Final Project Report

Team 23: Zach Rudder, John D'Ambrosio, Cedric Hollande

MEAM 5100

Dr. Yim

12/21/2023

Functionality

Approach to Competition

Our main approach to the competition was to create and build a robot that was as autonomous and robust as possible. Our original plan was to build off our Lab 4 robot and improve the control by incorporating sensors and finetuning our KP, KI, and KD values for our new motors. While we planned on using two motors to power the rear wheels and a servo to control the front wheel, we decided to use a caster wheel in place of the servo-controlled wheel and flip our robot, so the powered motors were at the front. We decided to do this to increase our stability and control over the robot, which would ultimately help us throughout the competition. To push the police car, we bought new DC motors with a significantly higher torque output than the provided yellow motors. We also purchased new wheels with better tires which increased our robot's traction, helping it push the heavy police car. To aid in autonomy, we purchased three Time of Flight sensors that we intended to place on the three sides of our robot. We ended up not competing in the competition and decided to only use two of the ToF sensors.

Approach for All Functionality

In terms of the full functionality of our robot, our goal was to make it as straightforward as possible. We decided to keep the basic motor controls that we had from Lab 4 (move forward/backward, turn left/right, stop) and simply add on buttons that could be pressed that would enable one of the three task-oriented modes, these being wall-following, beacon-tracking, and locating the police car. We used a state machine in our code to set different cases for each of the three modes and utilized our "Stop" button as a way to stop all movement and exit out of any specific mode. This allowed us to have easy control over our robot when we needed to move it, be able to quickly enter one of the specific modes, and stop the robot at any point.

Performances that Worked and Failed

After putting in many hours building and programming our robot, we were able to get it to perform wall following well and consistently, and also able to locate and push the police car using the Vive sensors. We spent the majority of our time finetuning our PID controller so that our robot would move in a straight line during normal movement and would also use its two Time of Flight sensors to maintain a certain distance from the wall when it was in wall-following mode. In addition to wall-following, we were also able to use the robot's two Vive sensors to locate the police car, rotate towards it, move to it, and push it the required distance. While this was less consistent than the wall-following due to the amount of noise present, we were still able to achieve the desired results. For beacon-tracking, we successfully built the detection circuits and mounted them on our robot, but we ran out of time before we were able to fully implement them. We believe we have a solid base for the software, but would need to improve our circuits so that the phototransistors would be able to pick up the signal from further away. Additionally, we were unable to get UDP sending once per second to work. We are unsure as to why, but when

trying to incorporate the timer interrupt for UDP, the rest of our functionality would stop working. Since we were in a time crunch at the end, we decided to forego this and ESP-NOW and focus on wall-following and the police car task.

Mechanical Design

Intended and Actual Mechanical Performance

Originally, we planned on implementing a front wheel with a servo to control the direction, however, we realized it would be more stable for us to have two driving wheels at the front and a caster ball at the back. We bought two 1200 RPM 12V DC motors which were not very expensive, had built-in encoders, were better quality than the yellow motors, and were very powerful, being able to generate more than enough torque to push the police car and drive the robot smoothly.

We decided to go for a structure with three levels, separated by spacers, to keep an organized layout of all components and circuits. Since the robot would not carry very large loads, we decided to go for ⅛ inch acrylic. To make sure this was the right decision, we first did basic and conservative calculations to make sure ⅛ inch acrylic would be strong enough to support the whole chassis and components (such as the heavy battery) based on the estimated weight, area, and material properties of acrylic.

The ground level of the robot held the Li-Po battery, power bank, two Time of Flight sensors – one at the front right corner and one at the very front – the servo for the IR sensors, and the motors. To ensure the motors were properly attached to the base and did not shake too much, we built fastener and press fit pieces that fit in the base well and kept the whole system rigid and compact. The motor “enclosers” took a lot of trial and error to achieve a rigid and sturdy design that resisted strong vibrations/shaking. This limits losses due to friction but mainly keeps the car as steady and straight as possible in motion. The wheels we bought did not have the same mounting shape as the motors, which meant we needed to create some sort of fastening connector that would keep the wheels spinning smoothly. This was a challenging task and was difficult to accomplish with laser-cutting. The best solution ended up being a shaft coupler with a screw and nut holding the motor D-Shaft very tightly, which lasted us the whole project without any major problems once built. To power the motors, we used a 14.7V LiPo battery, as we found that normal cell batteries did not provide enough current and drained too quickly for our purposes. This was mounted right behind the wheels on the first layer, creating a center of gravity in the middle of the robot. Initially, we placed the side Time of Flight sensor at the point on the edge closest to the wall, however, we noticed that this was not as effective for wall-following as having it mounted on the front right corner, so we shifted it over.

