AN-NAJAH NATIONAL UNIVERSITY

Computer Engineering Department

Hardware Graduation Project

Valet parking robot

Prepared By

Muheeb Hasan

Diaa Sharqawi

Supervised by

Dr. Samer Arandi

A report submitted in partial fulfillment of the requirements for bachelor's degree in computer engineering in the Faculty of Engineering Information Technology - Hardware Project

September,2024

Acknowledgements

We want to express our sincere gratitude to Dr. Samer for his hard work and priceless time. His astute counsel and contributions were crucial to this project's successful conclusion. We are also grateful to An-Najah University for giving us the chance to continue our education. A special thank you to our distinguished instructors in the Faculty of Engineering's Computer Engineering Department, who kindly shared their expertise with us and kept a close eye on our progress during the course of our studies. We owe a debt of gratitude to our families, whose steadfast encouragement and support enabled us to finish this project. We also extend our sincere gratitude to our friends for their unwavering support. Lastly, we would like to express our sincere appreciation to everyone who contributed to and motivated us during this project.

Disclaimer

Students from An-Najah National University's Faculty of Engineering's Computer Engineering Department wrote this study. It may contain grammatical and content mistakes and has not been changed or corrected—aside from editorial corrections—as a consequence of evaluation. The opinions presented in it, along with any conclusions and suggestions, belong only to the students. An-Najah National University disclaims all liability and responsibility for any effects resulting from the use of this report for purposes other than those for which it was ordered.

Contents

List of Figures

Abstract

When driving, parking is an everyday problem for many. It might be difficult and time-consuming to find a place. In order to combat this, our project builds a robot car that can function in parking lots (ground floor) There will be a model of the parking lot and a starting point where the car will be taken from the user by the robot (there is a fork in the front of the robot), and then it will park the car in the empty parking lot, where there is a camera at the top. A card is used to enter the car, causing the robot to locate the closest place and pick up the vehicle. The same card is used to call the robot, which returns the car when it's time to go. The robot car has a camera to detect empty spots and a forklift to lift automobiles. If the lot is full, it notifies to save needless travels(we use Raspberry Pi4 with Camera), The goal of this initiative is to improve parking by making it quicker and less stressful to find a spot.

1 Introduction

Robots are increasingly deployed in various scenarios, especially for tasks that are risky or physically demanding for humans. In our project, we developed a parking robot designed to assist people by parking their cars for them. The user simply leaves the vehicle at the garage entrance, and the robot handles the rest. Moreover, parking accidents, particularly while reversing, happen frequently. Our robot eliminates the need for human involvement, precisely parking and lifting the car autonomously. We created a prototype of this parking robot system, which we built and programmed to grab the car upon command.

The project was completed in two main phases. First, we focused on building the robot and constructing the garage, gathering the necessary hardware components, and setting up the garage door. Next, we programmed the robot to move, grab the car, and position it for parking. In the second phase, we developed the robot's decision-making system for determining the best parking spot. We implemented image processing to allow the robot to capture an image and park the car based on what it detects in the picture.

Our robot is cost-effective and operates using an infrared (IR) sensor. The robot follows the IR sensor along the main path, enters and exits the garage, and parks and retrieves the car at a 90-degree angle. The hardware part of the project was programmed using Arduino, while the image processing was done with a Raspberry Pi and its camera.

This report outlines our work in detail. In Chapter 2: Constraints and Previous Coursework, we discuss the challenges we faced during the early stages of the project and as it progressed. We also reflect on the coursework and past projects that helped us gain the knowledge and skills necessary for the project. Next, Chapter 3: Literature Review examines similar projects and highlights how ours stands apart.

Following that, we explore additional functionality in our project, focusing on specific components. Chapter 4: Methodology outlines the step-by-step approach, from assembling the robot and hardware, designing the layout, connecting the circuitry, and writing the code to testing the final product. In the next section, Results and Analysis, we present our findings and offer a detailed discussion. Finally, the last chapter, Conclusions and Recommendations, wraps up our key insights and suggests possible improvements for future work

2 Constraints, Standards/ Codes and Earliercourse work

2.1 Constraints

2.1.1 mechanical parts

We face several challenges due to our lack of experience with mechanical parts, starting with finding a business or store that sells them. Additionally, we struggle with assembling and connecting the different components properly.

