MEAM 510 FINAL PROJECT REPORT

Zhanqian Wu, Bowen Jiang, Enlin Gu, Yurui Wang
University of Pennsylvania
Philadelphia, PA 19104
{Zhanqian, jbwenJoy, Enlin, Yurui}@seas.upenn.edu

ABSTRACT

This paper presents a comprehensive exploration of a robotic system designed for diverse tasks, including communication, perception, and control. The project employs a behavior tree logic to manage modes for tasks such as wall-following, autonomous navigation, and manual control. Hardware components, including 18650 lithium batteries, motor drivers, and various sensors, contribute to the functionality of the robot. Additionally, the document details the mechanical design of the mobile base, its structural overview, and task-specific methodologies. The electrical design incorporates a sophisticated power management module, motor control circuitry, and sensor integration, ensuring optimal functionality. The core board features inter-ESP32 communication for data exchange, contributing to the modular architecture of the overall system. The paper also outlines the processor and code architecture, emphasizing sensor data retrieval, communication tasks, behavior tree implementation, and action functions. Various action modes, including wall-following, car-pushing, trophy-moving, and manual control, are discussed in detail, showcasing the versatility of the designed robot. The document concludes with a comprehensive appendix, including a bill of materials, robot photos, circuit diagrams, and dimensioned drawings. The project's success is demonstrated through video links that validate the system's capabilities. Overall, this paper provides valuable insights into the design, implementation, and performance analysis of a multifunctional robotic system.

***K*eywords** Mobile robotics · Behavior tree logic · Sensor integration · Power management · Inter-ESP32 comm

1 Functionality

1.1 Overview of Approach

Our robot is designed to realize the comprehensive task of communication, perception and control. The performance worked includes robot communications via ESP-NOW, vive XY via UDP, wall following, autonomous navigating to objects, police car moving and manual controlling.

In our design, We will use a behavior tree logic to change modes of different tasks. Upon setting mode in the web page, the robot system will act in the given mode and finish tasks automatically. We use ESP32S2 and ESP32C3 to realize all the software functions. The two ESP32s are connected by wires to complement serial communication. The ESP32s2 is used for getting data from sensors. The ESP32C3 is used for web page communication, UDP package broadcast, and ESP-now message sending. The details of software design is explained in Processor Architecture and Code Architecture section.

For the hardware design, our robot has a round shape with four layers of boards. Electrical components are placed on each layer, and the wires go through holes on the boards. The robot is driven by two wheels, and there are two cast wheels for supporting. We choose this design because differential control is reliable and relatively easy to realize.

For electrical design, our circuit power supply is an 11.1V 18650 lithium battery with DC-DC converter and LDO regulator, so that both 5V and 3.3V outputs can be provided. We use BTS 7960 H-bridge as motor driver to drive two 370 motors with encoders. We use MG996 servo to control the jaw in order to hold the trophy. For sensor module, we use IR photodiodes as the sensing elements to find trophies and applied TLV272 amplifier to strengthen signals. We use vive sensors to capture the position and orientation data of the robot from the base station. In addition, ToF

(Time-of-Flight) sensors are utilized On the side to measure distance between robot and obstacles like walls and trophies. The details of hardware and electrical design is listed in Hardware Design and Electrical Design sections.

1.2 Video Link

1. CHECK OFF: `https://www.youtube.com/watch?v=02UyDjv6waM`

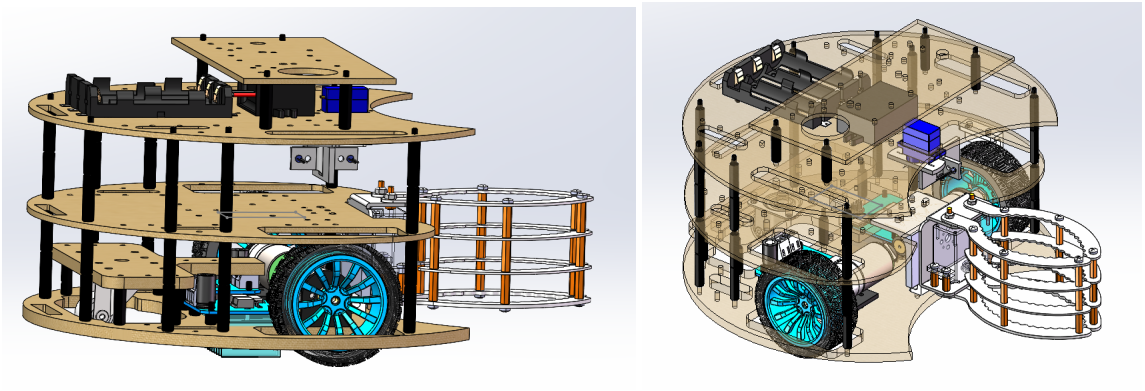
2. IR CIRCUIT TEST: `https://youtu.be/ctNg3ylUBPU`

3. VIVE CIRCUIT TEST: `https://youtu.be/0ITU11SvNJ4`

2 Mechanical Design

2.1 Structure Overview

Our mobile base design is shown in the figure 1:



(a) Mobile Base Overview 1 (b) Mobile Base Overview 2

Figure 1: Mobile base Design

As you can see from the figure, our mobile base has a round shape with two driving wheels being centrally symmetric about the circle. This enables our mobile base to rotate about the center of the circle by simply making both wheels rotate in opposite directions. Also, the round shape can protect our wheel and other important components from colliding with the environment or other items on the field.
We use **2** caster wheels in this project instead of **1** as we did in lab4. As we have more functionalities, our mobile base is heavier than that in lab4, therefore we use 2 caster wheels to bear the weight.
We use this kind of multi-deck design so that our mobile base is compact and each task can be done properly. The function of each deck is:

