MEAM 5100

Design of Mechatronics

--Fall 2023--

Final Project:

Grand Theft Autonomous

Rahul Birewar (rbirewar@seas.upenn.edu)

Sai Chand Gubbala (saichan8@seas.upenn.edu)

Aditya Rangamani (adivsr@seas.upenn.edu)

Submission Date: December 21, 2023

Functionality

Initially, the priority was to develop a functioning bot that could successfully achieve all the tasks required for the maximum graded evaluation score, with a goal of perfecting each task with hopes of bringing a versatile robot to the competition. This would have allowed us to team-up with a robot that possesses any combination of abilities to improve our chances of scoring. Unfortunately, many issues and other complications posed a challenge, resulting in schedule delays and leaving no time for competition preparation.

The drive system of our bot is a modified version of our Lab 4.2 bot. Now with two castor wheels, one on either side of the motorized wheels, the axis of rotation was centered about the bot, allowing it to perfectly turn in place. This was useful with the trophy tracking task. Instead of using two IR LEDs and a servo like Professor Yim recommended, we were able to see massive success with only a single IR LED rigidly mounted to the robot. As the robot rotated in place,, the bot would stop once the IR LED detected a specific frequency, and start moving forwards towards the emitter. This was only possible because of the shroud we designed and implemented, obstructing the IR LED's field of view to narrow the detection zone, resulting in quite a precise trophy detection system. (*See video in Appendix)*

For wall following, we used ultrasonic sensors on the front and sides of our bot. Initially, the side sensors were mounted at the bot's perfect 90°, directly above the wheels. This proved to be a major issue, especially when turning corners. As others suggested, we relocated our side sensor to the 45°, proving to be successful as shown in the video in the *Appendix*.

Finally, using the VIVE sensors with the base circuit and code provided, we were able to detect the location of our robot and move towards the police car. Using a standard reference as the shorter side/wall of the circuit , we first align the bot parallel to this side. Then we traverse in the y direction to align the robot along the police car. Finally after making a 90 deg rotation, we align the robot facing the police car. Then we proceed to move towards the police car and eventually push it.

Overall, our bot did not perform as expected at the competition, given we set quite high expectations for ourselves. We tried to perfect everything to the point where nothing was perfect due to time limitations and other challenges along the way. If we simplified our approach, we could've had more time to improve the robustness of our bot and in turn seen more success. We also

didn't have time to fix our UDP transmissions for VIVE readings and establish our ESPnow communications. Therefore, we decided to switch to fully autonomous during the competition for maximum communication factor.

After passing graded evaluation, we faced issues where our ultrasonic sensors would not detect anything after using the VIVE to push the police car. We concluded that updating our VIVE readings took a very long time and drastically slowed the code, thus the functionality of other sensors as well. Additionally, we tried implementing our gripper servos to find that they don't work. We are unsure whether this was a code issue, if we damaged the servos in handling, or if they were defective from the start.

Mechanical Design

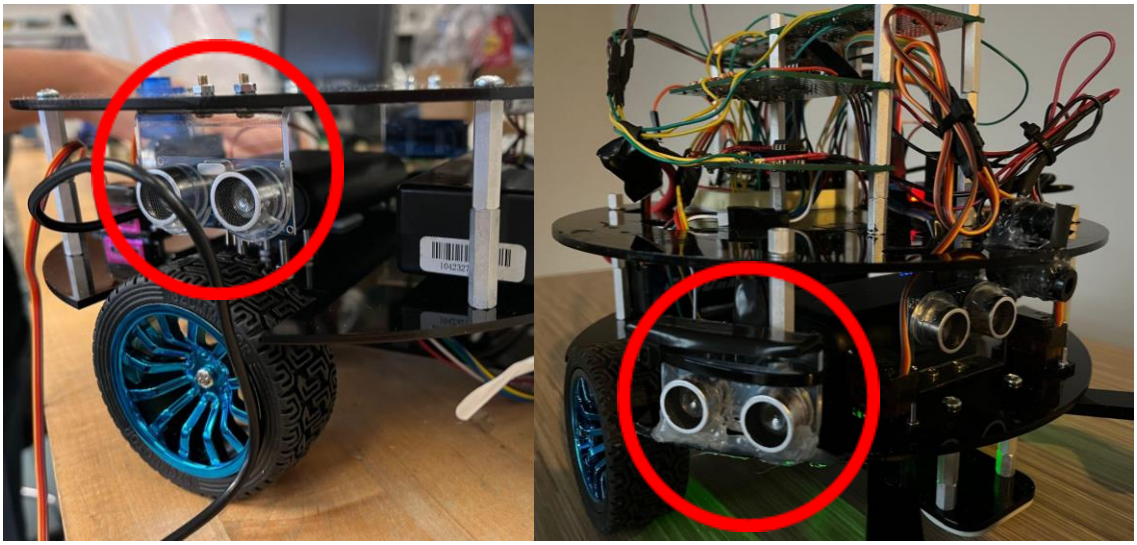
As mentioned, the drive system is a two wheel-two castor design, modified from our Lab 4.2 mobile base. The wheels are positioned along the center of the bot so that the axis of rotation is centered, allowing it to rotate in place. This worked exactly as intended. Since the motors were mounted on the underside of the bot (Layer 1), the height of the castor wheels had to be adjusted using standoffs to balance the bot. An additional acrylic piece (Castor Sheet) was made and incorporated for castor wheel rigidity. In order to ensure we can easily push the police car (at the time not knowing the actual weight of the police car), we decided to pick a motor with a lower speed in return for higher torque. Moreover, we learned from Lab 4.2 that we can achieve a greater power output with higher current in addition to voltage. Therefore, we chose a 6V DC motor at 100rpm in combination with a 7.4V LiPo battery for our drive system. The motor also came with a built-in encoder to use for our PID control.

Multiple layers were used to separate our batteries from electronics from the batteries, also allowing for ease-of-access to the circuits for wiring and debugging.

Throughout the final week, some design adjustments were made on the spot to fix various issues. When testing our wall following, we discovered the bot struggled to traverse over the tape, meaning the wheels were losing contact with the ground. Raising the castor wheels resolved this issue (as seen in the "Wall Following" video in the *Appendix*). Another big concern with the design was that the ultrasonic sensors may be too high and the emitted waves would either miss the wall or rebound off the top corner. Standoff heights were adjusted to lower the sensors as much as possible, but proved to not be an

issue after all. Instead, the issue was with the position of the sensor with respect to the bot's center. Initially, as the bot turned towards the wall, the right ultrasonic sensor ws reading an increasing distance rather than decreasing as it should have. Thus, the bot continued to move towards the wall until collision, preventing it from continuing. The ultrasonic sensor on the right side was then repositioned closer to the front at a 45 degree angle. This resolved the issue.

