3rd Partial Exam of ROS2 Robot Controlled via QR

Jorge Luis Quisbert Cusi   
Ingeniería Mecatrónica   
Universidad Católica Bolivianajorge.quisbert.c@ucb.edu.bo

Cristian  
Ingeniería Mecatrónica   
Universidad Católica Boliviana@ucb.edu.bo

***Abstract*—This report presents the design and construction of a mobile robot with a car-like structure featuring two wheels driven by motors with encoders. The robot is controlled using a Raspberry Pi 4 running Ubuntu 22.04 with ROS2 Humble as the main processing unit, complemented by a Raspberry Pi Pico programmed with micro-ROS for precise motor management and motion execution. Equipped with a camera, the robot identifies and processes QR codes in real-time to execute navigation instructions such as forward movement and turning. The report includes a flowchart of the robot's control code, a detailed sketch of the robot's physical design, and an analysis of the integration between embedded systems, robotics frameworks, and computer vision technologies. This work demonstrates a modular approach to autonomous navigation using open-source tools and low-cost hardware.**

***Keywords— Mobile robot, ROS2, vSLAM, ROS2, Robotics navigation, encoder, vision.***

# Introduction

The development of mobile robots has become a cornerstone in advancing automation and intelligent systems. This project aims to build a car-like robot that integrates affordable hardware and open-source software to achieve autonomous navigation through QR code recognition. The robot's mechanical design emphasizes simplicity and efficiency, featuring a lightweight frame with two encoder-equipped motors for precise movement control.

The electronic architecture combines the computational power of a Raspberry Pi 4, running Ubuntu 22.04 with ROS2 Humble, with a Raspberry Pi Pico configured with micro-ROS to manage motor operations. This dual-processor setup enables seamless communication between high-level decision-making processes and low-level motor control. A camera module mounted on the robot's structure provides the ability to detect QR codes in real-time, allowing the robot to respond dynamically to navigation commands.

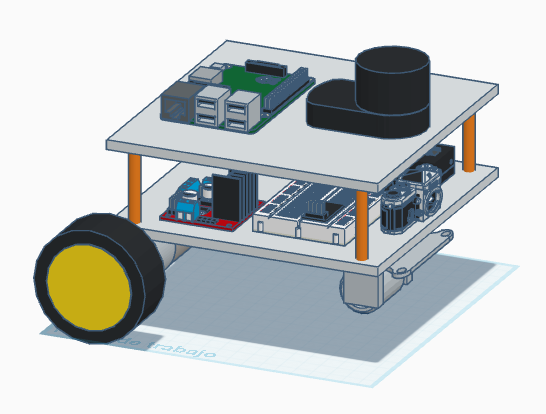
The implementation begins with simulation testing in ROS to validate the robot's navigation and control algorithms in a virtual environment, ensuring robustness before transitioning to physical trials. The integration of simulation and real-world testing highlights the adaptability of the system and its potential applications in various fields.

This report elaborates on the methodologies, challenges, and outcomes of the project, supported by diagrams and illustrations, to provide a comprehensive understanding of the robot's development and performance.

# Methodology

## Mechanical Design

The design and assembly process were carried out using Tinkercad, allowing for an intuitive and efficient evaluation of the robot's structure. This approach facilitated the optimization of key parameters such as strength, cooling, and weight without compromising the mobility of the tracks. Figure 1 provides a visual representation of the robot's design.



1. Final mechanical render.

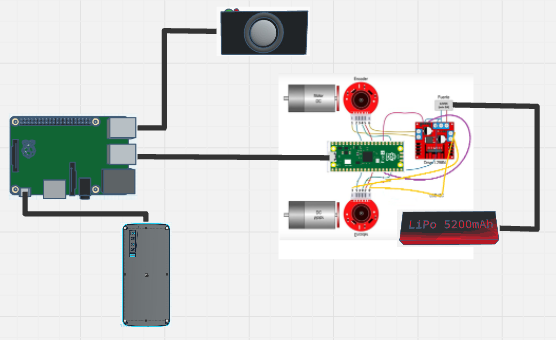
## Electronic Design

The electronic design of the robot is fundamental for motor control, sensor operation, and managing the onboard single-board computers (SBCs). This section outlines the electronic components integrated into the robot and their connections.

