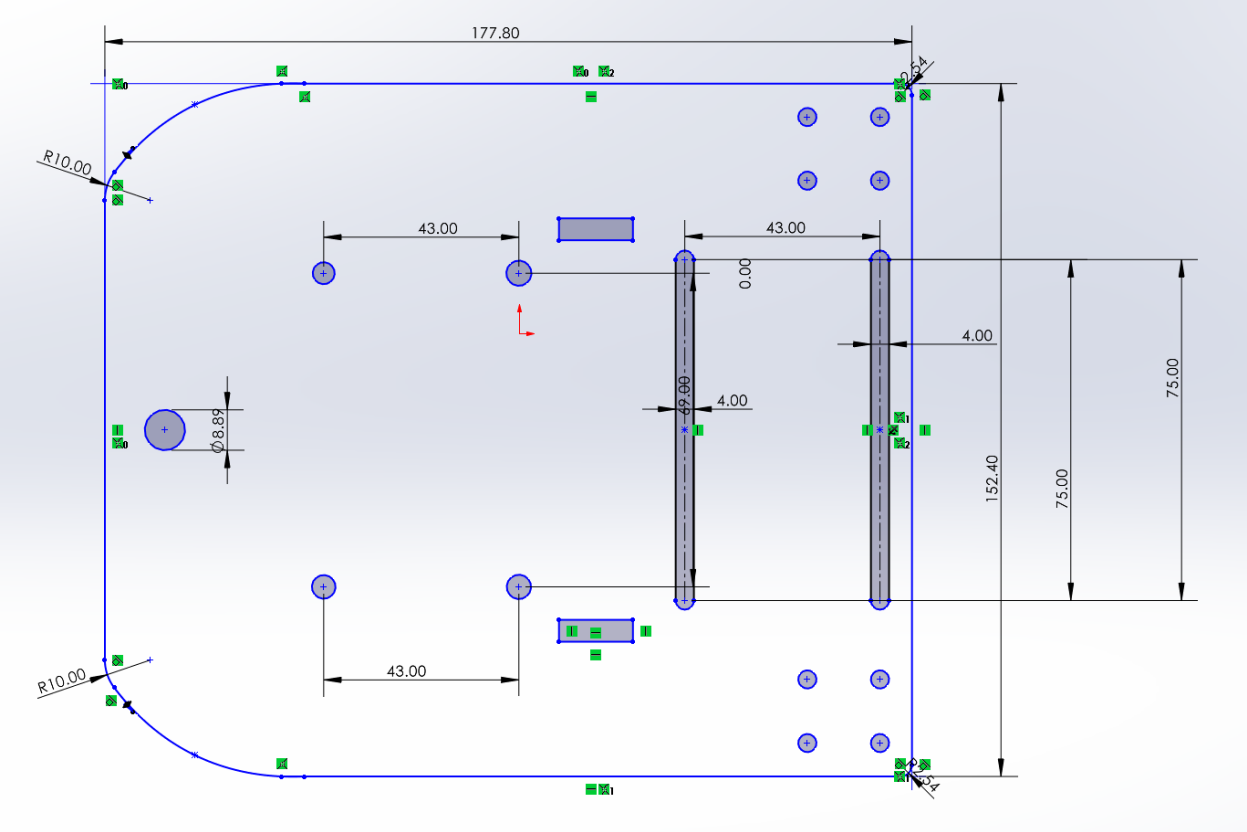
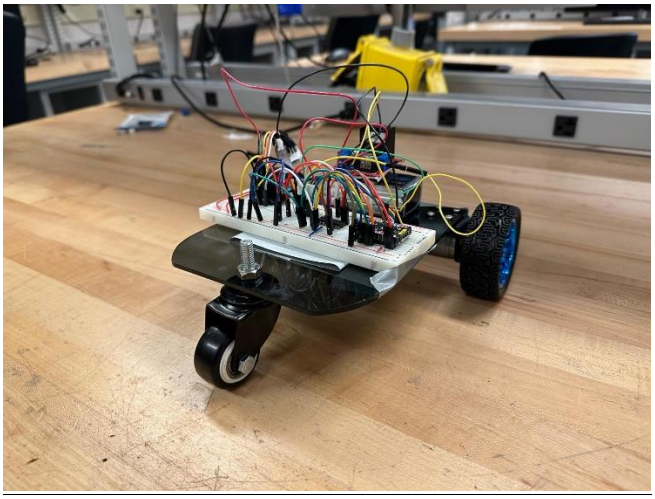
# 4.2 Report

## Mobile Base Design and Performance

Description of Mobile Base Approach



The above image showcases the draft design of the base plate for our robot. We made a single-layered bot, with all the components distributed by weight to ensure smooth driving.

