Wrocław University
of Science and Technology

Advanced Topics in Robotics Project Report

Remotely Controlled Mobile Tracked Platform

Faculty of Electronics, Photonics and Microsystems
Advanced Topics in Robotics

Students:
Azra Selvitop (276772)
Deniz Yildiz (276824)
Yiğit Temiz (276710)

Project Supervisor:
mgr inż. Roberto Orozco

Submission Date:

Grade: .....

Contents

1 Introduction

The "Remotely Controlled Mobile Tracked Platform" is a tank-like robot designed for precise remote operation on smooth and flat surfaces. This project aims to develop a stable and functional system for tasks such as inspection and observation in areas where human presence is inconvenient or unnecessary. The platform is controlled via Bluetooth using an Arduino-based system, allowing smooth and responsive movement through motorized tracks.

1.1 State-of-the-Art:

Tracked robotic platforms are widely used in industries such as defence, search-and-rescue, and autonomous logistics. Most modern systems integrate advanced sensors, AI-based navigation, or autonomous path planning. However, these solutions often require expensive hardware and complex software implementations.

1.2 Uniqueness of Our Solution:

Unlike high-cost industrial systems, our project presents a low-cost, simplified alternative for educational and experimental purposes. By focusing on Bluetooth-based remote control and manual operation rather than autonomy, we demonstrate how a functional mobile robot can be developed with minimal resources, making it an accessible and practical system for academic applications.

The development process focused on integrating durable hardware components to ensure stability. While initial plans included sensor-based automation, limitations in available power pins led to a streamlined design prioritizing motor control and wireless communication. Extensive testing ensured the platform’s mobility, responsiveness, and overall reliability.

Despite challenges related to power management, hardware compatibility, and time constraints, the project successfully delivered a fully operational prototype. This system demonstrates the potential of practical remote-controlled robotics and serves as a foundation for future enhancements, such as sensor integration or expanded functionalities. The project provided hands-on experience in motor control, wireless communication, and hardware integration, fostering essential engineering skills.

1.3 Team Structure

The team is structured as follows:

Scope	Member
Hardware Design and Integration	Azra Selvitop
Software Development (GUI & Bluetooth)	Yiğit Temiz
Documentation and Testing	Deniz Yıldız

Each person is responsible for their section. Once completed, it was integrated with the rest.

2 Solution

2.1 Idea of the Solution

The solution involves a remotely controlled tank-like robot that can be operated via a graphical user interface (GUI) over a Bluetooth connection. The robot is designed to be modular and cost-effective, using an Arduino Uno as the main controller and an L298N motor driver to control the motors. The GUI, developed in Python, sends movement commands (forward, backward, left, right) to the robot via Bluetooth, which are then executed by the motors. The system is powered by a 12V battery pack, ensuring portability and independence from external power sources.

2.2 Tools Used

In addition to extensive research online done by every member of the team, several components, libraries and software were used to finalize the project. The following subsections will cover each part.

2.2.1 Components

The system consists of the following components:

- **Arduino Uno PCB**: The main microcontroller responsible for processing commands from the GUI and controlling the motors via the L298N motor driver.
- **L298N Motor Driver**: Used to control the two motors of the Black Gladiator chassis. It receives signals from the Arduino and drives the motors accordingly.
- **HC-05 Bluetooth Module**: Enables wireless communication between the GUI and the Arduino. It receives commands from the GUI and transmits them to the Arduino.
- **Black Gladiator Tracked Robot Chassis**: The physical frame of the robot, equipped with two 33 GB 520 18.75 motors that provide movement.
- **12V Battery Pack**: Powers the entire system, ensuring the robot can operate independently.
- **Jumper Wires**: Used to connect the components (e.g., Arduino to motor driver, motor driver to motors).

2.2.2 Arduino Libraries

The Arduino code uses the following functions for communication and motor control:

- **Serial.begin(9600)**: Initializes serial communication with the HC-05 Bluetooth module at a baud rate of 9600.
- **Serial.available()**: Checks if there is data available to read from the serial buffer.
- **Serial.readStringUntil()**: Reads the incoming serial data as a string until a specific character is received.

2.2.3 Software for End-user

The solution below were used for coding and remote control.

- Visual Studio Code

Libraries used:

- **Tkinter:** We use tkinter to create a GUI for controlling the robot. It provides buttons and an intuitive layout for user interaction.
- **Messagebox:** This is used to show pop-up notifications (e.g., connection success or errors).
- **Serial:** The serial library handles communication with the HC-05 Bluetooth module, allowing us to send and receive commands.
- **Threading:** We use threading to handle incoming Bluetooth messages without freezing the GUI.