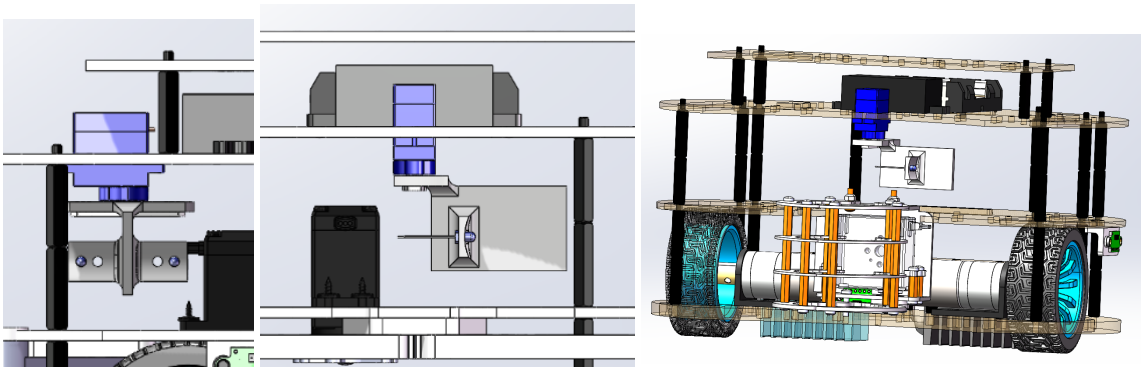
- **Base deck:** We put our motor, encoder, and motor control board on the base deck. This deck bears most of the weight of the car, so we chose 1/4 inch thickness board. The TOF sensors are also on this deck (hung from the ceiling, which is the board of the second deck.)
- **Caster Wheel Board:** Between the first and the second deck, we have a board that is connected to two caster wheels. As the height of the caster wheels is not the same as the height of the driving wheels, the height of this board is deliberately designed so that all decks on the mobile base are horizontal.
- **Second Deck:** We put the claw fixed on one side of the second deck, and the IR sensor is also on the second deck.
- **Third Deck:** The third deck is the deck for all circuits: Batteries, perfboards with ESP32 C3, ESP32 S2, vive and other circuits.

The reason for our choice of components is in the **Electrical Design** section. The dimensioned drawings are in the Appendix figure 17 18.

2.2 Task Mechanical Methodology

Wall Following: In this task we only use 2 TOF sensors: one is on the left and another one is on the left side (as our mobile base does wall following counterclockwise, there's no need for another TOF on the right). The TOF sensors are 64mm high, which is lower than the height of the wall on the field.
Vive Sensor: We decided to use two vive sensors on the top so that we can not only get our mobile base's current position but also get its orientation. Therefore we can know the angle between our current orientation and the proper direction to the goal (police car).

Trophy Navigation: In this task, we mount two IR phototransistors on a servo with an optical separator. To ensure that the beacon signal can reach our mobile base well, the designed height for two photosensors is 95mm from the ground, which is nearly the same as those emitters on the trophy. Moreover, the second deck on our mobile base is clear without any obstacles except a few standoffs, so nothing will block our phototransistors. Figure 2 demonstrates our design for this task.



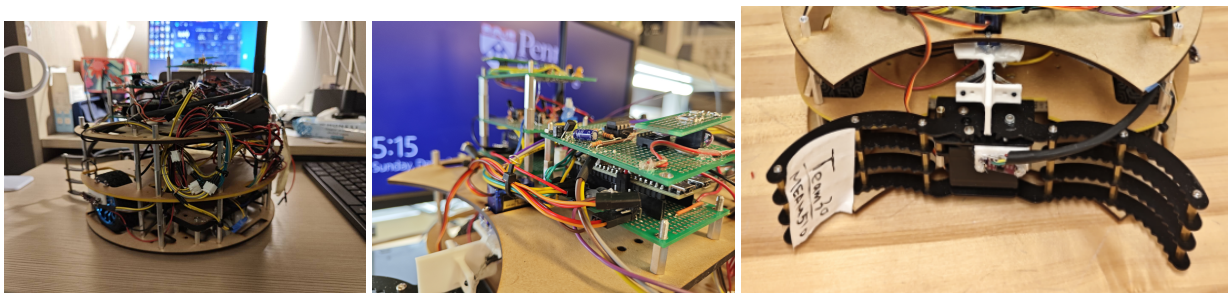
(a) IR, Front View (b) IR, Side View (c) IR, Clear Deck 2^{nd}

Figure 2: IR sensor

2.3 Actual Mechanical Performance

When assembling the mobile base according to our design, we have not encounter any interference, which means that the size of each part is all proper.
The performance of our gripper: as can be seen in figure 3c, when our gripper is open, it is nearly 180 degrees. Therefore, our gripper can push the police car in multiple directions without slipping.
However, there are still some differences between our design and the actual mobile base:

- **The circuit wire on the mobile base:** During the mechanical design stage, we left many holes on the mobile base for wires to come through, so that we could guarantee a clear sight for our IR sensors. In reality, we found that as the angle limit for the servo is 0 ~180 degrees, there's no need for us to use these holes. Thus, we just put all the wires on the rear of our mobile base.
- **The vive position:** In our design, we have prepared a fourth deck for the two vives we use. However, on the real mobile base, we found it more convenient to layer up the perfboards, and put the vives on the highest layer, which is shown in figure 3b.

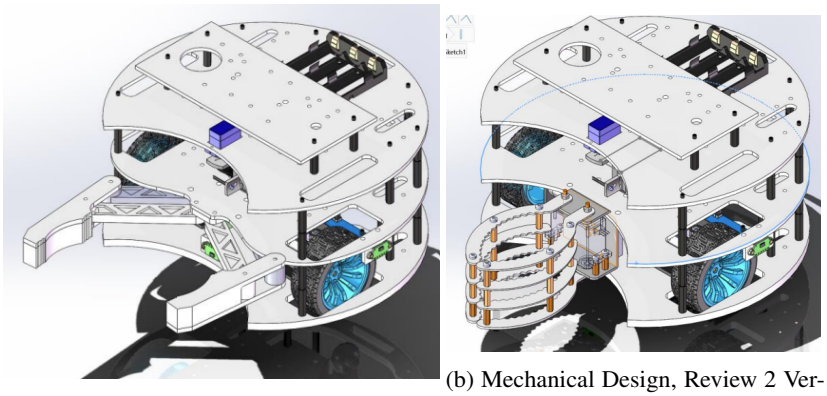


(a) The Wires on the Mobile Base (b) Vive on the Top (c) Gripper, Open State (180 degree)

Figure 3: Actual Mechanical Performance

2.4 Things Tried but Failed

Our mechanical design in review 1 (4a) and the revised design in review 2 (4b) is shown in figure 4:



(a) Mechanical Design, Review 1 Version

(b) Mechanical Design, Review 2 Version

Figure 4: Mobile base Design

As you can see, before the first review, our claw looked different from our final design. In the first stage, we designed a claw based on the four-bar mechanism. However, this claw design cannot be manufactured by laser cutting and we had to use 3D printing, which cannot promise high precision, resulting in hole size nonalignment and interference between parts. Therefore, we decided to change our claw to the one we're using now.

3 Electrical Design

The full circuit schematic 16 is shown in the Appendix

The robot's circuit architecture is meticulously designed to deliver a comprehensive and efficient robotic system. Fueled by an 11.1V 18650 lithium-ion battery, a sophisticated power management module employs both a DC-DC converter and an LDO regulator to ensure stable 5V and 3.3V outputs. The motor control module oversees two 370 motors with encoders, utilizing an H-bridge driver for precise control over movement, while servo motors manage the gripper and an IR detector. The sensor integration module integrates infrared sensors for proximity, an HTC Vive system for accurate position tracking, and a ToF sensor for distance measurement. Two ESP32 development boards serve as the central control, managing sensor data, computations, and communication. The communication module facilitates seamless data exchange among modules, while a safety module monitors battery health and implements an emergency stop mechanism. This modular architecture not only ensures optimal functionality but also provides a structured foundation for development and scalability.