The key features of the design include:

1. Wheel Mounts: Precise slots for mounting the powered wheels, castor wheel, and motors, ensuring alignment and stability.
2. Cutouts for Electronics: Slots and holes for securely mounting the motor drivers, microcontroller, and battery pack. The details of the electronic components are given in another section of this report.
3. Compact Layout: The design minimizes the robot's footprint while maintaining sufficient space for components.

For our mobile robot project, we chose a differential drive system for its simplicity, and ease of control. We used two independently controlled motors, which allowed the robot to move forward, backward, and rotate on the spot by varying the relative speeds and directions of the wheels.

Motor Selection:
We utilized the 500 RPM DC motors with encoders, as shown in the linked product: https://www.amazon.com/Encoder-Magnetic-Gearbox-Bracket-Reduction/dp/B07X5P1584/ref=asc_df_B07X5P1584?mcid=16cfc2bccb3c3e6e9dac01e59e80ecda&tag=hyprod-20&linkCode=df0&hvadid=693612458080&hvpos=&hvnetw=g&hvrand=12100250391631147989&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9198414&hvtargid=pla-1949760446346&psc=1

These motors were selected based on the following criteria:

1. Speed and Torque: The 500 RPM motor provides an optimal balance of speed and torque, which allowed the robot to traverse the track while maintaining good control.
2. Encoder Feedback: The integrated magnetic encoders gave us a precise measurement of wheel rotations.
3. Power Requirements: The motors were compatible with the power capabilities of our battery and motor driver setup.

Because of the simplicity and high maneuverability of differential drive, it was suitable for our project goals. This configuration was ideal for navigating the tight spaces around the obstacle.

Discussion of Performance and List of Improvements

The race time was 30.57 seconds.

The motion of the robot was close to what was expected based on our implementation. We implemented PD control and the robot's motion was fairly consistent in the forward and backward directions. There was, however, a slight turn in its motion, which could have been due to the following reasons:

i. Slight vibrations and shaking introduced because of the motors, leading to some instability in the mounting
ii. A slight lag between the button control and robot motion
iii. Lack of integral control

Some improvements that can be made are:

i. To add four extra buttons for diagonal motion (forward right, forward left, backward right, and backward left) to increase freedom of driving
ii. To distribute the weight on the base a little more evenly
iii. To give more support to the motors and prevent shifting and misalignment of wheels

For the final project, we will enhance the performance of our robot by implementing the above improvements, as well as shifting to a four-wheeled base and holonomic drive.

Final Version of Code Used on Mobile Base and Input ESP32

The final code is submitted to Gradescope.

The webpage for controlling the robot had five buttons- forward, backward, left, right and stop. The robot could be started by pressing any of the direction buttons. The stop button deactivated any pressed button and stopped the motors. When a button was clicked, it triggered a JavaScript function that sent an HTTP GET request to a server endpoint ('/update') with the selected direction. The function also updated the visual state of the buttons, highlighting the selected one. Once the buttons were pressed, different initial duty cycles were given to the motors based on the direction chosen.

Initialization:

The ESP32 was configured as a WiFi Access Point with a static IP address and network credentials. A web server (ws) was started on port 80. Two HTTP endpoints were defined:

- /: Serves the main control webpage (content from body).
- /update: Processes motor control commands (e.g., direction and stop).

GPIO pins for motor direction, PWM, and encoder channels were initialized. PWM channels were set up with a frequency of 5 kHz and 8-bit resolution for smooth motor speed control. Interrupts were attached to the encoder pins to handle pulse counting for RPM calculation.

Interrupt Handlers:
When the encoder signals changed, the interrupt handlers (handleEncoder1 and handleEncoder2) were triggered to count pulses. The pulse count reflected motor rotations, and the direction of counting depended on encoder signals.

Motor Control Logic:
The setMotorSpeed() function adjusted motor speed and direction based on input commands (e.g., forward, backward, right, left). PWM signals controlled the motor speed and GPIO pins managed direction. A "stop" command set PWM to 0 to halt the motors.

Main Loop:
The web server (ws.handleClient()) processed incoming HTTP requests. Every 100 milliseconds, the loop disabled interrupts to safely read encoder pulse counts, resets pulse count and calculated RPM for both motors based on elapsed time.

RPM Calculation:
The calculateRPM() function computes RPM from encoder pulses to provide real-time speed feedback for each motor.

Video for Part 4.2.2
PD Control: https://youtu.be/WQkMQWAOCzM

As shown in the video, when one wheel was held and the encoder values stopped getting updated, the other wheel also came to a stop.