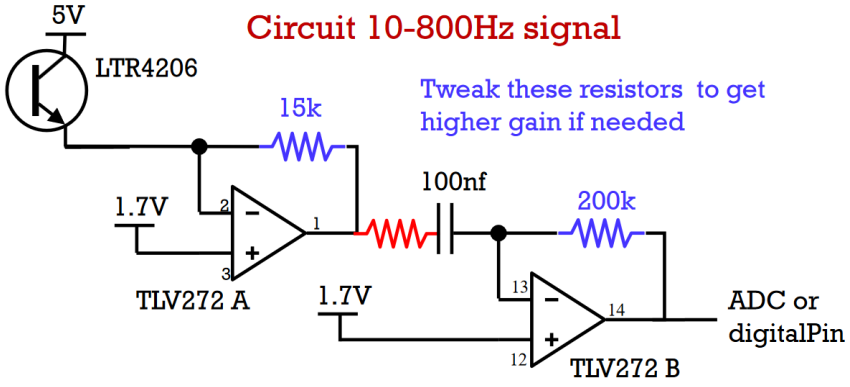
Before and After of Side Ultrasonic Sensor Placement:



Electrical Design

Beacon Tracking:

Initially following the circuit provided by Prof. Mark led to a lot of noise and inaccurate frequency readings at longer distances. With the circuit given by Prof. Yim attached below:



This circuit was later modified to remove the capacitor which removes the offset and the modified circuit has been attached in the *Appendix*.

The updated circuit contains 2 TLV-272 op-amps. One of these op-amps serves the initial purpose similar to the circuit given by Prof.Yim, however, the output of the TLV-272 is passed through another TLV-272 op-amp which acts as a comparator with no feedback. The 1.7V input voltage reference is given by a voltage divider set by the resistances 620 and 330Ω. The feedback resistances were left as is from the earlier circuit. There is a 100kΩ resistor going from V1out to V2in- which is to be tuned in the earlier circuit. The output of V2out was noted through an oscilloscope and noticed that with increasing distance the square wave received by the phototransistor has a minimum voltage of 1.7V and a maximum voltage of 2.4V. Knowing this we decided to create a comparator circuit to create a digital low for voltage under 2V and digital high for voltages over 2V. Keeping 2V as reference we created a voltage divider circuit with resistances of 1.5kΩ and 1kΩ. The earlier circuit would create negligible voltage difference between minimum and maximum voltages which was not enough for the microcontroller to detect logic-low and logic-high, hence the comparator approach was employed.

Vive Circuit:

We followed the vive circuit layout provided by Prof. Yim and the schematic for it is attached in the *Appendix*.

Using the provided schematic, we used two vive sensors. With 2 vives we were able to localize the bot by considering the average value of position returned by both vive circuits. The output of each vive circuit was connected to pins 4,5 respectively on the ESP32S2 microcontroller. The PD-70 photodiode was soldered onto a perfboard and placed on the top layer of the bot so that there is no obstruction in receiving signals from the mounted HTC vive. We were able to work with this circuit with little to no problem and were able to get the values of the coordinates of the bot with the code given to us. Using the positions we get and knowing the coordinates of the corners of the board. Using the horizontal edge of the board as one vector and the vector created by the line joining the two vives we can find the orientation of the bot using dot-product between the vectors, which is what we did for angular orientation.

Motor Circuit:

Given our task of pushing the police car, which requires a lot of torque, we chose a motor with low rpm but high torque. This results in out motor being 100rpm rate and 5.5kg-cm torque rated torque. With this we decided to employ an L-298N motor driver for its ability to drive multiple motors at once. In this motor driver we have 6 pins, 1 each for clockwise, counter-clockwise direction of each motor which totals 4, and the remaining 2 for PWM of each motor. The LiPo battery also is connected to the motor driver along with a common ground and a channel to 5V of the microcontroller. The circuit diagram for the motor circuit is attached in the *Appendix*.

Ultrasonic Circuit:

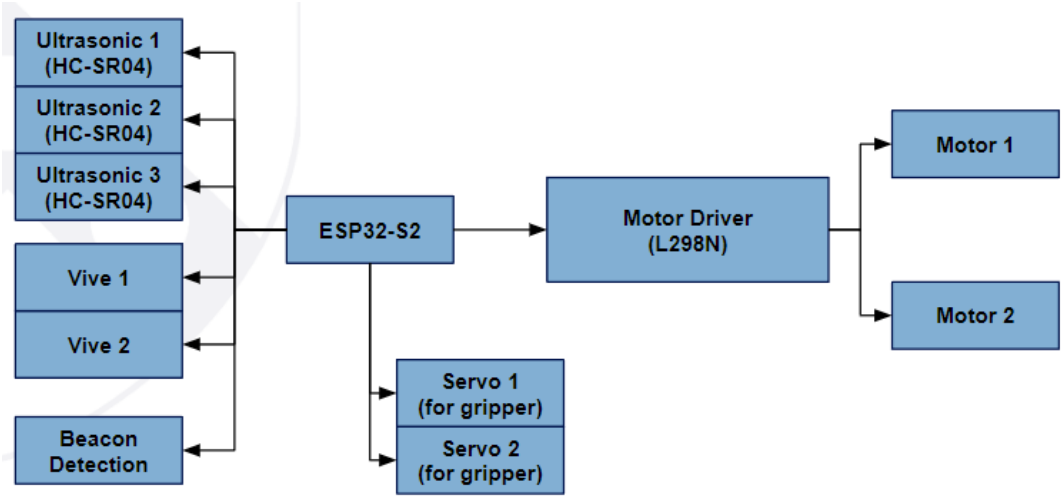
For the wall-follower circuit we used 3 ultrasonic sensors placed at the front, left, and right of the bot. With this we were able to define edge cases of corners where both the front and right sensor detected short distances to the wall. The ultrasonic sensor we used was HC-SR04 which has 4 pins, Vcc, trig, echo, GND. Vcc connects to 5V, GND connects to ground. The trig pin is the signal sent out by the transmitter supplied by the microcontroller and the echo pin reads the signal received from reflection. Upon calculating the time taken from the time signal was sent till received we can calculate the distance of the wall from the bot itself with this logic.

Servo Circuit:

Finally, for our grippers we used two 360° servos to independently control the movement of each gripper. The servos we used for this were MGS90 servos.

Processor Architecture and Code Architecture

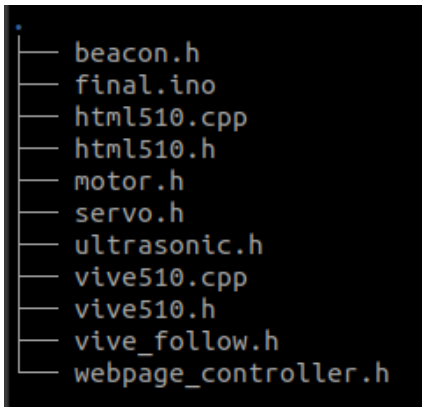
Block Diagram:



Software Approach:

We used one ESP32-S2 microcontroller. Initially we considered using ESP32-S2 for the majority of the bot operations along with ItsyBitsy to control the grippers. But with the large number of ports available on ESP32-S2, we decided to go with only one microcontroller.

We followed the approach of having an individual header file for each functionality and then calling each function in a final.ino file according to the mode requested from the webpage (from user).



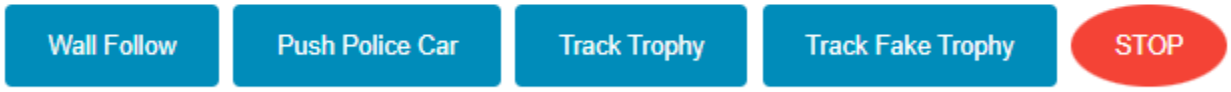
Bot Control:

We control the bot using a web page and use WiFi communication in access point mode to communicate the information from the website to ESP32-S2. The figure below shows the layout of the website with various buttons to select the mode of operation of the bot.

