KU LEUVEN

FACULTY OF ENGINEERING TECHNOLOGY

GROUP T LEUVEN CAMPUS

Autonomous robot for efficient traversal and mapping of a maze-like environment

RACE5

Pieter NOUWEN

Alexandru CIOCĂU

Fil DAEMS

Coach: Dimitri VARGEMIDIS

Paper submitted in partial fulfillment of the requirements of Engineering Experience 3 - Electronics and ICT Engineering

Academic Year 2024-2025

KU LEUVEN

Faculty of Engineering Technology Group T Leuven Campus — Paper submitted in partial fulfillment of the requirements of Engineering Experience 3 - Electronics and ICT Engineering
Academic Year 2024-2025

Autonomous robot for efficient traversal and mapping of a maze-like environment

RACE5

Daems Fil, Ciocău Alexandru, Nouwen Pieter

Bachelor in Engineering Technology: Electronics and ICT Engineering, Faculty of Engineering Technology, Group T Leuven Campus, Andreas Vesaliusstraat 13, 3000 Leuven, Belgium

Coach: Dimitri VARGEMIDIS

Engineering Technology: Electronics and ICT Engineering, Faculty of Engineering Technology, Group T Leuven Campus, Andreas Vesaliusstraat 13, 3000 Leuven, Belgium, dimitri.vargemidis@kuleuven.be

ABSTRACT

This paper presents the design, development, and evaluation of an autonomous racing robot, that is cost-effective and efficient. The primary objective was to create a functional and competitive robot capable of navigating an unknown track using onboard sensors and intelligent algorithms. Key components include a 12V-powered, differential-drive propultion-unit, an array of positional and proximity sensors (IR, TOF, IMU), and real-time communication between the robot, a remote controller and a Python-based control server (nRF24, WiFi/UDP). The robot maps its environment during the first laps and optimizes its performance in the subsequent laps. A web-based interface visualizes live telemetry for analysis and reporting. The implementation emphasizes low-cost, custom, in-house designed components without compromising core performance. Final testing demonstrated reliable navigation and competitive lap times, validating the system's design and control strategies.

1 INTRODUCTION

Self-driving vehicles are on a rapid rise. Taxi companies are developing driverless taxis revolutionizing transportation. Warehouse managers are trying to implement fully autonomous robots taking care of the hard manual labor. These innovations are only just a few of the many examples of how autonomous vehicles will have a big impact on our life in the near future. According to Ren and Xia (2023), rapid technological developments in artificial intelligence and the emergence of new sensors are turning these futuristic visions into a reality.

The objective of this project is to develop a cost-effective robot that drives autonomously. Its design will be tailored to meet specific functional requirements, which will be further elaborated upon in the requirements analysis. This project aims not only to deliver a functional prototype but also to provide insights and inspiration for future developers and fabricators working on similar designs.

This paper will guide you through our research and insights as we make a prototype of such a self-driving robot. The following topics will be discussed: Requirements analysis, design and materials, implementation, evaluation, discussion and a conclusion.

1.1 Requirements analysis

To simulate and validate the prototype, a test setup has been designed to evaluate the various aspects of the robot's functionality. The setup consists of an unfamiliar circuit that the robot must navigate while tracking and displaying its path. Additionally, the robot needs to read barcodes placed on the ground. As an added challenge, the engineering teams involved in this project will compete in a multi-lap race on the circuit, testing both the robot's efficiency and performance.

The test setup features an unknown course, requiring the robot's autonomy.

This requires great accuracy in the self positioning of the robot and in the detecting of obstacles such as walls, as well as a functioning, bug-free, algorithm that avoids undesired outputs stopping it from making progress in the track.

The algorithm should be optimized for speed and use the first lap as a means to gather information. Such that the robot can traverse the following laps as fast as possible. The laptime benchmark is difficult to anticipate as it is heavily dependent on the hardware and on the size and complexity of the maze which is unknown until the day of the competition. The robot is required to have a powerful enough powertrain and has to be nimble enough to complete easy maneuvers in an acceptable timeframe.

Moreover, the current choices of obstacle sensors are not affected by changes in the lighting or the loudness of the environment. However, it is important to assess their performance for the exact walls found in the maze (material, texture, color, etc). Only the barcode scanner is affected by dimly lit environments which can be addressed by equipping the robot with its own light source.

2 DESIGN AND MATERIALS

In this section, we present the components that make up the robot and provide a breakdown of all different subsystems and how they interact. Our design choices aim to strike a balance between performance, cost-efficiency, and reliability in the competition environment.

2.1 Drivetrain

The drivetrain is responsible for the mobility, speed and maneuverability of the robot. This subsection will outline the design choices to achieve a drivetrain that is suitable for navigating the maze-like circuit.

2.1.1 Chassis

The chassis is the mechanical platform on which the entire robot is build (Figure 2.1). It is designed to fit within a 20cm-by-20cm square and to be reliable enough to carry the whole design. The dimensions were chosen to allow for the implementation of reliable mechanical solutions while avoiding the challenges of designing and producing tiny mechanisms. The platform is also small enough to pass easily through the maze which has a track width of 35cm.

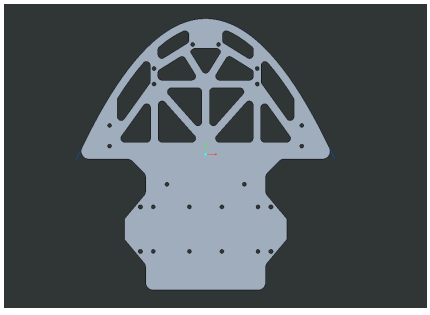

Figure 2.1: Chassis

The front of the chassis plate has a parabolic shape that allows the robot to slide without getting stuck if it ever collides with an obstacle. Furthermore, the wheels are mounted behind the parabolic shape to improve on the "aerodynamic" characteristic of our robot. The wheels

would need to have great traction as slip would make it less efficient and hinder the self positioning of the robot. Off-the-shelf lego wheels preformed better than a 3D printed version and were therefore used in the final design. In other words, this design minimizes the possibility of getting stuck if the car does collide, but also provides adequate protection for the drivetrain. Instead of a front wheel, the chassis incorporates two rounded downward extrusions that enable smooth sliding. This is a cost-effective and commonly used solution in robotics competitions like micromouse (a popular competition with a similar test setup where robots compete to solve a maze). On top of the robot fits a smooth shell to hide all the components and make the design more sleek and esthetically appealling.