3.1 Power Management Module

Power Source: 11.1V 18650 Li-ion battery Fig. 5.

18650 lithium-ion (Li-ion) batteries offer high energy density, longer lifespan, and rechargeability, making them suitable for mobile robots requiring frequent power cycles. Lithium polymer (LiPo) batteries, with their lightweight and flexible form factor, are advantageous for certain robotic applications but require careful handling due to safety concerns. Lead-acid batteries, though affordable and well-established, are heavy and have a lower energy density. Opting for 18650 batteries in robots provides the benefits of high energy density, longer cycle life, and reusability, making them a reliable and efficient power source for various robotic systems.

Figure 5: 18650 Battery we used for the Project

Voltage Regulation Fig. 6:

- **5A DC-DC Converter:** Steps down voltage from 11.1V to 5V .
- **300 mA LDO (Low Drop-Out) Regulator:** Further regulates voltage to 3.3V.

To calculate the required current, we need to know their operating current specifications.

MG996 Servo:

- Operating Voltage: 4.8V - 6.6V
- Stall Current: 2.5A

SG90 Servo:

- Operating Voltage: 4.8V - 6V
- Stall Current: 0.15A

ESP32:

- Operating Voltage: 5V
- Operating Current (varies): Let's assume around 80mA for estimation purposes.

$$\text{Total Current} = (1 \times 2.5A) + (1 \times 0.15A) + (2 \times 0.08A) = 2.81A \tag{1}$$

Therefore, with one MG996 servo, one SG90 servo, and two ESP32 boards, the total calculated current requirement is 2.81A. This falls within the maximum output capacity of the DC-DC module.

The 3.3v power supply is mainly for the voltage comparator, which consumes almost no current, so the use of LDOs can meet the requirements.

3.2 Motor Control Module

Motor Control Circuit in Fig 7

In order to increase the winding speed and the ability to push through obstacles, we need to level out the motor speed and torque.

Motors: Two 370 motors with encoders (N_0 (no-load speed) is 150 RPM and T_{stall} (stall torque) is 0.5 Nm)

$$\omega = \frac{2\pi \times \text{RPM}}{60} \tag{2}$$

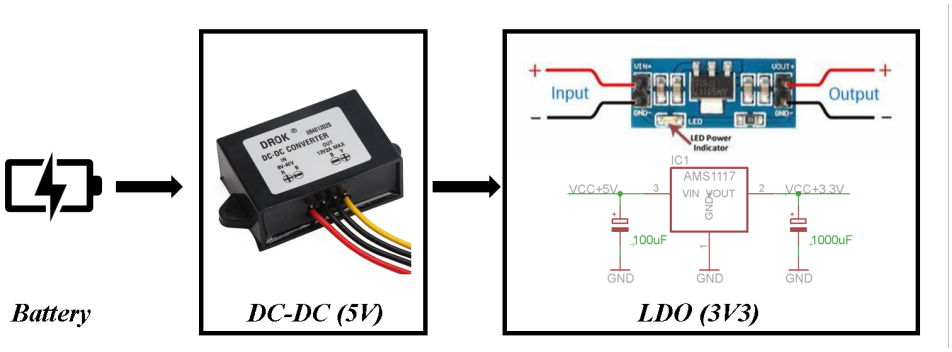


Figure 6: Schematic diagram of the power supply system

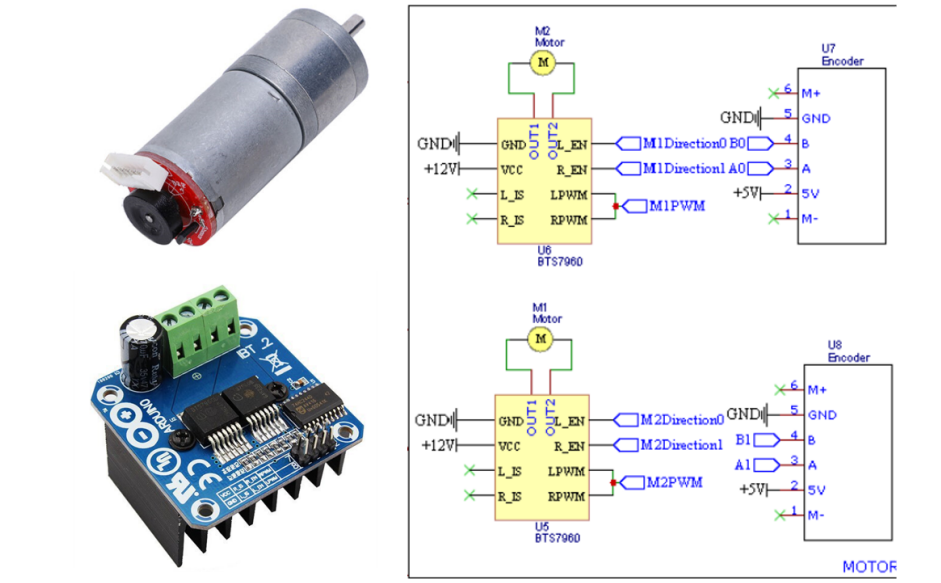


Figure 7: Motor Encoder and H-Bridge Controller

$$\tau = \frac{T_{\text{stall}}}{2}\left(1 - \frac{\omega}{N_0}\right) \tag{3}$$

$$\omega = \frac{2\pi \times 130}{60} \approx 4.25\,\text{rad/s} \tag{4}$$

$$\tau = \frac{0.5}{2}\left(1 - \frac{4.25}{150}\right) \approx 0.2\,\text{Nm} \tag{5}$$

Through calculations, we believe that the design specifications are met.

H-bridge motor driver module: The BTS 7960 is a robust H-bridge motor driver module widely used in robotics and automation projects. This H-bridge driver integrates two high-current half-bridges, allowing bidirectional control of DC motors or other high-current devices. It supports a wide voltage range, typically up to 27V, and is capable of delivering substantial current, making it suitable for driving motors with varying power requirements. The BTS 7960 features overcurrent and overtemperature protection, enhancing the durability of connected components.

