UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Visión Electrónica
https://doi.org/10.14483/issn.2248-4728

VISIÓN ELECTRÓNICA

A RESEARCH VISION

Educational robotic prototype for learning Object-Oriented Programming principles

Prototipo robótico educativo para aprender los principios de la programación orientada a objetos

Natalia Cañón Castillo[1], Juan D. Rodriguez[2], Juan Carlos Rincón[3], Henry B. Guerrero[4]

INFORMACIÓN DEL ARTÍCULO

Historia del artículo:
Enviado: 25/09/2025
Recibido: 01/11/2025
Aceptado: 17/11/2025

Keywords:
Differential-drive robot
LiDAR sensor
Linux
OOP
Python 3
Raspberry Pi

Palabras clave:
Robot móvil
Sensor LiDAR
Linux
POO
Python 3
Raspberry Pi

ABSTRACT

According to several reports, mobile robotics can be an effective tool for teaching and learning computer programming. In light of the fact that the object-oriented programming (OOP) paradigm is often not adequately incorporated by students in the corresponding classical courses, this paper describes a prototype mobile robotic system as an effective method for teaching and learning the concepts of object-oriented programming paradigms via tutored experiences. Students demonstrated high levels of enthusiasm during the development of experiments involving sensors and actuators in conjunction with objects, methods, classes, access modifiers, and in general OOP concepts applied to robot movement. A low-cost, smallscale mobile robot was constructed using a Raspberry Pi 3 B+ running Linux and suitable motors and sensors. As a result of combining OOP with mobile robots, a high level of interest in related topics was observed, even facilitating the teaching process.

RESUMEN

Diversos estudios indican que la robótica móvil puede ser una herramienta eficaz para la enseñanza y el aprendizaje de la programación. Dado que el paradigma de programación orientado a objetos (POO) frecuentemente no es asimilado de manera adecuada por los estudiantes en los cursos clásicos, este artículo presenta un prototipo de sistema robótico móvil como un método efectivo para enseñar

1. Industrial Engineering student, Universidad Distrital Francisco Jose de Caldas, Colombia. E-mail: ncanonc@udistrital.edu.co
2. Electronic Engineering student, Universidad Distrital Francisco Jose de Caldas, Colombia. E-mail: jdrodriguezr@udistrital.edu.co
3. Electronic Engineering student, Universidad Distrital Francisco Jose de Caldas, Colombia. E-mail: jcrinconn@udistrital.edu.co
4. Electronic Engineer, Universidad de los Llanos, Colombia. PhD in Mechanical Engineering, University of São Paulo, Brazil. Postdoctoral Researcher, BITPOINTER SAS, Colombia, in agreement with the Ministry of Science and Technology and Innovation of Colombia. Email: hbguerrero@ieee.org. ORCID: https://orcid.org/0000-0003-4243-4205

Cite this article as: N. Cañón-Castillo, J. D. Rodriguez, J. C. Rincón, H. B. Guerrero, "Educational robotic prototype for learning Object-Oriented Programming principles", Visión Electrónica, vol. 19, no. 2, 2025.

y aprender los conceptos del paradigma de programación orientada a objetos mediante experiencias guiadas. Durante el desarrollo de experimentos que involucraron sensores y actuadores en conjunto con objetos, métodos, clases y modificadores de acceso, en general aplicando conceptos de POO al movimiento del robot, los estudiantes demostraron altos niveles de entusiasmo. Se construyó un robot móvil de bajo costo y pequeña escala utilizando una Raspberry Pi 3 B+ con Linux, junto con motores y sensores apropiados. Como resultado de combinar POO con robótica móvil, se observó un nivel elevado de interés en temas relacionados, incluso facilitando el proceso de enseñanza.

1. Introduction

An ongoing research project aimed at developing autonomous navigation for agricultural mobile robotics led us to realize that object-oriented programming isn't as well exploited in mobile robotics as it should be. For students involved in mobile robotics projects, a thorough understanding of Object Oriented Programming (OOP) concepts is critical, and our research group has noticed that students have difficulty assimilating abstract concepts in general and for this reason, we designed a small-scale mobile robot to serve as a learning and teaching support tool for object-oriented programming.