2.1.2 robot movement

Because each DC motor is distinct and its performance can vary over time, the accuracy of the electrical component we identified was not adequate for determining the robot's actual position. As a result, we use infrared sensors to ensure that the robot consistently follows a predetermined path.

2.1.3 rotate the robot in a specific angle

Initially, we tried to control the robot's rotation by using a servo motor connected to the wheels. However, the mechanical design of the wheels did not provide enough flexibility to achieve a specified angle of rotation. As a result, we decided to put one wheel at the front and two wheels at the back of the robot. This configuration improves the rotation capability, but it still doesn’t allow us to accurately measure the angle. To address this, we use an ir sensor to ensure that the robot completes a full rotation.

2.1.4 raspberry pi

Our initial problem with the Raspberry Pi was that it would shut down after running the code for a few minutes. To resolve this, we installed a fan to keep the device cool, allowing it to operate for longer periods. Next, we needed to introduce a delay in the Arduino to ensure the camera captured the correct image. Finally, we faced issues with the connection between the Raspberry Pi and Arduino, as there were times when the Raspberry Pi wouldn’t receive all the signals from the Arduino. To address this, we had to keep the Arduino continuously sending messages until the Raspberry Pi responded

2.2 Earlier Courses

2.2.1 Arduino course

Through the Arduino course, we gained valuable knowledge about different hardware components used in our project, which was essential since a significant part of our work was based on Arduino.

2.2.2 CPU lab

In addition to showing us how to connect and solder the components together, that lab taught us to test each part before starting any work and to troubleshoot any problems that come up.

3 Literature Review

While parking may seem easy, many people struggle with it. To address this issue, various designs have been developed by individuals and businesses. One approach involves having a car stop parallel to parked cars when it detects an available parking spot. The robot's ability to guide the wheels about 95° in one direction allows the car's wheels to turn and align with the empty parking space [1]. To combat the problem of some drivers parking randomly rather than in designated spots, the start-up Stanley Robotics has introduced a parking robot used in airport parking to park passengers' cars [2]. Many similar initiatives share this concept and aim to achieve the same goal using different tools or designs. Our robot has successfully parked cars in the designated areas, but it has a unique design that sets it apart from other projects. It is also more cost-effective because it relies on wheel speed instead of expensive mechanical components to change its orientation.

4 Design and Implementation

Collisions are common when parking or reversing cars. In our project, we aim to organize parking using a parking robot. The car can be left at the garage door, and the robot can be called using a card. Our robot works by lifting the car and placing it in the parking area neatly. We built our robot in two stages, starting with gathering the hardware components and controlling them using Arduino. Then, we implemented image processing to help the robot determine the best place to park.

4.1 the garage design

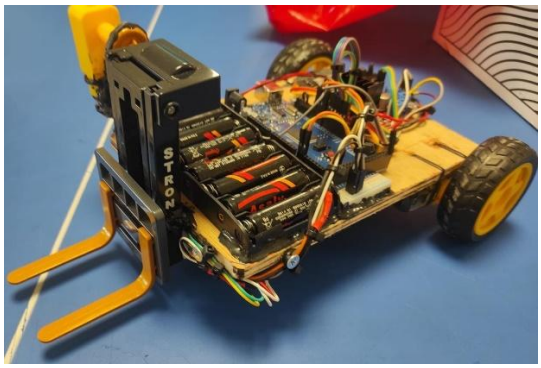

Figure 4.1.1: Robot design

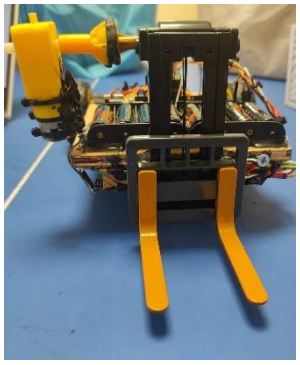

Figure 4.1.2: Robot crane

We have designed our robot-like forklift, constructed from wood and PLA (3D printer material). The body of the forklift is connected to the wheels and serves as a base for all the hardware components, as illustrated in Figure 4.1.2. Attached to the body is a crane, also made from PLA, which features arms that lift the car. The wheels enable the forklift to navigate to the designated location.

