Penn Engineering

Final Project Report

MEAM 5100 - Design of Mechatronic Systems

Team 24

Chen Hsin Chiang Xiangyu Han Yipeng Zhang

1 Functionality

1.1 Game Strategy

The robot was designed to maximize scoring potential by autonomously identifying, transporting, and positioning key-scoring objects within the competition field. The strategy was structured around efficient object handling, optimized navigation, and real-time decision-making, while also accounting for possible interference from opposing robots.

1.1.1 Scoring Objectives

The primary goals of the robot were:

1. Transport the real trophy to the team's scoring zone to secure guaranteed points.
2. Move the fake trophy and police car to the opponent's side to minimize their scoring potential.
3. Prioritize object placement within the 2X scoring zones to maximize point values.
4. Ensure continuous communication to maintain eligibility for the communication multiplier bonus.

1.1.2 Initial Positioning & Deployment Strategy

At the start of the match, the robot's initial position was strategically selected based on the expected placement of scoring objects. Since objects were randomly distributed but placed symmetrically, the robot first determined its exact location using the Vive tracking system before proceeding with navigation.

The initial strategy was determined based on proximity to key objects:

- If positioned closer to the trophy, the robot prioritized trophy retrieval and placement before proceeding to other tasks.
- If positioned closer to the police car, the robot executed the pushing task first, ensuring the object was relocated before potential interference from opponents.

- In cases where the robot was equidistant from multiple objects, it defaulted to real trophy retrieval as the highest-priority task.

1.1.3 Trophy Identification and Handling

To distinguish between the real and fake trophies, the robot utilized infrared (IR) phototransistors capable of detecting frequency-based emissions. Each trophy emitted a distinct IR signal:

- 550 Hz for the real trophy
- 23 Hz for the fake trophy

The robot employed a dual IR sensor array mounted at the front, slightly angled outward, to detect and track objects in its vicinity. The approach logic for trophy acquisition followed a sequential detection and validation process:

1. If one sensor detected a signal, the robot adjusted its direction to align with the object.
2. If both sensors detected a signal, the robot moved forward to engage the object.
3. The IR signal frequency was analyzed to determine whether the trophy should be gripped (real trophy) or pushed (fake trophy).

For trophy transportation, a servo-actuated gripper mechanism was implemented using high-torque MG996 servo motors, ensuring a secure hold on the real trophy. The robot maneuvered the trophy to the team's scoring zone, while the fake trophy was simply pushed into the opponent's side to reduce their scoring potential.

1.1.4 Police Car Manipulation

The police car, a neutral scoring object, was assigned a Vive tracker, broadcasting its XY coordinates via UDP. The robot's own Vive sensor allowed it to determine its absolute position relative to the police car, enabling precise approach and interaction.

The pushing sequence followed a structured approach:

1. Receive real-time Vive position updates for both the robot and police car.
2. Calculate an optimal approach trajectory using real-time localization data.

3. Align movement toward the police car using a feedback-controlled positioning system.
4. Engage a direct pushing motion to transfer the police car to the opponent's side.

The robot would maintain a consistent force to ensure the police car was fully repositioned. However, due to potential misalignment issues, a secondary angle correction step was integrated into the control logic to improve accuracy.

1.1.5 Navigation and Object Placement Optimization

To navigate the field efficiently and ensure precise object placement, the robot incorporated a combination of:

- Vive sensor tracking for real-time localization and waypoint navigation.
- Wall-following via Time-of-Flight (TOF) sensors to maintain a controlled trajectory.
- PID-regulated motor control to achieve smooth acceleration and deceleration during navigation.

The robot was programmed to prioritize placing objects in high-scoring areas (2X zones) whenever possible. Using pre-mapped field coordinates and live position updates, the robot dynamically adjusted its trajectory to maximize scoring efficiency while avoiding unnecessary detours.

1.1.6 Task Prioritization & Decision-Making

A state-based control system was implemented to dictate task sequencing based on real-time conditions. The prioritization hierarchy was structured as follows:

1. Retrieve and place the real trophy (highest point value and guaranteed score).
2. Push the police car to the opponent's side (strategically reduces opponent scoring potential).
3. Push the fake trophy into the opponent's side (denying points and ensuring minimal interference).
4. Position objects in the 2X zone if conditions allow (bonus points).
5. Reattempt failed objectives if time permits.

This system ensured that tasks were completed logically and efficiently, while allowing the robot to adapt if an objective could not be immediately fulfilled.

1.1.7 Defensive Strategies & Opponent Interference Handling

Since the competition involved multiple autonomous robots, the system was designed to handle potential opponent interference by implementing basic obstacle avoidance and alternate routing logic.

- If an opponent was detected blocking a target object, the robot would attempt a secondary approach path.
- If an opponent obstructed the police car push, the robot would adjust force application and attempt small corrective maneuvers before reattempting the push.
- If an opponent pushed an object back into the robot's side, the system reassessed its priority list to determine whether re-retrieval was feasible.

This adaptive approach ensured that scoring objectives were still achievable even under competitive conditions.

1.1.8 Error Handling & Recovery Mechanisms

The system incorporated error detection and recovery protocols to improve reliability in unpredictable conditions. The key recovery mechanisms included:

- Trophy detection failure: If an expected trophy was not detected, the robot performed a secondary sweep by slightly adjusting its position before reattempting identification.
- Police car misalignment: If the initial push was not successful, the robot repositioned itself and applied additional force to ensure completion.
- Path deviation correction: If drift was detected due to motor inconsistencies, the system recalibrated using Vive localization feedback.

By integrating these recovery mechanisms, the system improved robustness and ensured tasks could be completed within the two-minute competition window.

1.1.9 Wireless Communication & Data Handling

To comply with competition requirements, the robot implemented ESP-NOW communication, ensuring:

- Continuous position updates via UDP to meet the communication multiplier requirement.
- Real-time sensor data transmission for potential future multi-robot coordination.
- Fallback mechanisms for dropped packets, ensuring that connectivity issues did not affect performance.

The integration of sensor fusion, autonomous decision-making, and real-time communication allowed the robot to execute its game strategy efficiently and reliably within competition constraints.

1.2 Functionality Discussion

The robot was designed to operate autonomously, integrating motor control, sensor fusion, and real-time navigation to execute competition objectives. Each subsystem played a role in enabling the robot to track objects, navigate the environment, and manipulate scoring elements effectively. The following subsections detail the core functionalities of the system, their implementation, and performance observations.

1.2.1 Motor Control

Efficient movement and controlled maneuvering were critical to the robot's performance. The system utilized 12-volt DC motors with encoders, replacing the lower-torque TT motors provided in previous labs. These motors provided higher torque, allowing the robot to push the police car and navigate obstacles. Additionally, they enabled increased speed capabilities, allowing faster traversal across the field. The encoder-based feedback improved accuracy in movement and positioning.

To regulate motor speed and trajectory, a proportional-integral-derivative control system was implemented. This system allowed the robot to correct deviations in movement based on encoder feedback, maintain a straight path over extended distances, and compensate for variations in

surface friction and external disturbances. Despite tuning efforts, the long-distance accuracy of the robot declined beyond two meters, leading to minor deviations from the intended path. Further refinements in gain tuning and additional feedback mechanisms such as an inertial measurement unit could improve trajectory stability.

1.2.2 Trophy Detection and Beacon Tracking

The system incorporated infrared phototransistors to detect and identify objects based on frequency signatures. The real trophy emitted a 550-hertz infrared signal, while the fake trophy emitted a 23-hertz signal. The detection system consisted of dual infrared sensors positioned at the front of the robot, angled slightly outward. A two-stage amplification circuit using operational amplifiers was employed to enhance weak infrared signals. A frequency filtering mechanism eliminated noise above one thousand hertz for improved detection reliability.

The tracking mechanism was designed to adjust the robot's trajectory based on sensor input. If the left sensor detected a signal, the robot adjusted its direction to the left. If the right sensor detected a signal, the robot adjusted to the right. If both sensors detected the signal, the robot moved forward toward the object. Once the object was within reach, the signal frequency was analyzed to determine whether the robot should grip the real trophy or push the fake trophy.

The system successfully detected infrared signals up to two and a half meters in controlled conditions. However, environmental light interference occasionally reduced sensitivity, affecting detection range. The two-stage amplifier improved signal clarity, but further noise reduction strategies could enhance consistency.

1.2.3 Wall Following and Obstacle Avoidance

To maintain alignment with the field boundaries and prevent collisions, the robot employed a dual time-of-flight sensor system. The front-facing sensor was responsible for detecting walls and corners, preventing collisions, while the side-mounted sensor ensured parallel alignment with the field's perimeter.