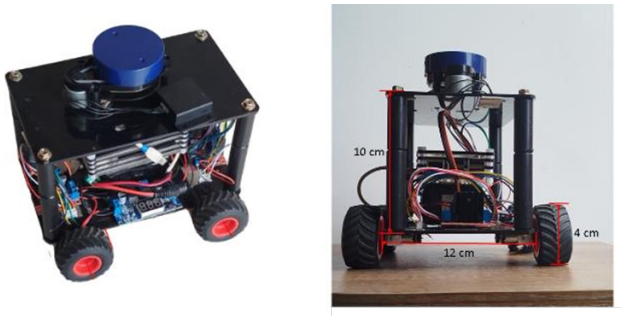
Diverse approaches in which mobile robots to support the learning and teaching of programming are shared in suitable publications like [1, 2, 3, 4], however the searching of formal literature reporting the taking of the advantages of the usage of mobile robots to teaching and learning OOP leads to find few papers which we consider as a god opportunity to undertake an interesting defiance: A mobile robot prototype for learning and teaching Object-Oriented Programming which is presented in this paper.

The prototype is described in section II, followed by a discussion of our considerations and future work in section III. Our acknowledgements and references are presented at the end of the paper.

2. General structure description of mobile robot

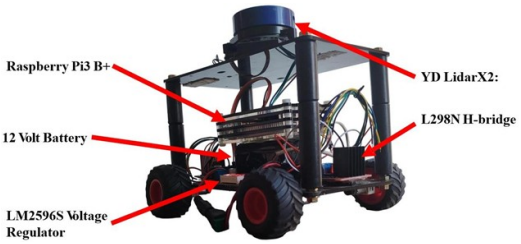
A four wheels differential drive mobile robot prototype was assembled and this is shown in Figure 1. It measures approximately 0.2 meters in length, 0.16 meters in width, and 0.15 meters in height. In terms of weight, it is approximately 0.6 kilograms. Our mobile robot has two floors structured to carry the vehicle elements. According to depicted in Fig. 2, the first floor is destined to allocate four DC motors, two L298 H-bridge driver boards, an LM2596S voltage regulator, a 12 Vdc battery as well as a 3b+ microcomputer, In accordance with Fig 2 over the second floor a LiDar sensor was allocated.

Figure 1: General view of Robot structure



Source: Own.

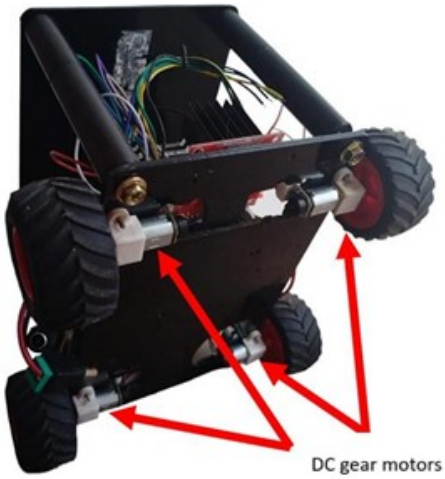
Figure 2: General view about elements distribution in the robot structure



Source: Own.

Figure 3 has been included to show how four dc motoreductors and four wheels were suitable assembled to the robot chassis.

Figure 3: Propulsion DC gear motors

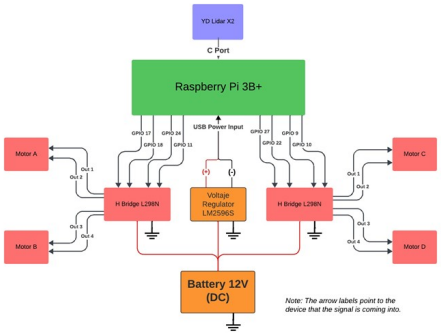


Source: Own.

Over the top of the robot structure, we allocated a YdLidarX2 sensor [5, 6]. According to [5], LiDAR sensing is basically remote sensing technology that emits a laser light beam with defined intensity and focus and measures the reflected beam arrival time detected by the photodiodes (PD) within the sensor. LIDAR sensors are capable of measuring distances to obstacles around them. As part of the YDLidar user manual provided by its manufacturer, instructions are provided about how to set up this sensor to work with Linux version working in this case over a Raspberry Pi 3b+, in particular we installed a Raspbian operating system. Although this manual specifies that it is only compatible with Linux Ubuntu 18.04, we tested YDLidar on Raspbian obtaining acceptable results.