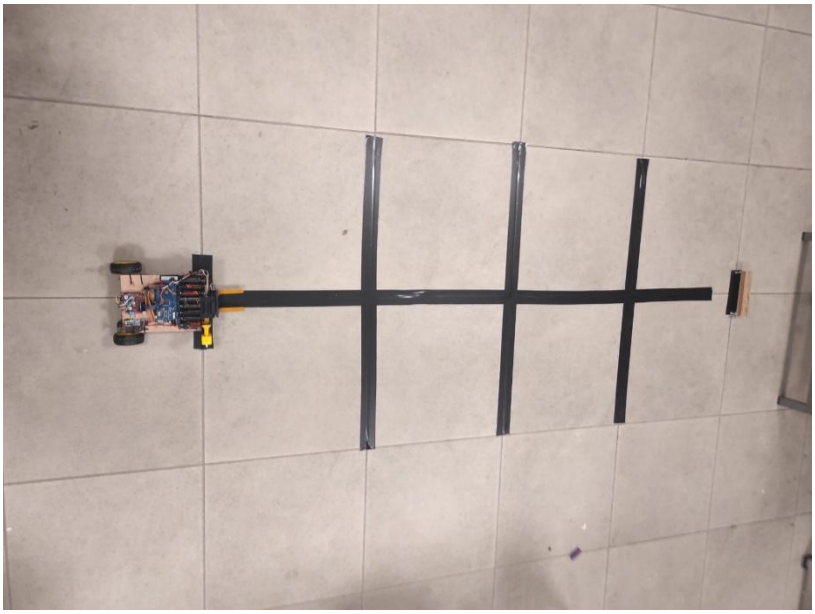

Figure 4.1.3: garage design

We have designed the garages to allow the robot to move easily, starting from the entry where the car waits to the robot's designated area. The main road is marked with a black line, which the robot follows to reach the car and return to its place. Additional black lines cross each garage and are positioned in front of each parking spot, enabling the robot to navigate backwards after parking or retrieving the car, as shown in Figure 4.1.3.

Each garage wall is marked with a picture that helps the robot determine whether the garage is occupied. When the robot sees the picture, it identifies if the garage is empty or full. If a car is parked, it covers the picture, indicating that the garage is occupied. Additionally, each picture includes a unique shape that is specific to each garage, allowing the robot to record where it parked the car. This enables the robot to return and retrieve the car when needed.

4.2 Hardware and Software Specifications

4.2.1 Hardware Components

Hardware component to order the robot:

For this part, we have used an Arduino Uno, HC-05 Bluetooth Module, RC522 RFID Module, and Buzzer.

- **Arduino Uno**

Figure 4.2.1: Arduino Uno

The Arduino Uno microcontroller, based on the ATmega328P, is designed for experimentation. It includes a reset button, a power jack, a USB connector, and fourteen digital input/output pins. In our project, we use the Arduino Uno to develop a robot ordering system. It connects with various components, including a buzzer, Bluetooth, and RFID modules, as shown in Figure 4.2.1.

- **HC-05 Bluetooth Module**

The HC-05 Bluetooth module is used for wireless serial communication. It is an excellent choice for establishing a wireless connection with a phone, as it supports both Master and Slave modes. The module operates with a current of 30 milliamperes and a working voltage ranging from 4 to 6 volts. It has a communication range of up to 100 meters. In this project, we use it to signal the robot when an order is received.

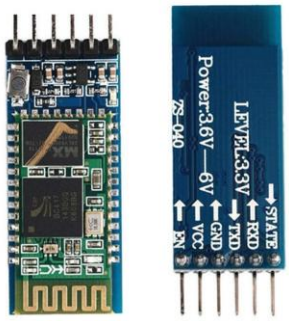

Figure 4.2.2: HC-05 Bluetooth Module

• RC522 RFID Module

The RC522 is a 13.56MHz RFID module based on the MFRC522 controller from NXP Semiconductors. It comes with an RFID card and key fob by default and supports communication via I2C, SPI, and UART. Commonly used in applications such as attendance systems for identifying individuals or items, we utilize it here to instruct the robot to park or retrieve the vehicle.