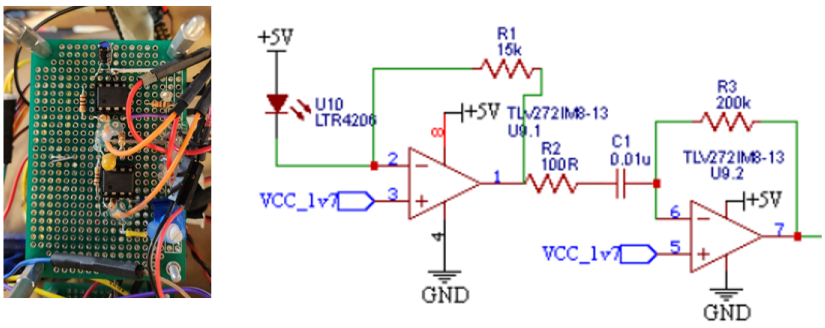


Figure 8: Infrared (IR) Sensor

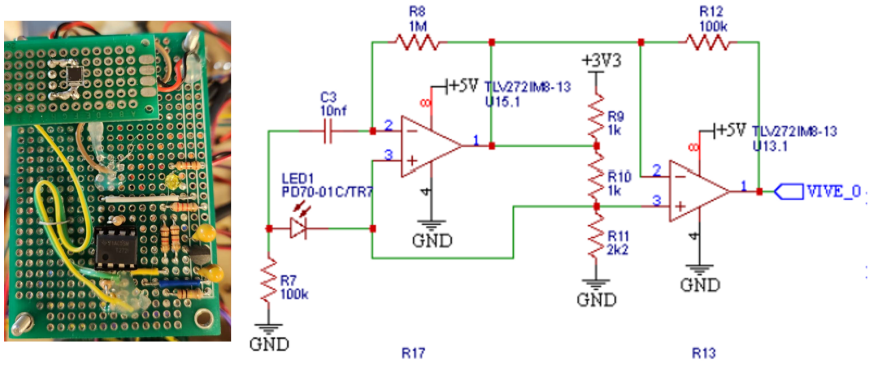


Figure 9: VIVE System

3.3 Sensor Module:

Infrared (IR) Sensors: Fig 8 The infrared (IR) sensor circuit designed for proximity sensing utilizes IR photodiodes as the sensing elements for detecting 550Hz and 23Hz light sources. These IR sensors generate electrical signals in response to the infrared light, and these signals are then processed through an amplification stage to strengthen the initially weak output. Incorporated into the circuit is the TLV272 operational amplifier, configured as a voltage comparator. The TLV272 plays a critical role in comparing the amplified signals with a reference voltage, allowing the circuit to discern between the specified frequencies. This design enables accurate detection of the 550Hz and 23Hz light sources, making it particularly effective for proximity sensing applications where precision in frequency discrimination is crucial for reliable performance.

HTC Vive: Fig 9 HTC Vive utilizes the Lighthouse tracking system for positional tracking in virtual reality (VR). The Lighthouse system employs two rotating laser emitters, called Lighthouse base stations, positioned in opposite corners of the room. These base stations emit laser beams that are picked up by sensors on the Vive headset and controllers. By measuring the time it takes for the laser beams to reach the sensors and calculating the angles, the system accurately determines the precise position and orientation of the VR devices in real-time.

ToF (Time-of-Flight) Sensor: The VL53L0X Time-of-Flight (ToF) sensor is a compact and versatile distance measurement device that utilizes laser-ranging technology to precisely measure distances within a range of millimeters to two meters. Featuring an integrated laser light source, the VL53L0X calculates distances by measuring the time it takes for the emitted light to travel to the target object and back to the sensor.

3.4 Core Board:

Inter-ESP32 Communication: Enables communication between the two ESP32s for data sharing and coordination.Because one ESP32 pin is not enough, so in this project, we use two ESP32s and use the communication protocol to exchange data, the circuit diagram is shown in Figure 10.

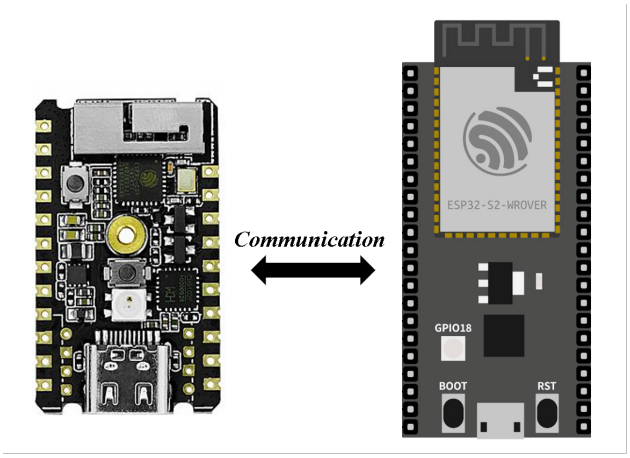


Figure 10: Inter ESP32 communication

3.5 Things We Have Tried But Failed

(1) At first, we intended to use a 7.4v supply, but the voltage was too low and the cart was too slow, so we increased the supply voltage.

(2) At first, we used a 5V LDO instead of a 5V DC-DC, but the current required by the servo was too high and the LDO heated up badly.

(3) We found that reflected infrared light from an object would cause the IR sensor to react incorrectly, so we added a spotting shield and isolation device

4 Processor Architecture and Code Architecture

Link to GitHub Repository: `https://github.com/jbwenjoy/Grand-Theft-Autonomous-30`

We divided our software tasks into the two different boards in Fig. 10. The code file architecture and division is shown in Fig. 11. Our code primarily consists of several modules: sensor information retrieval, various communication activities, behavior tree, and the implementation of actual actions. Among these modules, global variables are utilized to facilitate the interaction of information and connections, ensuring that the robot accurately selects behavior actions based on mode switches and acquire the correct information. We use several header files to define these modules in order to avoid to much code in the main function and .ino file. We will delve into the specific implementation approaches for each module.

4.1 Sensor Data Retrieval

There are four sensors in the robot system. For Vive sensors, we modify vive510.h and vive510.cpp provided in class to get sensor data on ESP32S2 board and then calculate the position as well as the heading angle of the robot. For ToF sensors, we install VL53L0X library and include header file VL53L0X.h to get its data on ESP32C3 board. For the bidirectional Hall encoders of two motors, we read two PWM waves similar to what we did in Lab4. Finally for photosensors, we write program on ESP32C3 to detect certain frequency signals as in Lab2.

It is worth mentioning that the computation of the robot's heading angle was very noisy during our test, especially when the robot is close to the center area of the field. This is probably because the distance between the two vive sensors are small. To solve this problem, we tried to add median filter and statistical outlier removal filter. But this also negatively affected the real-time performance. Therefore, we only take vive position as reliable data. The calculated heading angle is only a reference for roughly deciding the orientation.