The second level held the main electronic components, including the ESP32-S2, the motor driver, and the perf boards that held all of the wiring for the motors and encoders. On the third level, we mounted the two Vive sensors at the front and rear of the robot and also mounted two breadboards. One breadboard held the circuits for the Vive sensors, and the other held the circuits for the beacon detection.

Failures

In our project, we encountered a main mechanical failure where the wheels of our robot occasionally fell off. This issue was largely attributed to inconsistencies in the laser cutting process; the angle of the laser was not always perfectly set at 90 degrees across different machines, leading to a warp that compromised the stability of the wheel supports. Moreover, the choice of Acrylic for the wheel enclosures added to the challenge. Acrylic's brittleness was particularly problematic in areas with small fastenings that experienced the large force required to secure the wheel tightly to the motor's D-Shaft. Despite these difficulties, the overall design met our project requirements, and we continued using Acrylic, as the robot remained stable enough during all testing. However, resorting to 3-D printing would be a preferable alternative for applications requiring higher precision and accuracy.

Electrical Design

Intended and Actual Performance

To control our motors, we used an L298N motor driver. After performing research, we found that this was a popular choice for DIY mobile robot projects and could handle the 14V+ from our power supply with no issues. Additionally, it allowed for easy compatibility with our ESP32 board and has built-in protection features that protect the motors and microcontroller from voltage spikes. This allowed us to plug in our 14.7V power supply and power both of our DC motors without much worry. The motor driver worked as intended, however, our first one did stop working during testing and we had to replace it. The replaced driver did not have any more problems.

For wall-following, we used two VL53L0X Time-of-Flight sensors. We decided to use these ToF sensors because of their high accuracy and precision, fast measurement speed, and robustness against ambient light and object color. We believed having almost exact distance measurements would greatly help our robot’s performance on the wall-following task. While we encountered numerous problems setting up both sensors, including having a problem plugging two into the SDA and SCL ports, we were able to successfully use the sensors to perform wall-following once we found solutions. We encountered very slight noise from the sensors, but it was not enough to affect our wall-following capabilities since we were using a deadband range. Additionally, knowing the exact range was helpful for the police car task. Once we got

within “striking range” of the car, our robot would stop, back up, and accelerate into the car to push it back.

To locate the police car, we decided to use two Vive sensors (photodiodes). We used two rather than one so that in addition to determining our location, we could also calculate our orientation, which helped to move to the police car. We used the circuit that was provided by Dr. Yim in the lecture, which can be seen in the appendix. These circuits worked without much issue, however, we did burn up the first two photodiodes when trying to solder them to a perf board. We did encounter a great deal of noise when using the Vive coordinate readings, but we decided to filter this noise from a software approach rather than an electrical one.

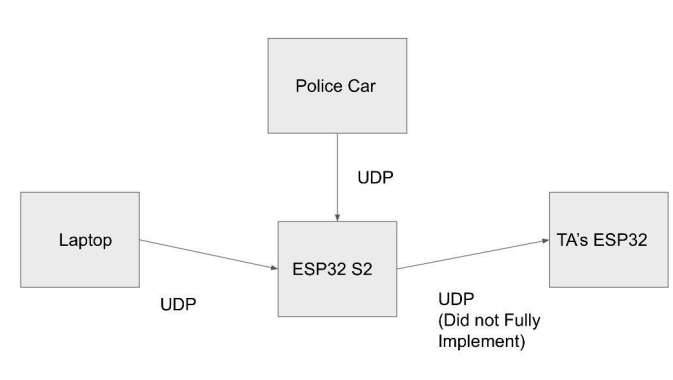
For beacon tracking, we used two phototransistors for the “Bang-Bang” method that was presented in the lecture. We mounted these on a servo and separated them with a piece of MDF. For the circuit, we used the circuit that was also provided in the lecture. However, during initial testing, we found that we were unable to record any frequencies that were emitted at a distance of more than a few inches, so we changed some of the resistors to increase the gain. This helped with increasing the range of the phototransistors, but we were still unable to record frequencies that were emitted more than one meter away. We were debating ways to increase the sensitivity of our circuit, or possibly use a software filter to amplify the signal further, but since we ran out of time and decided not to do beacon tracking, we did not pursue these ideas further.