2.1.2 DC motors

Two 12V direct current (DC) motors were the chosen actuators to drive our chassis. This decision was made to reduce the cost of the overall project whilst taking advantage of the power-increase due to the higher voltage. Despite their basic nature, brushed DC motors fulfill the requirements imposed by the project and can be easily controlled with an H-bridge integrated circuit (IC).

The selected DC-motors come coupled to an inline gearbox, reducing the rated speed to 200 rotations per minute (RPM) at 12V while simultaneously increasing the torque. This surplus of torque overcomes the additional drag of our bare-bones implementation of a *front-wheel*.

Figure 2.2: DC Motor

2.1.3 Motor controller

Initially, the simplest possible driver was tested. An H-bridge consists of 4 NPN transistors which can be opened and closed to adjust the direction as well as the speed of rotation, using pulse width modulation (PWM). However, it lacks advanced features that would allow one to tune the behavior of the motor more precisely. For this reason, the search for a new IC started.

An intelligent IC driver brings features such as slow or fast decay which can determine the braking behavior of the motor. This level of control is essential when trying to build the drivetrain of an autonomous robot which has to brake precisely to avoid colliding with the surrounding obstacles.

2.2 Positional sensors

Accurate self-localization of the robot is crucial to the accurate visualization of the maze-like course. To achieve this, a combination of sensors was chosen, which will be discussed in this section.

2.2.1 Wheel encoder

The wheel encoders are crucial for the self positioning of the robot. By keeping track of the rotational position of each wheel, the system can calculated the absolute position of the robot.

The initial implementation consists of a hall-effect based system. Hall effect sensors react to the magnitude of a nearby magnetic field. By placing small magnets on the wheels and fixing the Hall effect sensor on the chassis, the changing (rotating) magnetic field can be picked up and processed into a rotational position or angular speed. However, this approach proved to be problematic. It requires many magnets, placed close together, to obtain a satisfactory resolution. Furthermore, for a measurable output of the Hall effect sensor, it either requires more advanced circuitry or it has to be placed very close to the magnet-array. This, in turn, makes mechanical mounting tolerances very small. Because of the nature of the prototyping materials, this could not be guaranteed.

The implementation in the final design relies on photointerrupters (Optical sensors) (Figure 2.3). By pulling the drain of the phototransistors up, the passing of current results in logic-level voltage output. When the sensor is blocked by an object, the transistor will not conduct and vice-versa. Using this concept, an encoder wheel was developed (Figure 2.4). Thus, whenever the wheel passes through the sensor, the signal is high, otherwise the output is low and, hence, a square wave signal is obtained. By using 2 Whenever a rising edge is detected a ”step” is made.

To further improve the design, a second photointerrupter sensor is installed with a specific offset. Both signals are used in quadrature encoding to determine both the direction and speed of rotation. When setting the phase offset equal to half the width of a gap, the system allows for doubling the number of gaps in the encoder wheel, thus,

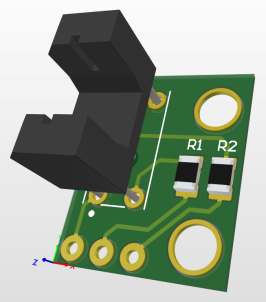

Figure 2.3: Photointerrupter PCB

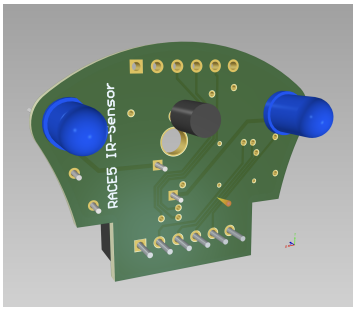

Figure 2.6: IR sensor PCB

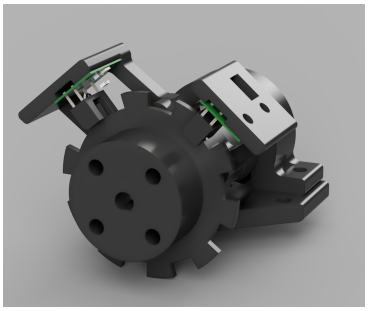

Figure 2.4: Encoder with motor-mount

doubling the encoder-resolution whilst still maintaining full functionality.

2.2.2 Infrared sensor

Both sides of the car feature a measuring sensor based on infrared (IR) waves. The choice was made to design and manufacture these completely custom, lowering the overall cost and stretch the available budget further. The IR Sensors rely on an IR light emitting diode (LED) and an IR phototransistor that detects the reflected light when an object (f.e. a wall) is nearby. The sensors-electronics are mounted on a printed circuit board (PCB), which fits onto a sensor-housing, creating a complete sensor-package.

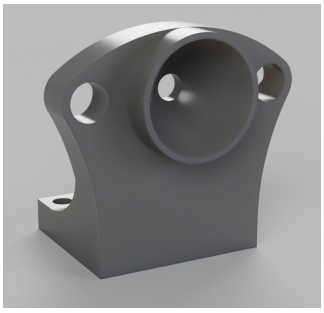

Figure 2.5: front of the IR sensor casing

When testing the first prototype of this sensor, a few downsides were discovered. The range of a single LED and a phototransistor to its side, was only 10mm to 70mm. This was not acceptable to our standards. When using two LEDs, to boost the light intensity, the range did not increase dramatically. This issue was traced to the phototransistor's field of view (FoV) being 120 degrees. To remedy this, a parabolic reflector was designed into the sensor housing to reflect a larger amount of light into the FoV of the phototransistor and increase its output to reflect a range of 10mm to 200mm.

This is more than sufficient for the readings on the side of the robot.

2.2.3 Time of Flight sensor

To achieve longer range sensing at the front of the robot, a time of flight sensor (TOF) is used. The VL53L0X is implemented on an off-the-shelve breakout-board (Figure 2.7). Designing a PCB with the surface mounted device chip (SMD-chip) and breaking them out to a header of sorts, is not worth the effort.