Figure 4 has been included in order to depicts devices connections. The YDLidar X2 uses a UART which facilitates the communication of its measurements to USB ports, consequently, we attached the Lidar sensor to a Raspberri Py 3b+ USB port which is represented as the C-port in Fig. 4. A 12 Vdc battery is used to supply two L298N H-bridges as well as a LM2596S regulator which regulates 5Vdc to supply the Raspberry pi3b+. According to depicted in Fig. 4 suitable Raspberry pi 3b+ gpio ports was connected to the control inputs of the H-bridges in order to drive the four robot dc motors.

Figure 4: Block diagram about used devices.



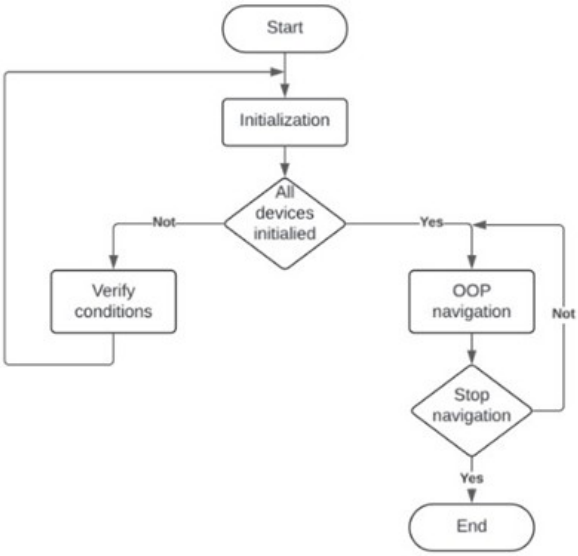
Source: Own.

The block diagram shown in Figure 4 provides an overview of the general operation of the proposed general system. By following the block diagram, we can see that LIDAR measurements are sent directly to the Raspberry pi 3b+ which processes the corresponding information to calculate control actions that correspond to PWM (Pulse Wide Modulation) signals for its application to two H-bridges that transfer energy to DC motors.

The Raspberry PI 3B+ microcomputer has a software application developed in the Python 3 programming language to control the navigation of the mobile robot. The programmed application is based on object-oriented programming concepts. The software application aims to control the movement of an autonomous robot according to the LIDAR sensor (YDLidar) detections to avoid obstacles by modifying the speeds of the robot motors.

In Figure 5 we show a flow chart which is included to describe the software application workflow, in accordance, after it starts an initialization subprocess is attended, during the initialization subprocess the LiDar sensor and g-pio ports are initialized. After the initialization subprocess, it is verified so that if some error is reported from the LiDar or even the gpio ports then the user has to verify correct connections and correct if it is necessary. If all devices report successful initialization then the OOP (Object-oriented programming) navigation subprocess is attended until it is requested to stop the application at all. When it id stop the application, dc motors and the LiDar are deactivated. OOP navigation subprocess will be detailed later.

Figure 5: General flow chart.

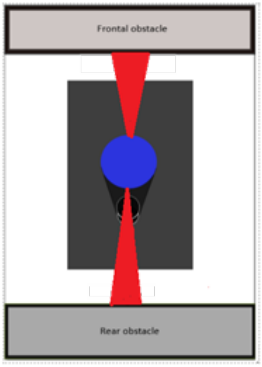


Source: Own.

Before to detail the working regarding the OOP subprocess included in flowchart at Fig. 5, let us to explain to mode in which we have undertaken our tests, in accordance with a first step, we allocate the prototype between two obstacles at its frontal and rear sectors, this situation is represented in Fig. 6.

After the robot is positioned like depicted in Fig. 6 then the algorithm represented in Fig. 5 is enabled to run and once all devices are successfully initialized, the OOP navigation subprocess is attended.

Figure 6: Start position of the robot prototype.