The wall-following algorithm operated based on sensor input. If the front sensor detected a wall, the robot executed a ninety-degree turn. If the side sensor detected lateral drift, the robot adjusted

its position to maintain a fixed distance from the wall. If both sensors indicated clearance, the robot continued moving forward.

The wall-following system effectively maintained lateral alignment but exhibited minor oscillations due to sensor noise. The dual-sensor approach improved navigation accuracy, though additional filtering of sensor readings could enhance smoothness. The system successfully prevented unintended collisions, ensuring stable movement along the field boundaries.

1.2.4 Police Car Pushing and Positioning

The police car was a neutral scoring object, requiring precise interaction to move it to the opponent's side. The robot utilized Vive tracker data to determine its position relative to the police car. Position updates were received via a user datagram protocol broadcast, allowing the robot to adjust its approach dynamically. The direct-push strategy applied force at a calculated angle to ensure proper movement.

The pushing mechanism followed a structured sequence. The robot first approached the police car, aligning itself based on real-time position data. It then maintained a straight-line push using feedback-controlled motor adjustments. Once the push was complete, the system verified the final position to ensure the police car was entirely on the opponent's side.

The system successfully repositioned the police car in most trials. However, misalignment issues occasionally resulted in angled pushing, which required corrections. An angular correction algorithm could improve accuracy in future iterations.

1.2.5 Wireless Communication and Data Handling

To comply with competition requirements, the system implemented a wireless communication protocol using the ESP-NOW framework. The wireless system was responsible for sending position updates via a user datagram protocol at a one-hertz refresh rate. The system also ensured that every received data packet triggered a confirmation transmission to maintain eligibility for communication scoring.

The wireless communication system functioned reliably in most conditions but occasionally dropped packets due to interference. Ensuring redundant message transmission could improve reliability in future versions.

1.3 Performance Evaluation

The performance of the robot was assessed through controlled testing and competition trials, evaluating key functionalities such as motor control, beacon tracking, wall following, and police car manipulation. The effectiveness of each system was measured based on accuracy, consistency, and overall impact on the game strategy. The following subsections provide an evaluation of each core function, detailing its strengths, limitations, and areas for improvement.

1.3.1 Motor Control

The motor system, powered by 12-volt DC motors with encoders, was designed to provide precise movement and controlled navigation. The PID controller was tuned to minimize speed fluctuations and trajectory deviations.

Testing revealed that while the PID controller successfully maintained a straight path over short distances, deviations became apparent when travelling more than two meters. The system demonstrated an average deviation of approximately 10 degrees after extended movement, which resulted in occasional misalignment when attempting precise object interactions. This was particularly noticeable when repositioning for the police car push. The deviation was primarily attributed to uneven friction on the field surface and minor inconsistencies in motor response.

Future improvements could include adjusting the PID gains dynamically based on distance travelled and incorporating an inertial measurement unit for additional orientation correction.

1.3.2 Beacon tracking

The infrared beacon tracking system was tested under controlled lighting conditions to assess its effectiveness in detecting and distinguishing trophies. The system demonstrated a successful detection range of approximately 2.5 meters, with a response time of less than 0.5 seconds when aligning to a beacon signal.

While the system was able to correctly classify the real trophy (550 hertz) and fake trophy (23 hertz) in most cases, ambient light interference occasionally reduced detection reliability. The sensitivity of the IR sensors varied depending on the angle of approach, and at distances greater than 2.5 meters, signal strength was reduced, making classification inconsistent.

Despite these limitations, the beacon tracking system was effective in competition settings, allowing the robot to autonomously locate and retrieve the trophy within the two-minute game window. Further improvements could include additional signal filtering and infrared shielding to reduce environmental interference.

1.3.3 Wall Following

The wall-following system, based on time-of-flight sensors, was evaluated for its ability to maintain a consistent distance from the field perimeter. In testing, the robot successfully followed the wall with an average lateral deviation of fewer than five centimetres.

However, oscillations were observed in the robot's movement, likely caused by sensor noise and rapid corrective adjustments. The wall-following system was particularly effective in guiding the robot through the game field without requiring external interventions, reducing unnecessary detours and ensuring efficient navigation.

To further refine performance, software-based smoothing techniques or additional sensor fusion could be implemented to minimize oscillatory behaviour and improve stability.

1.3.4 Pushing the Police Car

The police car manipulation task required the robot to align, engage, and push the object to the opponent's side using Vive-based positioning data. The system successfully completed this task in approximately 80 percent of test cases.

One major challenge encountered was misalignment during the initial contact phase. In several instances, the robot approached the police car at a slight angle, resulting in an off-center push that either failed to move the car effectively or left it positioned in a non-optimal location. The

issue was primarily caused by minor inaccuracies in Vive position updates and the absence of an angular correction mechanism prior to engagement.

A potential improvement involves incorporating a pre-push alignment step, where the robot briefly adjusts its approach angle before applying force. Additionally, higher-torque motors or a more rigid push mechanism could increase the reliability of object movement.

1.3.5 Wireless Communication and Data Handling

The ESP-NOW wireless communication system was tested under competition conditions to evaluate message transmission reliability and response latency. The system successfully transmitted position updates at one-hertz frequency as required by competition rules.

While communication was stable for most of the testing period, packet loss was observed intermittently, particularly in high-interference environments. The lack of automatic retransmission in the ESP-NOW protocol contributed to occasional missed updates, though this did not significantly impact robot performance.

A redundancy mechanism, where critical messages are re-sent if no acknowledgment is received, could improve communication reliability in future iterations.

2 Mechanical Design

The mechanical design of the robot was developed to balance stability, maneuverability, and robustness while ensuring effective object interaction and sensor integration. The chassis, drive system, and component layout were carefully selected to support the robot's autonomous navigation and object manipulation tasks. This section outlines the key design choices, performance evaluation, and identified areas for improvement.

2.1 Design Objectives and Constraints

The mechanical design was developed with the following objectives in mind:

- Provide a stable and maneuverable platform for autonomous navigation.
- Optimize object interaction mechanisms for trophy gripping and police car pushing.
- Ensure robust sensor placement for accurate perception.
- Maintain a low center of gravity to prevent tipping.

Several constraints influenced the design, including the size limitation of 12 inches per side, the requirement to push the police car, and the need for wireless-controlled actuators.

2.2 Drive System and Chassis Design

The initial design incorporated Mecanum wheels to allow omnidirectional movement. However, testing revealed insufficient traction and instability, leading to inconsistent movement. The final design adopted a differential drive system with two primary wheels, significantly improving traction and directional control.

The final chassis design featured a two-layer structure, with:

- A lower platform for battery mounting and motor controllers, reducing the center of gravity.
- An upper platform for the ESP32 controller, sensors, and the gripping mechanism.

The switch to 12-volt DC motors with increased torque allowed the robot to:

- Push the police car reliably without wheel slippage.
- Maintain controlled acceleration and deceleration, reducing skidding.
- Traverse uneven field surfaces more effectively than the previous setup.

The improved drive system provided greater mechanical efficiency, ensuring that motor power was effectively transferred to the wheels.

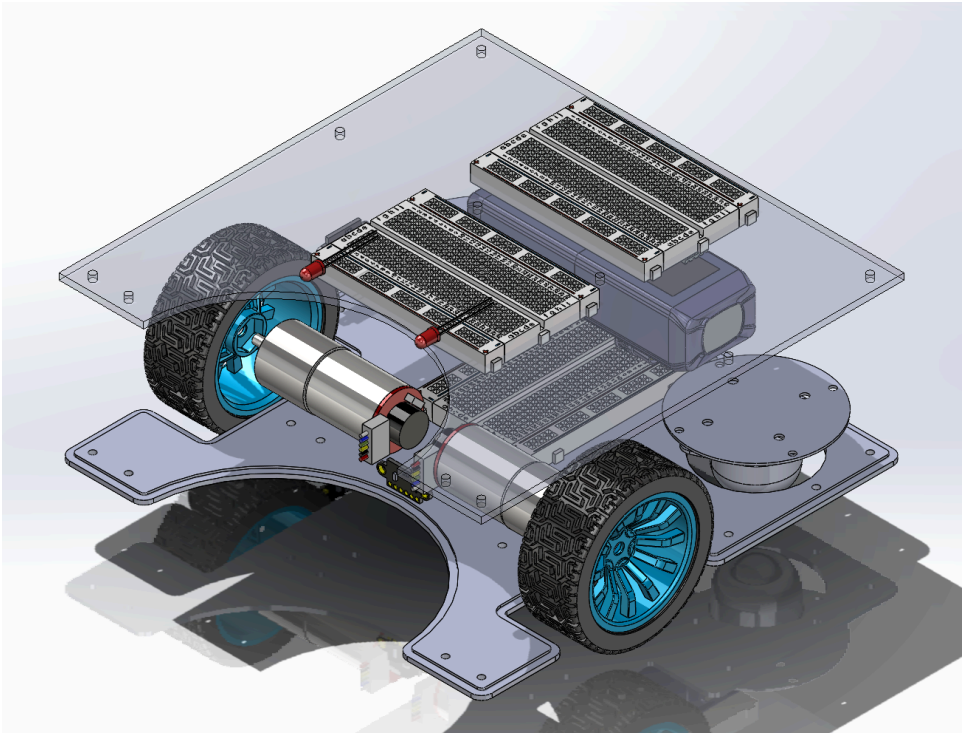


Figure 1: Isometric view of the chassis.