The use of a TOF-sensor, instead of the IR-sensor, is advantageous when looking at the sensing-range of both solutions. When comparing it to the mere 200mm maximum range of our IR sensor, having a, pre-calibrated, 1000mm range, is very useful for a front-mounted sensor. It allows the robot to optimize its speed depending on the available space on the track.

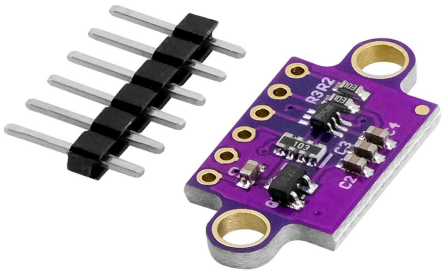

Figure 2.7: Time of flight sensor

2.2.4 Inertial Measurement Unit

An inertial measurement unit (IMU) is added to our car for backup-positional and rotational feedback. The measurements of the wheel encoders and distance sensor will be merged with this sensor, in order to create a redundant and robust position-feedback-system. The IMU used is a 6-axis MPU-6050 and is implemented as an SMD-chip on the PCB (Figure 2.8). In this text, both IMU and gyroscope will be used to refer to this component.

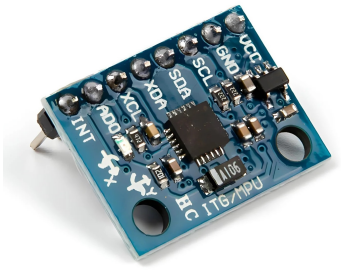

Figure 2.8: Inertial measurement unit

2.3 Barcode scanner

To scan the barcodes along the track, the robot is equipped with a barcode scanner mounted underneath its base. The barcode scanner comprises an LED and a light-dependent resistor (LDR), which work together to detect variations in light intensity. The LED's purpose is to eliminate external fluctuations in light intensity, ensuring reliable operation even in dimly lit environments. The LDR, a sensor designed to measure light levels, distinguishes between the white and black bars of the barcode based on their reflective properties. White bars reflect most of the incident light, while black bars absorb most of it. By analyzing these differences in reflected light, the LDR enables the robot to accurately identify the barcode patterns.

2.4 nRF24

The nRF24L01 (nRF) is a radio transmitter module designed for wireless communication and is used to communicate between the main processor (ESP32) and a remote controller. The nRF communicates with its microcontroller using the serial peripheral interface (SPI), operates wirellesly at a frequency of 2,513 GHz and is used to remotely start the robot. The module itself consist of a PCB with a built-in antenna. See Figure A.2 for an overview of all communications.

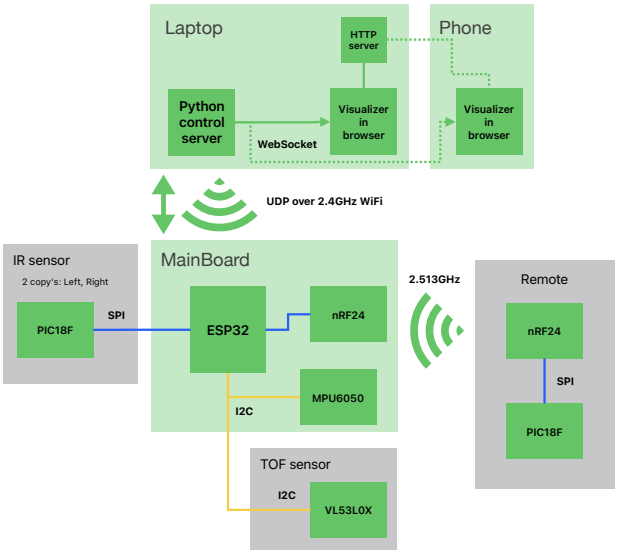

Figure 2.9: Communication Block Diagram

2.5 Overview of the design

The ESP32-S3-DevKitC-1 is the master microcontroller, orchestrating all the different modules operations and acting as the central hub for system intelligence. The PIC18F57Q43 is the microcontroller that will be used for the slave nodes. In Figure A.1 the complete block diagram of the car is depicted.

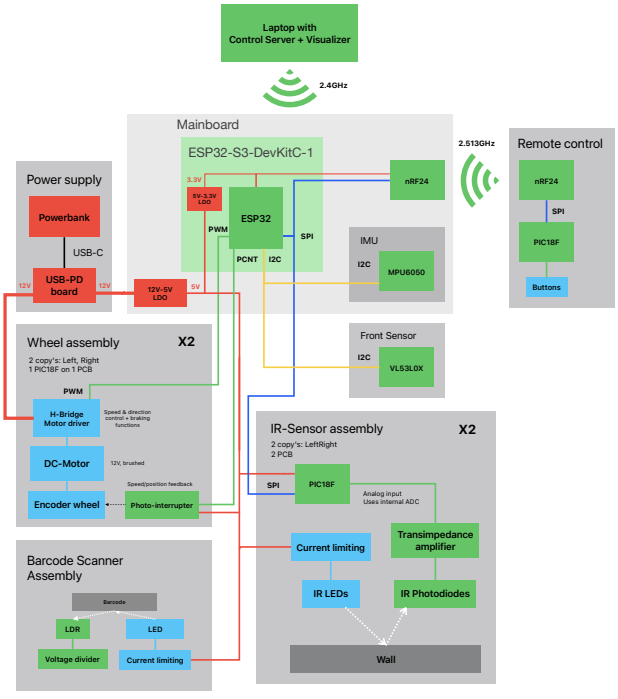

Figure 2.10: Complete Block Diagram

3 IMPLEMENTATION

The implementation phase focuses on combining all separate modules into a functional autonomous robot. This chapter details the steps taken to integrate the hardware components, develop the control software, and optimization of the system for stable and reliable autonomous navigation. Additionally, deviations from the initial drawings will be outlined and explained. Detailed images of the final assembly are provided in Appendix B.

Figure 3.1: Final Robot Assembly

3.1 Hardware Assembly

The first step involved assembling the hardware parts. The chassis was 3D-printed using a durable and lightweight PLA and PETG, ensuring stability while keeping the robot agile. The drive train was installed according to the specifications declared in the previous section. The electronic components, including the main PCB and sensors, were securely mounted on the chassis, maintaining proper weight distribution and avoiding unnecessary obstructions to sensor operation. In the final version, a smooth cover plate is mounted on top of the entire car, hiding the electronics and shielding the sensitive components from harm.