2.3 Details of the Solution

This section will explain all the details of our robotic system. It is divided into two parts: hardware and software

2.3.1 Hardware

First, refer please to figure 2.1 for flowchart of the mechanical/electronic elements.

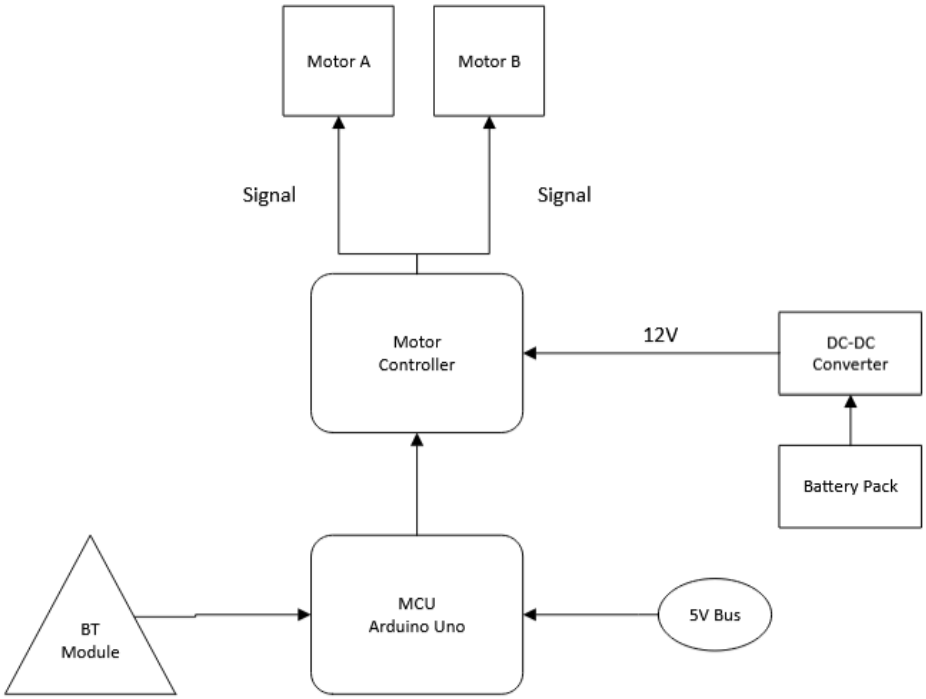

Figure 2.1: Block Diagram of robot's hardware.

The power supply uses three 18650 Li-ion cells connected in series to achieve the higher voltage required for the motors. The voltage was than adjusted to 12V by DC-DC converter. Regulated output was connected to the L298N motor controller board which additionally provided build in 5V regulator which output was used to power all the other components through common 5V bus. Additionally common ground bus was created with stabilizing capacitors and protection diodes added for more reliable operation and protection. The direction of movement and speed of the motors is controlled by 6 pins from MCU to the motor controller. The speed is controlled by PWM signal to enable pins and direction to the specific signal according to following table from datasheet, in figure 2.2.

The motor controller's operation is defined in Table 2.2, which details the input combinations required for different motor actions.

Function	ENA	ENB	IN1	IN2	IN3	IN4	Explanation
Move Forward	150	150	H	L	L	H	Left motor moves forward, right motor moves forward.
Move Backward	150	150	L	H	H	L	Left motor moves backward, right motor moves backward.
Turn Left	150	150	L	H	L	H	Left motor moves backward, right motor moves forward.
Turn Right	150	150	H	L	H	L	Left motor moves forward, right motor moves backward.
Stop Motors	0	0	L	L	L	L	All motors are stopped by cutting power and setting direction pins to Low.
Start Motors	150	150	X	X	X	X	Motors start with default speed, maintaining previous direction settings.

Figure 2.2: Motor Input Function Table

ENA: Controls Left Motor, **ENB:** Controls Right Motor, **IN1:** Left forward, **IN2:** Left Backward, **IN3:** Right Forward, **IN4:** Right Backward, **L:** Low, **H:** High, **X:** No change

2.3.2 Software

The software component of our project is designed to facilitate seamless communication and control of the tank-like robot using a dual-port Bluetooth connection. The system includes a Python-based GUI for user-friendly interaction and control commands, as well as a backend logic that enables smooth communication with the robot. Before achieving Bluetooth connectivity, we initially tested the motors and GUI functionality using a USB-B cable connected to the Arduino. Although this early stage did not allow for remote control, it was crucial for verifying motor functionality and understanding the logic behind movements and turns. These tests gave us valuable insights into motor directionality and the role of PWM signals, which were later applied to the wireless implementation.