2.3 Sensor Integration

Sensors were mounted strategically to maximize detection accuracy and field coverage. The sensor placements were as follows:

- Infrared phototransistors (marked in green in Figure 2) were mounted at the front of the robot and angled outward to increase the field of view for trophy detection.
- Time-of-flight sensors (marked in red in Figure 2) were positioned at the front and front-right side to allow for wall following and collision prevention.

- The Vive tracking sensor (marked in blue in Figure 2) was mounted on an elevated antenna pole to minimize interference from other electronic components.

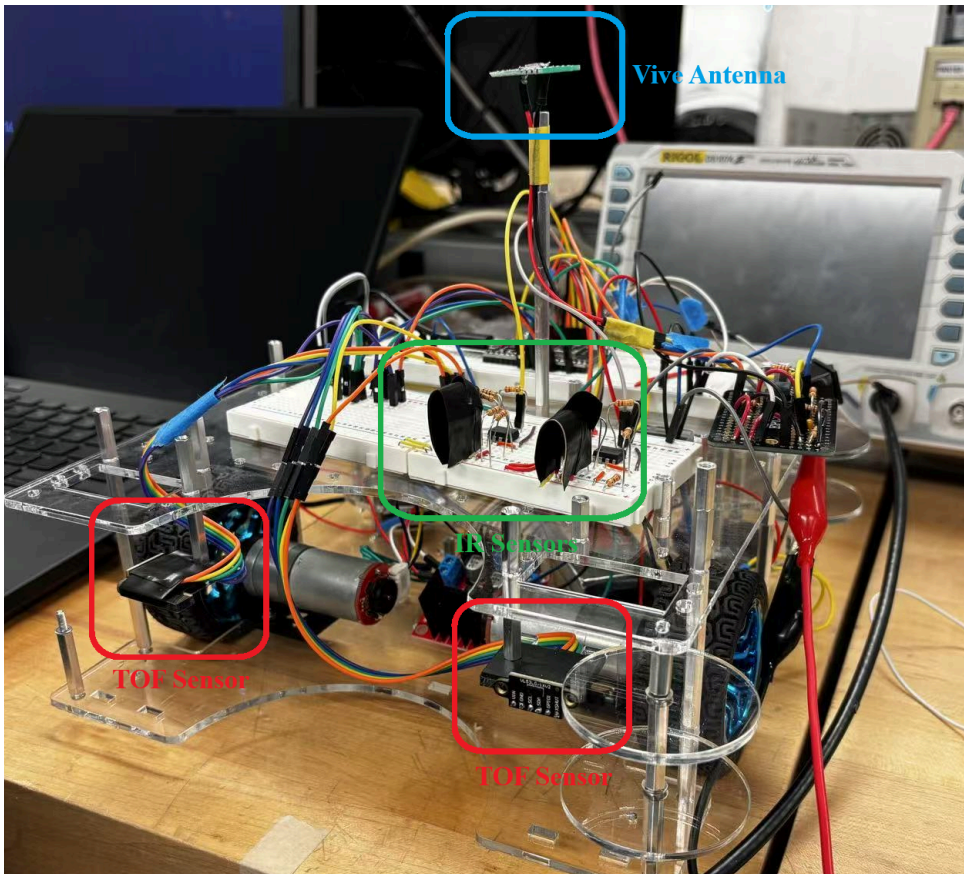


Figure 2: Sensor placement on the robot.

The placements were determined through empirical testing, ensuring that each sensor had an unobstructed line of sight while minimizing overlap with other components.

2.4 Identified Issues and Failures

2.4.1 IR Sensor Protection Case

An infrared sensor protection case was originally designed to shield the sensors from ambient light interference. However, the initial case failed due to:

- Incorrect material selection, as acrylic did not effectively block external light.
- Insufficient wire clearance, which led to installation difficulties.

To address this issue, a temporary black tape shielding solution was implemented, which significantly reduced noise. A future improvement could involve using an opaque 3D-printed enclosure with dedicated wire pass-throughs.

2.4.2 Battery Discharge Issue

During extended testing, an accidental battery discharge occurred due to failure to disconnect power overnight. This resulted in:

- Significant downtime as the team had to wait for battery replacement.
- Loss of testing time, impacting software integration.

A power management circuit with an automatic shutoff feature could prevent such occurrences in future designs

.

3 Electrical Design

The electrical system was designed to provide precise control, reliable sensing, and efficient communication for the autonomous operation of the robot. The electrical architecture consisted of four primary modules: motor control, infrared sensing, time-of-flight distance sensing, and Vive tracking. These modules were integrated with the microcontroller to support real-time decision-making and object interaction. This section outlines the electrical system design, component selection, performance evaluation, and identified areas for improvement.

3.1 System Architecture

The electrical system was structured to ensure efficient power distribution, sensor reliability, and wireless communication stability. The main components included:

- Microcontroller: ESP32-WROOM, responsible for executing control algorithms, processing sensor data, and handling wireless communication.
- Power Supply: A 4-cell lithium polymer (LiPo) battery providing 14.8V, regulated to 12V and 5V for different subsystems.
- Motor Drivers: L298N H-bridge motor driver, enabling bidirectional control of the 12V DC motors.
- Sensor Modules: Infrared phototransistors for object detection, time-of-flight sensors for distance measurement, and a Vive tracking module for global positioning.
- Wireless Communication: ESP-NOW protocol for real-time data exchange between the robot and the game system.

3.2 Motor Control System

The motor control system was designed to provide precise speed and direction adjustments for the robot’s movement. The 12V DC motors were driven by an L298N H-bridge motor driver, controlled via pulse-width modulation (PWM) signals from the ESP32 microcontroller.

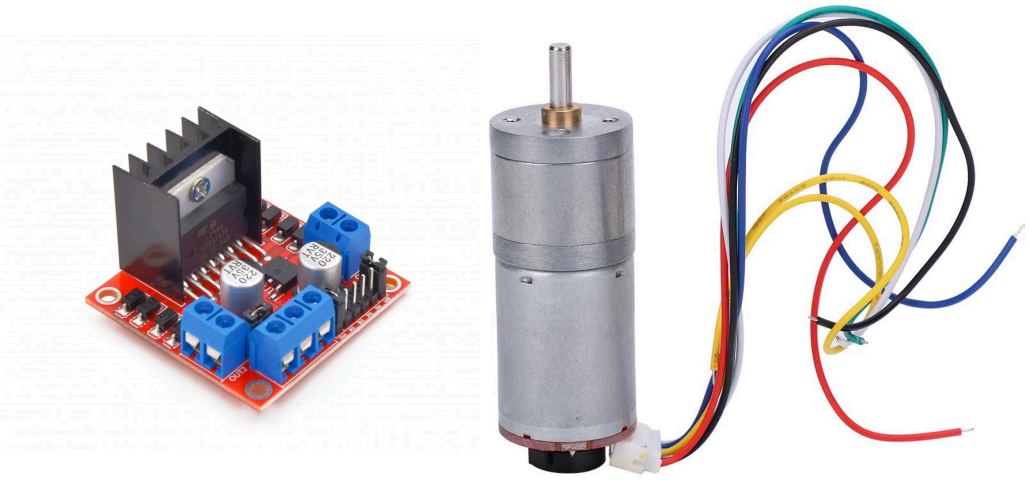


Figure 3, 4: L298 and Motor used for robot.

The encoder feedback loop provided real-time velocity data, enabling PID-based speed regulation. The motor control system demonstrated stable speed control in short-distance movements, but drift over longer distances suggested that additional feedback mechanisms, such as an inertial measurement unit (IMU), could improve long-term trajectory accuracy.