3.2 Software Development

The control software consists of a distributed system, containing the main microcontroller and an accompanying laptop, running a Python application: the control server and visualizer.

3.2.1 Sensor Data Processing

The ESP32 computes all sensors values into six floating-point values, the absolute position (x and y), the heading (theta) and three proximity values (north, east and west). All of which are based on the various sensor readings, with the heading being calculated using the IMU, the position based on the encoder values and the proximity relying on the TOF-sensor and custom IR-sensors.

3.2.2 Motor Control

Furthermore, the ESP32 will translate two floats, throttle (0...1) and steering (-1...0...1), into the correct motor controls. The steering method is adaptable to make operation more intuitive and the autonomous control scheme more stable. When the throttle input is zero, the motors are equally driven in oposite direction, in order to make the car turn in place. When a throttle input is applied, the motors only get influenced slightly, reducing or increasing its speed 15% accordingly to the direction of the requested adjustment.

3.2.3 Communication

The eight aforementioned values are exchanged with the control server, running on the laptop. Using the ESP32's built-in Wi-Fi capabilities, a 2.4GHz network is set up, allowing the laptop to connect to the ESP32. Both relay UDP packets over the network, providing high-speed communication in a reliable way. Although UDP does not guarantee delivery, the increased speed makes up for possible missing data points. Also, streaming live telemetry only relies on the latest data point, making packet loss a non-issue.

3.2.4 Control server

The control server is implemented in Python and runs on the laptop. It receives the sensor data from the ESP32 and processes it to determine the required throttle and steering.

The algorithm uses a simple proportional controller to adjust the throttle and steering values based on the proximity readings and the objective. If an obstacle is detected within a certain range, the robot will slow down and steer away from the obstacle, ensuring safe navigation. A second exploration-algorithm was devised to orchestrate autonomous navigation of the environment. This implementation is based on a couple of wall-following laps, making sure all grid cells are discovered, followed by an optimized lap-controller that can trace the shortest path and execute this path indefinitely.

3.2.5 Visualization

In addition, the control server provides a WebSocket with the latest telemetry-data. A webpage, run by a local HTTP server, connects to the WebSocket and displays the telemetry on a map. This allows for real-time monitoring of the robot's position and heading, as well as the proximity values. The map is updated in synchronization with the streamed UDP packets, providing a clear visual representation of the robot's movements and surroundings. Being browser-based, this application can be accessed from any device connected to the same network, making it easy to monitor the robot's performance from a distance.

3.3 Calibration and Testing

After the integration of hardware and software, a testing and calibration procedure was performed. The sensors were tested under different lighting conditions and adjusted to limit inaccuracies.

The motor control parameters were fine-tuned to ensure precise and uniform movements. As an example, the friction within each gearbox was not equal. This caused a speed delta between the two wheels which in turn lead to non-straight movement when no steering input was applied. Correcting for this offset, by scaling the motor outputs to match this difference in drag, resulted in very accurate straight-line behavior.

While multiple test scenarios validated the detailed obstacle detection capabilities of the robot, the calm and accurate motion of the drive-platform and the inherent flexibility of the software-stack, a full-scale deployment scenario would stress the complete implementation.

This full-scale testing involved running the robot in a controlled environment, gradually increasing the complexity of the setup. Edge cases, such as split paths and tight turns, were introduced to evaluate the robustness of the entire system. Debugging and software refinement were conducted iteratively to optimize performance.

3.4 Final Adjustments and Optimization

After initial testing, the implementation was refined by adjusting sensor thresholds and improving algorithm efficiency, also tuning its performance to the specific environment in which the car is required to operate. The final prototype demonstrated the ability to autonomously navigate and avoid obstacles in real-world scenarios.

The successful implementation of the autonomous robot serves as a foundation for further enhancements and potential future applications in fields such as warehouse automation, smart transportation, and robotics research.

4 EVALUATION AND VALIDATION

The evaluation consists of a series of tests designed to determine the performance metrics of the robot. The sensor modules are tested both individually and as part of the final implementation in order to create a reliable performance assessment.

4.1 Objectives of Evaluation

The objective of the evaluation is to assess the reliability, robustness and the optimal operating ranges of the sensor and actuator modules.Additionally, a complete benchmark of the entire robot is done to quantify the real-life metrics.

4.2 Test Setup Description

To benchmark the IR-sensors, reading were collected when detecting an obstacle positioned at a known distance. The test is repeated for different distances and lighting conditions, to compensate for the variable surroundings. An additional goal is to see if the values are fluctuating over time when the same obstacle is measured/detected.

The encoders are tested by moving the robot over a known distance and comparing the total number of steps multiplied by the resolution with the actual distance travelled. The robot is pushed by hand in a straight line over a distance of one meter, then the same test is repeated by driving the robot.

As for the path finding algorithm, it is first tested in a computer simulation where a virtual maze was constructed. The simulation was built to mimic real life. Thus, the virtual counterpart of the robot has to map the virtual maze using real-time simulated sensor data. This is done in order to test purely the logic and correctness of the algorithm while excluding problems caused by real life inaccuracies and imperfections.

The final implementation of the robot is tested in three textbook scenarios: driving in a straight path defined by maze walls, driving in square loop defined by maze walls and driving through a junction. This is done to see how the performance of the sensors translates in a real-life situation.

The first scenario is used to tune the system controllers used as well as to benchmark the ability of the robot to avoid the walls on a straight track. Furthermore, this test

gives insights into the stability of the system as well as how much the system oscillates relative to the reference path.

The second scenario tests the turning ability and decision making. The robot has to detect that a right or left turn has to be made and to execute it. Lastly, in the third scenario the robot is tested when traversing a junction. It has to detect the presence of a junction as well as to decide how to proceed.

