Wrocław University
of Science and Technology

Advanced Topics in Robotics Project

Remotely Controlled Mobile Platform by using Ackermann Geometry

Students: Tuğçe Avcu 276817
Marcin Kochalski 275514

Date: 24.01.2025

TABLE OF CONTENTS

Problem Statement and Motivation.....3
Project Photos…….……5
Solution……..……..9
Tools Used on the Project……10
Code Implementation……..11
Python GUI Code……12
Summary……..……14
Appendices……...…….15

Main Part

Problem Statement and Motivation

We started this project with the goal of building a remotely controlled mobile platform using Ackermann steering geometry. Our intention was to create a system that could mimic the steering behavior of real-world vehicles, while being controlled remotely through a graphical user interface (GUI). Initially, our plan was to use advanced technologies like ESP32, ROS, and Zephyr for real-time communication and motor control. However, in the process, major integration challenges arose, which turned us toward a simpler yet effective solution that included Arduino, the L298N motor driver, and the HC-05 Bluetooth module. This change allowed us to focus more on the functional aspects of the platform without getting bogged down by complex software compatibility issues.

This project was born from the desire to gain practical experience in robotics and control systems. Further, we needed to go a step further than theoretical knowledge into the application of the acquired knowledge on some hands-on, real-world problem. Interesting was the Ackermann steering geometry, for its wide application in automotive engineering; that was an opportunity to learn about its implementation in a small-scale, controlled environment.

We divided the work among team members based on our strengths during the beginning period. Some of us put more focus on the hardware aspect, like the setup of the motor and steering mechanism, whereas others worked on software aspects, which included developing the GUI and programming the microcontroller. The first try at integrating ESP32 and ROS resulted in compatibility issues related to wireless communication and real-time control. After discussing it thoroughly, we decided collectively to make the project simpler by using Arduino, which would provide us with a more stable and reliable platform.

In our final solution, Bluetooth communication was achieved by using the HC-05 module. The Python-based GUI was used as the main interface to send commands to the platform for movement, such as forward, backward, left, and right. In our design, the L298N was used for power and direction control of DC motors, and a servo motor for the steering angle based on Ackermann geometry. The platform was tested on a series of trials to ensure it was performing reliably and meeting all functional requirements.

Apart from the development of the fully functional mobile platform, the project taught us many lessons: problem-solving, teamwork, and adaptability. Transitioning from a complex to a simpler system proved to be a turning point to reinforce the importance of flexibility in engineering projects. At the end, we achieved a functional, remotely controlled Ackermann steering platform, and we are proud of the results.