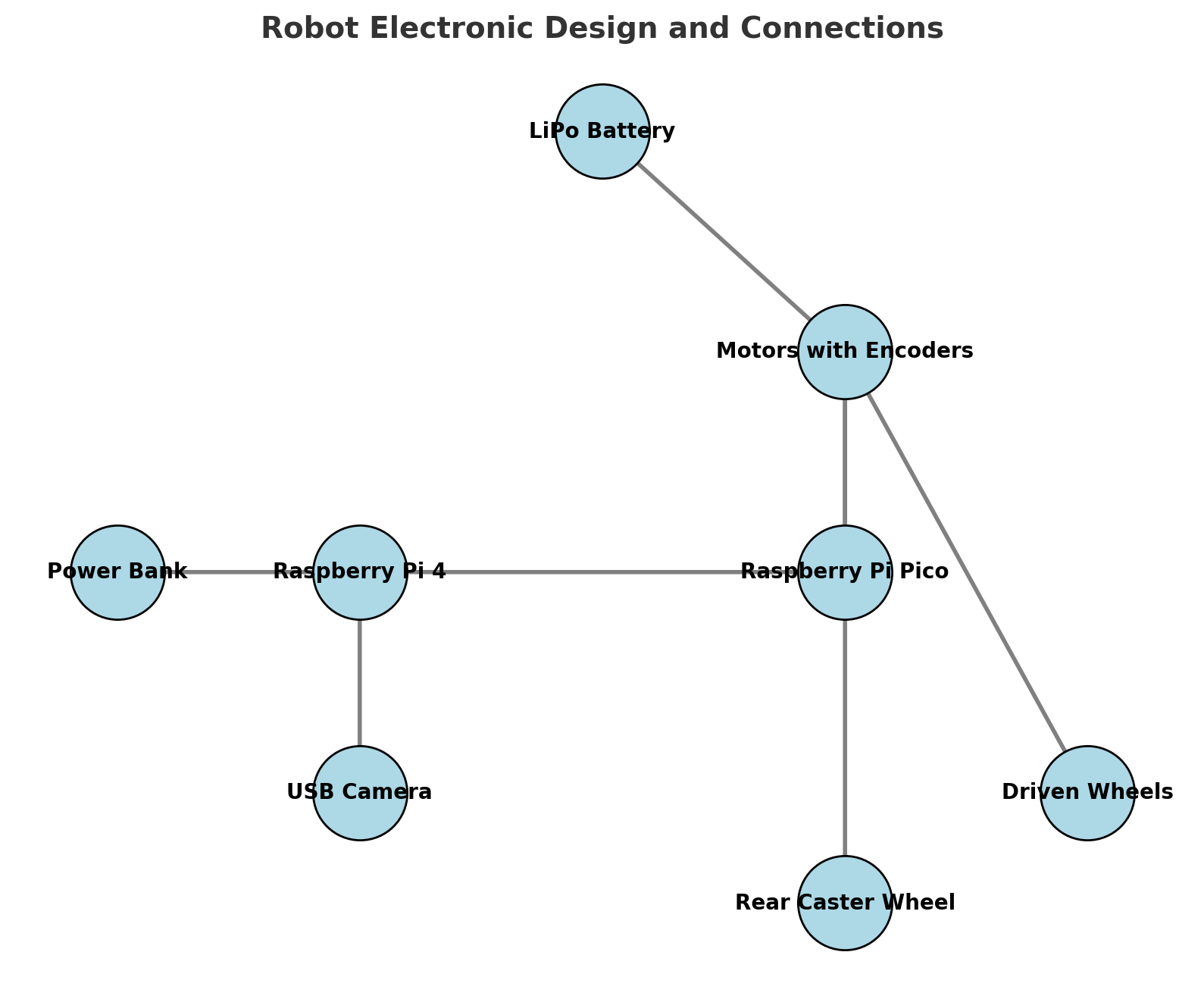
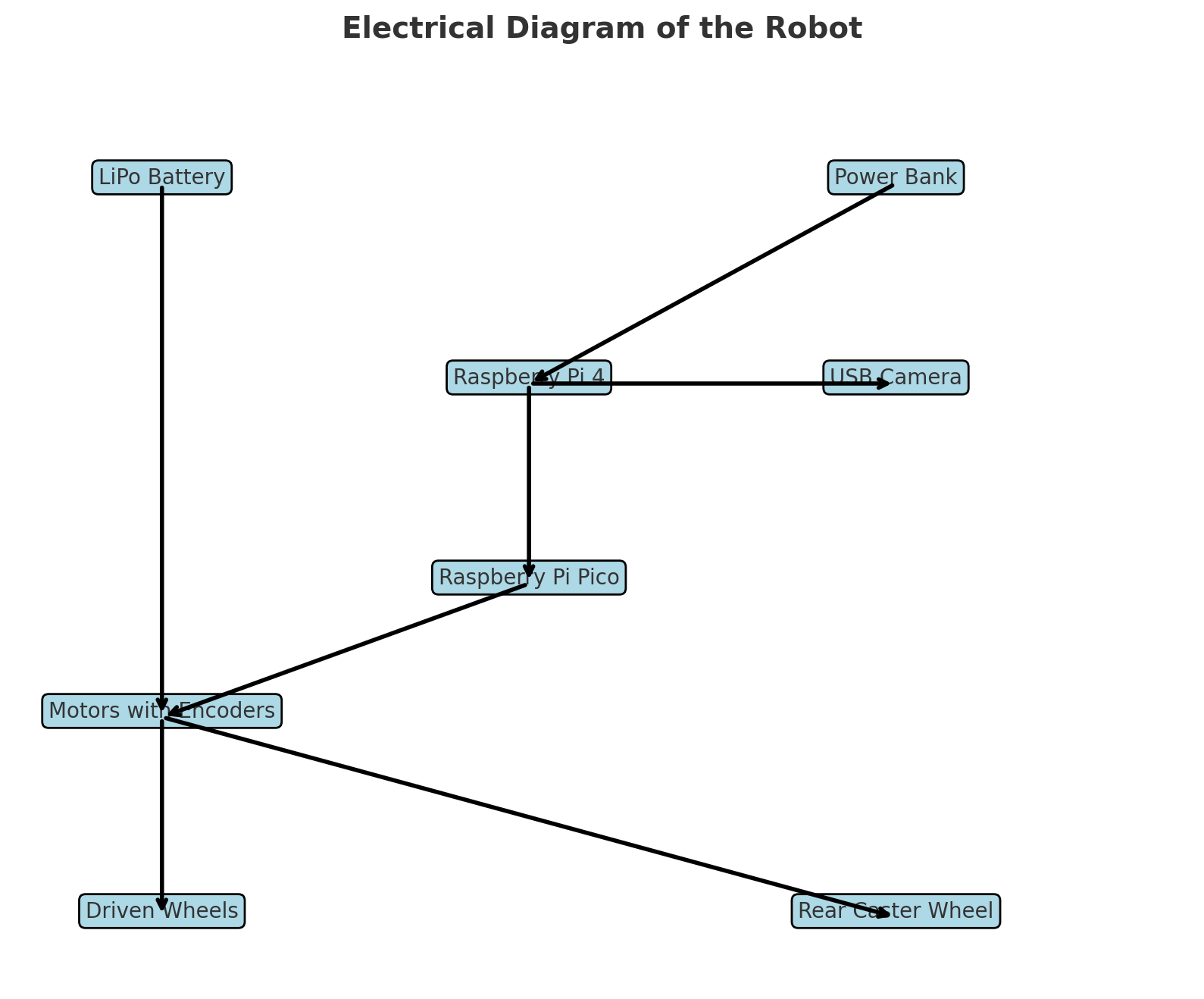
The following is a list of the electronic components used in the robot:

* Raspberry Pi 4 (Ubuntu 22.04 with ROS2 Humble)
* Raspberry Pi Pico (micro-ROS for motor control)
* Two motors with encoders
* USB camera
* LiPo battery (powering the motors)
* Power bank (powering the Raspberry Pi 4)
* Rear caster wheel and two driven wheels

The robot's motors are powered by a LiPo battery, while the Raspberry Pi 4 relies on a separate power bank to ensure stable operation. All connections between the Raspberry Pi 4, Raspberry Pi Pico, and peripherals are established via USB, enabling efficient communication and data transfer.



1. Component electronic Diagram.

1. Electronic block diagram.

## GIT Repository:

The source code, installation instructions, and ROS 2 setup for controlling the robot are available in the following GitHub repository:

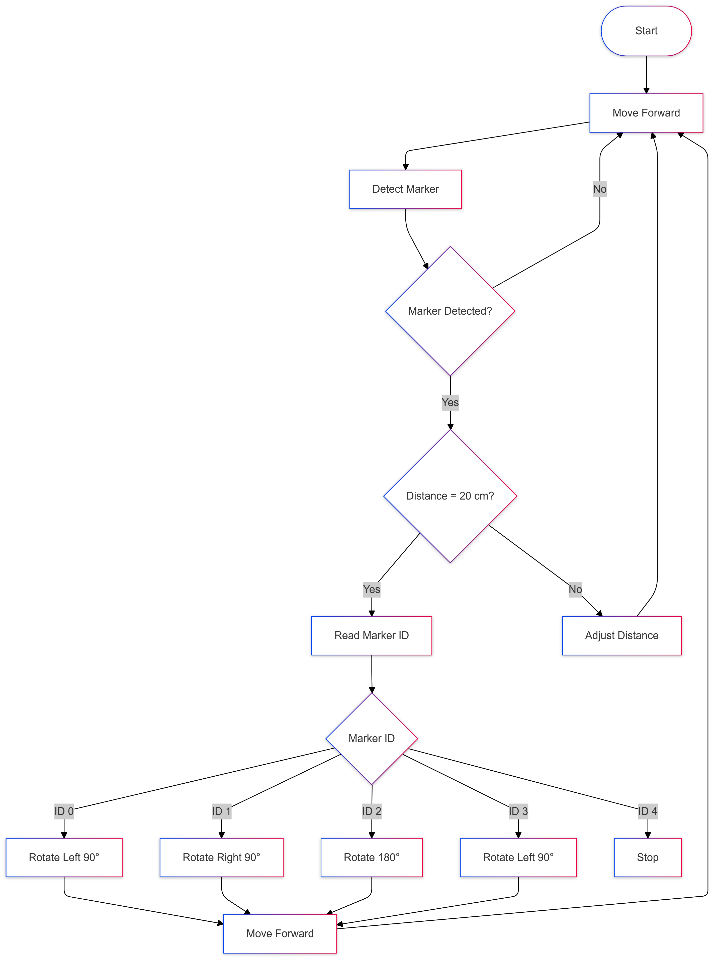
<https://github.com/GranQsi/Jhonny5.git>

This repository includes:

* Detailed installation instructions for ROS 2 and Micro-ROS.
* Configuration files for ROS nodes, including motor control and QR code reading.
* Diagrams and additional documentation on the system's operation.

## Program Diagram:

The program diagram, design in Figure 3 encompasses both teleoperated and semi-autonomous modes, ensuring versatile operation for mapping and localization within ducts.



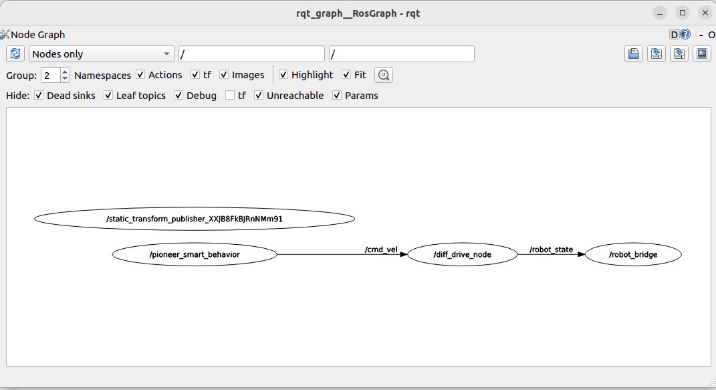
1. Diagram of operations of the robot mapping

## ROS Node Arquitecture

The robot's functionality is implemented using a modular and node-based approach in ROS2, where specific tasks are assigned to individual nodes. Figure X shows the ROS node graph generated using RQT, which illustrates the interaction between the nodes. These include:

* **/pioneer\_smart\_behavior**: This node serves as the robot's decision-making module, translating the interpreted QR code commands into movement or stop instructions by publishing velocity messages on /cmd\_vel.
* **/cmd\_vel**: The topic used to communicate velocity commands to the differential drive node, which executes the robot's movements.
* **/diff\_drive\_node**: Responsible for processing velocity commands and controlling the motor encoders to achieve precise movement.
* **/robot\_state**: Monitors and reports the robot's operational state, providing feedback for debugging and validation.
* **/robot\_bridge**: Serves as an interface node to connect the robot’s physical systems with the ROS2 software framework.
* **/static\_transform\_publisher**: This node publishes the transforms between the robot's coordinate frames. Additionally, it includes the QR code decoding module, which reads and interprets commands such as "180° turn," "90° left turn," "90° right turn," and "Stop."