4.3 Evaluation Metrics

The following metrics were used to assess performance:

ε_d: error accumulated by the encoders over the total distance travelled d
γ : accumulated error of z-axis gyroscope over the degrees turned
v_{opt}: optimal reference speed the robot
b_{ratio}: number of correct readings/10 per barcode
Δ_p: positional accuracy; difference in estimated position versus real position
w_{ratio}: obstacles detected / total obstacles
$t_{delay,\ UDP}$: communication latency; delay in transmission of UDP packets between laptop and robot
α_i: navigation accuracy; percentage of succesfull runs in scenario i

4.4 Results Summary

After being subjected to the testing scenarios described in section 4.2, the respective results are summarized in Table 4.1:

Metric	Average across 20 runs
$\varepsilon_{\text{avg},d=10m}$	12.83 cm
$\gamma_{\text{avg},\theta=360}$	22.24 degrees
v_{opt}	20 cm/s
b_{ratio}	9.15 correct barcodes
Δp	17.55 cm
$t_{\text{delay,UDP}}$	0.01 ms
α_1	85%
α_2	73%
α_3	97%

Table 4.1: Summary of Performance Metrics Across 20 Runs

4.5 Observations

The custom sensor modules performed as required. The IR sensors have shown great accuracy at medium range (1 – 24 cm) and consistently detected the walls/obstacles located on either side of the robot. Furthermore, despite the limited resolution of the encoder wheels (limitation imposed by the manufacturing technique used to produce them), the positional feedback was reliable.

In the first testing scenario the robot was capable of driving in a straight line while avoiding the walls defining the path. The robot showed oscillation when adjusting for the correct orientation but managed to remain in the middle of the track.

In the second testing scenario the robot was capable of detecting that a turn is necessary and turned in place by 90 degrees to either left or right before continuing to drive straight.

Finally, in the last testing scenario, the robot could detect that it arrived at a junction.

4.6 Limitations

After running the testing procedures described some limitations became apparent.

The readings of the IR sensors are not reliable when the obstacles are very close (less than one cm) or very far away (more than 24cm). Furthermore, the surface and colour of the maze walls affects the performance. Reflective surfaces or surfaces with uneven colour cause unreliable readings.

The encoders and gyroscope in the IMU accumulate error over time and have to be recalibrated. The gyroscope is especially sensitive to large rotational accelerations. If the robot turns abruptly, and the acceleration exceeds the maximum value of the gyroscope, then the gyroscope will clip at a maximum threshold. This problem occurs both when the robot corrects its trajectory to maintain a straight path as well as when it has to turn, hence why the positional feedback system is more efficient when the robot moves slower and the acceleration curve is less aggressive.

The ground floor also affects the performance of the robot. Very rough and very glossy surfaces negatively impact the drivability of the robot due to the loss of traction.

5 DISCUSSION

This research aimed to design and implement a differential-drive robot capable of autonomously navigating a maze. The key results from testing demonstrate that every part of the robot performs well, though there are small errors. These small errors accumulate over time

during the robots runs. Because the design is layered and the different modules rely on each other this accumulated error prohibits long driving and thus from autonomously completing the maze.

However, this does not mean that our design and research are of no significance. As mentioned all parts are working and parts that do not depend on others as much, such as the barcode scanner and the visualization, still perform reliable when driving manually with the robot. Furthermore the algorithm designed to solve the maze-like tracks performs great in the simulation.

Indicating that the theoretical framework is sound. The robot cannot drive autonomously but it lays a great foundation for further developments. It also shows how it can integrate extra functionalities into a design so it can be a perfect fit for a multitude of sectors and challenges.

To achieve a functional and fully autonomous design, recalibration is needed in order to accurately and reliably track the position of the robot. This holds true for the IMU which accumulates error over time and cannot reliably provide the absolute orientation unless it is recalibrated from time to time. An IMU with a better gyroscope would reduce the sensitivity to large rotational accelerations. Encoders with a higher resolution and a better weight distribution will also help reduce the errors. Replacing the two front gliders for a ball would reduce the drag. Bypassing even more of the unwanted slippage that adds to the small errors. This all ensures that the robots perceived location is also its actual physical location. However while these are helpful analyses for further studies, they highlight the trade-off between precision and the costs of a robot.

6 CONCLUSION

This project successfully demonstrated the design and implementation of a cost-effective racing robot. Through a careful selection of low-cost components and custom-built solutions, the robot achieved reliable performance while maintaining affordability.

The integration of multiple sensor systems, a distributed control setup, and a real-time telemetry interface allowed for efficient driving. In simulation the navigation proved to be efficient and the robot completing the course in competitive lap times. The robot's ability to learn and optimize after the first lap proved critical to its success. Iterative testing and calibration helped fine-tune both hardware and software aspects to maximize stability and accuracy. Despite certain challenges, such as sensor calibration and communication robustness, the separate modules met all key performance goals. However due to small imperfections and errors adding up when all the modules are combined autonomous driving for three full laps was not achieved.

Valuable lessons were learned regarding sensor integration, motor control, and real-world system optimization. Future work could explore enhancements like more advanced localization algorithms or improved hardware durability. Overall, this project provides a strong foundation for further developments in autonomous robotics. It highlights the importance of strategic design choices and adaptive software. Our robot exemplifies how innovation and engineering principles can converge to solve complex problems efficiently. The experience gained throughout this project will be instrumental for future endeavors in robotics and automation.

ACKNOWLEDGEMENTS

We are deeply grateful to the members of the other race teams for openly sharing their valuable insights and experiences. Their input greatly enhanced the development of key sections of this paper. The discussions around system optimization and design challenges were especially influential. This collaborative spirit made a significant difference in the depth and quality of the work. the openness and expertise, that was brought to the table, Is deeply appreciated.

LIST OF SYMBOLS

Symbols

λ_{ex}	Excitation wavelength
λ_{em}	Emission wavelength
$N(\dots)$	Noise process
ε_d	Error accumulated by the encoders
γ	Accumulated error of z-axis gyroscope
v_{opt}	Optimal reference speed of the robot
b_{ratio}	Number of correct readings per 10 barcodes
Δp	Positional accuracy
w_{ratio}	Obstacles detected divided by total obstacles
$t_{\text{delay,UDP}}$	Communication latency
α_i	Navigation accuracy

Acronyms

ADC	Analog-to-Digital Converter
CCD	Charge-Coupled Device
FOV	Field of View
IC	Integrated Circuit
IMU	Inertial Measurement Unit
IR	Infrared
LED	Light Emitting Diode
PCB	Printed Circuit Board
PETG	Polyethylene Terephthalate Glycol
PLA	Polylactic Acid
PWM	Pulse Width Modulation
SPI	Serial Peripheral Interface
TOF	Time of Flight
UDP	User Datagram Protocol

BIBLIOGRAPHY

(2021). The simultaneous localization and mapping (slam)-an overview. *Journal of Advanced Science, Technology and Technological Innovation (JASTT)*, 2(02):147–158.

