MEAM 5100 Group 1 Final Report

Author: Hanyu Gong, Yining Guo, Yuetong Hao

Functionality

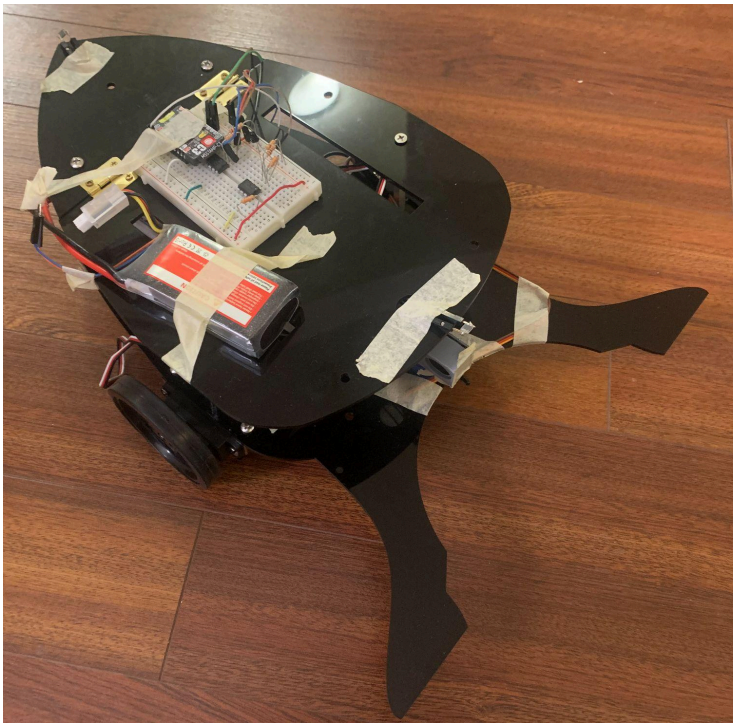


Figure 1: An overlook of our final design

High-level description of approach to winning the game and explanation of the design choices

Our team's final design draws inspiration from the streamlined shape of a beetle, incorporating tentacles with a curvature that complements the spherical nature of the beacons. The tentacles' front end features a broad, flat surface to enhance engagement with the police car during the pushing maneuver. Structurally, our beetle-like vehicle is composed of a dual-layered assembly. The foundational layer houses a C3 board, which is intricately programmed to manage actuator control, UDP broadcasting and reception, I2C communication, a web interface, Vive sensing, wall-following capabilities, and beacon tracking functionalities. Ascending to the top layer, it accommodates an additional C3 board, linked to the primary board via I2C, with its core responsibility being the transmission of ESP-now messages. The mobility of our beetle car is ensured by three wheels: two continuous rotation servo wheels at the forefront and a single caster wheel at the rear.

Our tactical blueprint for clinching victory in the competition primarily focuses on the rapid localization and subsequent displacement of the police car, followed by the pursuit and strategic repositioning of the fake trophy, contingent upon its discovered location. These tactical decisions are deeply rooted in the unique geometric configuration of our beetle car and are further augmented by a specialized algorithm dedicated to the efficient pushing of the police car. The intricacies of this algorithm and its operational dynamics will be elucidated in the subsequent section dedicated to detailed functionality.

Control the Robot:

The function of controlling the robot is achieved by steering two continuous servo motors. We extended the web control interface where there are two slider bars for manual control – one governs “forward, stop, and backward”, and the other governs the direction as “turning right, turning left, and moving forward”. The steering method is differential steering. We adopted continuous servo steering to achieve finer control of the turning angles since tasks such as wall following and navigation to objects require delicate control of movement.