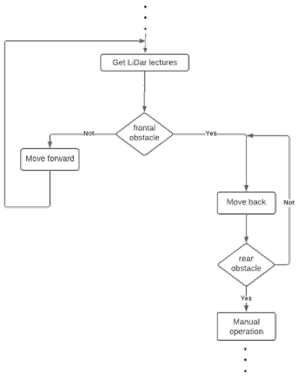


Source: Own.

OOP navigation subprocess is detailed in the flowchart in Fig. 7, in this subprocess at first the LiDar lectures are got to verify the presence of some obstacle at its front, if not, the robot is moved forward until it is detected some obstacle at least at 15cm in front, such that the robot is moved backward until it is detected some obstacle at least at 15cm in back, subsequently it is delivered the manual operation to the user.

The manual operation subprocess depicted in flowchart in Fig.7 is detailed in Fig. 8, in accordance, once the manual operation is delivered to the user the robot stops and proceeds to keep waiting for the pressing of a key from a remote keyboard our case we used the keyboard arrows to control manually the prototype, consequently up arrow determines the forward displacement, back arrow determines the backward displacement, left arrow determines the turning left, The right arrow determines the turning right and finally if the key “p” is pressed then the robot stops and even all application is closed.

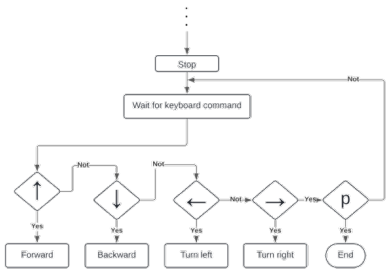
Figure 7: Flow chart regarding the OOP navigation subprocess.



Source: Own.

Flow charts allow to explain in sequence the working of our approach, however it in suitable to mention the corresponding role of the object-oriented programming. Figure 9 represents the classes that we used, accordingly, we implemented three classes which were denominated Car, Motores and MyLidar. The Car class has a method to run ordered the required method from another classes.

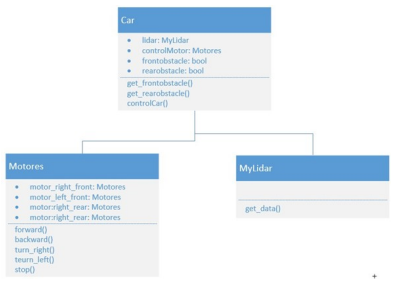
Figure 8: Flow chart regarding the OOP manual operation subprocess.



Source: Own.

Taking into account that python3 allows the importing of methods from another modules, again from depicted in Fig. 9, in the class Car the object lidar belongs to the MyLidar Class whereas the object controlMotor belongs to the class Motores, The Car class has the attributes front obstacle and rear obstacle, as wheel the methods get_frontobstacle() and get_rearobstacle(), get_frontobstacle() notifies when an obstacle is present at front of the vehicle and get_rearobstacle() notifies when an obstacle is present at rear of the vehicle. The method controlCar() is used to operate manually the prototype. Getter methods in class Car use the method get_data() in the class My_Lidar in order to get the measurements corresponding to those depicted in Fig. 6.

Figure 9: Classes diagram.



Source: Own.

In Fig. 9, the `Motores` class contains the objects `motor_right_front`, `motor_left_front`, `motor_right_rear`, and `motor_left_rear`, which correspond to the DC motors assembled in the prototype. Additionally, the `Motores` class includes the methods `forward()`, `backward()`, `turn_right()`, `turn_left()`, and `stop()`. Accordingly, when the

`controlCar()` method from the `Car` class is executed, the appropriate signals are applied to the H-bridges to drive the four motors.

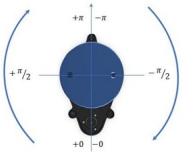
Figure 10 depicts the angular reference used by the LiDAR sensor to report its measurements. In this configuration, the LiDAR reports both the measured distance and the angle at which each measurement was obtained. For the left-side measurements, the reported angles satisfy:

$$0 < \alpha < \pi$$

while for the right-side measurements, the reported angles satisfy:

$$-\pi < \alpha < 0$$

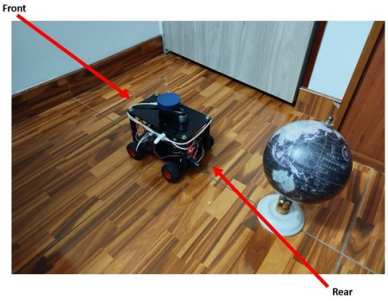
Figure 10: Angular sense in which LiDar sensor reports its measurements.