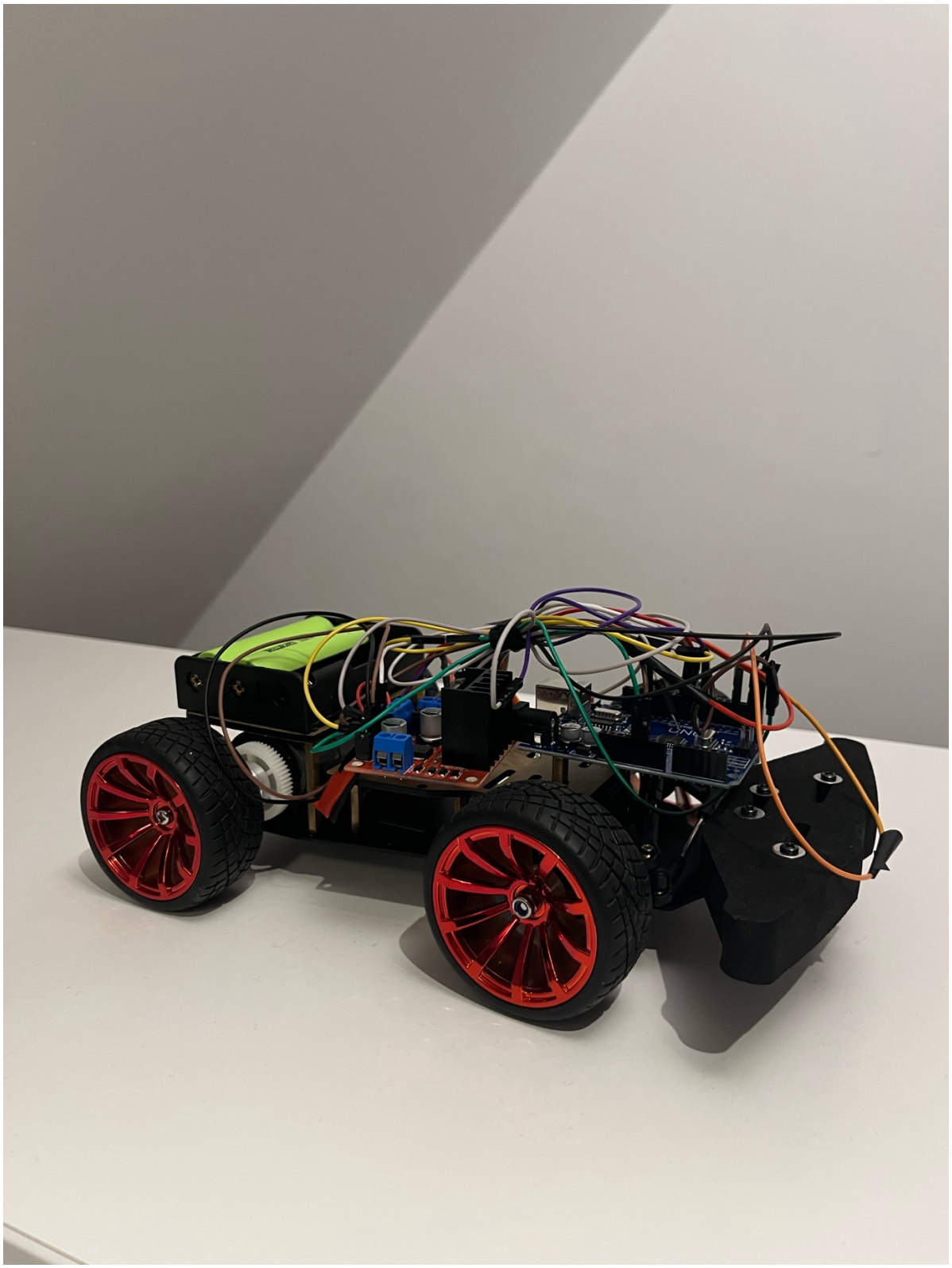


Fig. 1 Left view of the robot

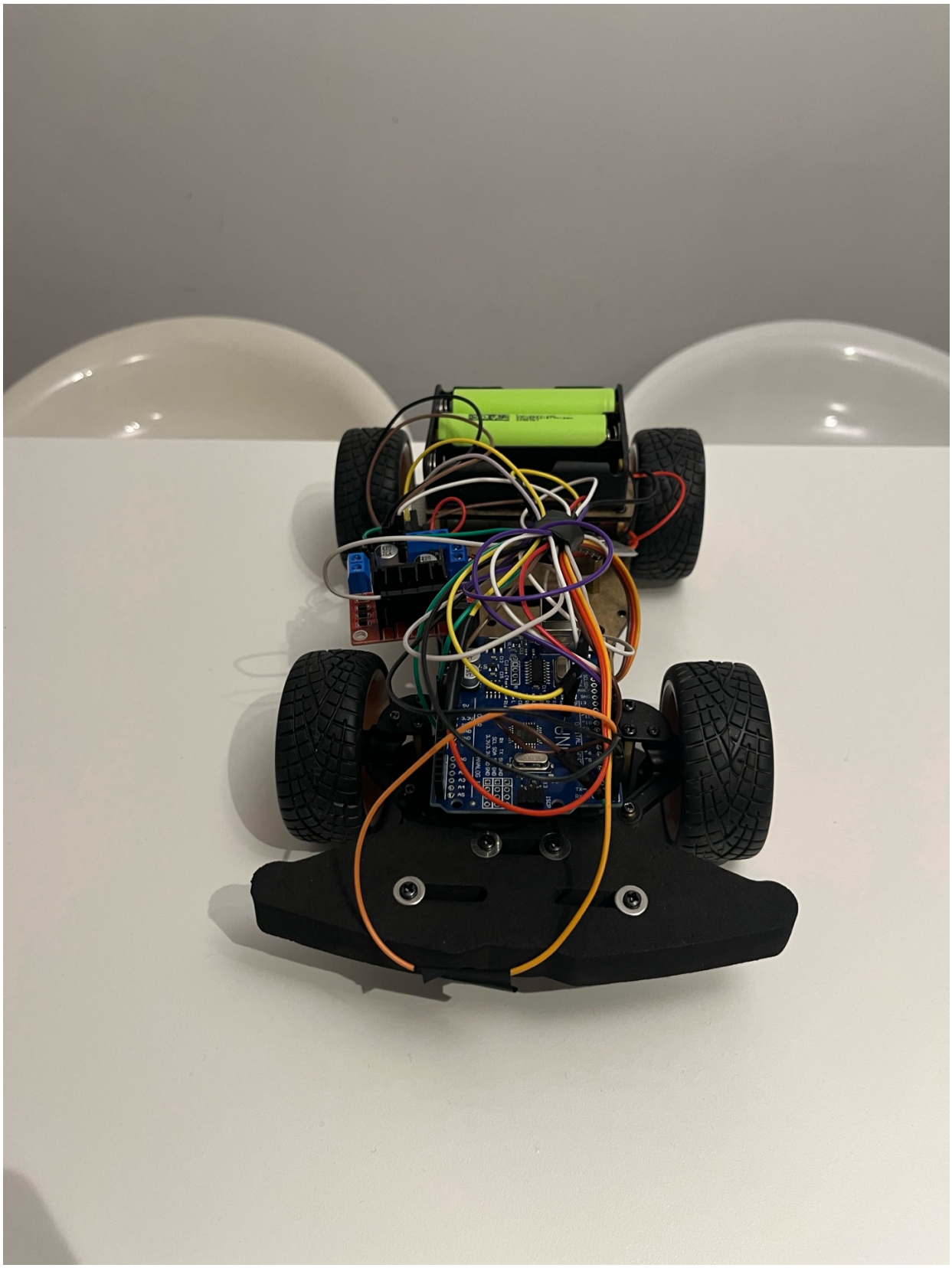


Fig. 2 Front view of the robot

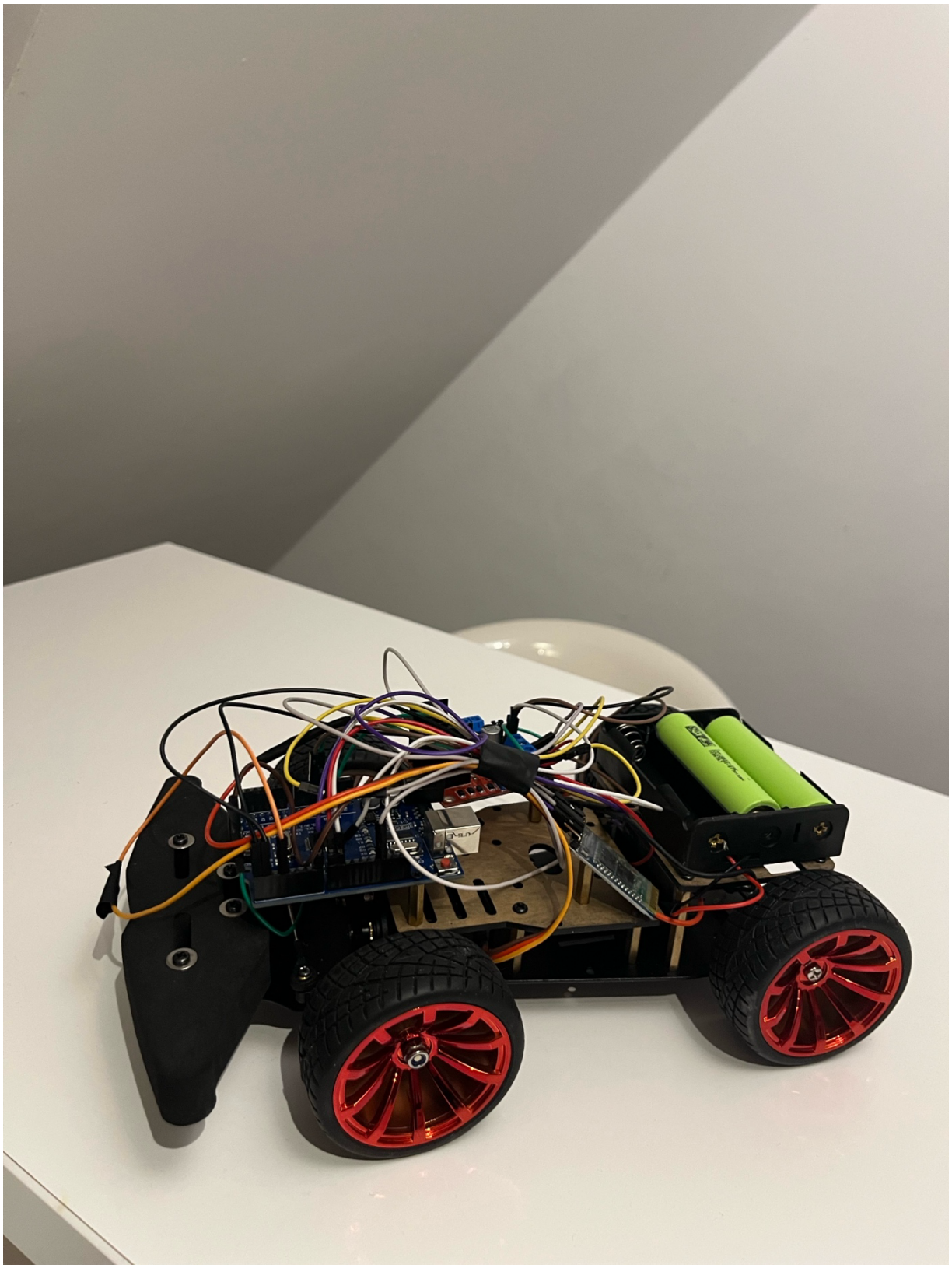


Fig. 3 Right view of the robot

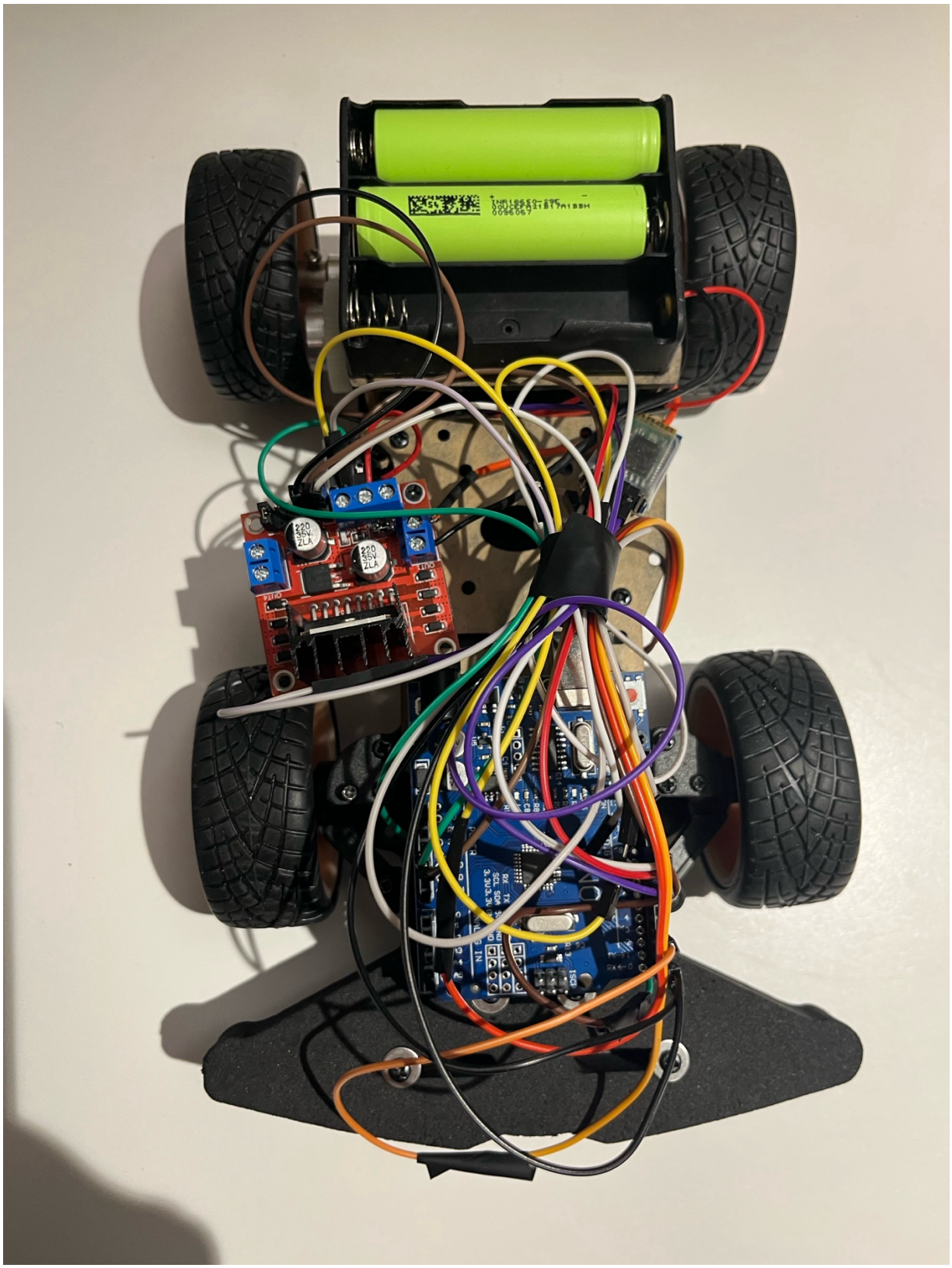


Fig. 4 Upper view of the robot

Solution

Overview of Solution

Our solution focuses on the functional remotely controlled Ackermann steering geometry platform that can make precise movement possible. Main parts include the following:

Ackermann Steering Geometry: Allows the model to make very realistic angle and movement similar to steering in vehicles.

Arduino Microcontroller: A brain for the whole system.

L298N Motor Driver: Interfaces the power supply with control logic to the DC motors of the platform.

HC-05 Bluetooth Module: Keeps the platform wireless to the user interface display.