The GUI allows the user to send commands such as forward, backward, left, and right, which are transmitted via Bluetooth to control the robot's movements in real time. It also includes additional functionality for stopping and restarting the motors. The code uses threading to handle incoming data from the robot without interrupting the GUI, ensuring a smooth user experience. This modular design made the software reliable, easy to extend, and robust enough to support the remote operation of the robot

```
1 | def connect_bluetooth():
2 |     global connection_port, command_port
3 |     try:
4 |         connection_port = serial.Serial("COM4", 9600, timeout=1)
5 |         command_port = serial.Serial("COM3", 9600, timeout=1)
6 |         messagebox.showinfo("Success", "Connected to HC-05 on COM3
(connection) and COM4 (commands)!")
7 |     except Exception as e:
8 |         messagebox.showerror("Error", f"Could not connect: {e}")
```

This function connects to the Bluetooth module on two ports: COM3 (for reading data) and COM4 (for sending commands).
If successful, it shows a success message; otherwise, it displays an error.

```
1 | def send_command(command):
2 |     global command port
3 |     if command_port and command_port.is_open:
4 |         try:
5 |             command_port.write((command + "\n").encode('utf-8'))
6 |             print(f"Command sent: {command}")
7 |         except Exception as e:
8 |             messagebox.showerror("Error", f"Failed to send command:
{e}")
9 |     else:
10|         messagebox.showerror("Error", "Please connect to Bluetooth
first!")
```

This function sends commands like FORWARD or STOP to the robot via the command port.

```
def read_messages():
    global connection_port
    while True:
        if connection_port and connection_port.is_open:
            try:
                if connection_port.in_waiting > 0:
                    message = connection_port.readline().decode('utf-
8').strip()
                    print(f"Received message: {message}")
            except Exception as e:
                print(f"Error reading message: {e}")
                break
```

This function continuously listens for messages from the robot on the connection_port (COM3).
Messages are read, decoded, and printed in the console.

```
def start_reading():
    thread = threading.Thread(target=read_messages, daemon=True)
    thread.start()
```

A new thread is created for read_messages(), allowing the GUI to remain responsive while monitoring Bluetooth data in the background.

```
def create_gui():
    root = tk.Tk()
    root.title("HC-05 Dual-Port Control Panel")
    root.geometry("450x600")
    root.configure(bg="#2E3440")
```

We create a main window with a professional design (#2E3440 for background color) and set its title and size.

```
btn_forward = tk.Button(control_frame, text="Forward", command=lambda:
send_command("FORWARD"))
btn_backward = tk.Button(control_frame, text="Backward", command=lambda:
send_command("BACKWARD"))
btn_left = tk.Button(control_frame, text="Left", command=lambda:
send command("LEFT"))
btn_right = tk.Button(control_frame, text="Right", command=lambda:
send_command("RIGHT"))
```

Buttons are provided for controlling the robot's movement. Each button sends a specific command (Forward, Backward, etc.).

```
btn_start = tk.Button(bottom_frame, text="Start", command=lambda:
send_command("START"))
btn_stop = tk.Button(bottom_frame, text="Stop", command=lambda:
send_command("STOP"))
btn_connect = tk.Button(root, text="Connect", command=lambda:
[connect_bluetooth(), start_reading()])
```

Start/Stop: These buttons control the robot's motors.
Connect: This button initializes Bluetooth communication and starts the message-reading thread.

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

The program starts by launching the GUI when executed.

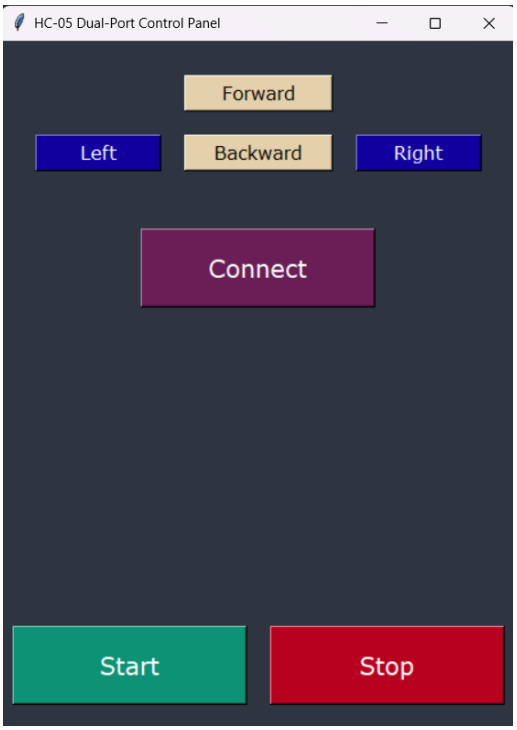

Figure 2.4.: Output Control Panel

2.4 Tests and Results

This section describes the challenges faced during the project and the performance of the robot in its final state.