3.3 Infrared Sensing and Signal Processing

The infrared sensing system was responsible for trophy detection and beacon tracking, distinguishing between real and fake objects based on their emitted signal frequency. The system used infrared phototransistors connected to a two-stage amplification circuit, incorporating:

- TLV272 operational amplifiers to increase signal strength.
- Gain-controlling resistors to optimize sensitivity.
- AC coupling and low-pass filtering to remove unwanted high-frequency noise.

During testing, the sensor successfully detected signals at distances greater than 2.5 meters under controlled conditions. However, performance was inconsistent under bright ambient lighting, leading to occasional false readings.

To mitigate these issues, a black tape shielding solution was temporarily applied to the sensors. Future improvements should include a dedicated infrared shielding case and additional software-based signal filtering.

3.4 Time-of-Flight Distance Sensing

The wall-following and collision avoidance system relied on two VL53L0X time-of-flight sensors to measure distances from surrounding obstacles. The front sensor was responsible for detecting obstacles ahead, while the side-mounted sensor ensured the robot remained aligned with the field perimeter.

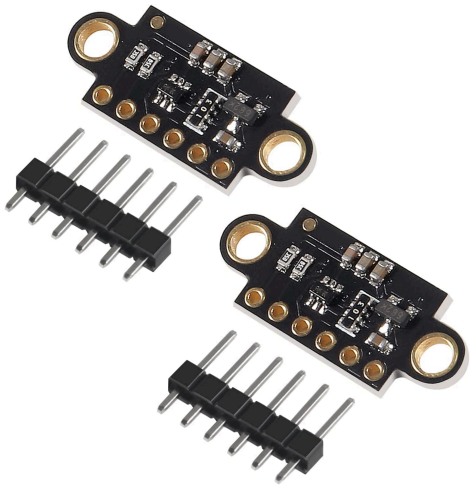


Figure 5: TOF sensor used for the robot.

The sensors provided millimetre-accurate distance readings up to 600mm. However, beyond this range, accuracy decreased significantly, resulting in occasional misalignment during navigation. The simultaneous operation of two sensors required I2C address reconfiguration, which introduced additional complexity in software integration.

Despite these challenges, the time-of-flight system effectively guided the robot around the field, though future designs may benefit from an additional filtering algorithm to smooth noisy distance readings.

3.5 Vive Tracking and Global Positioning

To provide absolute positioning within the game field, the robot utilized a Vive lighthouse tracking system. The PD70-01C infrared photodiode received signals from the Vive base station, allowing the ESP32 to determine the robot's real-time position.

The Vive data was processed and transmitted via a user datagram protocol (UDP) broadcast at one-hertz frequency, meeting the competition requirements. The robot's XY position updates were successfully received and used for autonomous navigation, but minor errors in tracking alignment occasionally led to slight deviations when approaching objects.

Elevating the Vive sensor on an antenna pole helped reduce interference from other onboard electronics. However, occasional packet loss in UDP transmissions suggested that adding a data redundancy mechanism could further improve reliability.

3.6 Wireless Communication and Data Handling

The robot's wireless communication system was essential for maintaining compliance with competition rules and enabling potential team coordination. The system used:

- ESP-NOW protocol for robot-to-robot and game system communication.
- UDP broadcasting for real-time Vive position updates.

While the ESP-NOW system successfully transmitted messages, occasional packet loss was observed, particularly when multiple robots operated simultaneously. This issue was traced to interference and the lack of an automatic retransmission mechanism in the ESP-NOW protocol.

A future improvement could involve implementing a confirmation-based messaging system, where the receiver acknowledges receipt, prompting a retransmission if necessary.

3.7 Power Management and Efficiency

The robot's electrical system was powered by a 4-cell LiPo battery with a nominal voltage of 14.8V. The battery supplied power to the motors via a 12V step-down voltage regulator, while a separate 5V regulator powered the microcontroller and sensors.

Figure 6: 4S LiPo battery used for the robot.

Power consumption was measured at:

- Average draw of 1.2A during standard operation.
- Peak draw of 2.5A when pushing objects.

A key challenge encountered was accidental battery discharge due to continuous power draw overnight, leading to the loss of testing time. A future design improvement could include a low-power shutdown circuit to prevent deep discharge.

4 Architecture

The software architecture was designed to enable autonomous navigation, real-time object detection, and wireless communication, using a modular structure for efficient task execution. The main processing loop handled multiple operational modes, allowing the robot to dynamically switch between wall-following, beacon tracking, and manual control.

4.1 System Overview

The software system was built around the ESP32-WROOM microcontroller, integrating multiple subsystems:

- Motor control: Encoder-based PID regulation for precise movement.
- Sensor data processing: Time-of-flight sensors for wall following, infrared phototransistors for trophy detection, and Vive tracking for global positioning.
- Wireless communication: ESP-NOW for low-latency messaging and UDP for position updates.
- Web-based control: An HTML-based manual interface for debugging and direct control.

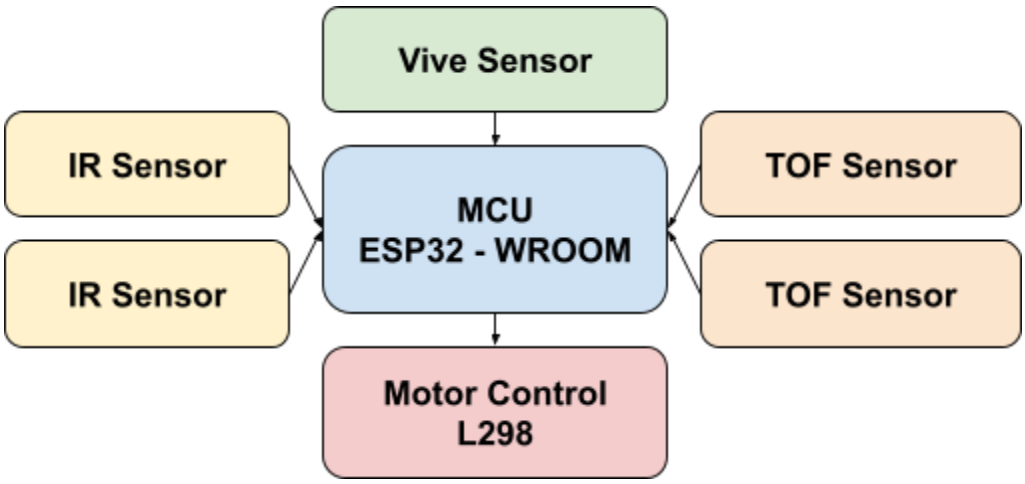


Figure 7: MCU Logic Block Diagram

4.2 Software Structure and Execution Flow

The software followed a hierarchical control structure, consisting of:

1. Initialization Phase
    - Set up WiFi, ESP-NOW, and UDP communication.
    - Configure PWM channels, motor driver pins, and sensor input modes.
    - Start the web server for manual control access.
2. Main Execution Loop
    - Web server polling, checking for manual control requests.
    - Reset sensor buffers to clear previous readings.
    - Read IR sensors for trophy detection.
    - Execute state-based control, selecting the appropriate mode:
        - Mode 0: Default behaviour, executing PID speed control.
        - Mode 1: Wall-following navigation using.
        - Mode 2: IR tracking mode for object interaction.
3. Interrupt Service Routines (ISRs)
    - Encoder feedback ISR for motor speed calculations.
    - ESP-NOW message ISR for receiving wireless commands.

The state machine structure ensured task prioritization, allowing the robot to respond dynamically to environmental conditions.

4.3 Motor Control and PID Regulation

The motor control system used PWM signals to regulate the speed of 12V DC motors, driven by an L298N H-bridge motor driver. The encoder feedback loop provided real-time velocity updates, allowing the PID controller to:

- Maintain a straight-line trajectory by adjusting motor speeds.
- Compensate for surface friction differences affecting movement.
- Prevent overshooting when stopping near objects.

The motor control was executed within `PIDControl()`, which computed the control output (`u[]`), applying speed corrections through `ledcAnalogWrite()`. The PID gains were manually tuned but exhibited minor drift over long distances, suggesting that adaptive gain tuning could improve performance.

```
608  void PIDControl() {
609    // Iterate over each motor in the system
610    for (int i = 0; i < numMotors; i++) {
611      // Calculate the velocity, which is the change in speed
612      int velocity = speed[i] - oldSpeed[i];
613      // Update the cumulative error by adding the current error (difference between desired speed and current speed)
614      summederror[i] += (desired - speed[i]);
615      // PID Control Calculation:
616      u[i] = KP * (desired - speed[i]) + KD * velocity + KI * summederror[i];
617      // Adjust summed error if the motor speed is more than desired speed
618      if (desired - speed[i] < 0) summederror[i] *= 0.5;
619      // Limit the maximum value of summed error to prevent integral wind-up
620      if (summederror[i] > 3000) summederror[i] = 3000;
621      // Limit the control signal to a maximum value
622      if (u[i] > 1500) u[i] = 1500;
623      // Ensure the control signal is not negative
624      if (u[i] < 0) u[i] = 0;
625      // Update oldSpeed for the next iteration
626      oldSpeed[i] = speed[i];
627    }
628  }
```

Figure 8: Code segment for PID controller.