Python-Based GUI: The GUI will provide an intuitive interface for remote control, letting the user manage speed, steering, and direction with ease.

This condition requires very accurate control over the steering angle. For the latter, we have utilized a servo motor controlled by Arduino. The to-and-fro movements of the platform are realized via an L298N motor driver, through which the amount of power is provided to DC motors. For the communication with the platform in order for wireless control to be achieved, a Bluetooth module will be used within the GUI.

Tools Used on the Project

Software:

Arduino IDE for programming the microcontroller.

Python (with Tkinter) for creating the GUI.

Hardware:

Arduino Uno microcontroller.

HC-05 Bluetooth module for communication.

L298N motor driver for controlling motor power.

Servo motor for steering.

DC motors for platform movement.

Details of the Solution

The solution has gone through significant evolvement throughout the project. First, we intended to use ESP32 with ROS and Zephyr for higher-order communication and control capabilities. However, difficulties in system integration shifted us towards Arduino-based control with Bluetooth communication.

The final setup involves:

Communication: The HC-05 module transmits commands from the GUI to the Arduino. The Arduino interprets these commands and executes the corresponding actions, such as moving forward, backward, or turning.

Motor Control: L298N-the motor driver for DC motors is applied in order to give speed and direction input from Arduino.

Steering Mechanism: Servo motor for giving steering angle for Ackermann geometry in order to take proper and smooth turns.

Code Implementation

Arduino Code

The Arduino code was designed to process commands received via Bluetooth and control the motors accordingly:

```
#include <Servo.h>

Servo steeringServo;
int motorPin1 = 5;
int motorPin2 = 6;
int enablePin = 9;

void setup() {
  steeringServo.attach(3);
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(enablePin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available()) {
    char command = Serial.read();
    if (command == 'F') {
      digitalWrite(motorPin1, HIGH);
      digitalWrite(motorPin2, LOW);
      analogWrite(enablePin, 200); // Speed control
    } else if (command == 'B') {
      digitalWrite(motorPin1, LOW);
      digitalWrite(motorPin2, HIGH);
      analogWrite(enablePin, 200);
    } else if (command == 'L') {
      steeringServo.write(45); // Turn left
    } else if (command == 'R') {
      steeringServo.write(135); // Turn right
    } else if (command == 'S') {
      digitalWrite(motorPin1, LOW);
      digitalWrite(motorPin2, LOW);
    }
```

The Arduino initializes communication with the HC-05 Bluetooth module.
It continuously listens for commands from the user interface (GUI).