Failures

The electrical component of this project was the most challenging for us. Numerous times throughout the project, our progress would get delayed by six or more hours to fix an electrical issue that popped up. One of the main problems we encountered was that our connections were not stable. To fix this, we soldered many of the components and wires, but after soldering and rewiring, ran into a new issue where our motors no longer turned on. After a few more hours troubleshooting this and finally fixing it by replacing the motor driver, we thought we would be ready to start working on the software again. However, we were delayed further by two new issues: we could no longer read both of our ToF sensors and all WiFi capabilities on our board had stopped working. These two issues combined took us over a day and a half to correct, but we were finally able to find workable solutions. In the end, we did not have any major electrical failures that we were unable to fix, but we did encounter many problems along the way that delayed us more than we were expecting.

Processor Architecture and Code Architecture

Block Diagram of MCUs



Software Approach

To develop all of the many software components for this project, we decided to work on each task separately, dividing them into their own file. We developed the software for wall-following, beacon tracking, and locating the police car independently from each other, and then combined all of them into one main file at the end. This was the easiest to ensure that parts of the code did not get lost or disrupted. Additionally, this modular approach allowed us to rapidly iterate and test our functions independently. Our main file is split into four sections: the first section is where all pins are initialized, global variables are declared, and instances of classes are declared, the second section is where all the functions for the various tasks are held, the third section is the setup, and the fourth section is the loop. Everything is grouped based on the task and/or functionality. For example, all the pin definitions for the motors are grouped, and all the setup functions are grouped above the main setup. The main loop is controlled by a state machine which has four cases: wall-following, beacon tracking, locating the police car, and the default. Whenever the specific button for a task is pressed on the webpage, it changes the current state and starts performing that task. Before any button is pressed, or after the "Stop" button is pressed, the state changes to the default state, where we have manual control over the robot's movements from the webpage. The use of the state machine proved to be effective, as we were able to reliably control the robot from our webpage.

In terms of wall following and overall control, we found that we needed to entirely retune our PID controller from Lab 4, as the new motors and new mechanical designs resulted in a brand-new system. Our controller class was very similar, but we added two key changes: an integral anti-windup and a *dt* parameter. The integral anti-windup would constrain the integral error between our maximum and minimum actuation values, which were -2047 and 2047, respectively. Doing this prevented our integral term from blowing up the controller, allowing for much faster response times, very minimal steady-state error (< 10%), and increased robustness to disturbances and voltage changes. The addition of the *dt* term helped our controller become

even more stable and reactive, as it would now calculate the errors as a function of time, allowing it to react as fast as our feedback sensors were sampling, which was 10 milliseconds. This term produced plots that reflected an ideal, second-order response, and allowed our controller to quickly respond to disturbances, which was essential for smooth and reactive wall-following. The only change we had to account for here was the KD term, as the derivative error became very sensitive due to the division of the 10 milliseconds term. After numerous hours of tuning, we eventually settled on a high KP, moderate KI, and very minimal KD, as the mass of the robot and friction of the tires added more than enough dampening.

For the wall following, we decided to add a second PID controller so that our robot's motors could adjust speed in real-time. We chose a desired distance and used our reading from the ToF sensor to calculate the error. We then created an instance of our PID class and chose a small KP and KI value to have a quick, slight response to the changes in distance. We chose smaller values so that our wall following would be smoother and less oscillatory. We then created a fuzzy logic controller and only applied the wall following control if the robot was outside of a 25-millimeter deadband range on either side of our target value. To apply the control, we subtracted the wall following control signal from the left motor control signal and added it to the right motor control signal. This allowed the left motor to speed up if the robot was too far, and the right to speed up if the robot was too close.

To locate the police car, we used the coordinates that we read from our front and rear Vive sensors to calculate the orientation of the robot. To do this, we took the inverse tangent of the difference in the Y-coordinate values over the difference in the X-coordinate values. This gave us the angle of our robot's heading. We then used the two coordinate points to calculate the midpoint of the robot. Once we had this, we then found the angle formed by the line connecting the center of our robot with the police car, once again using the inverse tangent method. We found the difference between these two angles to determine how the robot should rotate. We rotate our robot, either clockwise or counterclockwise depending on the sign of the angle difference, whenever the angle difference is greater than a threshold. Once this difference is within that target range of values, we stop rotating the robot and move it straight forward. Once it reaches the police car, it reverses slightly before ramming into the police car at a high speed and repeats this motion multiple times to ensure that it has moved the police car far enough.