5V Pin Limitation -- A lack of sufficient 5V pins on the Arduino Uno prevented the integration of additional components, such as a proximity sensor. This limitation required us to focus on motor control and Bluetooth communication instead.

ESP32 Usage Issue-- Initially, we planned to use the ESP32 microcontroller for enhanced functionality. However, due to insufficient GPIO knowledge and lack of documentation during early stages, significant time was spent troubleshooting its configuration. Eventually, we switched to Arduino Uno to avoid further delays.

Bluetooth Module Instability-- We initially used the SPP-C Bluetooth module, but it proved unreliable, with frequent connection drops during tests. To address this, we switched to the HC-05 Bluetooth module, which provided a more stable and consistent connection. The HC-05's ease of use and reliable dual-port operation (for sending commands and receiving responses) significantly improved the system's performance.

Hardware Placement Challenges-- Ensuring that the components were securely mounted on the robot's chassis was a challenge. During sharp turns, there was a risk of components becoming unstable or dislodged. Extra effort was made to design a more stable hardware layout to minimize this issue.

Motor Speed and Battery Life-- Despite setting the motor speed to a PWM value of 150 (a relatively low speed), the robot moves faster than anticipated. This may affect precision in future applications and should be addressed in further adjustments. Additionally, the robot has a short battery life due to the 12V power consumption of the motors and Bluetooth module.

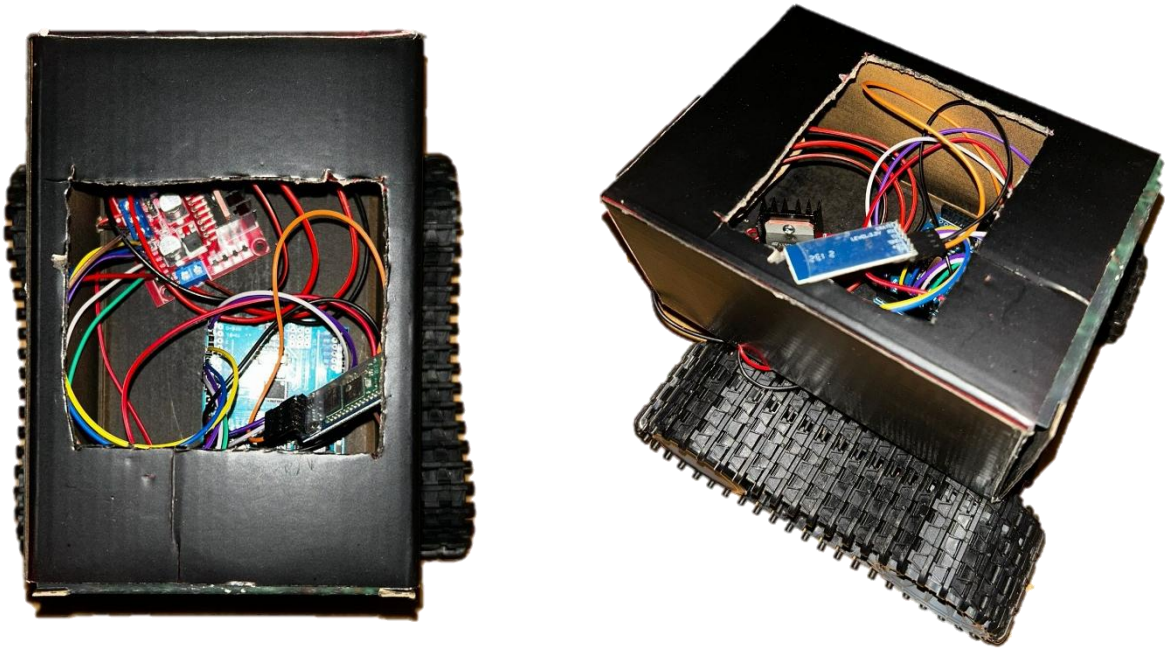

Figure 2.5: Finished Robot View

3 Summary

Overall, the project was a success, as the fundamental objectives were achieved, and the robot operates reliably in its final state. The general concept of creating a remotely controlled tank-like platform was confirmed, and the team worked effectively to overcome challenges. While each team member had autonomy to work independently, periodic meetings were held to address issues, brainstorm solutions, and test the system. Coordination was strong, with two or all three members meeting in person frequently to ensure progress was on track.