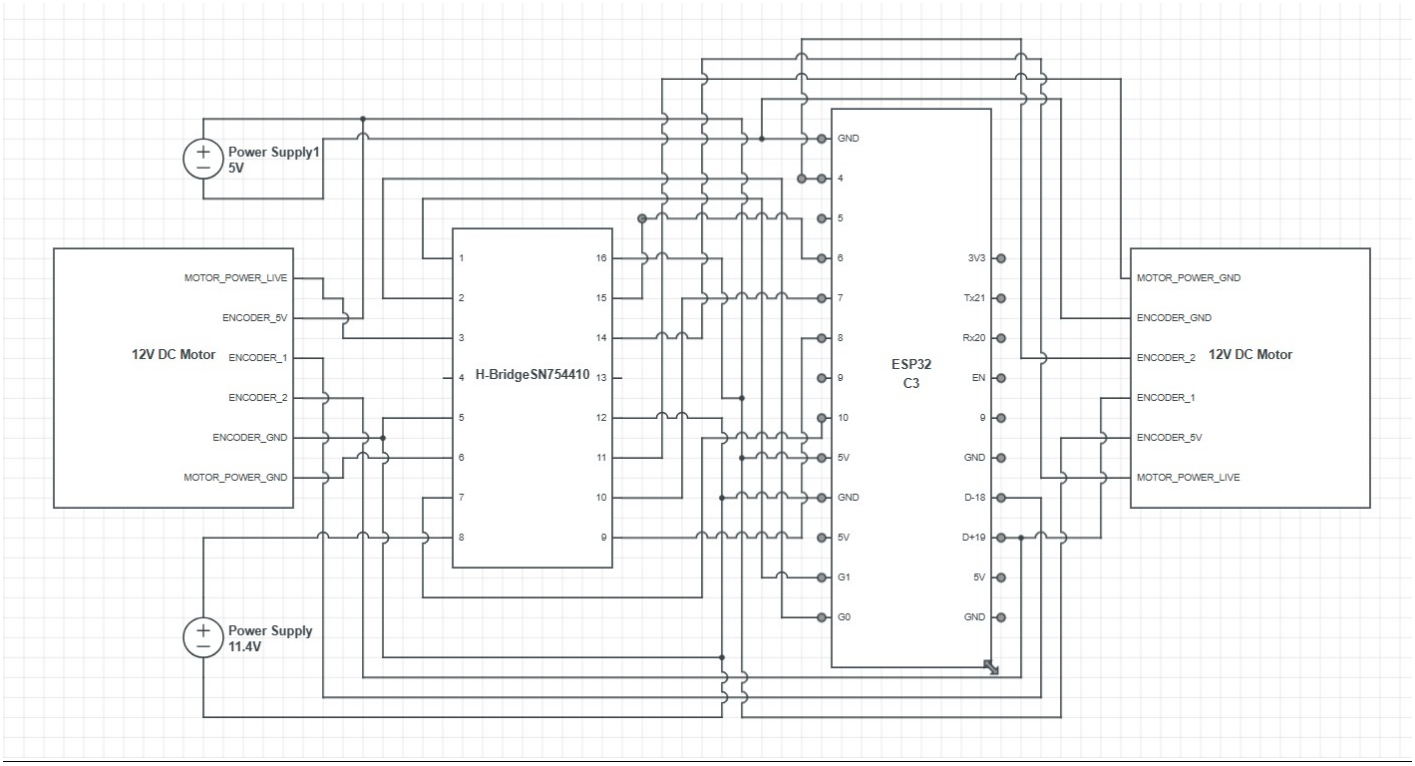
Straight Line Motion: https://youtu.be/hliANWhLX7Y

Bill of Materials

| Serial No. | Description | Material/Component | Quantity |
|---|---|---|---|
| 1 | Motors | Metal | 2 |

| | | | |
|---|---|---|---|
| 2 | Wheels | Plastic + Rubber | 2 |
| 3 | Castor Wheel | Rubber + Metal | 1 |
| 4 | Chassis | Acrylic | 1 |
| 5 | Battery | 11.1V LiPo | 1 |
| 6 | Microcontroller | ESP32-C3 | 1 |
| 7 | Voltage Divider | L298N Motor Driver | 1 |
| 8 | H-Bridge | SN754410 | 1 |

Circuit Diagram



Division of Work

For this lab, we divided the work into three main areas: code, circuits and design.

- The code to make the website and control the robot on the ESP32 Wi-Fi was done by Samhitha. The PID code was done by Saayuj.
- The circuits that used the SN754410 motor driver, ESP32C3, voltage regulator and battery were assembled by Saayuj.
- The CAD drawings and laser-cutting of the robot body were done by Gia.
- The assembling of the robot was done by Gia and Samhitha.

References

- MEAM 5100 Lecture Slides
- Google
- ChatGPT (for debugging)
- DigiKey
- Datasheets:
    - https://zeeebattery.com/products/zeee-3s-lipo-battery-2200mah-50c-xt60
    - https://www.digikey.com/en/products/detail/texas-instruments/SN754410NE/380180