Grand Theft Autonomous

Team 09

Set Mode:



Current Mode: Following Wall

Motor Control:

Almost all functionalities of the robot use the motor control header file to move / rotate the bot as required. Hence we kept the architecture of the code as modular as possible to be able to use the motor functions across various operations like wall following, beacon tracking, police car pushing.

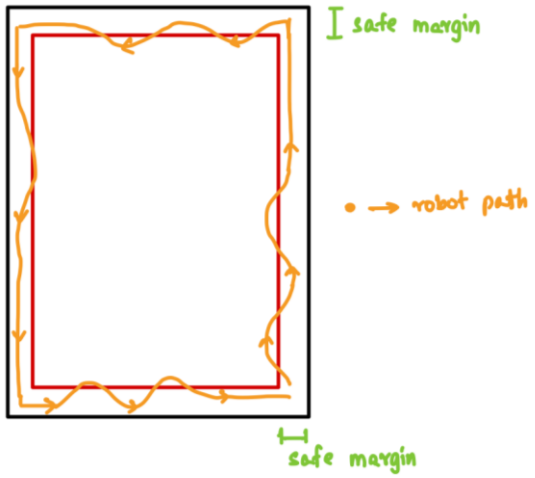
setupMotorPins() assigns pins on ESP32-S2. This function assigns pins to output mode for PWM, motor direction (CW and CCW). It also assigns another set of pins to input mode to read the encoder values from individual motors.

Further, we have directionForward(), directionBackward(), directionLeft(), directionRight() and directionNoMotion() to set the pins to either HIGH or LOW and accordingly control the direction of motion of the bot. The speed at which the motors rotate is controlled by driveMotors(), which takes in two values of duty cycle to control two motors independently. We send PWM signals to the motor through a motor driver using LEDC command. Additionally, we have positionControl(), which precisely rotates the bot by a desired angle. This function uses PD control to achieve the target rotation specified. We tuned the PD values for this operation using the Ziegler–Nichols method.

Wall Following:

For wall following, we initially considered using three ultrasonic sensors to perform the operation. But later, we decided to use only two ultrasonic sensors (front and right) for the operation with a compromise of being able to follow the circuit only in one direction. To eliminate noise from the readings, we implemented averageFilterFront() and averageFilterRight() to average out the readings on a sample size of 40 and the new values are updated as they come in.

wallFollow() function takes in the readings from the front and right sensors and gives out commands to the motors. We implemented a PD control which calculates the required duty cycles of PWM for left and right motors. We tuned the PD values for this operation using the Ziegler–Nichols method. We set the threshold safe distance to be 20cm at the front and a safe zone between 4 to 25 cm on the right. When the front sensor detects an object / wall within the threshold, it rotates the robot in the CCW direction. If all threshold margins are respected, the robot simply proceeds in the forward direction. If the robot is too far away / close to the wall, it corrects itself by moving towards / away from the wall until the safe threshold values are maintained. The figure below shows a rough idea of the wall following procedure.



Police Car pushing:

For pushing the police car, we used the information generated by the vive sensors. We used two onboard vive sensors to determine the orientation of the robot. For measuring the orientation, we used the shorter side of the circuit as our reference.

For the police car location, we use the UDPreceive() function in final.ino to get the x, y coordinates. For reaching the police car after getting its coordinates, we employed a strategy where we cover the distances in y direction and then in the x direction. Once the robot aligns itself in the y direction to the police car, the robot rotates by 90 deg CW or CCW (depending on which side of the police car we are on), then the robot covers the remaining distance in the x direction and pushes the police car.

Here as well, we use averageFilterX1(), averageFilterY1(), averageFilterX2(), averageFilterY2() to average out the readings on a sample size of 40. Here we initially considered implementing a feedback control but later decided to omit it in the final code as the results were quite satisfactory even without using a feedback control.

Beacon / Trophy Tracking:
We used a single IR phototransistor rigidly mounted at the front of the robot. We used rising edge interrupts to measure the frequencies emitted by the real trophy (550 Hz) and fake trophy (23 Hz).

We keep track of the current and previous time whenever the state changes and then take the time difference between the events to get the time period and frequency.

We initially considered using a servo mounted IR phototransistor scanning approach which sweeps a 180 deg to detect the frequencies. Later we decided to use the robot itself to do the sweeping where it rotates in the CCW direction until it locks onto a frequency. The function beacon_track() takes the input as the desired frequency to lock onto. Once the robot locks onto a 23 Hz / 550 Hz frequency, it proceeds towards it. We use directionForward(), directionLeft(), driveMotors() to control the robot during these steps.

Grippers:
We used two independent servos for each gripper hand. closeGripper() closes the gripper to grab the trophy (real / fake). openGripper() opens the grippers wide apart which are used when pushing the police car. To move the servo motor to a certain angle, we mapped the 0 to 180 deg to the PWM resolution (duty cycle)

Observation (Learnings):
Initially, we placed the lateral ultrasonic sensors on the left and right side of the robot. This method proved to be faulty as the robot was struggling to achieve the desired functionality and kept getting bumped into the wall due to noise in the measurements, particularly when the robot is rotating. Initially we checked if there are any issues with the algorithm. After multiple attempts, we decided to place the lateral ultrasonic sensor at an angle of 45

deg wrt the robot's forward / longitudinal direction. This solved the issues of noise and the robot was able to perform wall following as intended

Organizing the codes proved to be very helpful as it avoided the hassle of putting all codes in one file and ending up in a huge file that's difficult to debug. Further using a finite state machine approach, we restricted the robot to have only a certain functionality enabled, thereby avoiding unexpected interruptions by other codes for different functionalities. Also, when using a header file multiple times, it's important to use "pragma once" or "ifndef" to avoid re-declaring the variables. Also when trying to make a certain code modular (to be able to use across various other codes), it's important to make that function / code modular and as generic as possible, so it was important to make decisions on the architecture about what each function should compute and what arguments have to be passed

Retrospective

The biggest struggle with the class was having sufficient equipment and materials that were fully functional. Many times there were oscilloscopes, multimeters, and other equipment that were not working properly. With more than 100 students in the class, it was difficult to get access to equipment on time to be able to finish the labs. Furthermore, many times the GM ministore ran out of crucial components for the current labs.

Nevertheless, I think we can all agree we appreciate the hands-on experience we received. This topic is difficult to learn through lectures, so the practical practice we got from labs and projects were very beneficial to our understanding of this subject matter and ability to successfully build mechatronic systems.