Python GUI Code

The GUI code enables the user to send commands to the platform using a graphical interface:

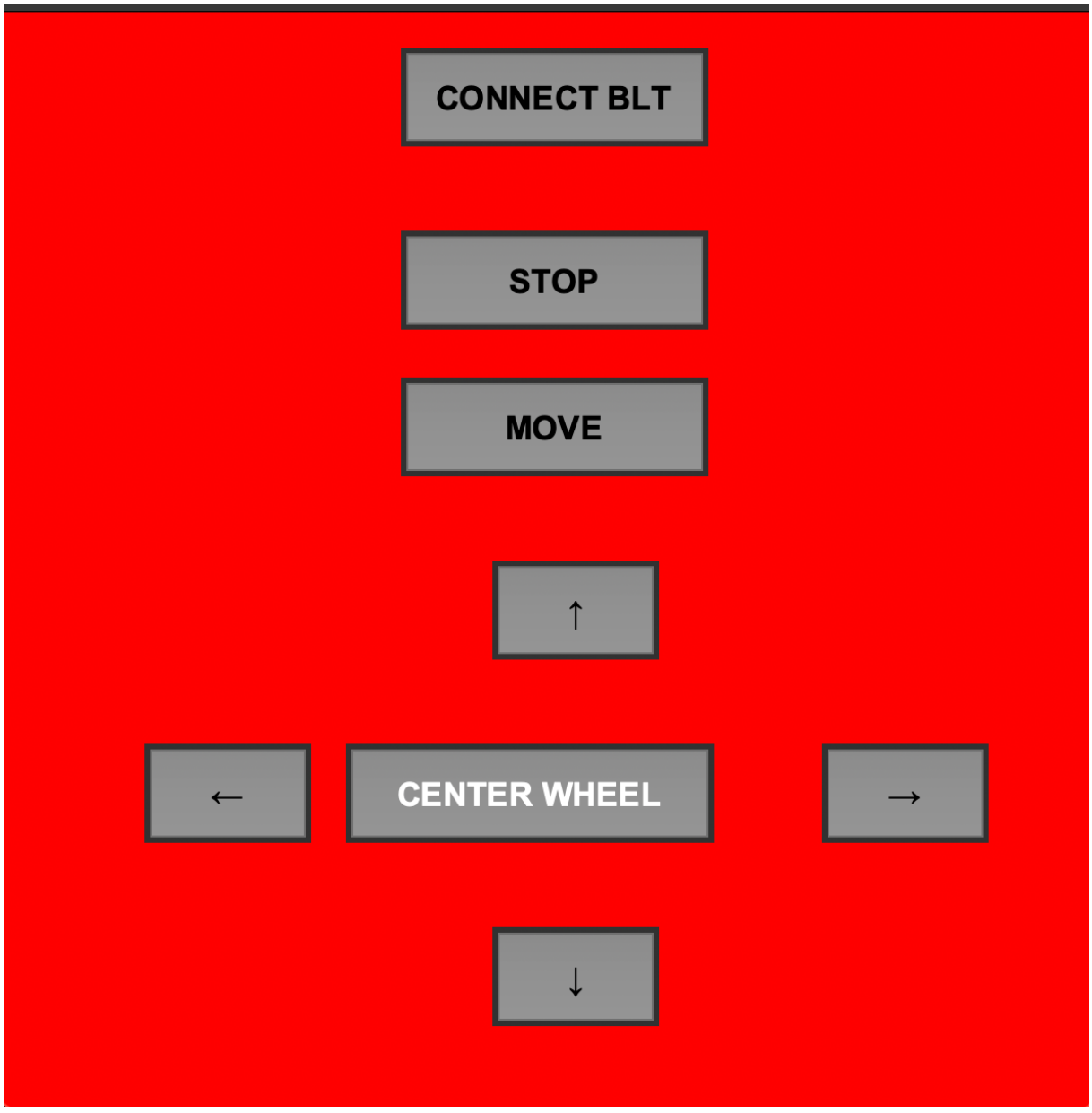
appcode › create_gui

```python
import tkinter as tk
from tkinter import messagebox
import serial
import time

# Bluetooth port ve baud hızı ayarları
BLUETOOTH_PORT = "COM6"  # Doğru port adresi
BAUD_RATE = 9600

# Global seri bağlantı
bt_connection = None

# Bluetooth'a bağlanma fonksiyonu
def connect_bluetooth():
    global bt_connection
    try:
        bt_connection = serial.Serial(BLUETOOTH_PORT, BAUD_RATE)
        time.sleep(2)  # Bağlantı kurulum süresi
        messagebox.showinfo("Connected", f"HC-05'e {BLUETOOTH_PORT} connected!")
    except Exception as e:
        messagebox.showerror("Error of Connection", f"Bluetooth is not connected: {e}")

# Komut gönderme fonksiyonu
def send_command(command):
    global bt_connection
    if bt_connection and bt_connection.is_open:
        try:
            bt_connection.write((command + "\n").encode())  # Komutu gönder
            print(f"Command sent: {command}")
        except Exception as e:
            messagebox.showerror("error", f"Komut gönderilemedi: {e}")
    else:
        messagebox.showerror("no connection", "Lütfen önce Bluetooth bağlantısını kurun!")

# Tkinter arayüzü
def create_gui():
    root = tk.Tk()
    root.title("HC-05 Kontrol Paneli")
    root.geometry("600x600")
    root.configure(bg="red")

    button_font = ("Arial", 18, "bold")

    # Bluetooth bağlantı düğmesi
    btn_connect = tk.Button(root, text="CONNECT BLT", font=button_font, bg="green", fg="black", width=12, height=2,
                            command=connect_bluetooth)
    btn_connect.place(x=220, y=20)


    # Komut butonları
    btn_stop = tk.Button(root, text="STOP", font=button_font, bg="blue", fg="black", width=12, height=2,
                         command=lambda: send_command("STOP"))
    btn_stop.place(x=220, y=120)

    btn_start = tk.Button(root, text="MOVE", font=button_font, bg="orange", fg="black", width=12, height=2,
                          command=lambda: send_command("START"))
    btn_start.place(x=220, y=200)

    btn_forward = tk.Button(root, text="↑", font=button_font, bg="yellow", fg="black", width=5, height=2,
                            command=lambda: send_command("FORWARD"))
    btn_forward.place(x=270, y=300)

    btn_left = tk.Button(root, text="←", font=button_font, bg="yellow", fg="black", width=5, height=2,
                         command=lambda: send_command("LEFT"))
    btn_left.place(x=80, y=400)

    btn_right = tk.Button(root, text="→", font=button_font, bg="yellow", fg="black", width=5, height=2,
                          command=lambda: send_command("RIGHT"))
    btn_right.place(x=450, y=400)

    btn_backward = tk.Button(root, text="↓", font=button_font, bg="yellow", fg="black", width=5, height=2,
                             command=lambda: send_command("BACKWARD"))
    btn_backward.place(x=270, y=500)

    # Adding Center Wheel button (move it down to avoid overlap)
    btn_center = tk.Button(root, text="CENTER WHEEL", font=button_font, bg="purple", fg="white", width=15, height=2,
                           command=lambda: send_command("CENTER"))
    btn_center.place(x=190, y=400)  # Adjust y to avoid overlap with other buttons

    root.mainloop()

if __name__ == "__main__":
    create_gui()
```