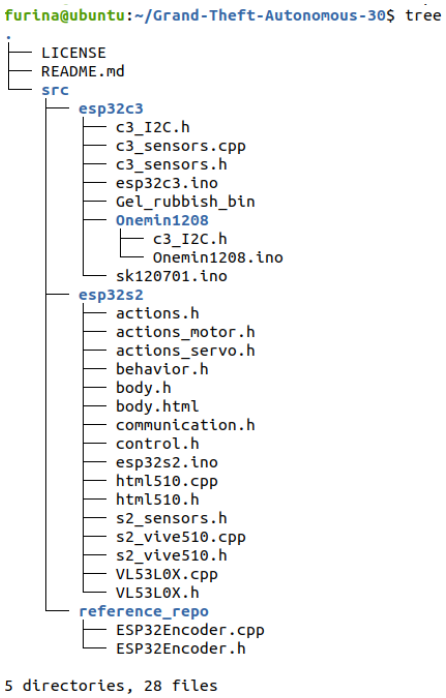


Figure 11: Code File Structure

4.2 Communication

There are three main communication tasks in this project: receive commands from website, broadcast the (x,y) coordinates of the robot and send ESP-now to staff. In website communication, there are behavior mode switches (wall following, police car pushing, trophy moving and manual control mode), action switches (forward, back ward, left, right, stop, speed setting and turning rate setting for manual control mode), jaw state switches (open and close), and speed control slider, as displayed in Fig. 12. In every receiver function of web page communication, an ESP-now message is sent to staff. For broadcasting coordinates, UDP package is sent through ESP32C3 WiFi AP mode.

4.3 Behavior Tree

The behavior tree establishes the complete behavioral logic of the robot system. It runs in each iteration of the main loop. First, command is given through the website to decide the general mode of robot. If the robot is in fully automatic mode, it will detect which action mode to take base on the sensor signals, i.e. the position of the trophies, the position of the police car. If the general mode is manually specified, i.e. wall-following, trophy-moving, car-pushing, it will directly go ahead into that action mode. The structure of the behavior tree program is as Fig. 13.

4.3.1 Wall-following

In wall following mode, the robot moves to the wall and adjust its orientation using Vive position and direction data from two Vive sensors. By utilizing information from both frontal and side-facing ToF sensors, the robot can determine the distance between itself and the walls. This information allows the robot to maintain a proper posture and adjust its direction while moving forward. PID control is also used in our program. The behavior algorithm is in Alg. 1.

4.3.2 Car-pushing

In police car pushing mode, the robot moves to the police car and align with the police car and target direction using Vive data. Then the robot will push the police car straight forward to the other side. When it gets stuck or the police car deviates from the original path, it will readjust the position and orientation by redoing the aligning progress so that it can continue pushing the car from another direction back to the center line. The behavior algorithm is shown in Alg. 2.

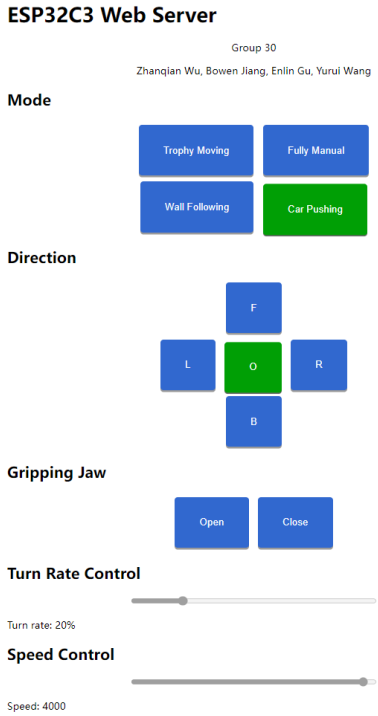


Figure 12: Control Website

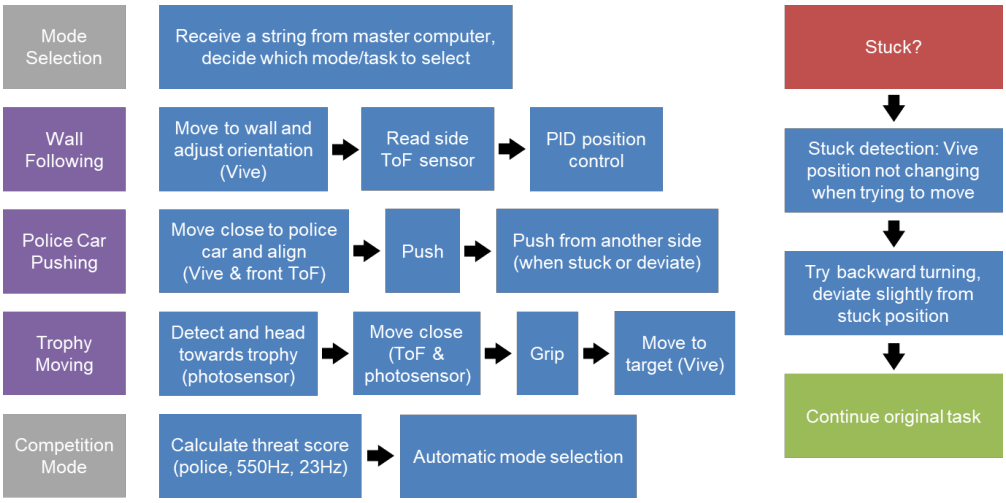


Figure 13: Behavior Tree

4.3.3 Trophy-moving

After enteringthis mode, the robot autonomously rotates at the same place to search for the IR signal of a specific frequency emitted by the trophy. When the trophy's direction is detected, it ceases rotation, aligns itself with the trophy, and advances towards it. The distance to the trophy is measured using ToF sensors. During the approach, if the orientation deviates, the robot initiates rotation again to rediscover the trophy, repeating the process. When it reaches the trophy, the robot stops, closes the jaw to catch the trophy, and takes it to the scoring area. During this process, Vive sensors are employed to calculate the orientation of the robot. The behavior algorithm is in Alg. 3.

Algorithm 1 Behavior Algorithm for Wall-following

```
Initialize dist_to_turn_thres ← 350                                   ▷ distance threshold in mm
Initialize previous_wall_theta
Initialize turn_cycle_count ← 0
procedure WALLFOLLOWING
    if tof_front ≤ dist_to_turn_thres AND NOT needTurnFlag then      ▷ If front distance is too small, turn right
        Set needTurnFlag ← TRUE
    end if
    if OrientationReached(previous_wall_theta + 90, vive_theta) AND needTurnFlag then   ▷ If finish turning, set
needTurnFlag to false, and reset PID
        Set needTurnFlag ← FALSE
        Set previous_wall_theta ← vive_theta
        Reset PID
    end if
    if needTurnFlag then                                              ▷ If need to turn, turn right at the same place
        Turn right at the same place
        Increment turn_cycle_count
        if turn_cycle_count ≥ 3 then
            Set needTurnFlag ← FALSE
        end if
    end if
    if NOT needTurnFlag then      ▷ If don't need to turn, follow the wall, and keep updating previous_wall_theta
        Move forward
        Set turn_cycle_count ← 0
        Set previous_wall_theta ← vive_theta
    end if
end procedure
```

4.3.4 Manual control

In this mode, we use the website to control the robot. The command of moving forward, backward, turning left, right, stop, and jaw actions are implemented through button inputs. The command of setting speed and turning rate are implemented through slider inputs and the values of speed and turning rate are also displayed on the website.