Wall Following:

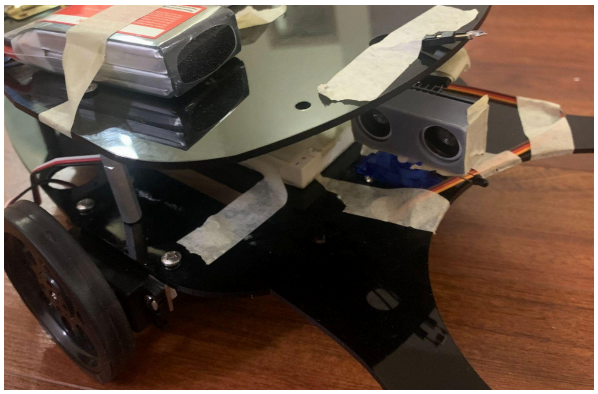


Figure 2: A close-up shot of the ultrasonic unit

The wall following function is accomplished by combining the robot control with an ultrasonic sensing unit as shown in Figure 2. An SG90 servo motor is mounted on the base layer of the beetle, and the ultrasonic sensor is connected to the servo motor with a 3D-printed case. The servo motor enables the ultrasonic sensor to rotate 90 degrees between the front and the right constantly to scan and store its frontal and right distances to its nearest obstacles. If it sees an obstacle within 30cm to its front, the wall following function will steer it to turn 90 degrees to the left. The wall function also enables it to maintain a 20cm to 30cm distance to its right side by steering it 10 degrees to the right or left when the right wall distance is too large or too small.

Police car navigation and pushing:

Two vive-sensing photodiodes are mounted on the top layer of the beetle(refer to Figure 1) to sense xy-coordinates of the robot's head and tail in the field. The police car navigation function is a two-step process including 1) alignment of the direction and 2) pilot to the police car. The direction alignment is achieved by turning the beetle to the left 30 degrees at a time until its alignment with the police car is less than 20 degrees. Then it will move forward to reach the police car. Once its Euclidean distance to the police car is less than 500, the beetle will steer back and forth to push the police. By doing so it will gain extra momentum and exert a stronger push.

Robot communication via ESP-NOW:

Since an MCU board cannot broadcast/receive UDP messages and send ESP-NOW messages at the same time, we adopted a second MCU on the top layer to integrate the ESP-NOW message sending with I2C master mode. Every time a command is made by the main MCU(slave), the master sends an ESP-NOW message to the TA's MAC address.

Autonomous navigation to trophy:
To achieve the function of trophy/fake seeking, we replicated the circuit from Lab 2:Beacon and mounted a photoresistor on the baselayer pointing forward with a comparable height to the trophy/fake. Our design strategy is similar to the police car tacking that the beetle would steer left constantly until it achieves an alignment with the trophy/fake and then move forward.

Overall performance, what worked/what didn't and why
The overall performance of the system was satisfactory, with most functionalities working robustly. However, one notable exception was the trophy/fake navigation feature. While the detection circuit itself operated effectively during individual testing, it encountered difficulties when integrated into the overall navigation and steering system. The detection circuit demonstrated its functionality by successfully detecting signals up to a 2-meter distance when tested independently, and we were able to amplify the detected signal with a larger gain. However, when attempting to integrate this functionality into the navigation and steering components, we encountered challenges. Specifically, there was inconsistency in reconciling the continuous analog readings from the IR sensor with the steering of the wheels.

The issue likely stemmed from the complex interaction between the IR sensor readings and the steering mechanism. It's possible that the continuous analog readings were not effectively translated into appropriate steering commands or that there were inconsistencies in the integration logic. Further investigation and refinement of the integration process may be needed to ensure a seamless interaction between the trophy/fake navigation feature and the overall system.

Mechanical Design

Intended Performance
The intended performance of the car was to successfully push the police car and hold trophies. The design of our car allows it to both push and hold trophies. The wheels were intended to fit tightly together with the motors, avoiding any possibilities of motor idling. The steering function was achieved by implementing differential drive, which is different from our approach in lab 4 where we used the Ackerman mechanism. We chose to use differential drive because we believed it would have more precise rotation, allowing accurate tracking of the objects. Two servo motors were used to power our robot; a servo motor was chosen because the power level is adjustable, and we believed it allows for more maneuverability.

Actual Performance
The actual performance was not satisfactory at first, as there were multiple issues. The first issue was the noise from the servo motor. Without implementing an encoder, the wheels were rotating at different speeds. However, the fix was simple—by tuning the input power value of the motor. We had to often adjust the input value since the ideal value changes constantly as we change our power source and add more electronics. The steering function performed as intended; the differential drive allows for accurate angles of rotation. Since we had to calibrate

the servo motors at the same speed, the motors do not run at full power. Due to this, the robot did not have enough power to push the police car at first. We analyzed the issue and realized the main issue was not the motor but the weight of our robot. The robot was too light compared to the police car, so by adding more weight to it, we were able to push the police car eventually.