4.4 Sensor Processing and Object Detection

The robot used a combination of infrared, time-of-flight, and Vive tracking sensors for autonomous operation.

- Infrared Detection (`IRDetect()`)
    - Detected 550 Hz and 23 Hz signals for trophy identification.
    - Used a low-pass filter to remove noise, improving accuracy.

```
695  void IRDetect() {
696    // Iterate over each sensor. Assuming there are 2 sensors connected.
697    for (int i = 0; i < 2; i++) {
698      // Read the current value from the IR sensor
699      unsigned long currentReading = analogRead(IRpins[i]);
700
701      // Check if the current sensor reading is above the threshold and was previously below it
702      // This indicates a rising edge, which is the start of a high pulse
703      if (currentReading >= highThreshold && lastAnalogValue[i] < highThreshold) {
704        // Calculate the duration of the previous pulse
705        pulseDuration[i] = micros() - lastRisingEdge[i];
706        // Record the time of this rising edge
707        lastRisingEdge[i] = micros();
708        // Mark that we're currently in a high pulse
709        isHighPulse[i] = true;
710
711        // Calculate the frequency based on the pulse duration
712        frequency[i] = 1000000.0 / pulseDuration[i]; // Convert pulse duration to frequency
713        // Print the frequency to the Serial Monitor for debugging
714        Serial.print("Frequency of sensor ");
715        Serial.print(i + 1);
716        Serial.print(" is: ");
717        Serial.print(frequency[i]);
718        Serial.println(" Hz");
719      }
720      // Check if the current sensor reading is below the threshold and we were previously in a high pulse
721      // This indicates a falling edge, which is the end of a high pulse
722      else if (currentReading < highThreshold && isHighPulse[i]) {
723        // Mark that the high pulse has ended
724        isHighPulse[i] = false;
725      }
726
727      // Store the current reading to compare in the next iteration
728      lastAnalogValue[i] = currentReading;
729    }
730  }
```

Figure 9: Code segment for IR detection.

- Wall Following (`wall_following()`)
    - Read distance data from VL53L0X TOF sensors using `read_dual_sensors()`.
    - Adjusted heading using proportional correction based on sensor input.

```
794  void wall_following() {
795    // Forward Logic
796    if (10 < TOF_Front && TOF_Front < 90000 && 10 < TOF_Side && TOF_Side < 90000) {
797      // Working Range for TOF sensor
798      if (TOF_Side > 200) {
799        // Too far from wall
800        // Turn slightly right
801        Serial.println("Turn Left A Bit");
802        mRight();
803        ledcAnalogWrite(LEDC_CHANNEL1, 1200); // motor1
804        ledcAnalogWrite(LEDC_CHANNEL2, 1200); // motor2
805        delay(8);
806        mForward();
807        delay(10);
808        mStop();
809      }
810      else if (TOF_Side < 100) {
811        Serial.println("Too Close to wall");
812        mLeftR();
813        ledcAnalogWrite(LEDC_CHANNEL1, 1200); // motor1
814        ledcAnalogWrite(LEDC_CHANNEL2, 1200); // motor2
815        delay(10);
816        mForward();
817        delay(10);
818        mStop();
819      }
```

Figure 10: Partial code segment for wall following logic.

- Vive Position Tracking
    - Updated XY coordinates via UDP broadcasting at 1 Hz frequency.
    - Occasionally lost precision due to jitter, suggesting a need for smoother filtering.

```
855  void viveSend() {
856    if (millis()-vive_ms>1000/FREQ) {
857      vive_ms = millis();
858      if (WiFi.status()==WL_CONNECTED) {
859        neopixelWrite(RGBLED,255,255,255);  // full white
860        UdpSend(vive_x, vive_y);
861      }
862    }
863
864    if (vive1.status() == VIVE_RECEIVING) {
865      vive_x = vive1.xCoord();
866      vive_y = vive1.yCoord();
867      neopixelWrite(RGBLED, 0, vive_x/200, vive_y/200);  // blue to greenish
868    }
869    else {
870      vive_x=0;
871      vive_y=0;
872      switch (vive1.sync(5)) {
873        break;
874        case VIVE_SYNC_ONLY: // missing sweep pulses (signal weak)
875          neopixelWrite(RGBLED,64,32,0);  // yellowish
876        break;
877        default:
878        case VIVE_NO_SIGNAL: // nothing detected
879          neopixelWrite(RGBLED,128,0,0);  // red
880      }
881    }
882    delay(20);
883  }
```

Figure 11: Code segment for Vive receiver.

4.5 Wireless Communication and Web Interface

The robot communicated using two networking protocols:

- UDP Broadcasting
    - Sent Vive position updates to comply with competition requirements.
    - Operated in a stateless mode, meaning lost packets were not recovered.

```
632  void UdpSend(int x, int y){
633    char udpBuffer[20];
634    sprintf(udpBuffer, "%02d:%4d,%4d",teamNumber,x,y);
635    UDPTestServer.beginPacket(ipTarget, UDPPORT);
636    UDPTestServer.println(udpBuffer);
637    UDPTestServer.endPacket();
638    Serial.println(udpBuffer);
639  }
640
641  void handleUDPServer() {
642     const int UDP_PACKET_SIZE = 14; // can be up to 65535
643     uint8_t packetBuffer[UDP_PACKET_SIZE];
644
645     int cb = UDPTestServer.parsePacket(); // if there is no message cb=0
646     if (cb) {
647        packetBuffer[13]=0; // null terminate string
648        UDPTestServer.read(packetBuffer, UDP_PACKET_SIZE);
649        if (atoi((char *)packetBuffer) == 0) {
650          police_x = atoi((char *)packetBuffer+3); // ##,####,#### 2nd indexed char
651          police_y = atoi((char *)packetBuffer+8); // ##,####,#### 7th indexed char
652          Serial.print("From Team ");
653          Serial.println((char *)packetBuffer);
654          Serial.println("Police Car data: ");
655          Serial.println(police_x);
656          Serial.println(police_y);
657        }
658     }
659  }
```

Figure 12: Code segment for UDP server.

- ESP-NOW
    - Used for low-latency sensor updates and real-time data exchange.
    - Occasionally experienced packet loss, requiring retries.

```
974  void espSend() {
975    uint8_t message[50]; // Max ESPnow packet is 250 byte data
976    message[0]=2;
977    message[1]=1;
978    message[1]=3;
979    // put some message together to send
980    sprintf((char *) message, "Team 24 ESPNow" );
981
982    if (esp_now_send(peer1.peer_addr, message, sizeof(message))==ESP_OK)
983      Serial.printf("Sent '%s' to %x:%x:%x:%x:%x:%x \n",message, peer1.peer_addr[0],peer1.peer_addr[1],peer1.peer_a
984    else Serial.println("Send failed");
985
986    // delay(500); // ESPNow max sending rate (with default speeds) is about 50Hz
987  }
```

Figure 13: Code segment for ESP-NOW receiver.

A web-based manual control interface was implemented using an HTML server hosted on the ESP32. The following handler functions processed user inputs:

- `handleForward()` – Move forward
- `handleBackward()` – Move backward
- `handleLeft()` – Turn left
- `handleRight()` – Turn right
- `handleStop()` – Stop all movement

The web control system provided debugging capabilities, but due to network latency, real-time responsiveness was limited. Future iterations could benefit from WebSocket-based messaging for reduced lag.

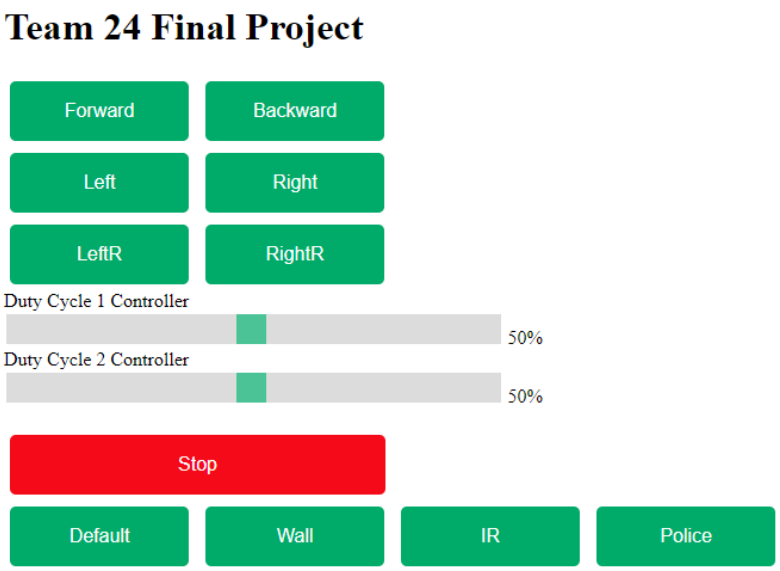


Figure 14: HTML page for the robot controller.

