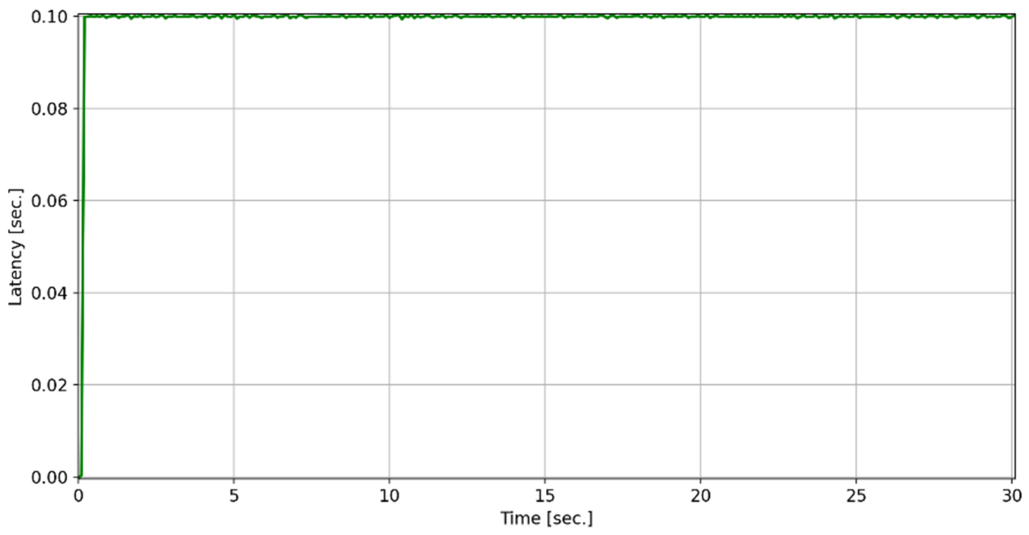


Fig. 18. Communication response times at a frequency of 10 Hz

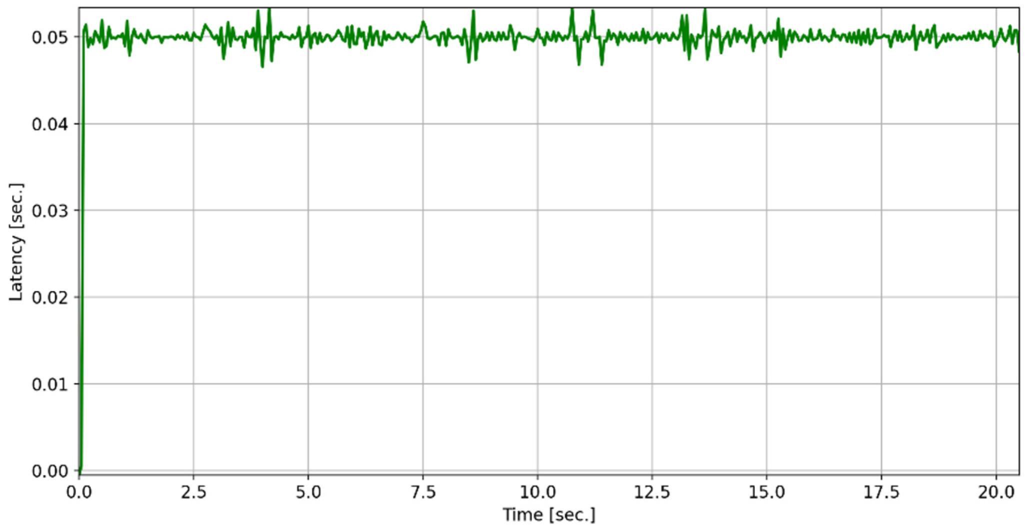


Fig. 19. Communication response times at a frequency of 20 Hz

4 DISCUSSION

The main goal of the tool presented in this document is to provide an open-access, low-cost, ready-to-use solution that enables users to enhance their robot programming skills within a robust programming environment. This is achieved by leveraging the connectivity platform offered by ROS. ROS primarily supports programming languages such as C++ and Python and also offers compatibility with MATLAB and Lua. Although the literature mentions options for educational robots that incorporate ROS, such as EUROPA, Robotont, and PlatypOU [20–22], these robots are often proprietary, and reproducing them can be a laborious task, not to mention the potential obstacle posed by their cost.

In contrast, our proposal, based on the CrowBot BOLT, is highly accessible and strikes a good balance between cost and performance. A significant advantage of our proposal is the opportunity it provides for users to interact with a mobile robot in real-world scenarios, enhancing their learning experience compared to traditional tools that often lack a tangible practical component [15, 17, 18]. No similar commercial options were found that provide such accessible access to working with the Robot Operating System.

In terms of benefits, our proposal has certain limitations. For example, obstacle detection is limited to a single point at the front of the robot, while other suggestions include sensors such as depth cameras and 2D lidar scanners [24]. However, this limitation is offset by the inclusion of multiple sensors, enabling a wide range of applications. Additionally, it's important to note that even ROS-based proposals with more advanced hardware exhibit similar constraints, despite the seemingly limited local processing capacity of an ESP32-based system [20]. These limitations can be overcome by processing information remotely on a central computer.