Things that we tried but failed (and thus learned from)

Throughout our time working on the robot, many ideas in our initial design did not end up being implemented due to complexity or insufficient time. Initially, we designed the robot to be assembled through pressfit (see Appendix). As we worked on the electronic parts and functions, it turns out that press fitting simply does not offer enough space and openings for all the sensors we need. The need to constantly change the pressfit design was also very time-consuming. We ended up using a simplified design by attaching two pieces of laser-cut acrylic together, using spacers and screws. This design was utilized by many other groups as well. It allows easy attachment of a breadboard and various sensors, and easy wire connections since it is an open space. Most importantly, the two-layer design gives us a much easier time to adjust/fix the circuit. The CAD design for the two-layer design was simply a 2D drawing and was not saved. A main concept we had in our initial design was to introduce a claw mechanism to our robot so it's able to grab and secure the trophies (see Appendix). The claw was designed to be powered by another servo motor. However, during the process, we soon realized that there is simply no room for such a claw design, on top of all the sensors and circuits. We fell back to the two-layer design and added an extra claw shape in the front.

Electrical Design

- Intended and actual performance

Actuators (Motors)

Since we have previously employed both continuous servo motors and non-continuous servo motors, we possess a clear understanding of their mechanisms, enabling us to manage them effortlessly. By transmitting the PWM signal, we can precisely control the position of the standard SG90 servo motor within a 180-degree angle range. Mounted with an ultrasonic sensor, the standard servo serves as the robot's "head," turning in different directions to enable distance scanning functionality. Unlike the standard servo motor, the use of the PWM signal with our continuous servo FS5103R allows us to regulate the speed and direction of the continuous servo motors. We opted for continuous motors to serve as the mobility module of the robot instead of DC motors, as used in Lab 4, because we believed they could provide more precise control over the wheels during autonomous navigation. Both standard servos and continuous servo motors performed as expected in terms of functionality. However, it is noteworthy that our two continuous servos exhibit slight variations in their reaction to the same PWM signal, necessitating some fine-tuning.

Sensors

Ultrasonic sensors and IR sensors are used in the system. The ultrasonic sensor is intended to measure distances by emitting ultrasonic waves and then listening for an echo. When the emitted sound wave hits an object, it bounces back to the sensor, which calculates the distance

based on the time delay of the echo. On the other hand, the IR receiver is intended to detect IR LED emitted light. Both sensors work as we expected, the ultrasonic sensor played a major role in our wall following function after calibration. Noise from the ultrasonic sensor can be generated when the servo motor turns fast, thus we programmed the servo to turn between the front and right every two seconds allowing enough time to capture accurate frontal and right distances. We have also noticed some inconsistencies and noise issues with the IR sensor which is discussed in more detail in the noise section.

Noise

Our team encountered a couple of noise issues when using the IR sensor. We suspect that the major source of noise for IR sensors comes from ambient light, especially sunlight, which contains IR radiation. Indoor lighting, such as incandescent and fluorescent lights, can also emit IR radiation. These noise signals can be problematic when a large sensing distance is required, and we go through the amplification process. We did not notice much motor noise in standard servo motors. However, when we initially connected our continuous servos directly to the MCU, we found that they generated too much stall current every time we changed the motor direction (e.g. changing from moving forward to moving backward). We addressed this issue by adopting Professor Yim's suggestion, using a separate voltage source (7.4 Lipo battery) to power those continuous servo motors.

Things that we tried but failed (and thus learned from)

During the construction of our circuit, our team also tried to implement the example circuit introduced in Lecture 21 for detecting the trophy/fake. However, we still experienced high-frequency noises, which resulted in an inaccuracy detection of 23Hz and 550Hz. Thus, we suspect the unreliable performance of the system is coming from the incorrect combination of the resistor choices. An attempt was also made to employ a set of IR remote sensors for beacon tracking. However, we faced limitations as these sensors could only sense a specific frequency at 37MHz, thus the IR remote sensors proved unsuitable for our tracking needs. Subsequently, we explored phototransistors to achieve accurate beacon tracking.