5 Retrospective

The development of the autonomous robot presented numerous engineering challenges, requiring iterative problem-solving across mechanical, electrical, and software subsystems. Throughout the project, the team gained experience in sensor integration, motor control, wireless communication, and autonomous decision-making. This section reflects on key technical and process-based lessons learned, identifies challenges encountered, and discusses potential future improvements.

5.1 Lessons Learned

5.1.1 Mechanical Design

- The transition from Mecanum wheels to a differential drive system significantly improved traction and movement stability.
- Weight distribution and center of gravity considerations were critical in ensuring consistent object manipulation.
- The police car pushing mechanism required additional reinforcement to ensure stable force application.

5.1.2 Electrical System

- Infrared signal processing was more sensitive to ambient light than initially expected, necessitating additional shielding and filtering.
- Power management failures, such as battery over-discharge, highlighted the need for an auto-shutdown system.
- ESP-NOW and UDP communication conflicts required a revised networking approach to avoid connectivity drops.

5.1.3 Software Development

- Implementing a modular code structure improved debugging efficiency and allowed for faster iteration on control algorithms.

- The state-machine approach effectively handled mode switching but could be further optimized for smoother transitions between tasks.
- PID tuning for motor control worked well at short distances but introduced minor drift over longer travel paths.

5.1.4 Project Workflow

- Early hardware prototyping helped refine mechanical components, but late-stage software integration caused delays in final testing.
- More structured testing procedures would have improved bug tracking and sensor calibration consistency.
- Dividing tasks into hardware, software, and testing subteams increased development efficiency but required better cross-team communication.

5.2 Challenges Encountered

Several key challenges emerged throughout the project, requiring iterative problem-solving and redesign efforts.

5.2.1 Sensor Performance Issues

- Infrared sensor interference from ambient light occasionally produced false readings, affecting object detection reliability.
- The Vive tracking system introduced jitter, impacting precision in autonomous movement.
- Time-of-flight sensors required calibration adjustments, as range accuracy degraded in certain orientations.

5.2.2 Communication and Control Conflicts

- ESP-NOW messages occasionally dropped, requiring a packet confirmation system.
- PID motor control introduced minor overshoot, leading to slight misalignment when stopping near objects.

- The robot occasionally struggled with initial police car alignment, requiring manual intervention in some trials.

5.2.3 Competition Constraints

- Testing time was limited due to lab availability, impacting fine-tuning opportunities.
- Field surface variations introduced friction inconsistencies, affecting movement stability.
- Manual control debugging was hindered by web interface latency, making real-time adjustments slower.

5.3 Performance in Competition

The final robot was successfully able to complete all major game objectives, including trophy retrieval, police car pushing, and wall-following navigation.

- Strengths
    - Trophy detection and handling were executed with high reliability.
    - The wall-following system worked effectively, preventing unnecessary collisions.
    - The ESP-NOW wireless system maintained communication, enabling rule compliance.
- Limitations
    - Minor misalignment during police car pushing required reattempts in some cases.
    - PID drift over long distances caused small trajectory deviations.
    - Vive tracking jitter introduced slight inconsistencies in object approach angles.

Despite these challenges, the robot achieved successful object interactions and navigation, demonstrating robust autonomous behaviour.

5.4 Future Improvements

To further enhance system performance, several design modifications and refinements are recommended:

5.4.1 Mechanical Enhancements

- Reinforce the police car pushing mechanism to ensure more stable force application.
- Use an adjustable weight distribution strategy to fine-tune stability.
- Optimize chassis rigidity to minimize structural vibrations affecting sensor readings.

5.4.2 Electrical and Sensor Upgrades

- Implement an automatic power shutoff circuit to prevent over-discharge.
- Redesign the IR sensor shielding case to further reduce ambient light interference.
- Incorporate an IMU for additional trajectory correction during long-distance movement.

5.4.3 Software and Control Refinements

- Develop an adaptive PID controller that adjusts gains dynamically based on travel distance.
- Introduce a Kalman filter for Vive tracking to smooth jitter in position data.
- Optimize ESP-NOW message handling to reduce dropped packets in high-interference environments.

5.4.4 Testing and Workflow Improvements

- Conduct earlier system integration testing to identify major issues sooner.
- Develop structured debugging checklists to improve issue resolution speed.
- Increase documentation of software modules to facilitate future modifications.