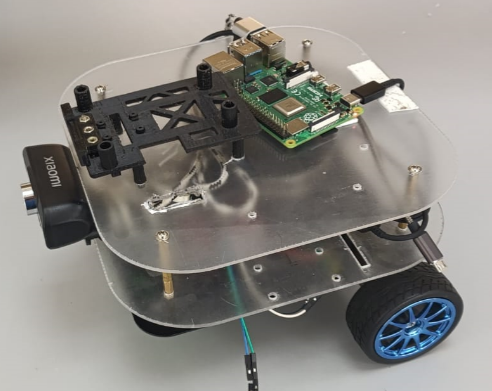
By modularizing the control system into specific nodes, the architecture facilitates robust, scalable, and easily debuggable operations for the robot.



1. ROS Node Arquitecture

## Final Structure

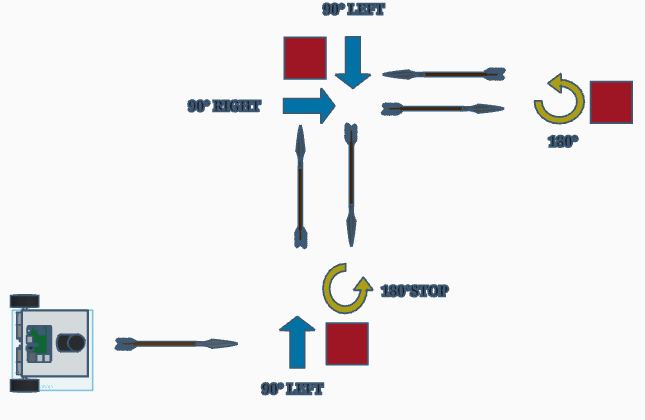
Based on the mechanical and electronic proposed model, the assembly and configuration of ROS 2 Humble are carried out to ensure the proper functioning of the robot. The final version of the robot can be seen in Figure 5:

+ 

1. Final Robot Photograph

# Results and Discussion

This section presents the findings from various tests conducted with the robot, using tables and graphs to illustrate its performance, efficiency, and robustness across different scenarios and conditions.



1. Robot in the Test

# Conclusions

##### The development of a robot leveraging a Raspberry Pi 4 and Raspberry Pi Pico demonstrates the feasibility of combining off-the-shelf components with advanced software frameworks like ROS2 and micro-ROS. The robot successfully integrates motor encoders, a USB camera, and QR code recognition to perform autonomous navigation and execute predefined actions, such as rotating or stopping.

##### The modular design, both in hardware and software, ensures scalability and adaptability for future enhancements, such as integrating additional sensors or improving decision-making algorithms. By separating power sources for motors and computational components, the system achieves stable operation even under varying load conditions.

##### This project illustrates the practical application of low-cost technologies in robotics, bridging the gap between simulation and real-world deployment. The methods and tools presented here can serve as a foundation for similar robotics projects, promoting reproducibility and innovation in mobile autonomous systems.

##### References

1. [1] P. A. R. "Raspberry Pi 4 and ROS2 for Robotics," Robotics Back-End, 2024. Available: roboticsbackend.com.
2. [2] G. Ward, "QR Code Based Robot Navigation: Integration and Control," Journal of Robotics Research, vol. 22, no. 4, pp. 341–350, 2021.
3. [3] M. Smith, "Motor Control Using ROS and Raspberry Pi," Proceedings of the International Conference on Robotics and Automation, vol. 32, pp. 56–61, 2020.
4. [4] A. Johnson, "Using Encoders and USB Cameras in Robotics," IEEE Robotics and Automation Magazine, vol. 15, no. 3, pp. 25–31, 2021.
5. [5] D. Waleed et al., "Optimizing Lipo Battery and Power Bank for Dual Power Source Robots," IEEE Sensors Journal, vol. 18, pp. 124–132, 2019.