Source: Own.

The figure 11 illustrates the front and rear regions of the robot, as well as the initial position for our tests, therefore, in the front there is a wall as an obstacle, while in the rear there is another obstacle. Figure 12 shows an image while the mobile robot moves forward during corresponding task described in flow chart in Figure 7. Figure 13 shows an image while the mobile robot moves backward during corresponding task described in flow chart in Figure 7. Figure 14 shows an image while the mobile robot moves under manual operation during corresponding task described in flow chart in Figure 7.

Figure 11: Front and rear regions of the mobile robot.

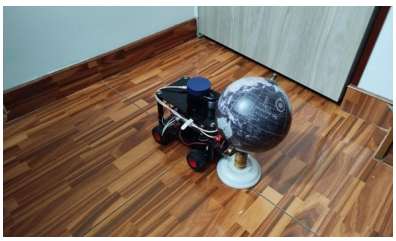


Source: Own.

Figure 12: Mobile robot moving forward

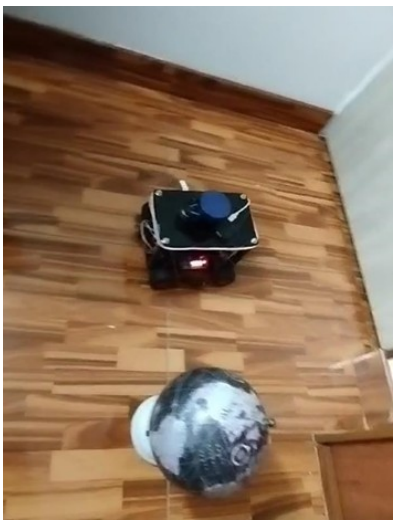
Source: Own.

Figure 13: Mobile robot moving forward



Source: Own.

Figure 14: Mobile robot moving during manual operation



Source: Own.

3. Conclusions and Future works

The described mobile robot was operated using remote access to the Raspberry Pi 3b+ in the robot, as a future work we are looking for the usage of a remote graphical user interface since currently we are accessing just the Raspbian terminal to control the robot. According to initial demonstrations to

OOP students, we consider that our approach has shown promising results since we noted that initial OOP concepts like objects, attributes, methods, and access modifiers were adopted more quickly in comparison to experiences in the past without a robot, noting that in the most of students, happiness during the learning was expressed.

Acknowledgments

The authors sincerely thank the Colombian Ministry of Science, Technology, and Innovation (MINCIENCIAS) and the Francisco José de Caldas Fund for their support through the allocation of resources in Call 934 of 2023, contract 332-2023, project 99209. In compliance with the disclosure requirements of this contract, the authors refer to MINCIENCIAS and the Francisco José de Caldas Fund in this work. The authors also thank the LASER research group at the Francisco José de Caldas District University for their guidance and advice during the development of this research project. Special thanks go to BitPointer S.A.S., beneficiary of project 99209, in particular to its manager, Juan Fajardo Barrero, and its administrative director, Alexandra Reyes Agudelo, for their valuable contributions and commitment to this initiative.

References

[1] O. O. Ortiz et al., "Innovative Mobile Robot Method: Improving the Learning of Programming Languages," *IEEE Trans. on Education*, vol. 60, no. 2, 2017.

[2] D. N. A. Jawawi et al., "Introducing computer programming using mobile robots," in *10th Asian Control Conference (ASCC)*, 2015.

[3] J. M. Rodríguez-Corral et al., "Application of robot programming to teaching OOP," *Int. Journal of Engineering Education*, vol. 32, 2016.

[4] R. Moran-Borbor et al., "Desarrollo de un robot sumo como material educativo," *Revista Habitus*, vol. 1, no. 2, 2021.

[5] I. K. Alam Bhuiyan, *LiDAR Sensor for Autonomous Vehicle*, Technical Report, 2017.

[6] H. Borrero-Guerrero et al., "A differential drive mobile robot controlled by ROS," *Visión Electrónica*, vol. 17, no. 2, 2023.