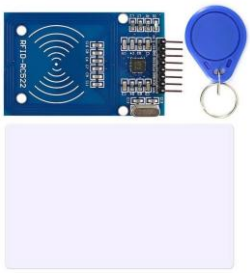

Figure 4.2.3: RC522 RFID Module

• Buzzer

A buzzer or beeper is a type of audio signaling device commonly used in timers and alarm systems, powered by DC voltage. In our system, it is connected to the Arduino and activates when the RFID reader detects a card. This audio signal confirms that the card has been successfully read and that the robot is approaching.

Figure 4.2.4: Buzzer

Hardware component for the robot:

For this part, we have used an Arduino Mega, HC-05 Bluetooth Module, 2 L298N Dual H Bridge, 3 DC Motor, 4 IR Sensor and Buzzer.

- **Arduino Mega**

Figure 4.2.5: Arduino Mega

The Arduino Mega is a microcontroller based on the ATmega328P, designed for experimentation. It features 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a USB connection, a power jack, and a reset button. In our project, we used the Arduino Mega to program and control the robot. It is connected to the Bluetooth module, buzzer, two H-Bridge motor drivers, and IR sensors. Additionally, it communicates with the Raspberry Pi via a serial connection.

- **HC-05 Bluetooth Module**

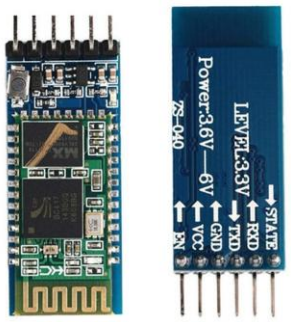

Figure 4.2.6: HC-05 Bluetooth Module

As mentioned earlier, this Bluetooth module is used for wireless communication. In this case, it receives signals from the first module located at the garage entry. When the robot receives this signal, it moves to the entry to either retrieve the car or return it if it is already parked.

- **L298N Dual H Bridge**

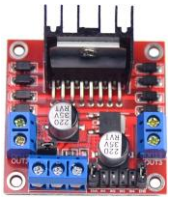

Figure 4.2.7: L298N Dual H Bridge

An H-Bridge is a pulse width modulation (PWM) circuit that allows current to flow in either direction, making it ideal for controlling both the direction and speed of motors. We used one H-Bridge to control the DC motors attached to the robot's wheels, managing its movement. Another H-Bridge was used to control the forklift mechanism, enabling it to lift the car.

- **DC motor**

Figure 4.2.8: DC motor

A DC motor is an electric motor that converts electrical energy into mechanical energy using direct current. In our project, the DC motors operate at 12 volts. We connected two DC motors to the robot's wheels and controlled them via an H-Bridge to manage movement.

- **IR Sensor**

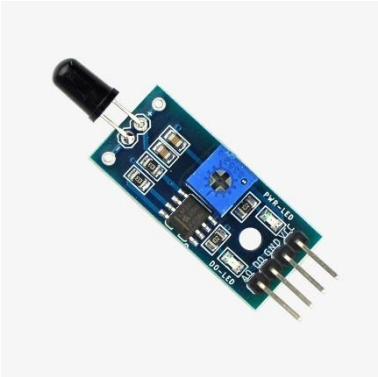

Figure 4.2.9: IR

Infrared (IR) sensors are electronic devices used to detect and measure infrared radiation in their environment. When infrared photons hit a white surface, they reflect back and are captured by photodiodes, causing voltage fluctuations. In contrast, dark surfaces absorb infrared light, preventing it from reflecting, leaving the photodiode without any light. We utilized these sensors to detect the black line, enabling the robot to follow it.