AL-Furati, S. and Rashid, A. T. (2020). Shortest distance orientation algorithm for robot path planning using low-cost ir sensor system. In *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pages 1–6, Istanbul, Turkey.

El Sallab, M. A., Perot, E., and Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 303–312, Venice, Italy.

Mahmoudi, R., Maskeliūnas, R., and Damaševičius, R. (2023). Simulated autonomous driving using reinforcement learning: A comparative study on unity's ml-agents framework. *Information*, 14(5):290.

Ren, J. and Xia, D. (2023). *Autonomous Driving Algorithms and Its IC Design*. Springer Nature Singapore.

APPENDICES

APPENDIX A : OVERVIEW BLOCK DESIGN

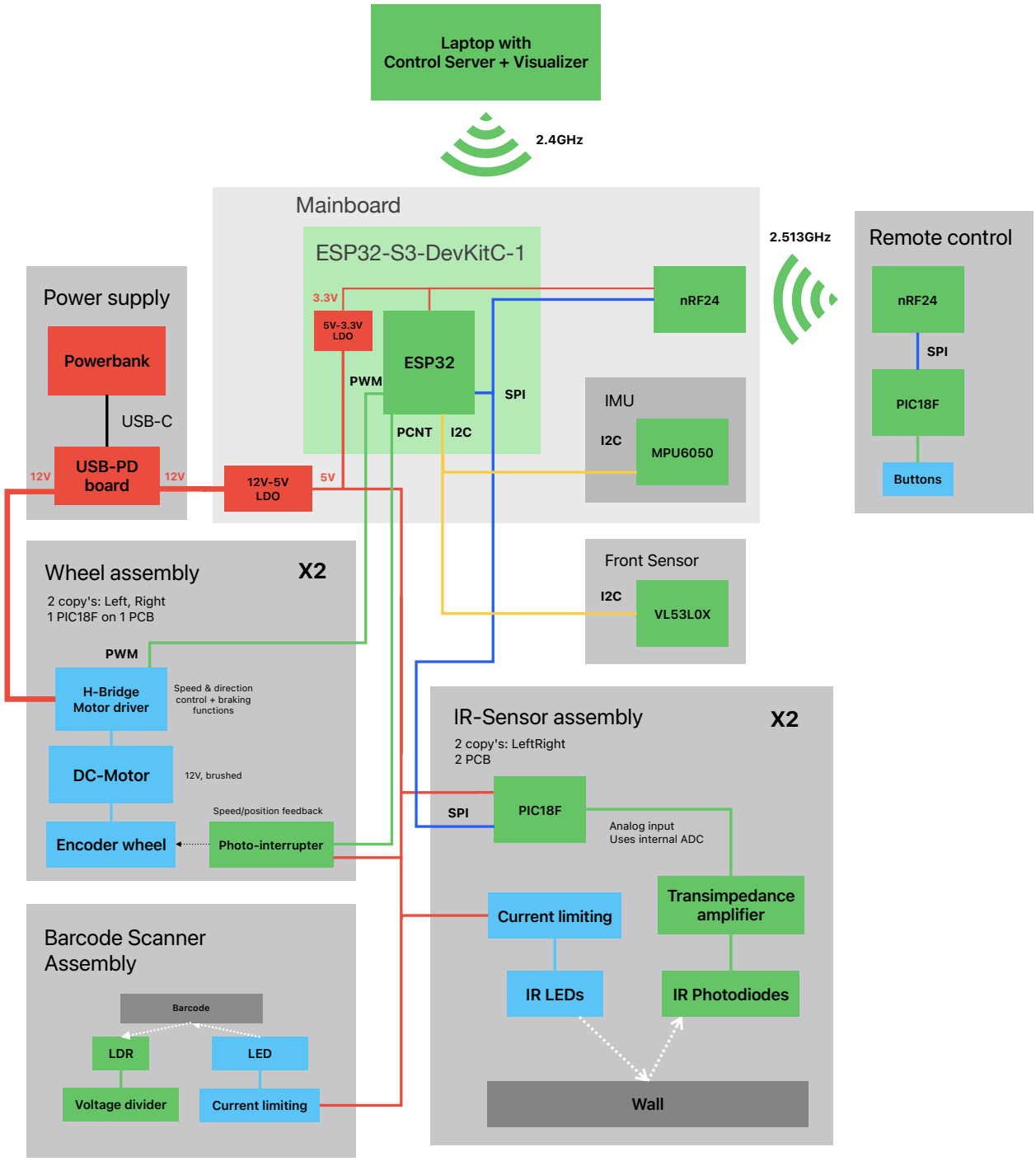

Figure A.1: Complete Block Diagram

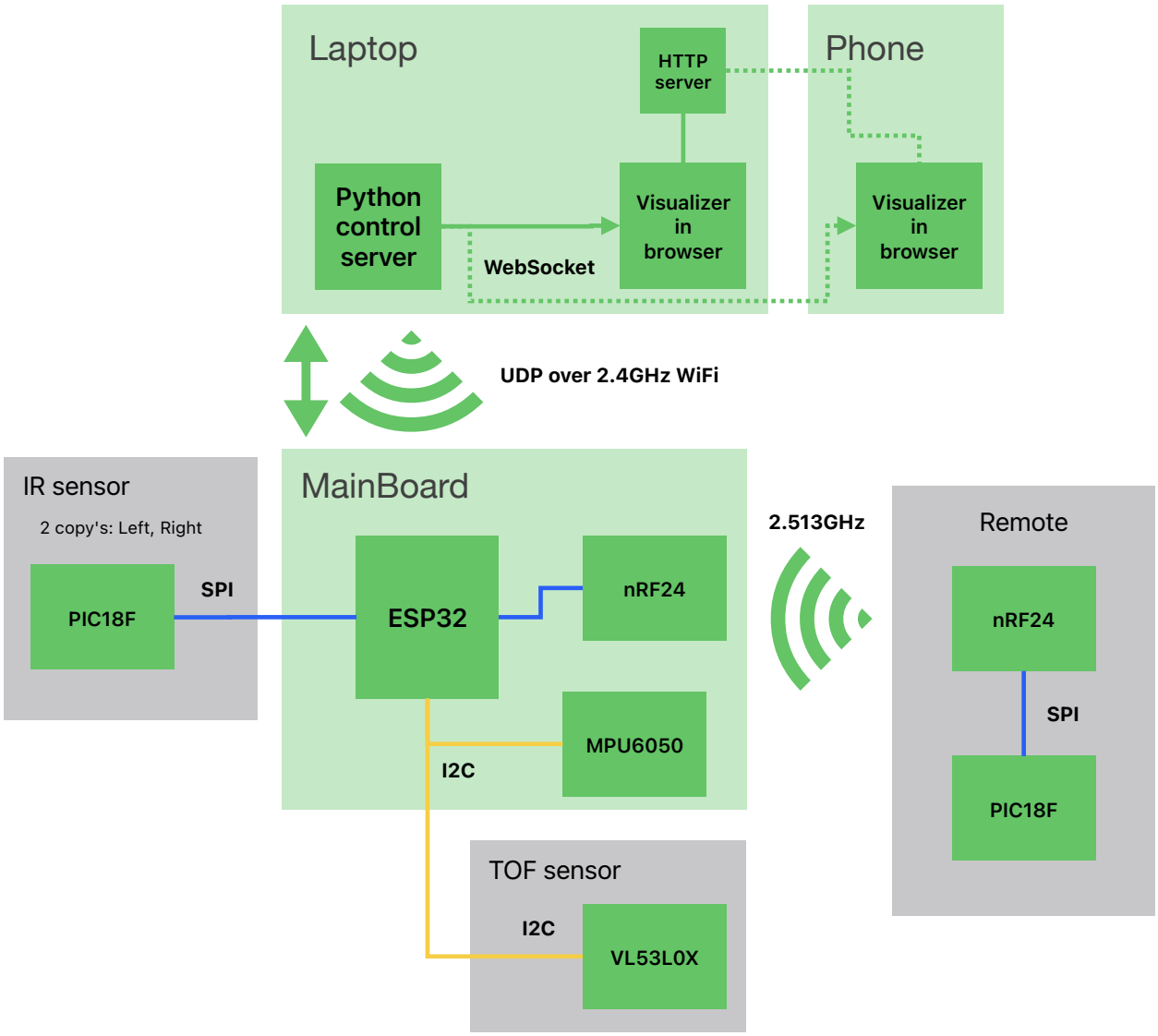

Figure A.2: Communication Block Diagram

APPENDIX B : FINAL IMPLEMENTATION IMAGES

Provided in this part of the appendices are images of the final robot assembly.