The Python GUI uses the serial library to establish a connection with the HC-05 Bluetooth module.
The user interacts with buttons in the GUI to send commandss.

Each button triggers a function that sends the corresponding command to the Arduino.

The Arduino processes these commands to control the motors and servo, executing the desired movement.



Tests and Results

Functionality Tests: Verified movements.

Bluetooth Range: Stable communication within a 10-meter range.

Stability Tests: Continuous operation for over an hour without failure.

GUI Usability: Commands executed seamlessly with no noticeable delay.

Summary

Evaluation of the Project

The project successfully demonstrated the development of a remotely controlled mobile platform using Ackermann steering geometry. Despite initial challenges, the final system performed reliably and met the functional requirements.

Possible Enhancements

Better functionalities about moving the robot left-right from keyboard.
Extended Bluetooth range for broader usability (maybe with more advanced bluetooth module/board)
Adding sensors for obstacle detection and navigatoin.
Enhanced GUI features for better control.

Appendices

Project Plan and Deviations

The original plan involved using ESP32 and ROS. However, due to integration difficulties, the project was restructured to use Arduino, HC-05, and a Python GUI.

Tasks and Distribution

Hardware Design: Managed by Marcin Kochalski

Software Development: Managed by Tuğçe Avcu

Testing and Debugging: Collaborative effort.

Report: Collaborative effort.