Processor architecture and code architecture

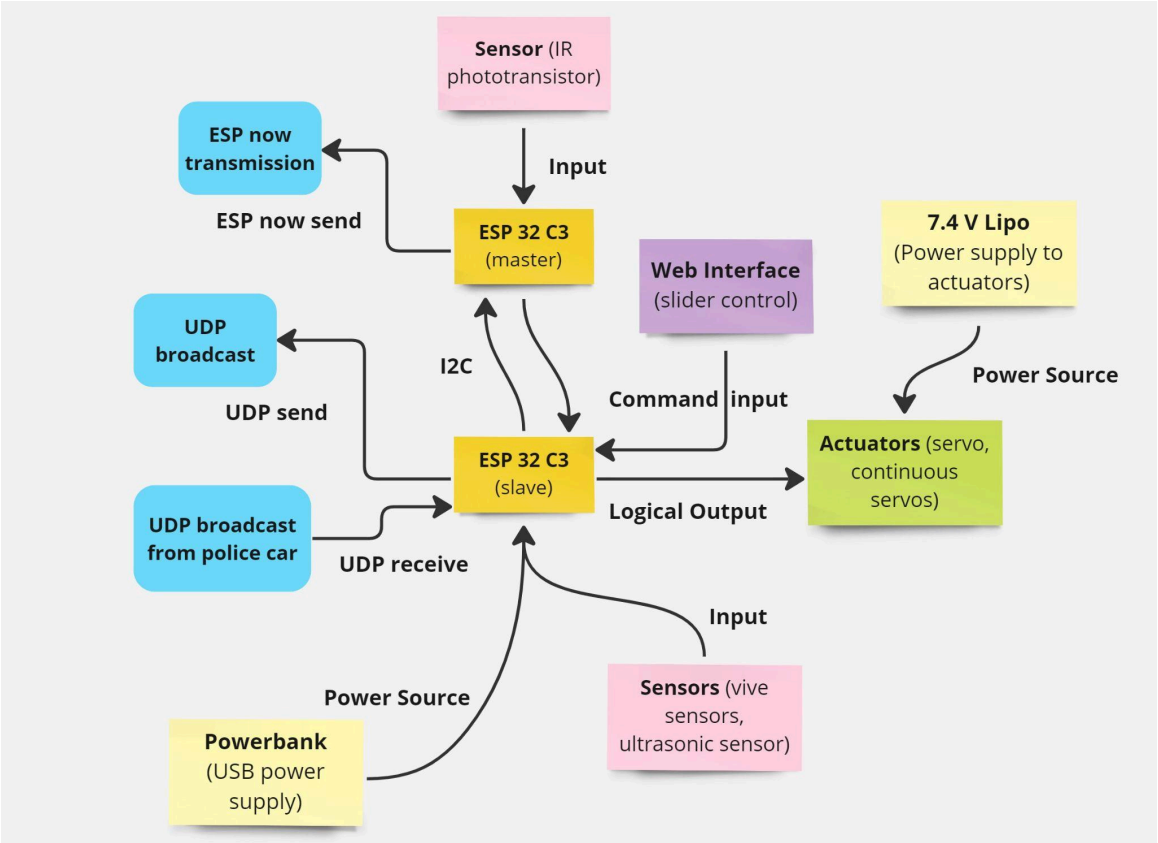


Figure 3. Block diagram showing MCU's and other components logically connected

Software approach:
The software architecture employed in this project incorporates a diverse range of components, including UDP broadcast and receiving, ESPnow transmission, I2C, web interface, Vive sensing input, ultrasonic sensor digital input, IR phototransistor input, and actuator control specifications. Our programming foundation is rooted in the lab 4 implementation for the web-controlled car, featuring a sophisticated web-based slider user interface that caters to various functionalities.

Our journey started with a modification of the template code, drawing inspiration from lecture sample programs like vive2test.ino for Vive sensing. Despite our limited understanding of the underlying mechanics of Vive sensing, the provided code guided us successfully through this implementation. Subsequently, we delved into motor actuations, aiming to utilize three ledc channels corresponding to robot head, left, and right motor controls. Through testing and calibration, we developed functions such as moveLeft(), moveRight(), moveForward(), and moveBackward(), establishing the foundation for our robot's autonomous navigation.