Failures

The software implementation also proved challenging, albeit less error-prone than the electrical aspect. The issue that took the longest to fix was the PID controller, which affected general movement control and wall following. For a while, we had no term to account for the integral windup, causing our controller to blowup with even the slightest integral term, giving us suboptimal performance. Once that issue was fixed, we continued to experience strange and inconsistent behavior from our controller. In an attempt to make it more robust, we decided to

increase the sampling rate of our encoder readings, thus giving us a better response to the error changes. This still did not resolve our issues, so we ultimately added a *dt* term to our PID calculations, which finally produced the second-order system output we were looking for, in addition to making the response much faster. This further improved our wall following by making it more responsive and less oscillatory. Another issue we ran into was sending our coordinates using UDP once per second. Every time we tried to implement it in our final code, it caused all of our other systems to fail or work incorrectly. We decided not to use this for the graded evaluation and also decided not to incorporate ESP-NOW due to a time constraint. The final issue we ran into was with Vive localization. We were incorrectly calculating the target angle between our robot and the police car. Once we flipped that and normalized our angle values, we resolved the issue, although some trouble persisted due to the ambient noise that caused the coordinates to be very inconsistent.

Retrospective

Reflecting on our project, it felt like we faced every possible challenge, and even some we didn't expect – like that Monday when the S2 Wi-Fi wouldn't cooperate the entire day. We decided to stick with a workflow similar to what we used in the previous lab, where we each focused on one area and then came together every few days to troubleshoot. We started off working independently, each diving into our respective parts once we had our game plan laid out. After a few days, we'd regroup, share our progress, discuss any issues we were running into or might hit later, and figure out our next steps. In the last week, things ramped up. We were all in the lab, at least 10 hours a day, digging into every issue we could find, whether it was with the software, the hardware, or the mechanical design.

Overall, we all felt that we struggled most with the electrical aspect of the project and the class in general. All coming from MEAM with little to no experience in electronics, it was a much sharper learning curve than the software was. We spent many hours troubleshooting electronic issues we would run into, only to find out that it was a mistake as simple as not properly wiring the Vcc and ground of an op-amp. We all enjoyed the freedom of Lab 4 and the final project greatly. It was a very exciting experience to be able to brainstorm ideas, purchase necessary components, build the individual aspects, and finally be able to put everything together into a working robot. Throughout the entire class, we learned many valuable lessons such as reading datasheets, C/C++ fundamentals, and how to read the value of a resistor.

Appendix

Bill of Materials

I. 1 ESP32-S2-Saola-1
II. 1 SG90 Servo
III. 2 VL53L0X v2 TOF Sensors
IV. 1 L298N Motor Driver
V. 2 PD70-01C Photodiodes
VI. 2 LTR4206 Phototransistors
VII. 2 Orange Wheels
VIII. 2 12V DC Motors with Built-In Encoders
IX. 1 14.7V LiPo Battery
X. 1 Miady Power Bank
XI. 1 Ball Caster Wheel
XII. 3 Perf Boards
XIII. 2 Breadboards
XIV. 8 TLV272 Op-Amps