The milestones and deadlines provided structure and motivation, keeping the project moving forward. If additional time had been available, we could have addressed further enhancements, such as optimizing motor speed control, extending battery life, and integrating sensors for additional functionality. Cooperation with the project supervisor was highly productive and encouraging, as they offered valuable guidance and timely advice, helping us navigate the technical challenges we faced.

In conclusion, the project provided valuable hands-on experience, improved technical skills, and demonstrated the potential of robotics for practical applications. The experience was both educational and rewarding.

3.1 Applicability and Limitations of Project

Functional Assumptions	Design Assumptions
The robot is primarily designed for remote manual control via Bluetooth.	Requires stable Bluetooth connection with a PC running the GUI to operate.
Only usable on smooth and flat surfaces.	The chassis and motor system are not equipped for uneven terrain or outdoor environments.
Cannot operate autonomously; user control is required.	Limited to manual commands, as sensors for autonomy (e.g., proximity sensors) are not integrated.
Requires periodic recharging due to short battery life.	Uses a 12V power supply with limited runtime, especially during prolonged usage or high-speed operation.

Table 3.1: Functional and design assumptions of the robot.

3.2 Possible Enhancements

Power Management-- Replacing the battery with a higher-capacity option to extend operating time.

Motor Speed Control—Implementing more precise PWM control or adjustable speed settings to improve movement precision.

Autonomy—Integrating proximity sensors or other environmental sensors to enable autonomous navigation.

Terrain Adaptability-- Upgrading the chassis and motors for better performance on uneven surfaces or outdoor environments.

Wireless Communication-- Exploring alternatives like Wi-Fi modules or enhanced Bluetooth modules for a longer range and more stability.

User Interface-- Adding more intuitive features to the GUI, such as live feedback on battery status or motor activity.

4 Appendix

This chapter outlines the original project schedule and task distribution, as well as an evaluation of how closely the project followed these plans. While the project was successful, significant deviations from the schedule and initial plan occurred due to unforeseen challenges. These deviations, however, provided valuable lessons in flexibility, problem-solving, and adapting to new constraints.

4.1 Original Project Plan

[go to the next page]

TERMINATOR — Project Charter

Project description

The "Remotely Controlled Mobile Tracked Platform" is a tank-like mobile platform designed to be remotely operated on smooth, normal surfaces. This project aims to develop a reliable and functional platform that can be controlled from a distance for various practical applications, such as remote observation and monitoring.

The platform will include a robust tracked system for stable movement on flat surfaces, with a remote-control interface enabling operators to guide the platform's movements and control its functions. It will also have the capacity to carry small payloads of equipment such as cameras or sensors, allowing it to be used in tasks such as inspection or observation in areas where direct human interaction is inconvenient or unnecessary.

This project will leverage key principles of electronic and computer engineering, focusing on motor control, wireless communication, sensor integration, and power management. The objective is to create a versatile and expandable system for future enhancements and specialized uses.

TERMINATOR – Project Schedule

Schedule and Milestones

- Milestone 1: Initial Meeting and Platform Assembly Review: November 22, 2024

  Objective: Present the platform construction progress, discuss any challenges, and confirm readiness for movement control development.

- Checkpoints:

  - October 25, 2024 - Platform Assembly: Start assembly of tracks, motors, and main body structure.
  - November 8, 2024 - Power System Setup: Secure batteries and check power distribution to components.
  - November 8, 2024 - Motor and Microcontroller Connection: Complete wiring between the ESP32, motor drivers, and proximity sensor.
  - November 22, 2024 - Initial Code Test: Run initial code to evaluate basic movement functions and confirm obstacle detection works.

- Milestone 2: Final Integration and Testing Review: December 20, 2024

  Objective: Demonstrate completed remote control functionality, refined sensor responses, and any additional features.

- Checkpoints:

  - December 5, 2024 - Remote Control Setup: Configure Bluetooth or Wi-Fi for remote control functionality.
  - December 8, 2024 - Obstacle Detection Tuning: Adjust proximity sensor sensitivity.
  - December 13, 2024 - Additional Features Integration: Add camera, LED indicators, or buzzer (if applicable).
  - December 15, 2024 - Full System Test: Conduct comprehensive testing of remote control, sensors, and any additional features.

The project adhered to the original milestones, but several tasks were delayed due to unforeseen challenges, including switching from ESP32 to Arduino Uno, replacing the unstable SPP-C Bluetooth module with HC-05, addressing 5V pin limitations that prevented sensor integration, and troubleshooting motor speed control issues. Despite these deviations, the team successfully reached all objectives by adapting the plan and prioritizing critical tasks.