Consistent with our approach in lab 4, we fine-tuned the web interface JavaScript code, linking each function using handleslider() in the setup() code, and aligning them with specific control sliders for enhanced manipulation of the robot during the competition. The web slider bar

includes robot mode switching (for autonomous motion) and other sliders for manual control. Moving forward, we implemented the executive code, meeting the primary functionality requirements, namely, wall following, police car tracking, and beacon tracking.

We also implemented functionality for scanning distances in the surroundings. This implementation is crucial for wall-following behavior, enabling the robot to adjust its movement based on detected distances from obstacles. The wall following functionality consists of three parts: scan(), watchsurrounding(), and go(), where watchsurrounding() enables the scanning of front and right distance by turning the servo motor on the robot's head to the right orientation and calling the scan() function that initializes the ultrasonic sensor and calculates the distance sensed by it. The go() function then determines which way to go based on the scanned distance results, where the code is implemented such that it follows and follows back to the wall.

Transitioning to police car and beacon tracking, our code integrates Vive sensing input to track the positions of Vive controllers. The trackPolice() function is specifically designed to detect patterns or movements indicative of a police car and respond accordingly, including making turns, stopping, and adjusting the robot's direction based on calculated angles and distances from Vive trackers. A unique approach is also implemented in the trackPolice() function to make the robot push back and forth when it is close enough to the police car. The implementation also includes the UdpSend() function, responsible for sending UDP packets with Vive tracker data to the broadcast channel IP address. This feature plays a crucial role in communication with the police car and systems in the game competition, facilitating collaborative or competitive interactions.

The integration of I2C communication adds another layer of versatility to the project. This enables the ESP32 C3 to communicate with another C3 MCU in the system (master-slave), facilitating message transmission. By utilizing this, we can send our command message from the main MCU (with the web interface) to the sub-MCU that sends ESP-now message packets to the police car. Specifically for the sub-MCU, we implemented the provided lecture sample code of ESPnow sending along with the I2C master. Once a message (with team number and message string length) has been received from the main MCU, the sub MCU will send it to our targeted receiver (police car) with a specified MAC address.

Things that we tried but failed (and thus learned from):
In our pursuit of refining the system, we encountered challenges and explored various approaches, some of which proved unsuccessful. Here are notable instances where our attempts did not yield the desired outcomes:

We initially experimented with utilizing ESPnow and UDP sending concurrently, aiming to eliminate the necessity for I2C communication. However, this approach did not yield the desired results since the UDP protocol somehow blocked ESPnow, leading us to reconsider and eventually incorporate I2C into the communication framework.

In the process of tracking Vive sensors and police cars, we experimented with different methods for computing angles. Initially, we explored slope computation and arccos for determining the angle between the vector of two Vive sensors and the vector of the police car to Vive sensor. Unfortunately, these methods proved to be inaccurate. Eventually, we found that arctan provided the accurate angle computations needed for our purposes.

Our attempt at implementing a straightforward wall-following method encountered challenges. The initial strategy involved the robot turning 90 degrees upon detecting an obstacle. However, this simplistic approach led to issues where the robot consistently moved away from the wall without effectively following it. To address this, we introduced the follow-back function, allowing the robot to trace back and properly follow the wall.

In the final part of the project, we tried our best to build the beacon tracking functionality. However, despite being able to analogRead the beacon IR frequency with correct values, we could not figure out a way to incorporate the data reading in controlling the robot. A certain disturbance was found that when motors turn, the frequency measurement is disabled due to time delay.

Retrospective

In our final project for the class, we encountered various challenges and learning opportunities that contributed to our overall understanding of sensor integration and signal processing. One of the most crucial lessons we gleaned was the significance of comprehending the intricacies involved in these processes. Our problem-solving skills were honed through troubleshooting issues related to circuit integration and steering mechanisms. Moreover, we underscored the importance of thorough testing and validation before the full system integration to ensure its seamless functionality.

The hands-on experience with real-world applications and practical problem-solving proved to be among the best aspects of the class for us as a team. We appreciated the exposure to amplification techniques for sensors and the valuable insights gained from addressing challenges in integrating various components into a cohesive system. Professor Yim's guidance played a pivotal role, and we found applying his suggestions particularly beneficial in overcoming project-related difficulties.