- **Raspberry Pi4 with Camera**

Figure 4.2.10: Raspberry Pi4 with Camera

The Raspberry Pi is an affordable, credit-card-sized computer that can be connected to a TV or computer monitor. It includes a camera connection and is capable of running programs written in Python and Scratch, making it suitable for various applications, including robot programming. In our project, we utilized the Raspberry Pi to enable the robot to observe the garage using a camera.

We installed the operating system on the Raspberry Pi, along with the necessary image processing packages and Visual Studio Code for Python programming. Additionally, we connected an Arduino Mega to the Raspberry Pi via a serial connection, allowing them to interact seamlessly. The Arduino controls the robot, while the Raspberry Pi processes data based on the camera's observations.

4.2.2 Software Components

We began by connecting the robot's body and its component parts to ensure proper functionality. Next, we attached the necessary electronics to program the robot at the garage entrance. We utilized the Arduino IDE to program both the Arduino Mega and the connected components. Finally, we employed a Raspberry Pi and Python to program the robot to detect whether the garage is empty or occupied.

Arduino IDE

The Arduino Integrated Development Environment (IDE) is software that facilitates code editing, compilation, and debugging. Powered by the Java Platform, the Arduino IDE allows users to write and upload code to compatible Arduino boards. In our project, we utilized the Arduino IDE to write the necessary code for the robot's operation.

Raspberry Pi4 Operating System and Visual Studio code:

Raspberry Pi OS (formerly known as Raspbian) is a Debian-based operating system designed for the Raspberry Pi. To enable the camera for our robot to observe the garage, we downloaded and installed this OS. Raspberry Pi OS supports programming languages such as Scratch and Python, allowing us to utilize Visual Studio Code for Python programming. We also installed the necessary libraries for image processing to enhance the robot's capabilities.

4.3 Arduino Code

We use Arduino IDE to write two codes:

- **Scanner code:**

First, we include the library for RC522 RFID which is called <MFRC522.h>toread the card number and then send the value of the card using Bluetooth which uses serial communication, we include <SoftwareSerial.h> library to use software serial nested of using TX, RX pins on the Arduino. For each card number, we send a character nested of sending the whole number.

```ino
#include <SPI.h>
#include <RFID.h>

#define RFID_PIN_SDA 53
#define RFID_PIN_RST 49

RFID rfid(RFID_PIN_SDA, RFID_PIN_RST);

void setup()
{
  Serial1.begin(9600);
  SPI.begin();
  rfid.init();
}

void loop()
{
  if(rfid.isCard())
  {
    if(rfid.readCardSerial())
    {
      String cardId = String(rfid.serNum[0]) + String(rfid.serNum[1]) + String(rfid.serNum[2]) + String(rfid.serNum[3]);

      for(int i = 0 ; i < cardId.length() ; i++)
      {
        Serial1.write(cardId[i]);
      }

      Serial1.write("*");
    }
    rfid.halt();
  }
```

Figure 4.3.1: RFID Bluetooth code

Robot Code Overview

For the second part of our project, we utilized the Arduino Mega. The robot reads values from the IR sensor to navigate from the end of the garage to its entrance. At each parking space, the robot stops to check the parking space number and determine if it is empty.

To assess the availability of the parking space, we used a Raspberry Pi camera. When the robot reaches each parking space, it sends two messages to the Raspberry Pi via USB for serial communication. The first signal informs the Arduino whether the parking space is empty or occupied.

```
Serial.println("Waiting for Raspberry PI feedback");
bool isParkingSpaceChecked = false; // Flag to track parking space check

while (true) {
    if (Serial.available() > 0) {
        raspberryPI_Value = Serial.read(); // 'a' = empty, 'b' = occupied

        Serial.println("Enter park ID (0, 1, or 2):");
        while (Serial.available() == 0); // Wait for Raspberry Pi input
        park_id = Serial.parseInt(); // Read park ID

        // Check parking space status
        switch (raspberryPI_Value) {
            case 'a': // Empty parking space
                Serial.println("Found empty parking space");
                found_parking_space = true;
                isParkingSpaceChecked = true;
                break;

            case 'b': // Occupied parking space
                Serial.println("Moving to the next parking space");
                isParkingSpaceChecked = true;
                break;

            default:
                Serial.println("Unknown status received.");
                break;
        }

        // Exit loop if parking space has been checked
        if (isParkingSpaceChecked) {
            break;
        }
    }
}
```