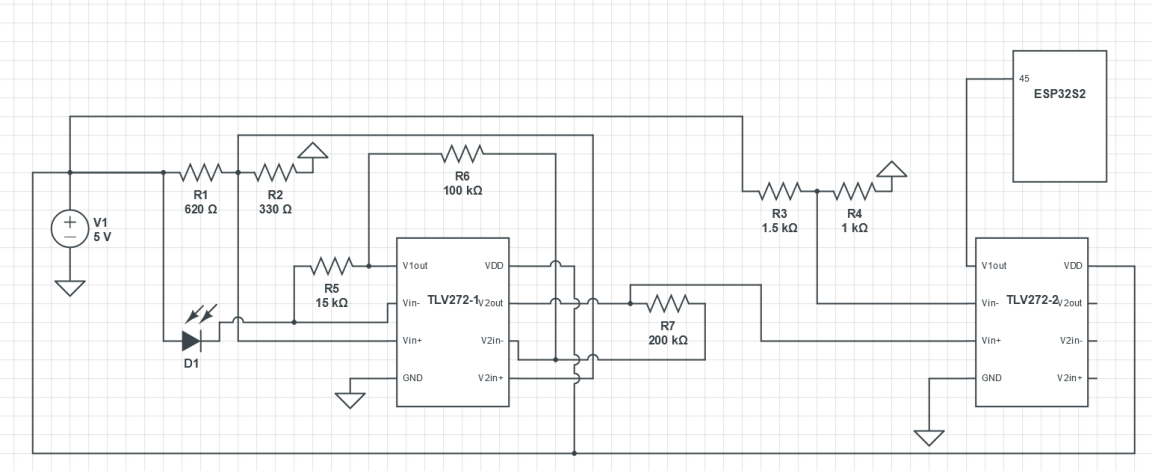
Appendix

Bill of Materials (BOM)

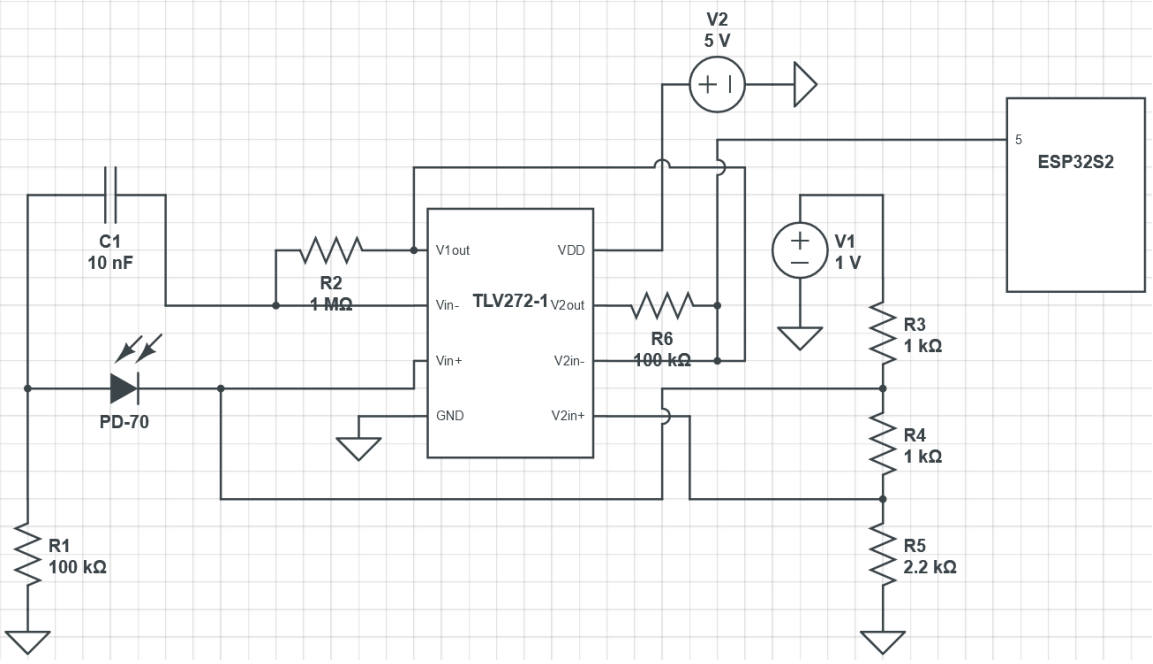
COMPONENT NAME	PRODUCT #	UNIT COST	QUANTITY	TOTAL COST	VENDOR
Mechanical Structure					
Acrylic (1/8")	-	-	*1*	-	-
Spherical Castor Wheel	-	$1.00	*2*	$2.00	Class Lottery
6V Motor, Encoder, Wheel Kit	*25SG-370CA-45-6*	$19.52	*2*	$39.04	Amazon (Garosa)
Servo (Pack of 2)	*MG90S*	$9.99	*1*	$9.99	Amazon (Maxmoral)
Spacers/Stand-offs	-	-	*16*	-	GM Ministore
Electrical Circuitry					
Perfboard	-	-	*4*	-	GM Ministore
Breadboard	-	-	*1*	-	GM Ministore
ESP-S2	-	$8.00	*2*	$16.00	GM Ministore
Motor Driver (Pack of 4)	L298N	$9.99	*1*	$9.99	Amazon (BOJACK)
Ultrasonic Sensors (Pack of 3)	HC-SR04	$7.99	*1*	$7.99	Amazon (EIKS)
LiPo Battery (Pack of 2)	-	$34.18	*1*	$34.18	Amazon (Zeee)
Power Bank	-	-	*1*	-	-
TOTAL				**$119.19**	

Circuit Schematics

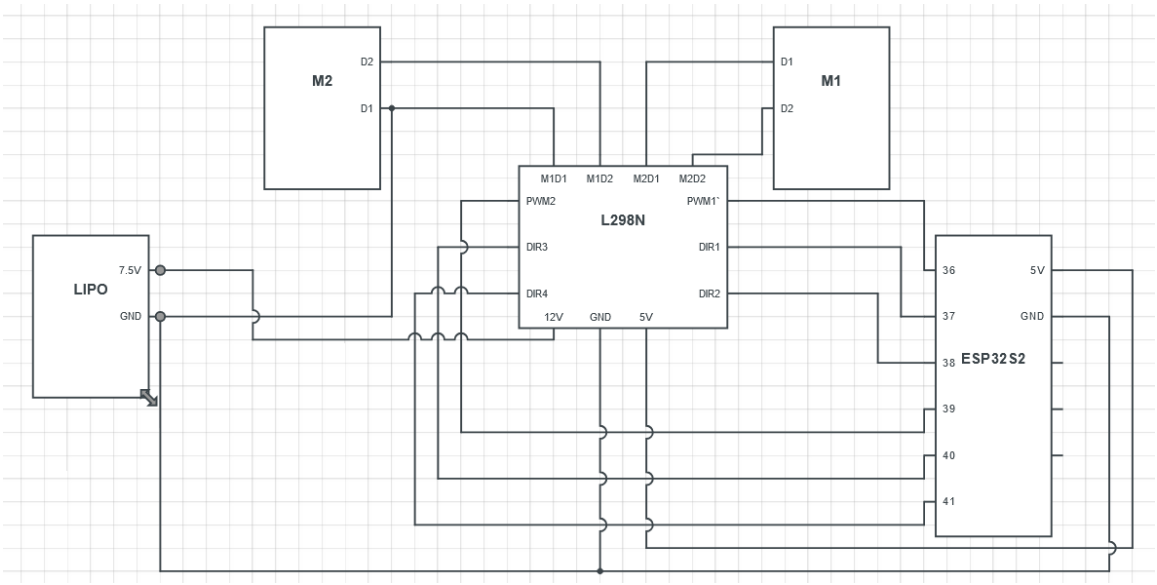
Beacon Tracking Circuit:



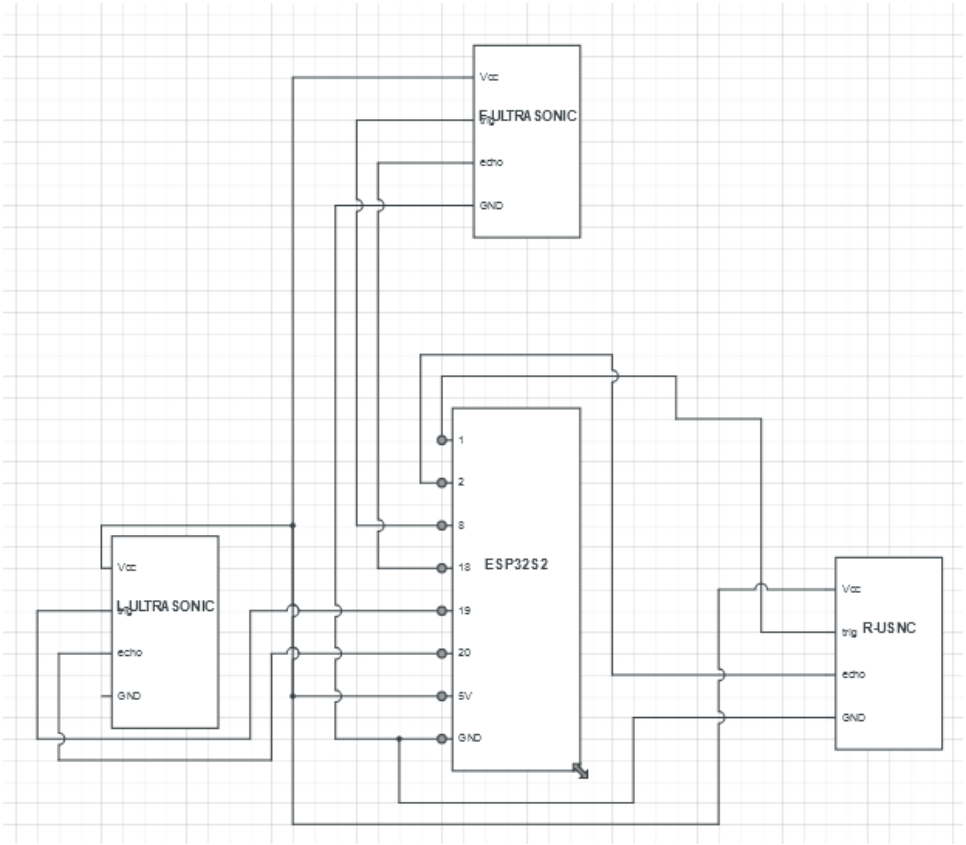
Vive Circuit:



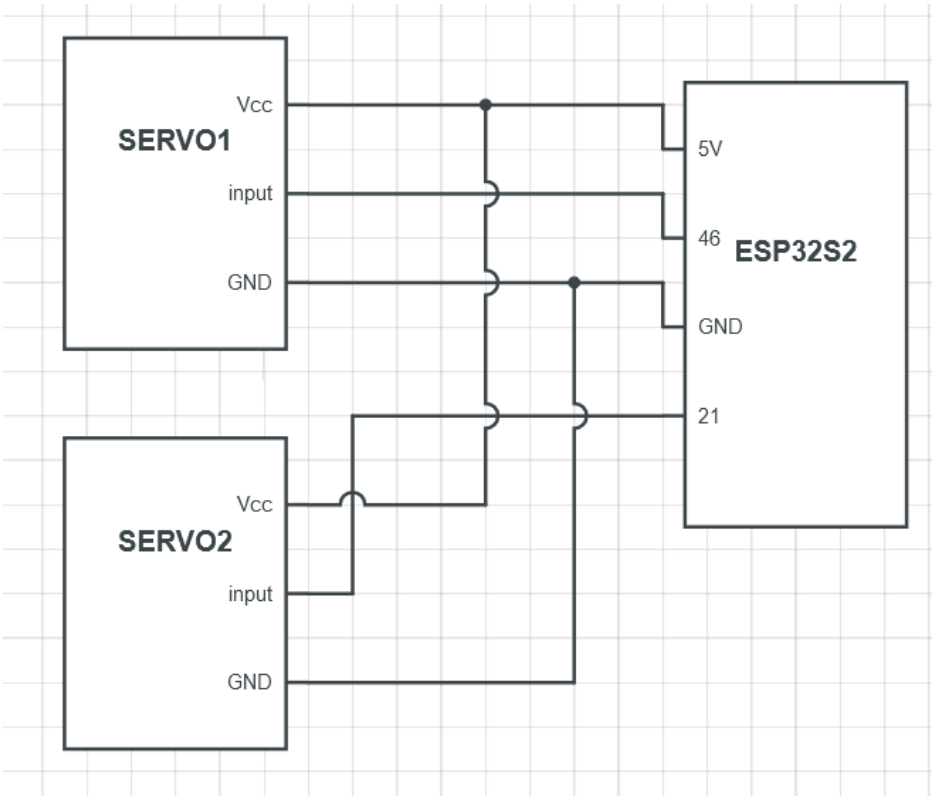
Motor Circuit:



Ultrasonic Circuit:

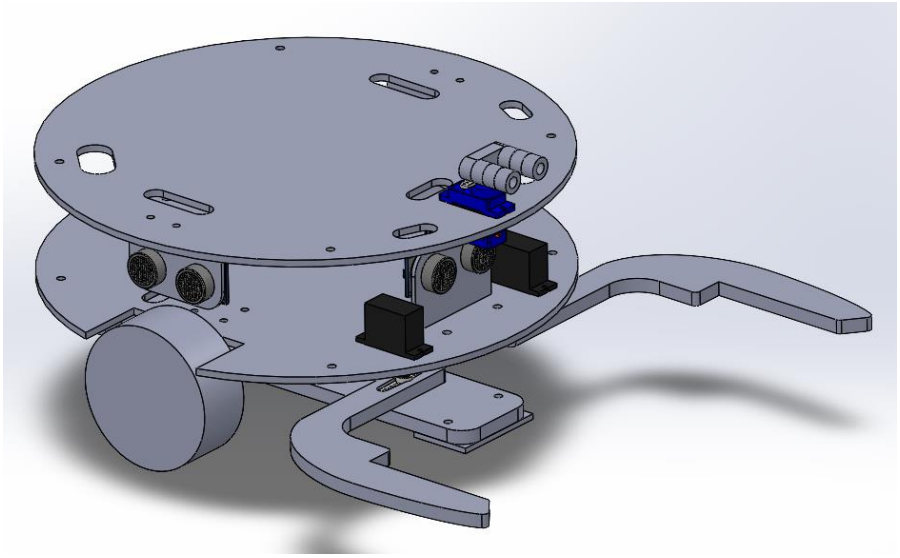


Servo Circuit:

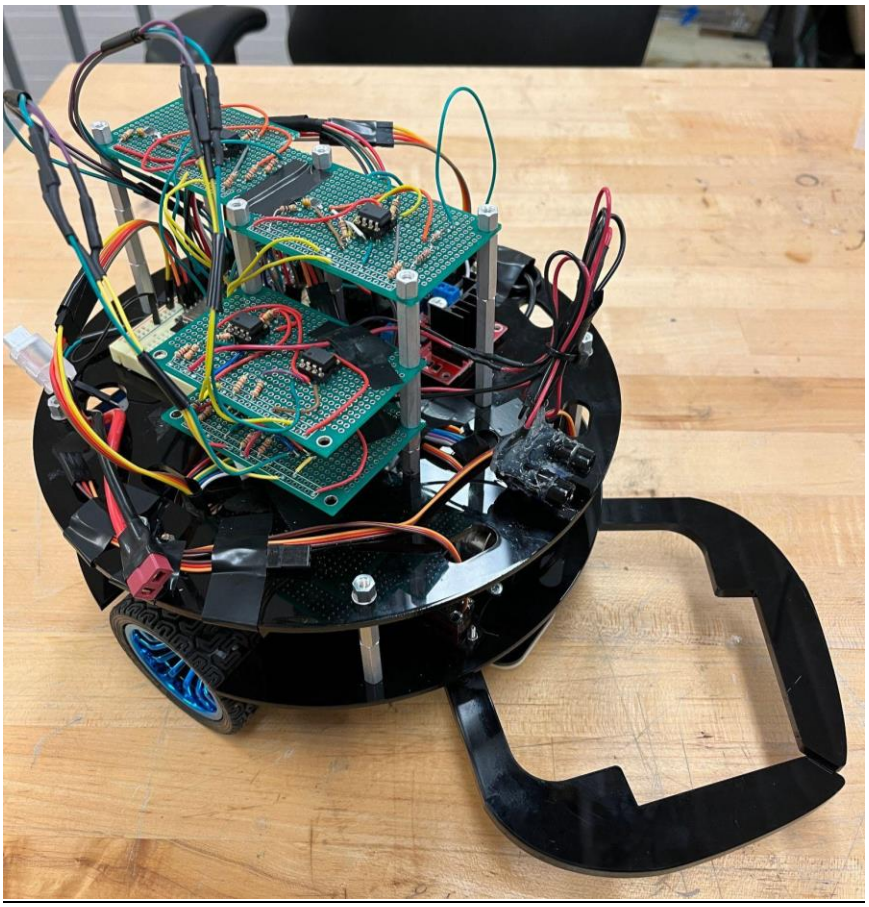


Full Robot Picture

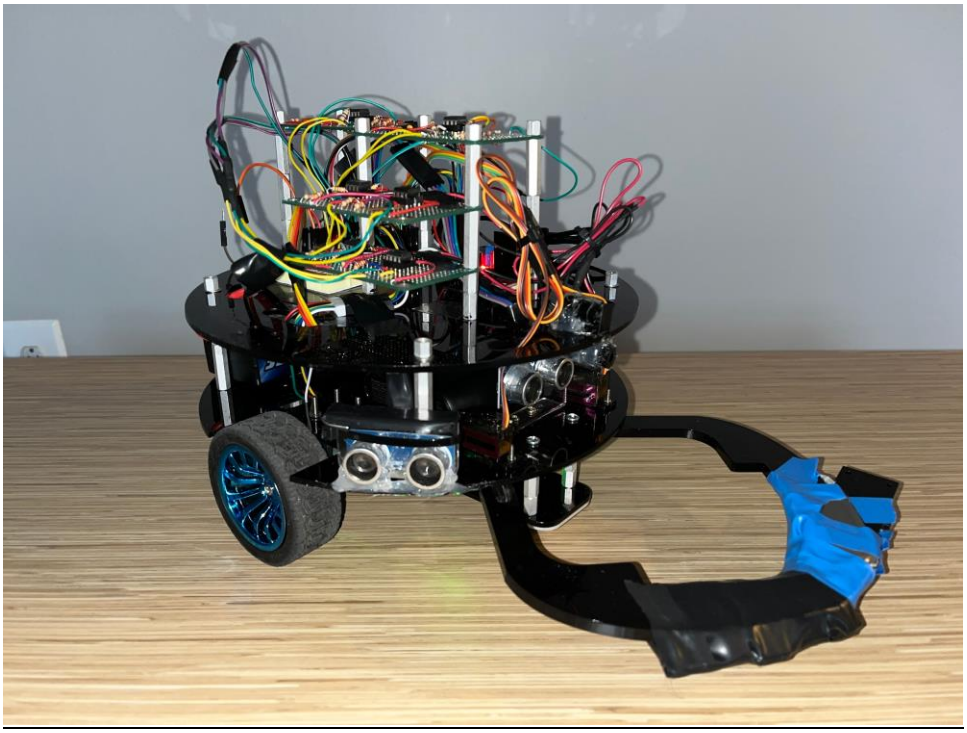
CAD Assembly:

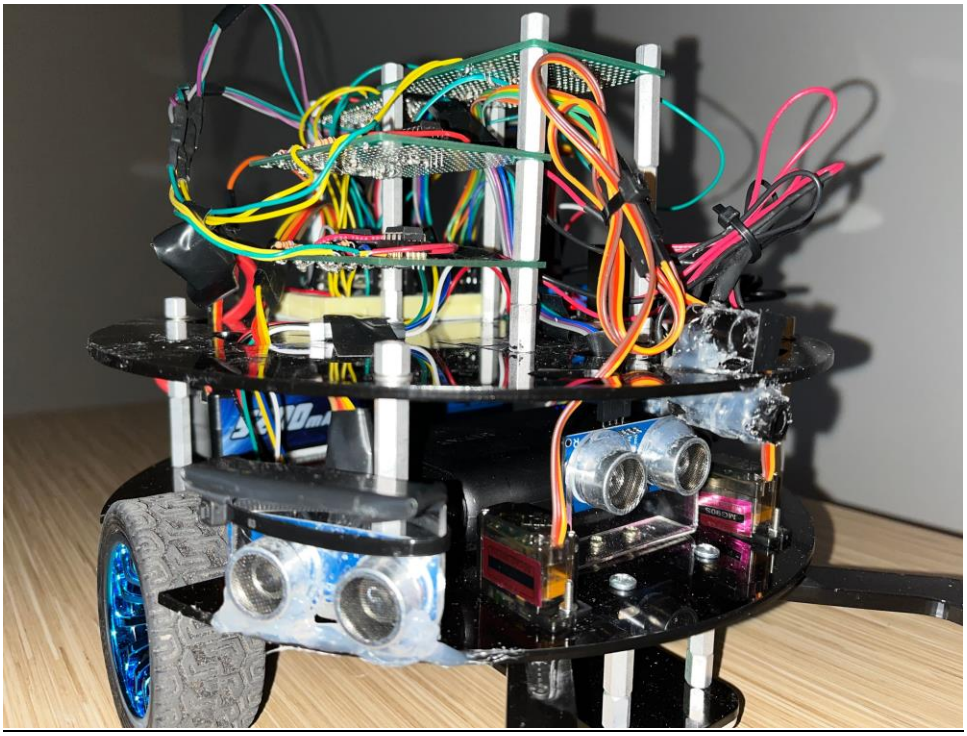


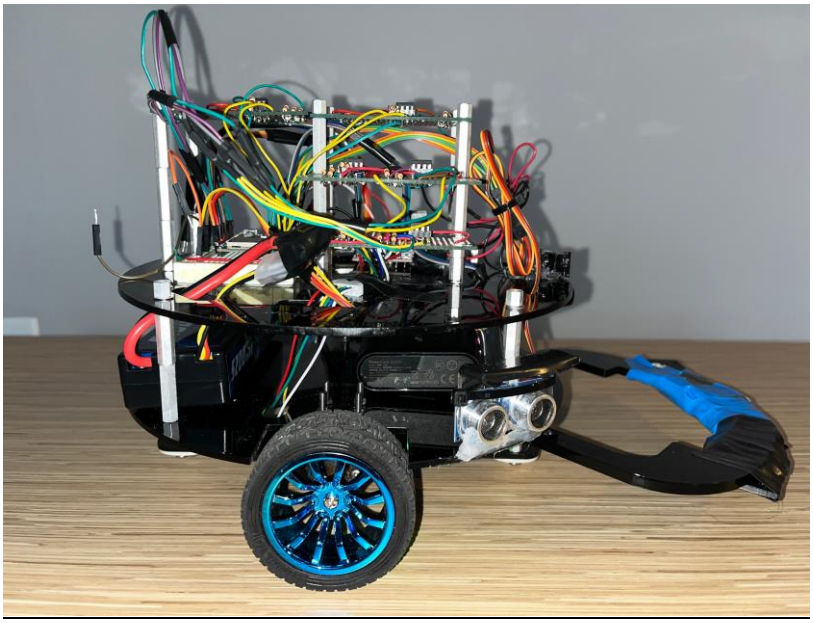
Pre-Graded Evaluation (Wednesday, December 13th):

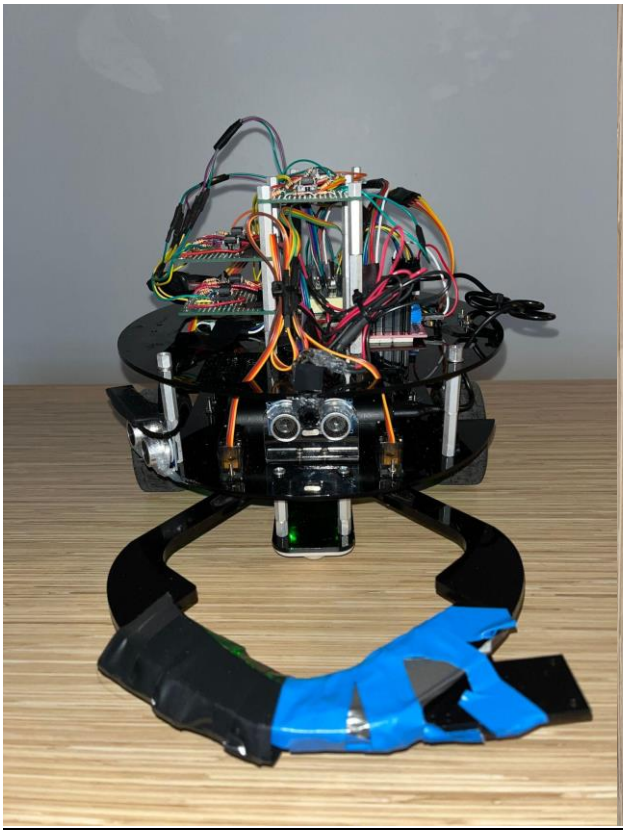


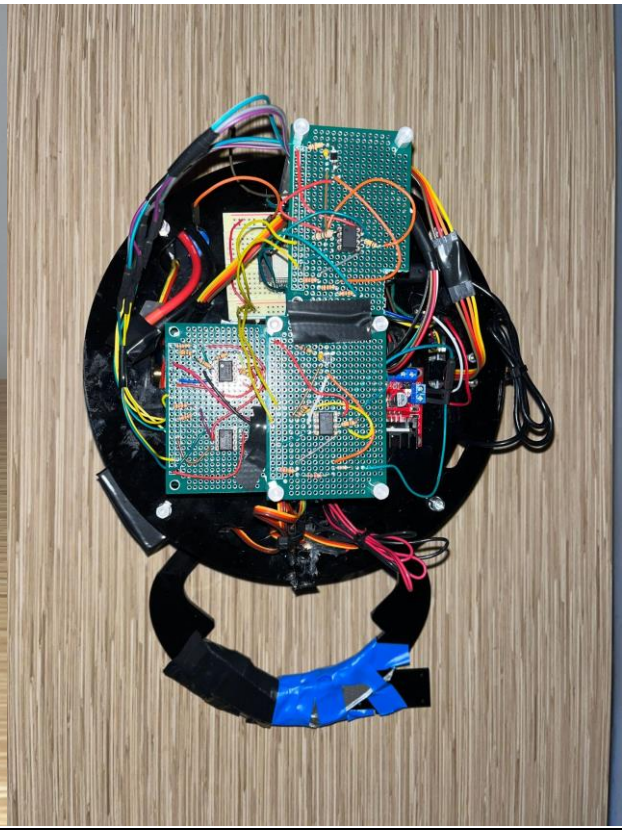
Post-Competition (Sunday, December 17th):