Reflecting on our experiences, we express a wish for improvement of the course in certain areas: More detailed guidance or additional resources on integrating complex sensor functionalities with steering mechanisms would have been beneficial. Collaborative problem-solving sessions among classmates could provide opportunities to share insights and strategies, fostering a more collaborative learning environment.

In our view, the class placed a commendable emphasis on the practical application of theoretical knowledge in real-world scenarios. The opportunity to explore alternative solutions and refine approaches through iterative testing and feedback was invaluable. The iterative nature of engineering projects was acknowledged, encouraging continuous improvement and

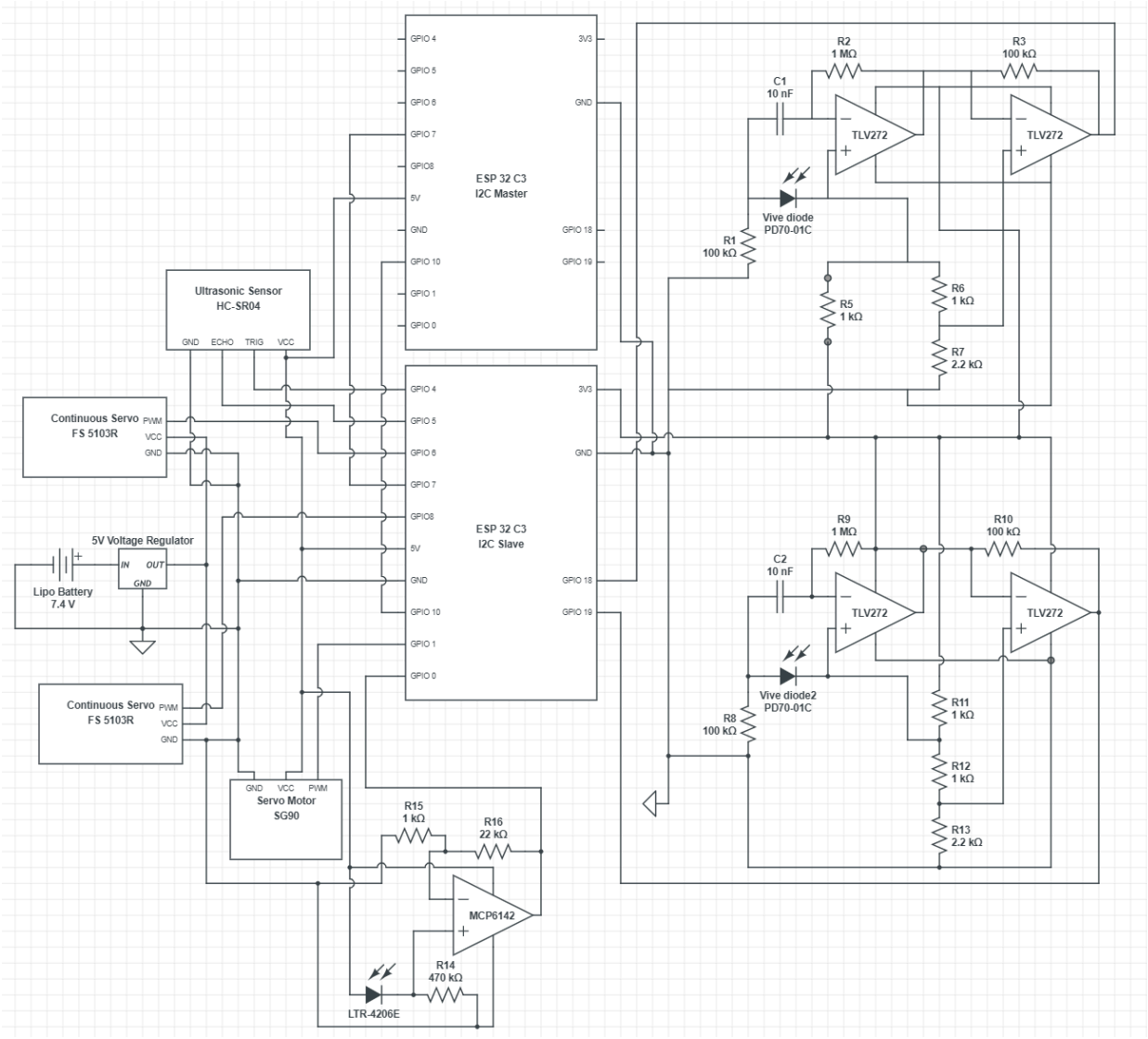
the ability to learn from failures. As a team, we appreciate the holistic learning experience provided by the class, contributing to our growth as aspiring engineers.