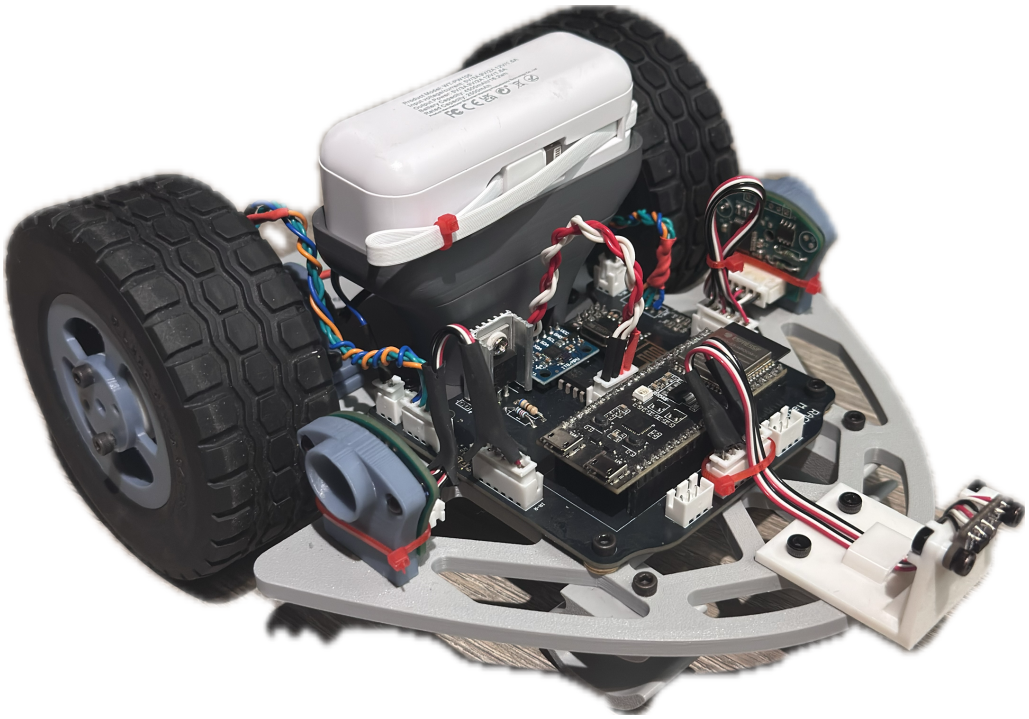

Figure B.1: Robot Assembly 3/4 Front View

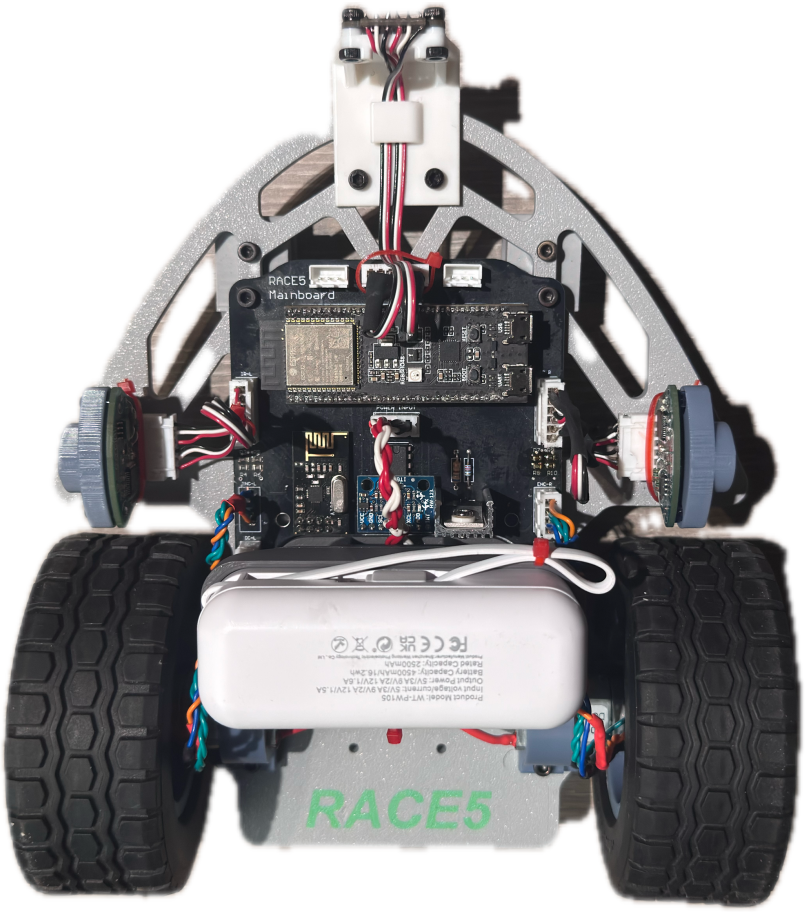

Figure B.2: Robot Assembly Top View