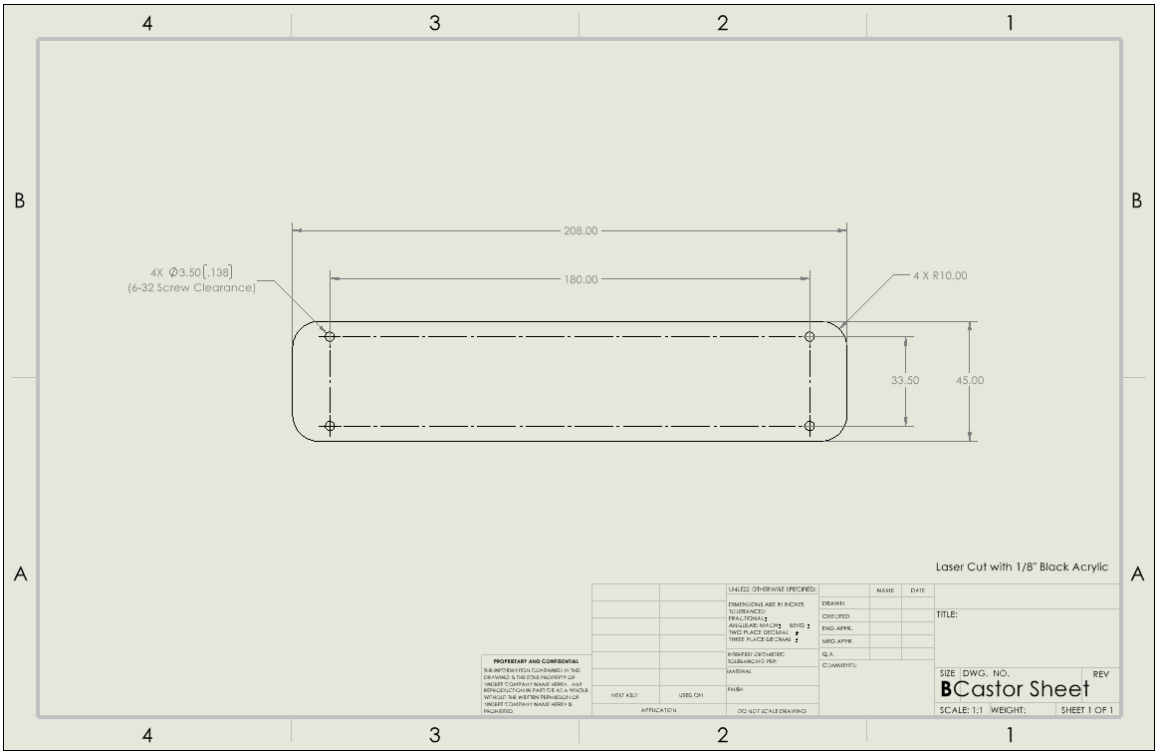




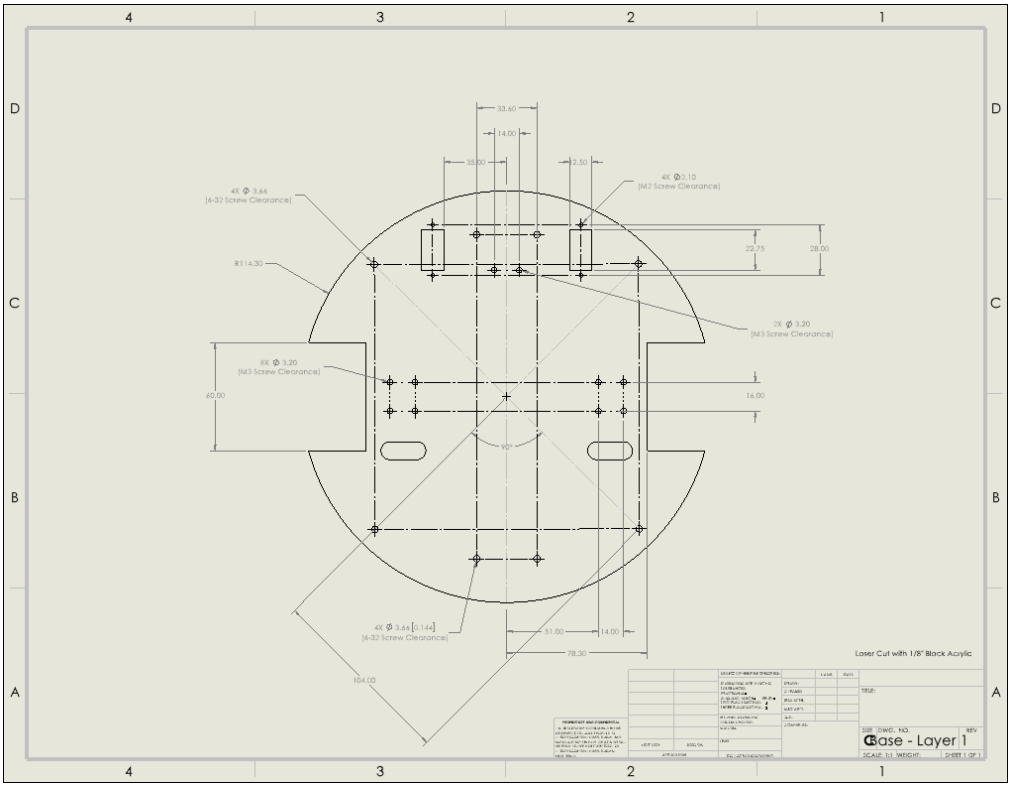


CAD Drawings

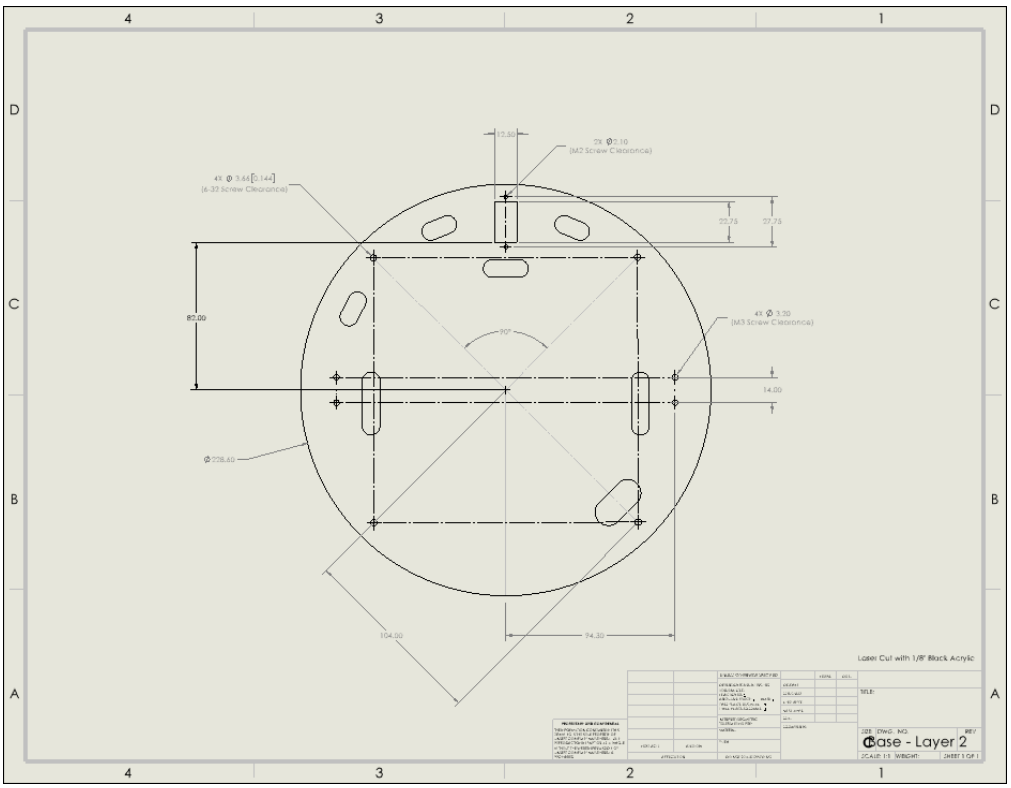
Castor Sheet:



Layer 1:



Layer 2:



Data Sheets

6V DC Motor (100RPM) w/ Encoder
MG90S Micro Servo 9g
L298N Motor Driver
HC-SR04 Ultrasonic Ranging Sensor

Videos

Wall Following:
https://drive.google.com/file/d/1eXDIELYGZb_qaybQwbGqnP2qwAMy05SG/view?usp=drive_link

Trophy Detection:
https://drive.google.com/file/d/1X9TpJK47fb97HGBgW0tev_KAa8v3xSlb/view?usp=drive_link

Police Car Pushing:
https://drive.google.com/file/d/1ny1efacAa69xrnA9_mroTXmiEWODy4Ce/view?usp=drive_link

Competition Videos

Match #9:
https://drive.google.com/file/d/1jaB1-s9k06LJGEAU6O8cUPNXGz_ctIpU/view?usp=drive_link