Appendix

BOM (Bill of Materials)

Part:	Amount:	Cost:	Website:
Wheels	1	$5	GM lab
Continuous Servo Motors	2	$36.75	Adafruit
Lipo Batteries	Pack of 2	$21.59	Link
IR Infrared Obstacle Avoidance Sensor	Pack of 10	$8.79	Link
Ultrasonic Sensor	Pack of 2	$0	BE lab
ESP32 S2	2	$22.95	Digikey
ESP32 C3	2	$18	Digikey
Castors with wheels	2	$4	GM lab
Total	$123.07		

Schematic of electronic circuit

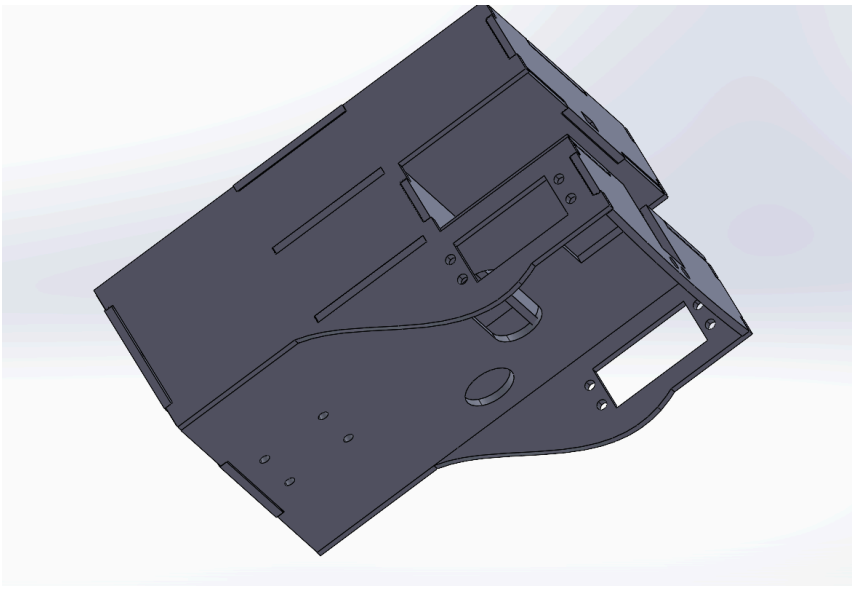


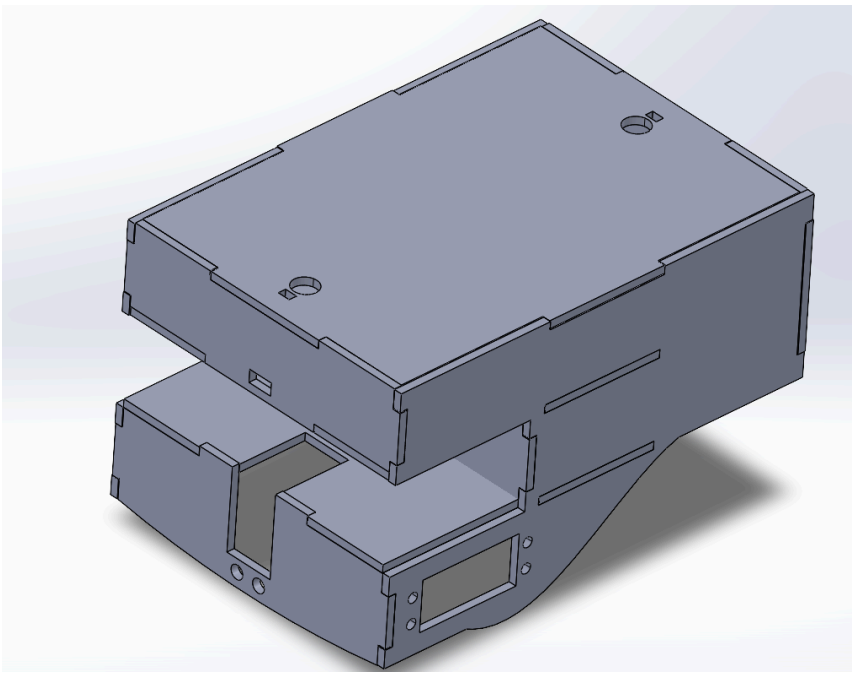
Nice photo or rendering of full robot

See above in “Functionality” section