Figure 4.3.3: check empty parking space

the second signal will tell the Arduino the number of the parking.

```
while (true) {
    Serial.println("What is this park ID? (0, 1, or 2)");
    while (Serial.available() == 0); // Wait for Raspberry Pi input
    short park_id = Serial.parseInt(); // Read park ID starting from 0 to 2

    // Check if the car park ID matches the input park ID
    if (car_park_id == park_id) {
        break; // Exit the loop if park IDs match

    } else if (car_park_id < park_id) {
        moveForwardToParkSpace(0); // Move forward to the park space
        delay(500); // Pause for half a second

    } else if (car_park_id > park_id) {
        moveBackwardToParkSpace(0); // Move backward to the park space
        delay(500); // Pause for half a second
    }
}
```

Figure 4.3.2: check parking number code

4.4 Image processing Code

We wrote the image processing code using Visual Studio Code on the Raspberry Pi. This code activates the Raspberry Pi camera to capture images, enabling the robot to decide where to park the car. The image detection process is divided into two parts.

1. **Parking Space Detection**:

- When the robot stops in front of a garage, the code uses the OpenCV library to analyze the image.
- We convert the image colors to HSV format using the `cv2.cvtColor()` function to facilitate color detection.
- We define the HSV color range that the robot needs to identify. The new image highlights the specified color while converting other colors to black using the function `cv2.bitwise_and(img, img, mask=mask)`.
- After that, we apply filters to enhance the shapes visible in the identified color.
- Using `cv2.findContours`, we contour all shapes in the image and assess which shape has an area less than a specified threshold (this value depends on the distance of the camera from the wall).

2. We start by counting the vertices of the detected shape using the approxPolyDP() function. If we find a shape with four vertices and an area within the specified range, the robot concludes that the garage is full. Conversely, if no such shape is detected, the robot determines that the garage is empty, indicating that there is no car.

3. **Garage ID Detection**:

    - The original image captured before any modifications undergoes additional filtering to identify the garage ID.
    - This process involves detecting other shapes in the image to ascertain the specific garage.

the same image so every garage has a shape. the data stored from the detection of the shapes will send in a serial connection to Arduino.

4.5 Circuit Layout

To control the movement of the robot, we utilized the Arduino Mega to program various components:

1. **Infrared (IR) Sensors**:

   Four IR sensors are connected to pins 8, 9, 10, and 11 on the Arduino Mega. These sensors are used to detect the black color of the line, ensuring that the robot follows the correct path.

2. **H-Bridges**:

   Two H-Bridges are employed and connected to the pulse width modulation (PWM) pins on the Arduino.

   One H-Bridge controls the direction and speed of the two DC motors responsible for the robot's movement.

   The second H-Bridge operates the forklift's DC motor.

3. **Bluetooth Communication**:

   To establish a connection between the Arduino Mega on the robot and the Arduino Uno at the door, we use a Bluetooth module (HC-05). The RX and TX pins of the Bluetooth module are connected to A8 and A9 on the Arduino Mega, respectively.

4. **Power Supply**:

   The Arduino Mega and the H-Bridges are powered using a 12V supply, ensuring adequate power for all components.

5. **Raspberry Pi Connection**:

   Finally, we connect the Raspberry Pi to the Arduino Mega using a USB cable for data communication.