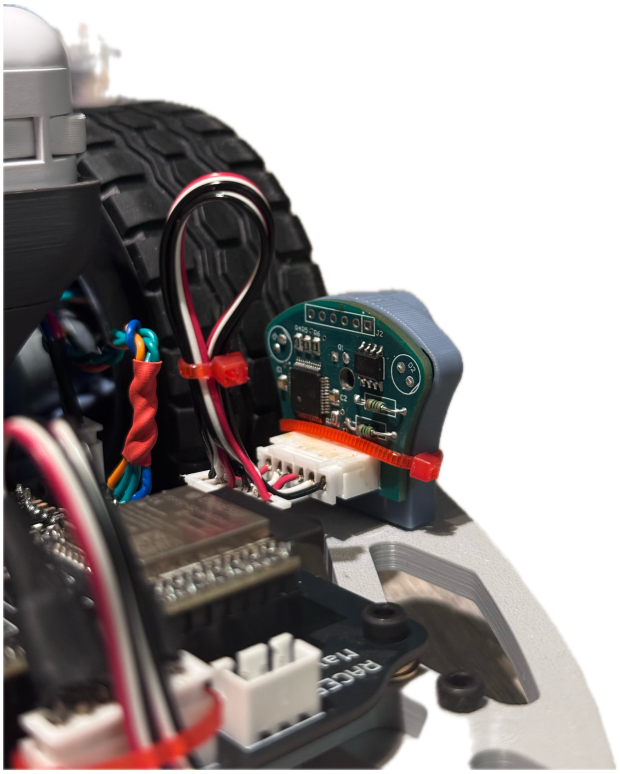

Figure B.3: Robot Assembly IR Sensor View

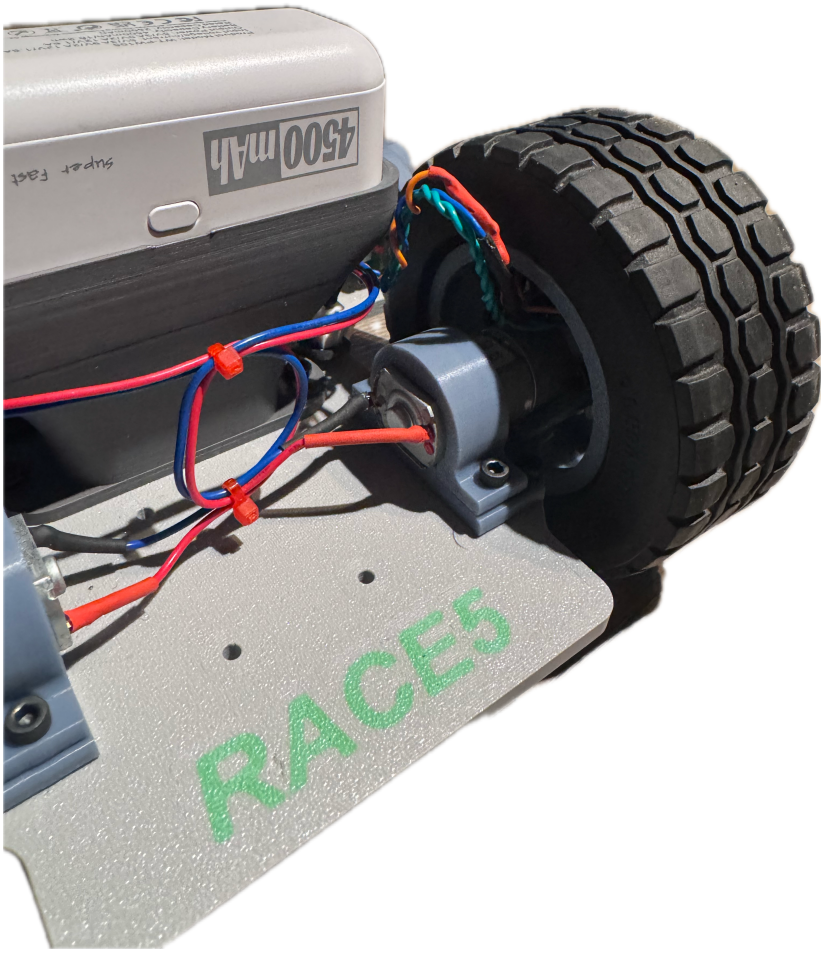

Figure B.4: Robot Assembly Motor & Encoder View