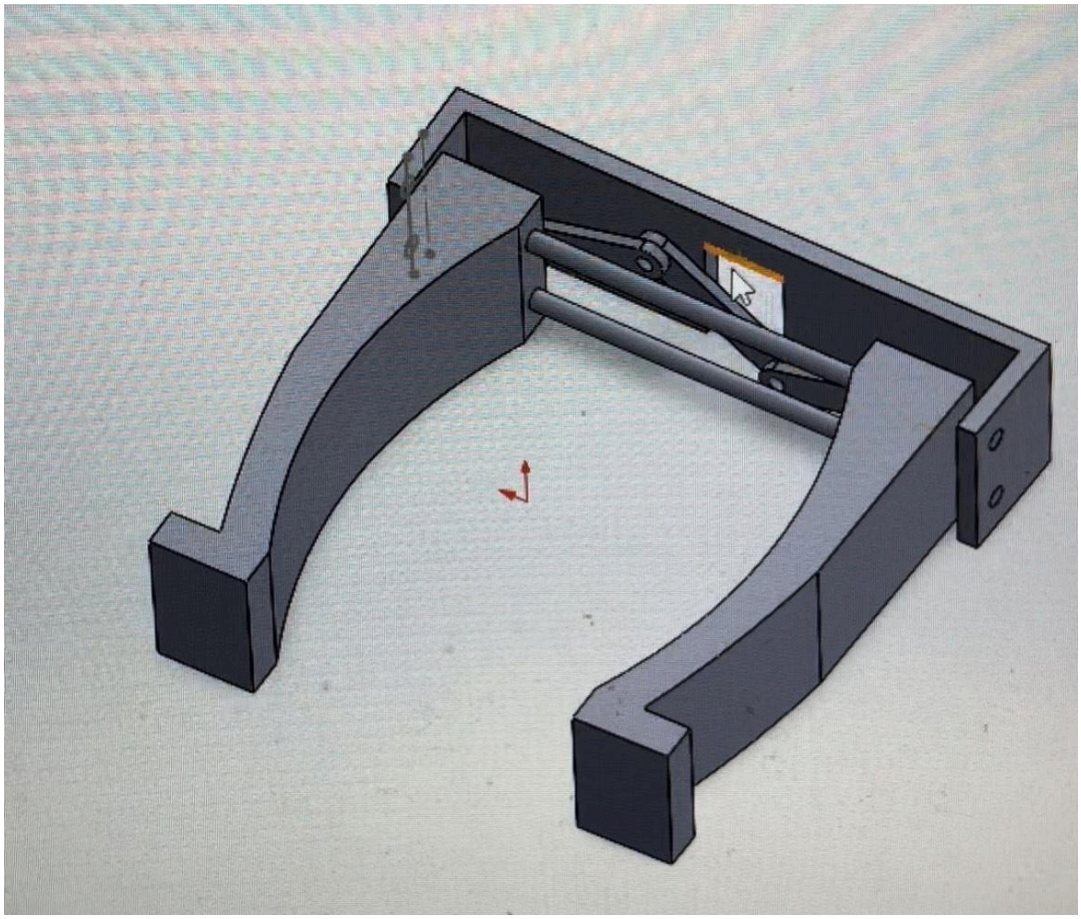
CAD drawings (or mechanical drawings) highlighting anything special

CAD file of initial pressfit design





CAD for claw



All data sheets for all components that are not parts supplied by the ministore

1. FS5103R Continuous servo
2. HC-SR04 Ultrasonic sensor

Links to videos of functionality

1. Wall following
2. Track and push police car

Fun competition video link

Link to the Final competition round 2 Group 1 pushing the police car:
https://youtu.be/7DT5nEVvP1I?si=7WFDIJYPNye2yf1E