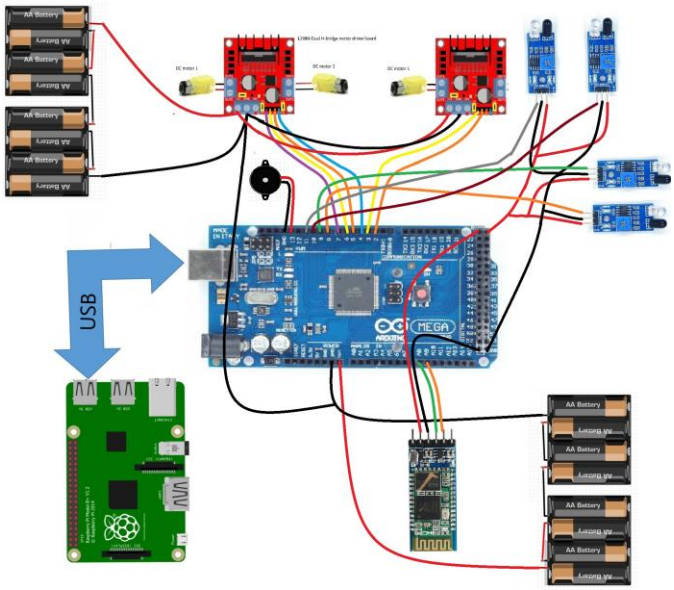

Figure 4.5.1: Robot implementation

To read the user card we use RC522 RFID and send the value of the cardby Bluetooth (HC-05) which is connected to pins 8, 7.

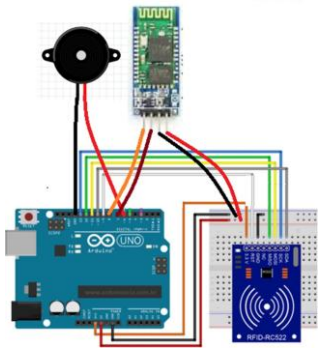

Figure 4.5.2: scanner implementation

5 Results and Analysis

We compiled the source code for each step of the project, ensuring that everything functioned correctly before proceeding to the next phase. This systematic approach greatly assisted us in identifying and resolving any errors.

In the final stage, our project successfully met all requirements, including:

- RFID User Card Reading: The system effectively reads the user card using the RFID scanner.
- Signal Transmission: Upon scanning the card, a signal is sent to the robot via Bluetooth to retrieve the car.
- Parking Management: The robot parks the user's car in an available parking space.
- Retrieval Process: When the user scans their card a second time, the robot efficiently brings the car back to them.

However, we encountered some challenges:

- Forklift Functionality: The forklift occasionally failed to lift the car properly, which required adjustments and troubleshooting.
- Raspberry Pi Accuracy: The Raspberry Pi sometimes provided inaccurate results during image processing, impacting the detection of empty parking spaces.

Step1: The robot go to the entry of the garage to take the car

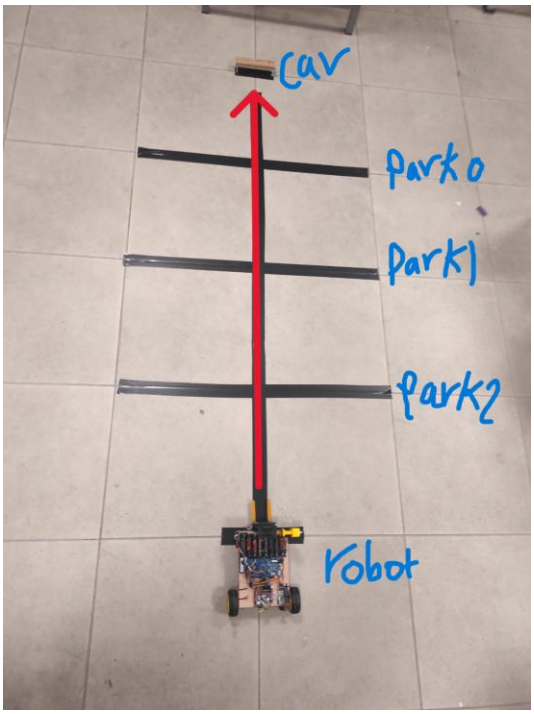

Figure 5.1: the robot moves to bring the car

Step2: the robot carries the car:

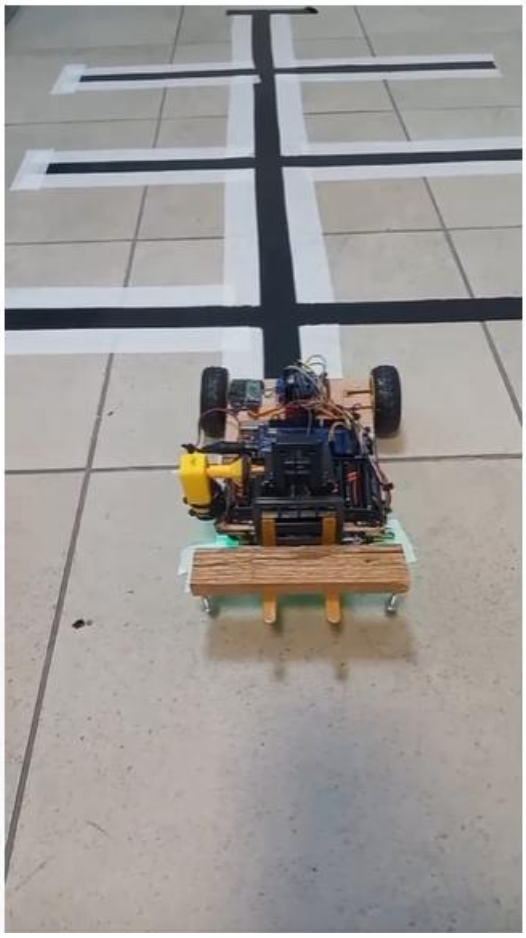

Figure 5.2: forklift lifts the car from the ground

step3: after reach to the first parking space the robot check if there is acar in the parking space if not the robot moves as shown to directing to thecar to an empty parking space.

Figure 5.3: park the car

the same steps done on reverse order to bring the user car back to him

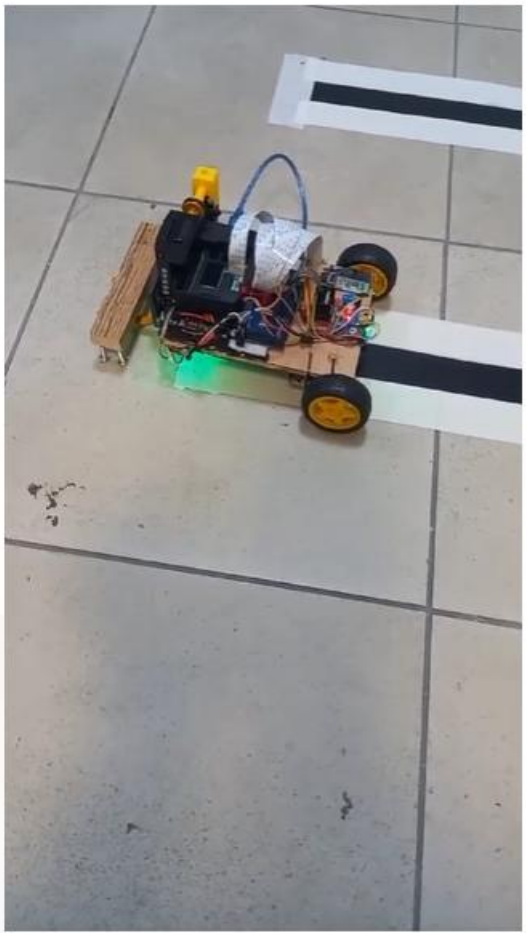

Figure 5.4: park the car

6 Conclusions and Recommendation

In our project, we developed a robot designed to park cars in a garage using unique cards assigned to each user. Users can retrieve their cars by scanning their cards at the garage entrance. To enhance the project, we propose several improvements. First, we could integrate a mobile application to replace the physical cards, which would increase user convenience and streamline the parking process. Additionally, reducing the size of the robot would minimize its footprint, allowing for more efficient use of space and enabling the parking of additional cars within the garage. Finally, implementing Wi-Fi connectivity between the two Arduinos instead of Bluetooth would significantly extend the range of communication, improving overall system reliability and performance.

Bibliography

[1] LYON HAS THE WORLD'S FIRST ROBOTIC PARKING LOT. (2015). Retrieved May 15, 2022, from https://1cars.org/16090-in-lyon-the-first-parking-lot-with-robots-was-opened.html

[2] B. KHAROSHEH and T. JAWABREH, "Parking Robot," AN-NAJAH NATIONAL UNIVERSITY,2022.