It is true that our tool is not designed for performing complex applications in unstructured environments, despite the significant benefits [25]. Instead, it is more focused on the educational and training domains in a classroom setting, which makes it easier to assess advanced control systems [28]. The architecture based on the 'rosserial' package has proven to be effective in evaluating various control strategies [28], [29]. Finally, our tool is freely available and supported by practical examples that help users effectively utilize the capabilities of the mobile robot using the Robot Operating System.

5 CONCLUSIONS

In this study, we have introduced an open-access tool that facilitates the installation of ROS on a mobile robot through the rosserial package. This solution offers access to robot speeds, RGB LEDs, a buzzer, a programmable button, and proximity, light, and line sensors. Our results demonstrate the effectiveness of these techniques, and stable latency is achieved at a frequency of 10 Hz.

While there are similar or even more advanced proposals available on the market, none of them offer the same level of immediate reproducibility that our approach provides. Our approach simply requires the acquisition of a robotic kit. This work introduces an opportunity to make a powerful tool like ROS accessible to students and developers with minimal investment, providing tangible results that enhance the learning experience in the field of robotics.

However, it is important to highlight the limitations of our proposal, including limited local processing capacity, a restricted number of sensors, relatively low communication speeds, and the requirement for the exclusive use of the CrowBot BOLT robot. These limitations inspire us to tackle and overcome these challenges in our future work.

6 ACKNOWLEDGMENT

The authors thank the Universidad Indoamérica and CICHE Research Center. The results of this work are part of the project "Educación 4.0 aplicada a la enseñanza de competencias STEAM."

7 REFERENCES

[1] M. Ullah *et al.*, "Serious games in science education. A systematic literature review," *Virtual Real. Intell. Hardw.*, vol. 4, no. 3, pp. 189–209, 2022. https://doi.org/10.1016/j.vrih.2022.02.001

[2] A. K. Barianos, A. Papadakis, and N. Vidakis, "Content manager for serious games: Theoretical framework and digital platform," *Adv. Mob. Learn. Educ. Res.*, vol. 2, no. 1, pp. 251–262, 2022. https://doi.org/10.25082/AMLER.2022.01.009

[3] L. A. Mamolo, "Students' evaluation and learning experience on the utilization of Digital Interactive Math Comics (DIMaC) mobile app," *Adv. Mob. Learn. Educ. Res.*, vol. 2, no. 2, pp. 375–388, 2022. https://doi.org/10.25082/AMLER.2022.02.006

[4] A. Baltynova *et al.*, "Pedagogical conditions for the training of future teachers based on digital educational technologies," *Int. J. Emerg. Technol. Learn.*, vol. 18, no. 18, pp. 121–137, 2023. https://doi.org/10.3991/ijet.v18i18.43209

[5] N. N. S. P. Verawati, N. Ernita, and S. Prayogi, "Enhancing the reasoning performance of STEM students in modern physics courses using virtual simulation in the LMS platform," *Int. J. Emerg. Technol. Learn.*, vol. 17, no. 13, pp. 267–277, 2022. https://doi.org/10.3991/ijet.v17i13.31459

[6] J. Guggemos, S. Seufert, S. Sonderegger, and M. Burkhard, "Social robots in education: Conceptual overview and case study of use," in *Orchestration of Learning Environments in the Digital World*, 2022, pp. 173–195. https://doi.org/10.1007/978-3-030-90944-4_10

[7] T. Belpaeme, J. Kennedy, A. Ramachandran, B. Scassellati, and F. Tanaka, "Social robots for education: A review," *Sci. Robot.*, vol. 3, no. 21, 2018. https://doi.org/10.1126/scirobotics.aat5954

[8] J. Varela-Aldás, J. Buele, J. Jadan-Guerrero, and V. H. Andaluz, "Teaching STEM competencies through an educational mobile robot," in *Lecture Notes in Computer Science*, 2020, pp. 560–573. https://doi.org/10.1007/978-3-030-50506-6_38

[9] K. Youssef, S. Said, S. Alkork, and T. Beyrouthy, "Social robotics in education: A survey on recent studies and applications," *Int. J. Emerg. Technol. Learn.*, vol. 18, no. 3, pp. 67–82, 2023. https://doi.org/10.3991/ijet.v18i03.33529

[10] J. López-Belmonte, A. Segura-Robles, A.-J. Moreno-Guerrero, and M.-E. Parra-González, "Robotics in education: A scientific mapping of the literature in web of science," *Electronics*, vol. 10, no. 3, p. 291, 2021. https://doi.org/10.3390/electronics10030291

[11] E. Ferreira, M. J. Silva, and B. da C. Valente, "Collaborative uses of ICT in education: Practices and representations of preservice elementary school teachers," in *2018 International Symposium on Computers in Education (SIIE)*, 2018, pp. 1–6. https://doi.org/10.1109/SIIE.2018.8586692

[12] O. Engwall and J. Lopes, "Interaction and collaboration in robot-assisted language learning for adults," *Comput. Assist. Lang. Learn.*, vol. 35, nos. 5–6, pp. 1273–1309, 2022. https://doi.org/10.1080/09588221.2020.1799821