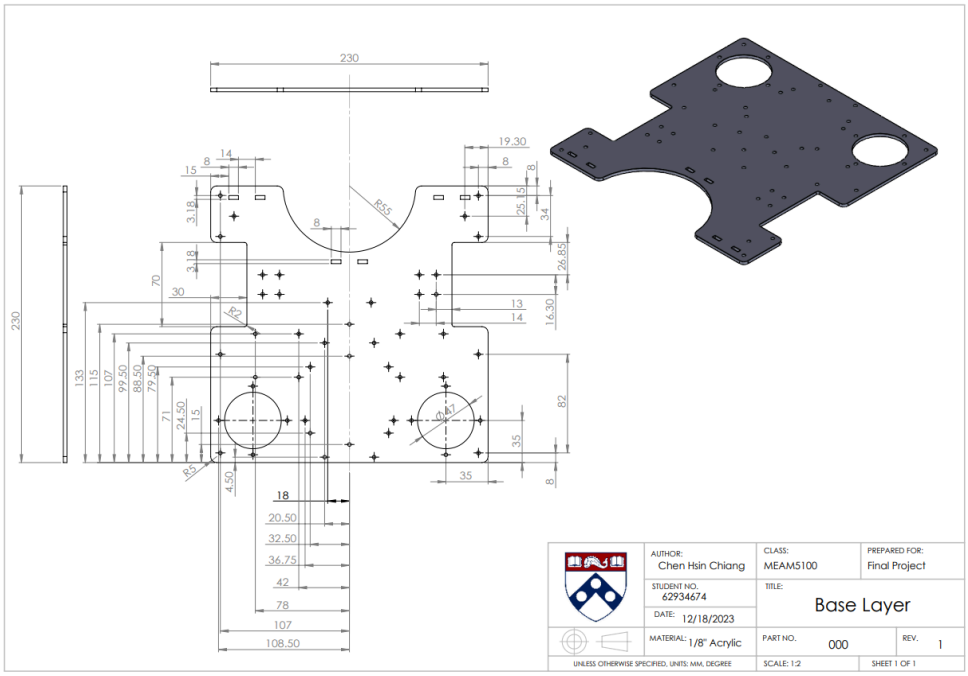
6 Appendix

A1: BOM for the Project

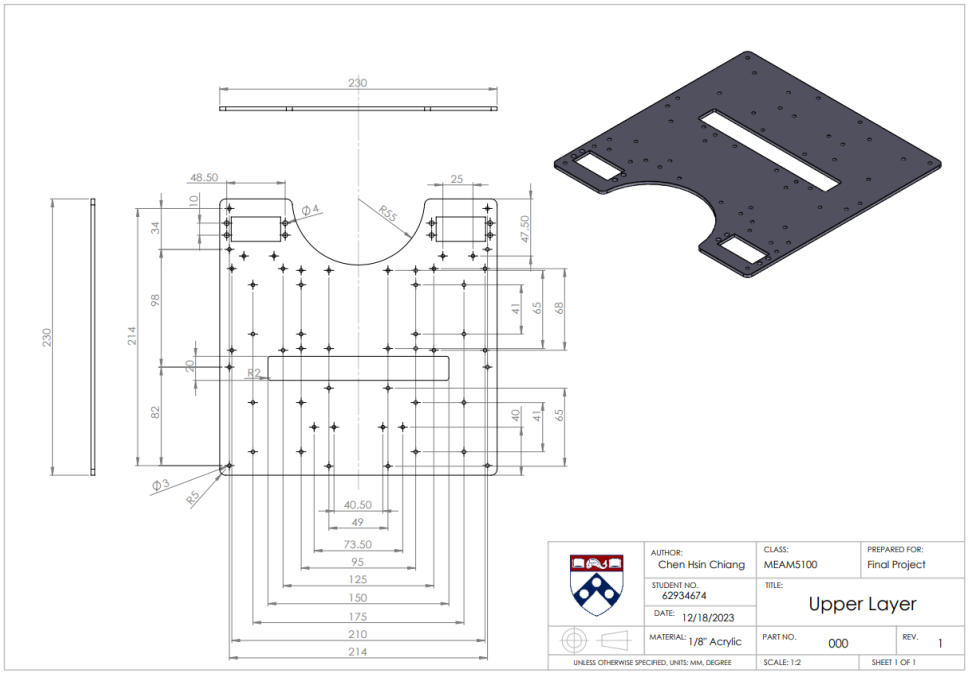
Table 1: BOM for the final project

Category	Part Name	Unit Cost ($)	Quantity	Total Cost ($)	Notes
Microcontroller	ESP32-WROOM	6.00	1	6.00	Main processing unit
Power System	4S LiPo Battery	28.00	1	28.00	14.8V rechargeable battery
Drive System	Motor-Wheel Set	34.00	1	34.00	Includes 2 motors and wheels
Motor Driver	L298N Motor Driver	3.00	1	3.00	Dual H-Bridge motor driver
Sensors & Detection	VL53L0X Time-of-Flight Sensor	8.00	2	16.00	Wall-following navigation
	TLV272 Op-Amp IC	0.00	3	0.00	IR signal amplification
	Vive Sensor Diode	0.00	1	0.00	Used for Vive tracking
	PD70-01C IR Sensor	0.00	1	0.00	Trophy detection
	LTR4206 IR Sensor	0.00	2	0.00	Additional IR detection
Total Cost				**87.00**	

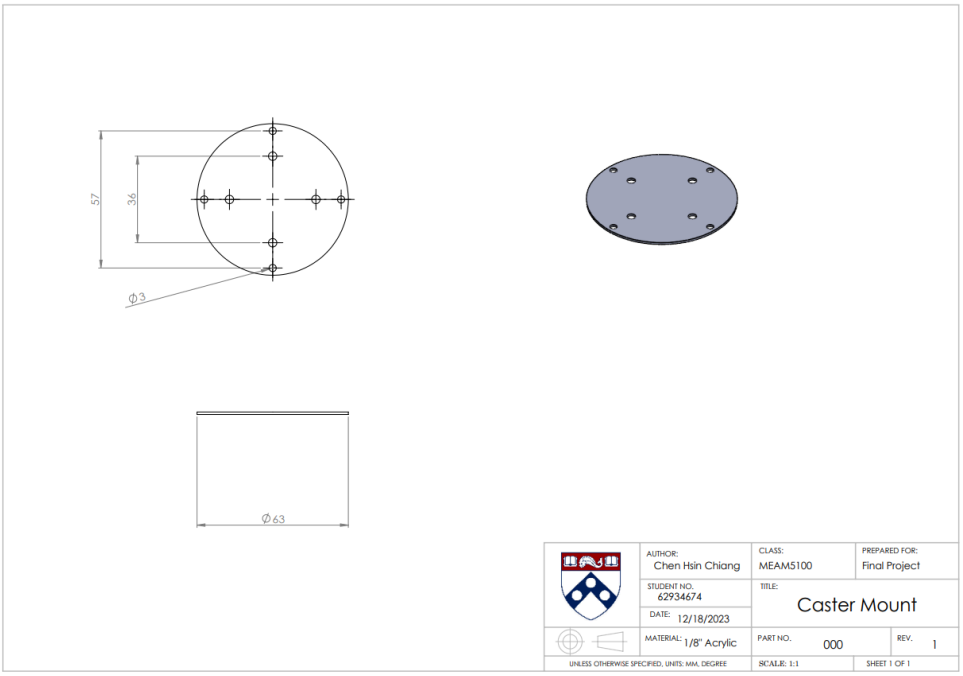
A2: 2D Dimensional Drawing for the Base Layer



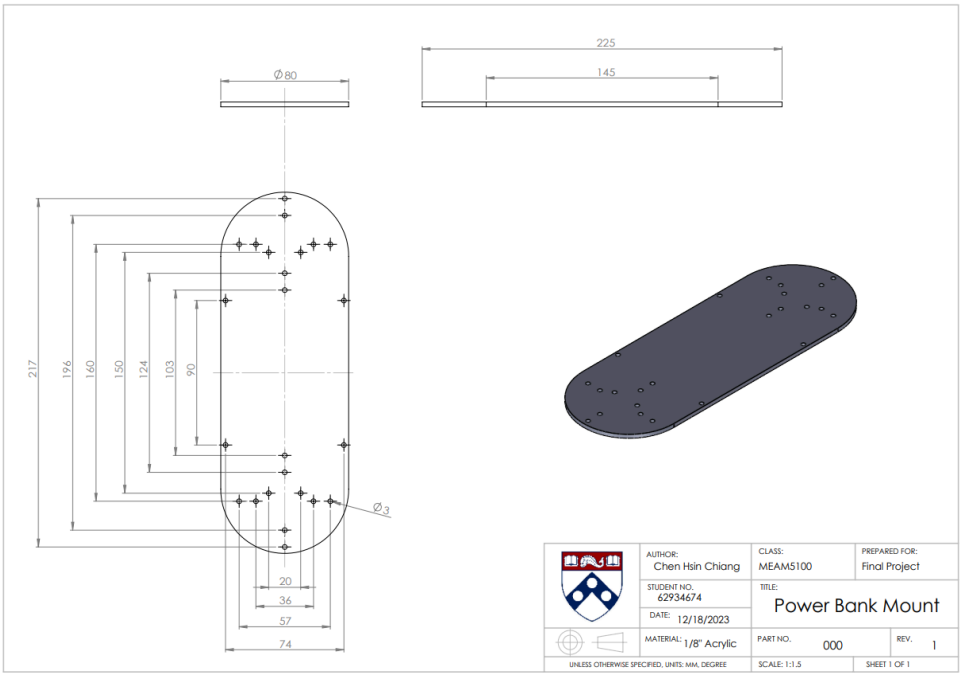
A4: 2D Dimensional Drawing for the Upper Layer



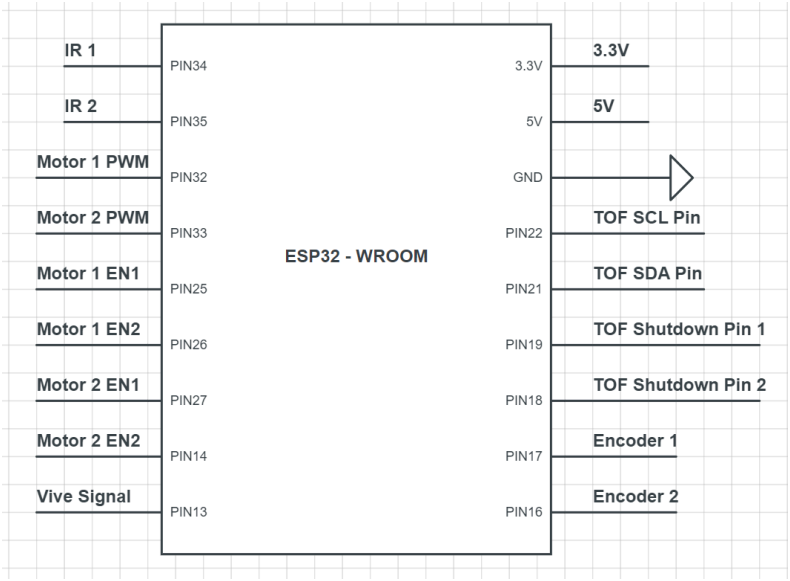
A5: 2D Dimensional Drawing for the Caster Mount