Schematics of Electronic Circuits

I. Vive Circuits

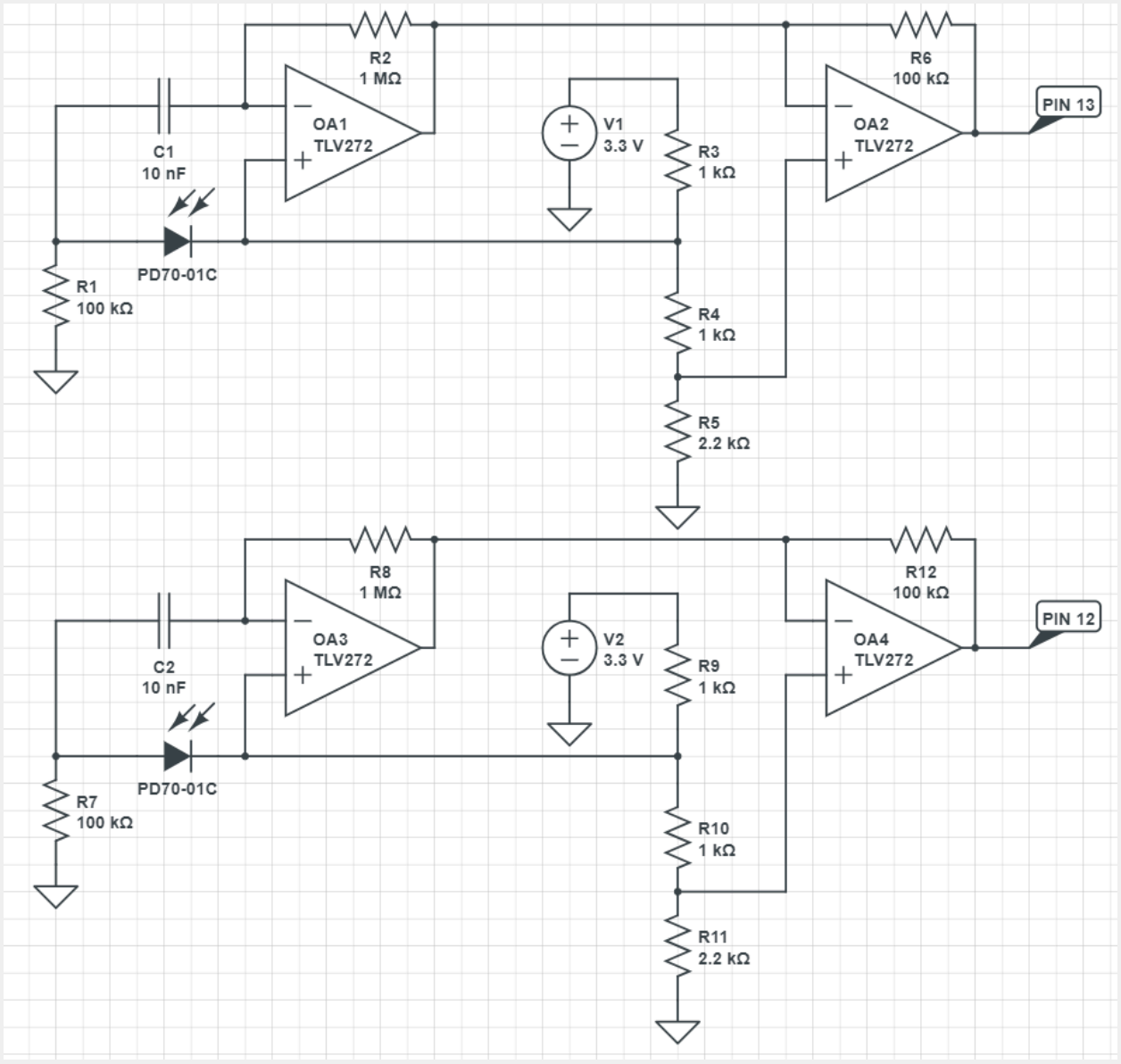


Figure 1: The top circuit is for the front Vive, and the bottom is for the rear Vive. ESP32 is not shown.