[13] J. Barnes, S. M. FakhrHosseini, E. Vasey, C. H. Park, and M. Jeon, "Child-robot theater: Engaging elementary students in informal STEAM education using robots," *IEEE Pervasive Comput.*, vol. 19, no. 1, pp. 22–31, 2020. https://doi.org/10.1109/MPRV.2019.2940181

[14] S. Tselegkaridis and T. Sapounidis, "Simulators in educational robotics: A Review," *Educ. Sci.*, vol. 11, no. 1, p. 11, 2021. https://doi.org/10.3390/educsci11010011

[15] J. M. Cañas, E. Perdices, L. García-Pérez, and J. Fernández-Conde, "A ROS-based open tool for intelligent robotics education," *Appl. Sci.*, vol. 10, no. 21, p. 7419, 2020. https://doi.org/10.3390/app10217419

[16] A. Kerem Erdogmus and U. Yayan, "Virtual robotic laboratory compatible mobile robots for education and research," in *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2021, pp. 1–6. https://doi.org/10.1109/INISTA52262.2021.9548503

[17] G. A. Casan, E. Cervera, A. A. Moughlbay, J. Alemany, and P. Martinet, "ROS-based online robot programming for remote education and training," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6101–6106. https://doi.org/10.1109/ICRA.2015.7140055

[18] Z. B. Rivera, M. C. De Simone, and D. Guida, "Unmanned ground vehicle modelling in Gazebo/ROS-based environments," *Machines*, vol. 7, no. 2, p. 42, 2019. https://doi.org/10.3390/machines7020042

[19] A. Dobrokvashina, R. Lavrenov, E. Magid, Y. Bai, and M. Svinin, "How to create a new model of a mobile robot in ROS/Gazebo environment: An extended tutorial," *Int. J. Mech. Eng. Robot. Res.*, vol. 12, no. 4, pp. 192–199, 2023. https://doi.org/10.18178/ijmerr.12.4.192-199

[20] M. F. Zakaria, J. C. Shing, and M. R. Md Tomari, "Implementation of robot operating system in beaglebone black based mobile robot for obstacle avoidance application," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 7, no. 6, p. 2213, 2017. https://doi.org/10.18517/ijaseit.7.6.3221

[21] G. Karalekas, S. Vologiannidis, and J. Kalomiros, "EUROPA – A ROS-based open platform for educational robotics," in *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, 2019, pp. 452–457. https://doi.org/10.1109/IDAACS.2019.8924409

[22] R. Raudmäe *et al.*, "ROBOTONT – Open-source and ROS-supported omnidirectional mobile robot for education and research," *HardwareX*, vol. 14, p. e00436, 2023. https://doi.org/10.1016/j.ohx.2023.e00436

[23] M. Rácz *et al.*, "PlatypOUs—A mobile robot platform and demonstration tool supporting STEM education," *Sensors*, vol. 22, no. 6, p. 2284, 2022. https://doi.org/10.3390/s22062284

[24] T. B. Minh, H. Thanh Luan, D. X. Phu, T. Quang Nhu, and B. M. Duong, "MiniRos: An autonomous UGV robot for education and research," in *2021 International Conference on System Science and Engineering (ICSSE)*, 2021, pp. 170–175. https://doi.org/10.1109/ICSSE52999.2021.9538463

[25] E. C. Camacho, N. I. Ospina, and J. M. Calderón, "COVID-Bot: UV-C based autonomous sanitizing robotic platform for COVID-19," *IFAC-PapersOnLine*, vol. 54, no. 13, pp. 317–322, 2021. https://doi.org/10.1016/j.ifacol.2021.10.466

[26] Electrow, "CrowBot BOLT-open source programmable smart robot car STEAM robot kit." https://www.elecrow.com/crowbot-bolt-programmable-smart-robot-car-steam-robot-kit.html [Accessed Jul. 07, 2023].

[27] D. Herrera, F. Roberti, R. Carelli, V. Andaluz, J. Varela, and J. Ortiz, "Modeling and path-following control of a wheelchair in human-shared environments," *Int. J. Humanoid Robot.*, vol. 15, pp. 1–33, 2018. https://doi.org/10.1142/S021984361850010X

[28] W.-Y. Wu and Y.-C. Liu, "Autonomous guided robotic systems in regulating indoor environmental quality," in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2018, pp. 188–193. https://doi.org/10.1109/AIM.2018.8452331

[29] M. K. Diab, H. H. Ammar, and R. E. Shalaby, "Self-driving car lane-keeping assist using PID and pure pursuit control," in *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, 2020, pp. 1–6. https://doi.org/10.1109/3ICT51146.2020.9311987

8 AUTHORS

José Varela-Aldás, Ph.D., Associate Professor, Centro de Investigación en Ciencias Humanas y de la Educación – CICHE, Universidad Indoamérica, Ambato 180103, Ecuador.

Guillermo Palacios-Navarro, Ph.D., Associate Professor, Department of Electronic Engineering and Communications, University of Zaragoza, Teruel 44003, Spain.