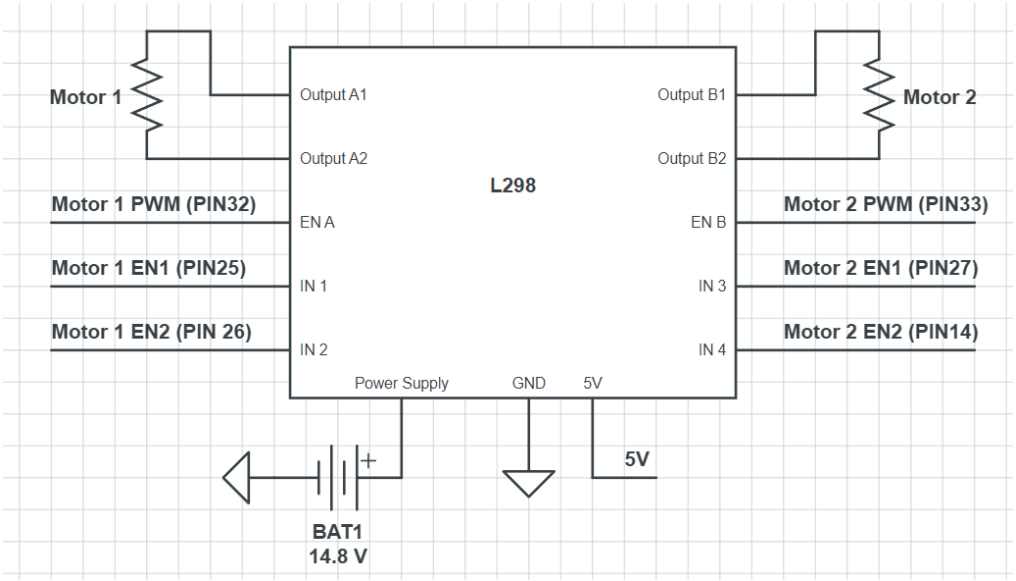
A6: 2D Dimensional Drawing for the Power Bank Holder



A7: Schematics for the ESP32 WROOM

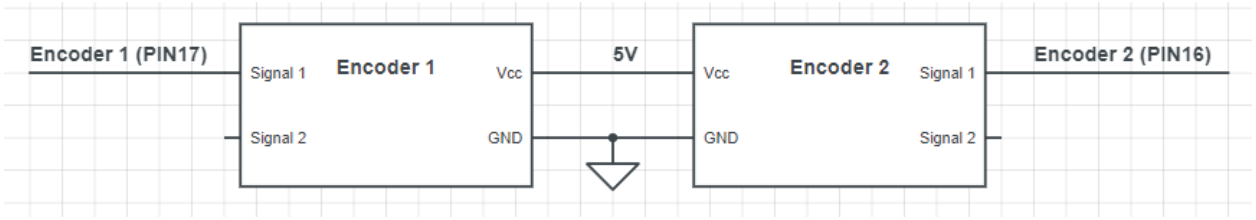


A8: Schematics for the Motor Controller

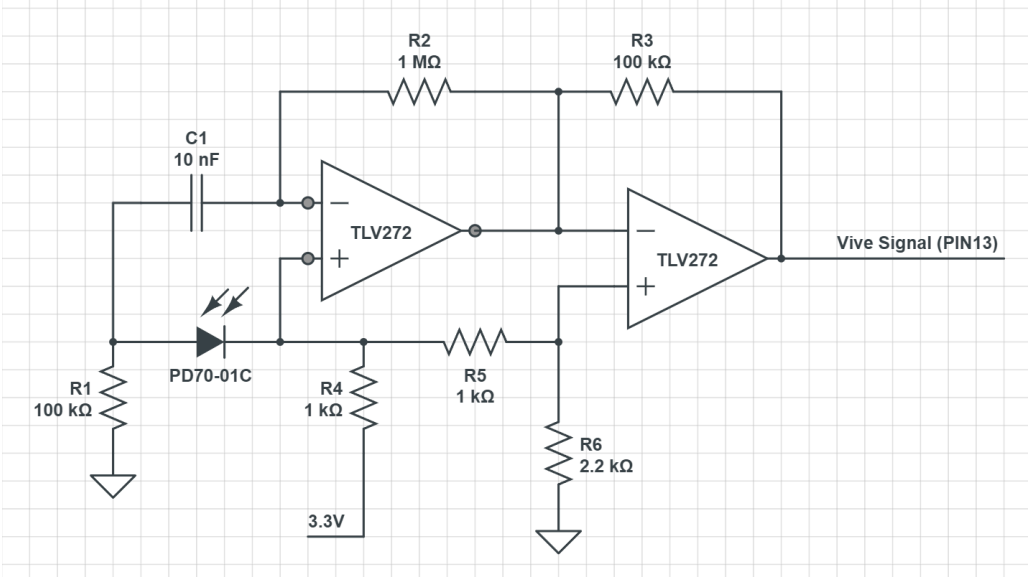


*Note: Modeling Motors as Resistors.

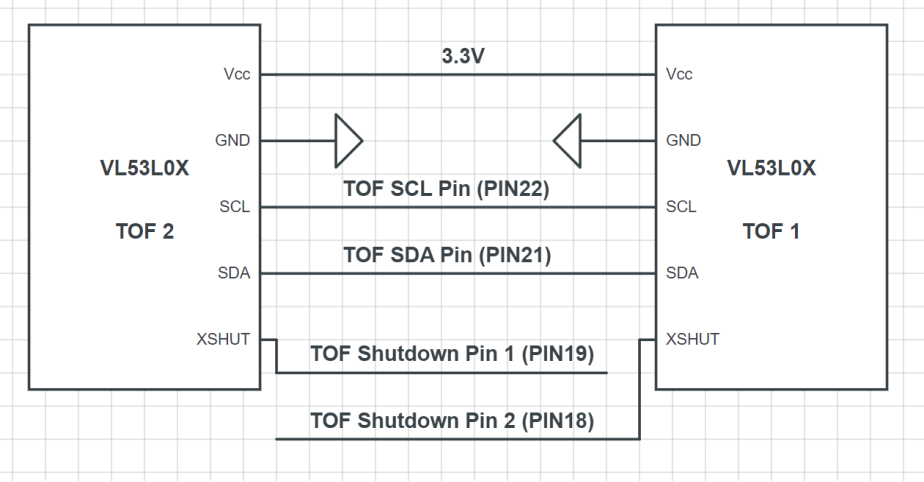
A9: Schematics for the Encoder



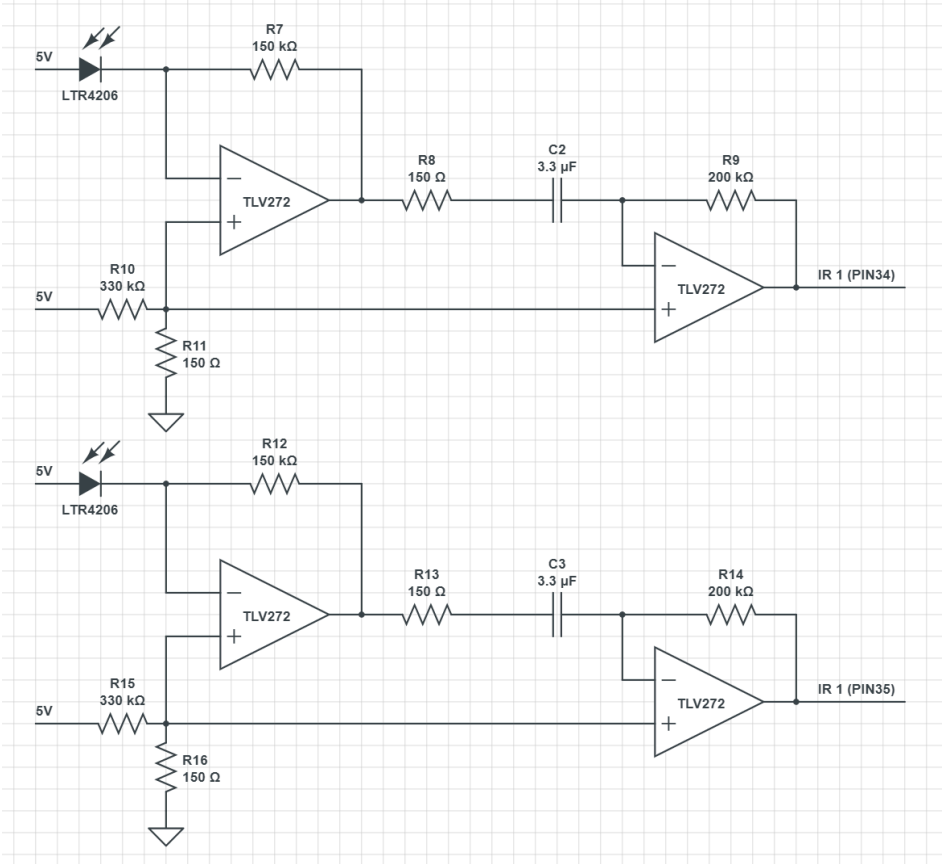
A10: Schematics for the Vive Sensor



A11: Schematics for the TOF Sensors



A12: Schematics for the IR Sensors



A13: Data Sheet for Electronic Components

- ESP32 WROOM
- L298
- Motor with Encoder
- TLV272
- TOF VL53L0X

A14: Video Demonstration for Robot Functionalities

- Wall Following
- Beacon tracking
- Police Car Tracking