II. Beacon Tracking Circuits

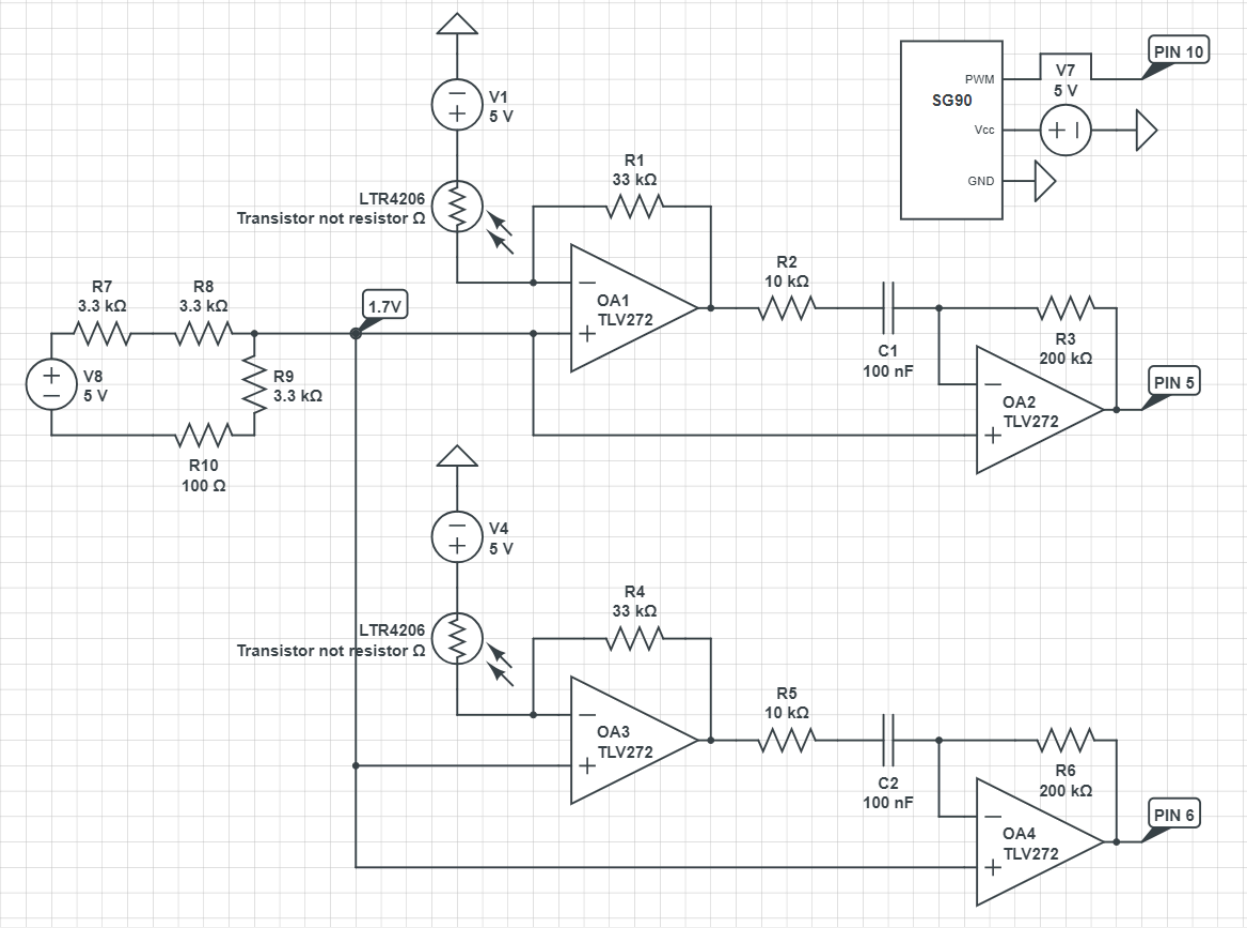


Figure 2: The top right circuit is for the servo that held the phototransistors. The top circuit is for the left phototransistor, and the bottom circuit is for the right phototransistor. ESP32 is not shown.