4.4 Action Functions

There are three levels of actions:

1. Basic actions that directly specify motor speeds and servo angles.
2. Medium-level actions, like moving, turning, etc, which are combinations of basic actions and PID feedback.
3. Upper-level actions, including wall-following, car-pushing, etc.

All action functions are encapsulated within different classes, facilitating convenient invocation by the main program. The upper-layer behavior actions are already defined in behavior.h. We also have action.h to define medium-level actions like movements, PID speed calibration and jaw actions, and actions_motor.h/actions_servo.h to define the most basic actions like setting motor speed and servo angle.

Algorithm 2 Behavior Algorithm for Car-pushing

```
Initialize carPushingInitFlag, carPushingApproachCarFlag, carPushingApproachAlignFlag
procedure CarPushing
    if NOT carPushingInitFlag then                                              ▷ Initialization
        Set carPushingInitFlag ← TRUE
        Set carPushingApproachCarFlag, carPushingApproachAlignFlag ← FALSE
        Set push_target_xy ← target_region
        Release Trophy
    end if
    if Is close enough to target then                                           ▷ Stop at target
        Stop action
        Reset flags and set mode to NOTHING
    else
        if NOT carPushingApproachCarFlag then                                   ▷ Approach the car
            Set approach_target_xy ← police_xy - offset
            Update current position and heading using Vive
            Move to approach_target
            if Is close enough to police car then
                Set carPushingApproachCarFlag ← TRUE
            end if
        else if NOT carPushingApproachAlignFlag then    ▷ After approaching, align with the car and target
            Calculate approach_target_theta
            Turn to heading (approach_target_theta)
            if Is at target heading then
                Set carPushingApproachAlignFlag ← TRUE
            end if
        else                                                                    ▷ Start pushing
            Move forward
            Calculate derivation from center line
            if Derivation > threshold then                  ▷ If the car deviates from center line
                Reset alignment flags
            end if
        end if
    end if
end procedure
```

Algorithm 3 Behavior Algorithm for Trophy Moving

procedure TROPHYMOVING(trophy_target_x, trophy_target_y, trophy_freq, approach_only)
  if !trophyLockedFlag **then** ▷ Find trophy of desired frequency
    Turn left at the same place
    if ir_left_freq = trophy_freq AND ir_right_freq $\neq$ trophy_freq **then**
      Turn left at the same place
    else if ir_left_freq $\neq$ trophy_freq AND ir_right_freq = trophy_freq **then**
      Turn right at the same place
    else if ir_left_freq = trophy_freq AND ir_right_freq = trophy_freq **then**
      Stop and delay
      Lock the trophy
    end if
  end if
  if trophyLockedFlag AND !trophyConfirmedFlag **then** ▷ Confirm and lock the trophy
    Confirm trophy direction and update flags
  end if
  if trophyLockedFlag AND trophyConfirmedFlag AND !approachTrophyFlag **then** ▷ Move to the trophy
    Approach the trophy
    Adjust orientation based on IR frequencies
    Update approachTrophyFlag when close enough
  end if
  if trophyLockedFlag AND approachTrophyFlag AND !trophyGrabbedFlag **then** ▷ Grip and detect success
    Attempt to grab the trophy
    Adjust tof_threshold and reset flag as needed
  end if
  if trophyLockedFlag AND approachTrophyFlag AND trophyGrabbedFlag **then** ▷ Trophy delivery
    Move to the target position
    Release trophy if lost during movement
    Reset flags and mode when at target position
  end if
end procedure

5 Conclusion

The mechanical design of the mobile base, with its round shape and multi-deck structure, demonstrated a thoughtful approach to functionality and task-specific requirements. The electrical design, encompassing a sophisticated power management module, motor control circuitry, and sensor integration, contributed to the overall reliability and efficiency of the robotic system.

The behavior tree logic proved instrumental in managing different operational modes, ranging from wall-following and car-pushing to trophy-moving and manual control. The modularity of the architecture, particularly the inter-ESP32 communication, facilitated seamless data exchange and coordination between various components.

The detailed discussion on action functions, spanning basic, medium-level, and upper-level actions, highlighted the hierarchical organization of the system's capabilities. The successful implementation of these actions in diverse scenarios underscored the adaptability and versatility of the designed robot.

The inclusion of video links and an extensive appendix, featuring a bill of materials, circuit diagrams, and dimensioned drawings, adds transparency and completeness to the presented work. The project's success in achieving its objectives is evident in the videos demonstrating various functionalities.

In essence, this paper contributes valuable insights into the intricacies of designing and implementing a multifunctional robotic system. The combination of mechanical, electrical, and software components in a well-integrated framework demonstrates the feasibility of creating versatile robots for real-world applications. Future work may involve further optimization, scalability, and the integration of additional features to enhance the robot's capabilities in various environments.

6 Appendix

FINAL PROJECT TEAMWORK

Purchase Order

Item #	Description	Payer	URL	Qty	Unit price	Total price	Arrived
1	DC 12V DIY Encoder Gear Motor	JBW	Amazon.com: DC12V Encoder Gear M	2	$18.46	$36.92	y
2	TOF	GEL	Amazon.com: Onyehn 3pcs VL53L0X T	1	$11.99	$11.99	y
3	MG996R Servo (2 Pcs)	GEL	https://www.amazon.com/gp/product/	1	$13.99	$13.99	y
4	High Power Motor Driver (3 Pcs)	JBW	https://www.amazon.com/gp/product/	1	$12.79	$12.79	y
5	ESP32-s2	GEL	https://www.amazon.com/gp/product/	2	$8.00	$16.00	y
6	Direct Current Converter (12V to 5V 3A)	WYR	https://www.amazon.com/gp/product/	1	$6.79	$6.79	y
7	Direct Current Converter (8-40V to 5V 5A)	JBW	https://www.amazon.com/gp/product/	1	$6.37	$6.37	y
8	Connector Kit	WYR	https://www.amazon.com/gp/product/	1	$9.99	$9.99	y
9	2.54mm Male & Female Header Socket	GEL	https://www.amazon.com/DAOKI-2-54	1	$6.29	$6.29	y
10	18650 Battery Holder (12Pcs)	WYR	https://www.amazon.com/Battery-Sto	1	$8.99	$8.99	y
11	Heatsink	WYR	Amazon.com: M.2 Copper Heatsinks C	1	$7.19	$7.19	y
12	WiFi Antenna	JBW	https://www.amazon.com/dp/B07TDN	1	$8.49	$8.49	n
13	3D Printing Parts	-					y
14	Laser Cutting Parts	-					y
15	Screws, nuts, glue.	-					y