III. Motors and Encoders Circuits

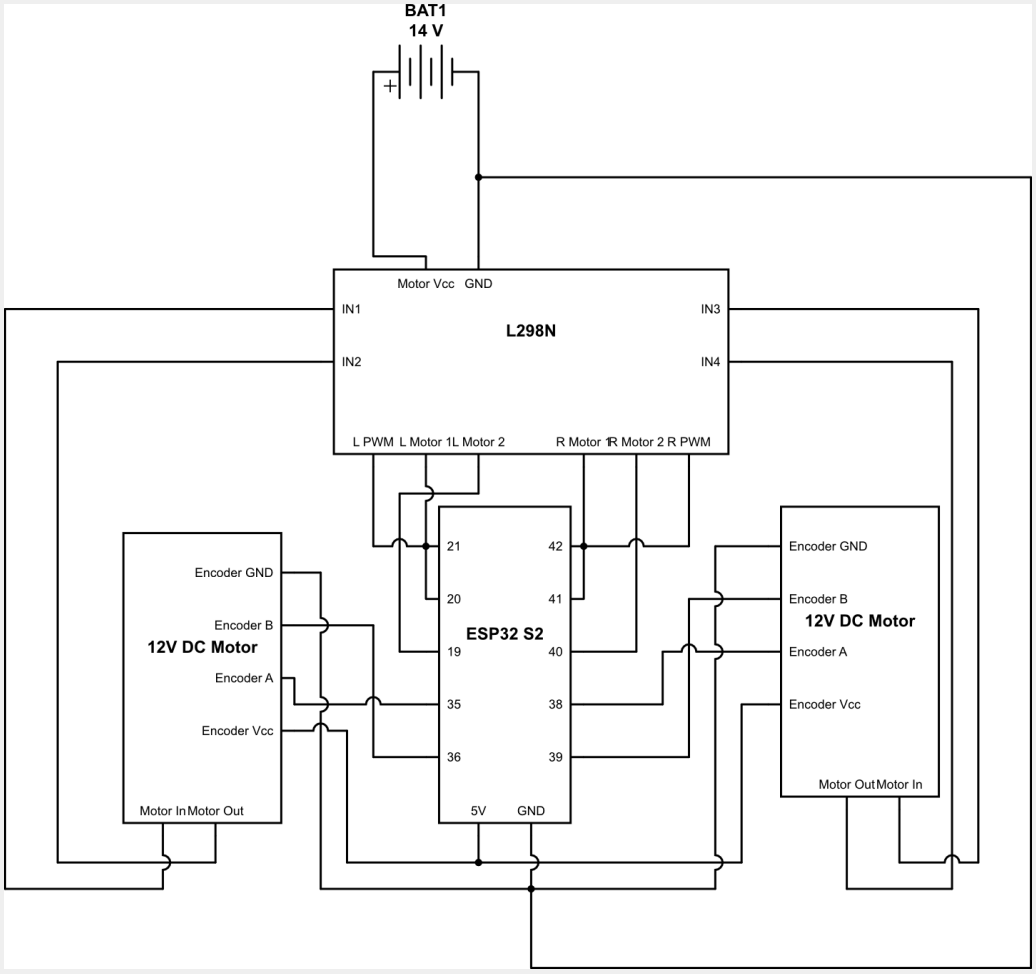


Figure 3: The circuit for the two motors with built-in encoders.

IV. TOF Sensor Circuits

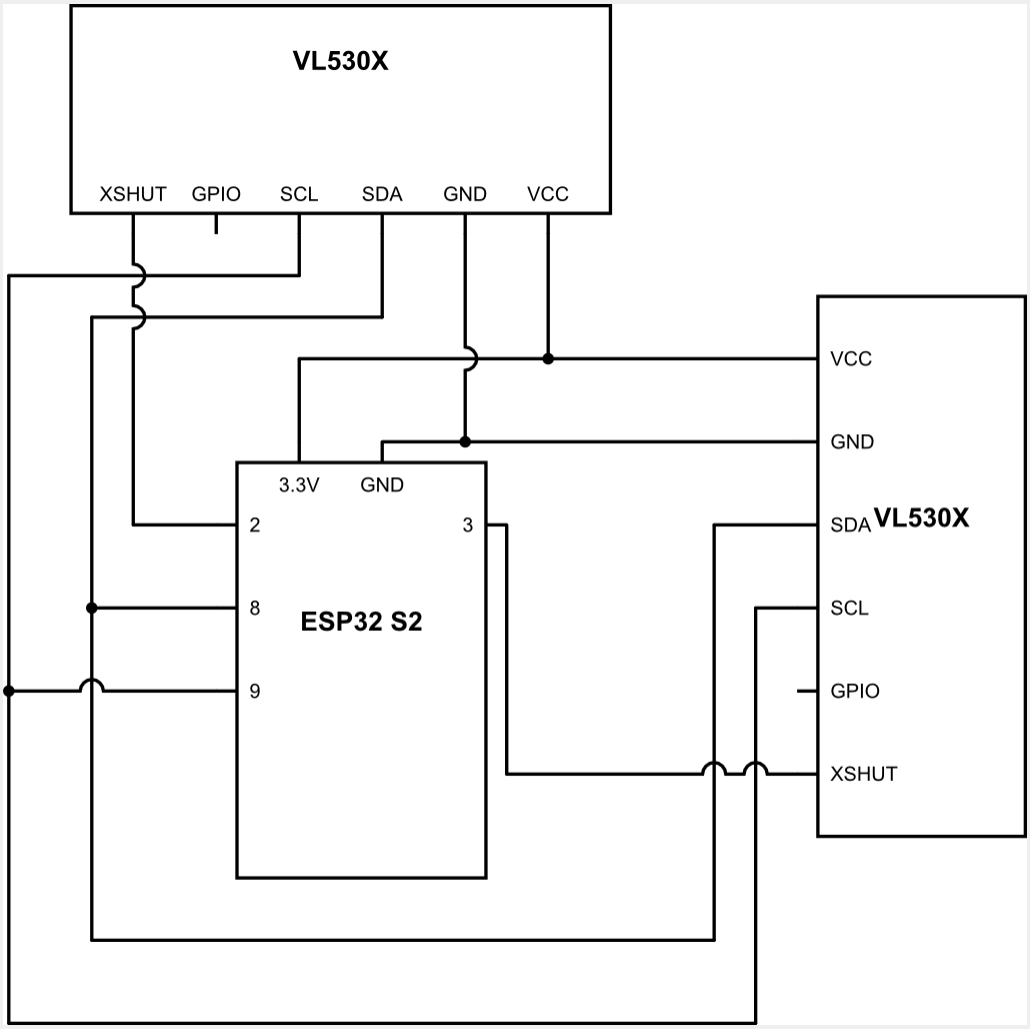


Figure 4: The circuit for the two ToF sensors.