Subtotal	$145.80
Shipping & handling	$0.00
Tax rate	6.00%
Sales tax	$8.75
	$154.55

Figure 14: Bill of materials

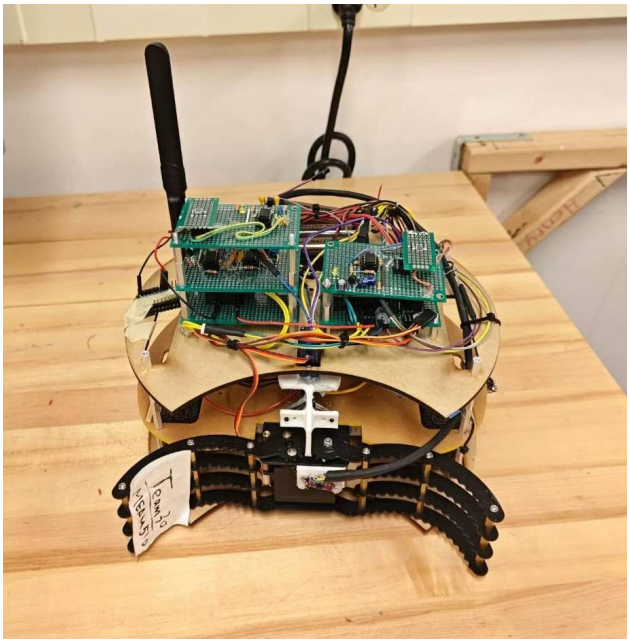


Figure 15: Robot photo

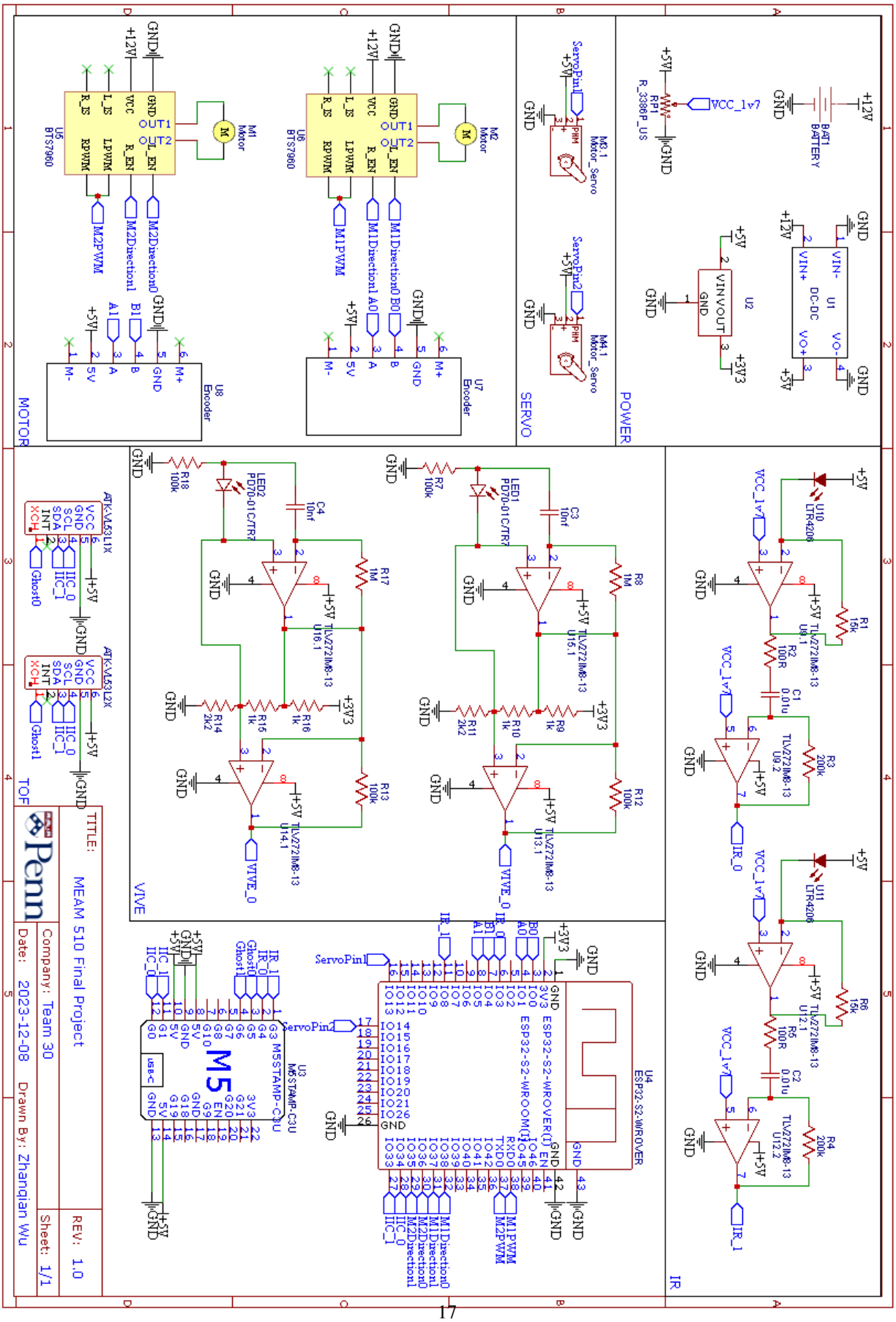


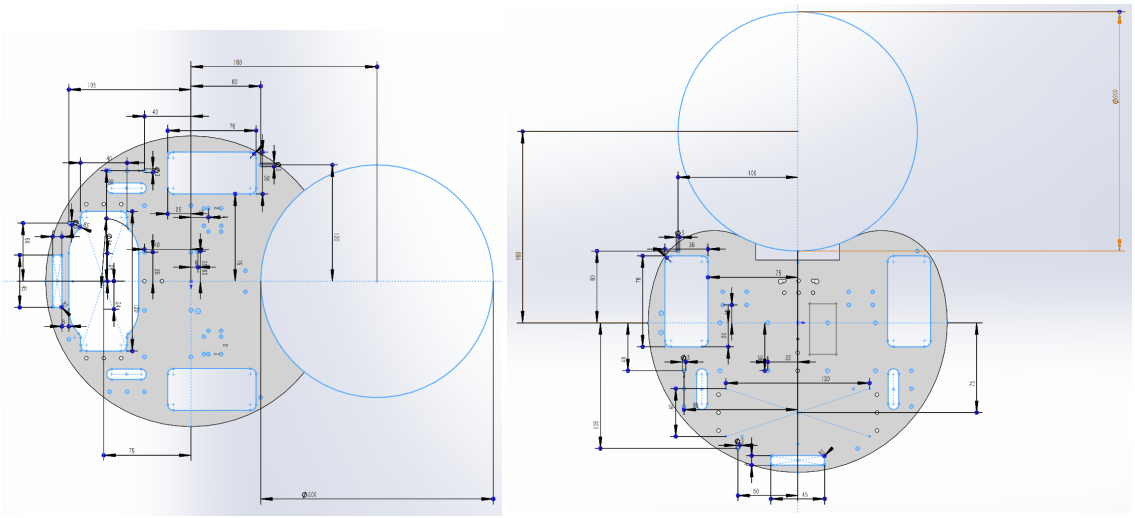
Figure 16: Circuit Diagram

Video Links:

1. CHECK OFF: `https://www.youtube.com/watch?v=02UyDjv6waM`

2. IR CIRCUIT TEST: `https://youtu.be/ctNg3ylUBPU`

3. VIVE CIRCUIT TEST: `https://youtu.be/0ITU11SvNJ4`



(a) Dimensioned drawing for Base Board

(b) Dimensioned drawing for 2^{nd} Board

Figure 17: Dimensioned drawing 1

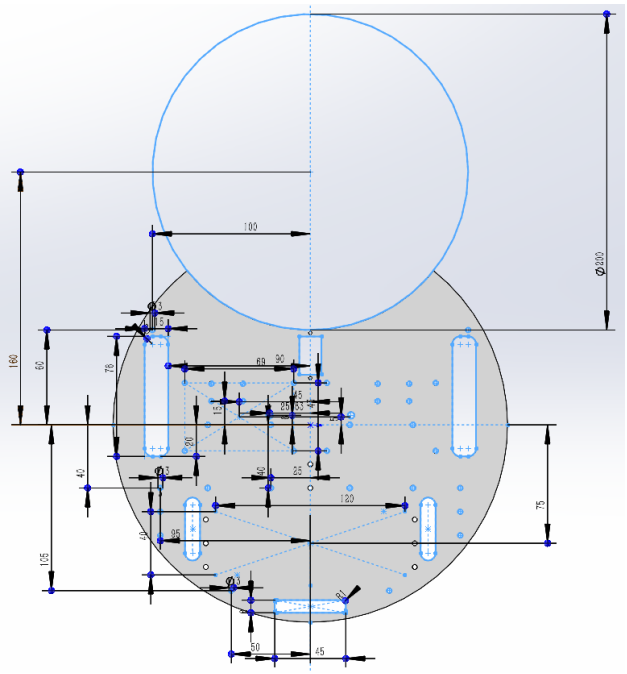


Figure 18: Dimensioned drawing for 3^{rd} Board

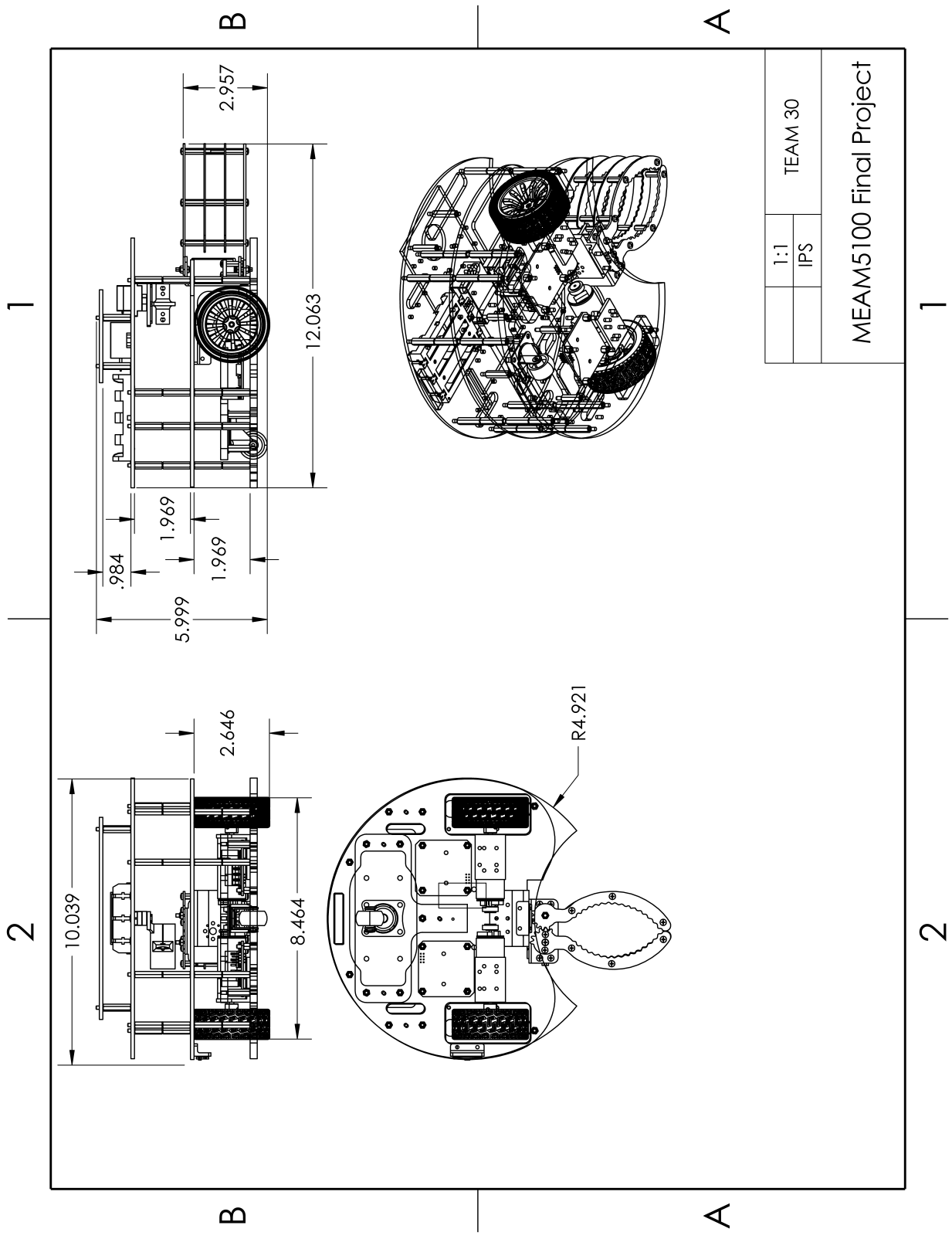


Figure 19: Dimensioned drawing for The Whole Mobile Base