V. General Circuit

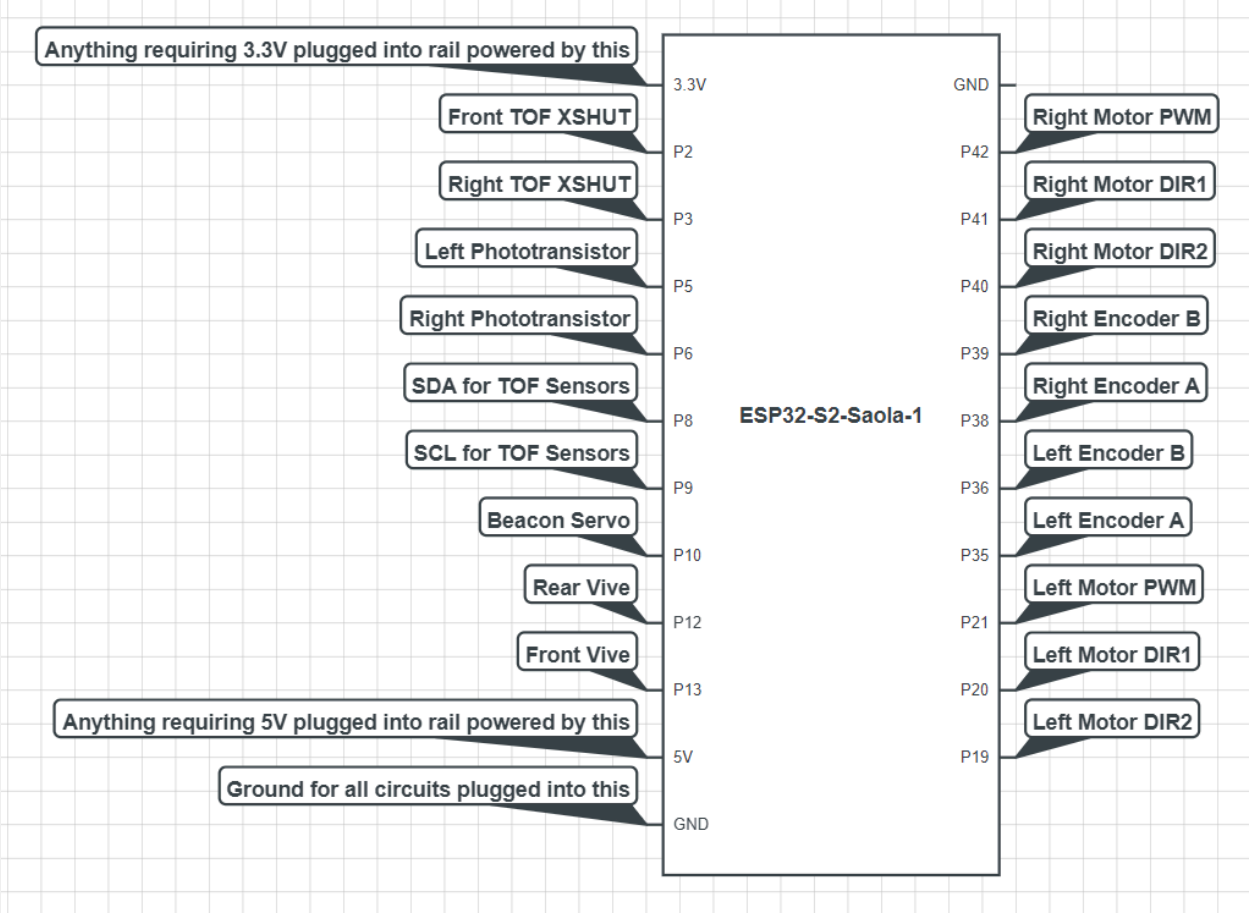


Figure 5: The microcontroller that controlled all other circuits. Not shown in Figures 1 and 2, not all pins showed in Figures 3 and 4.

Full Robot

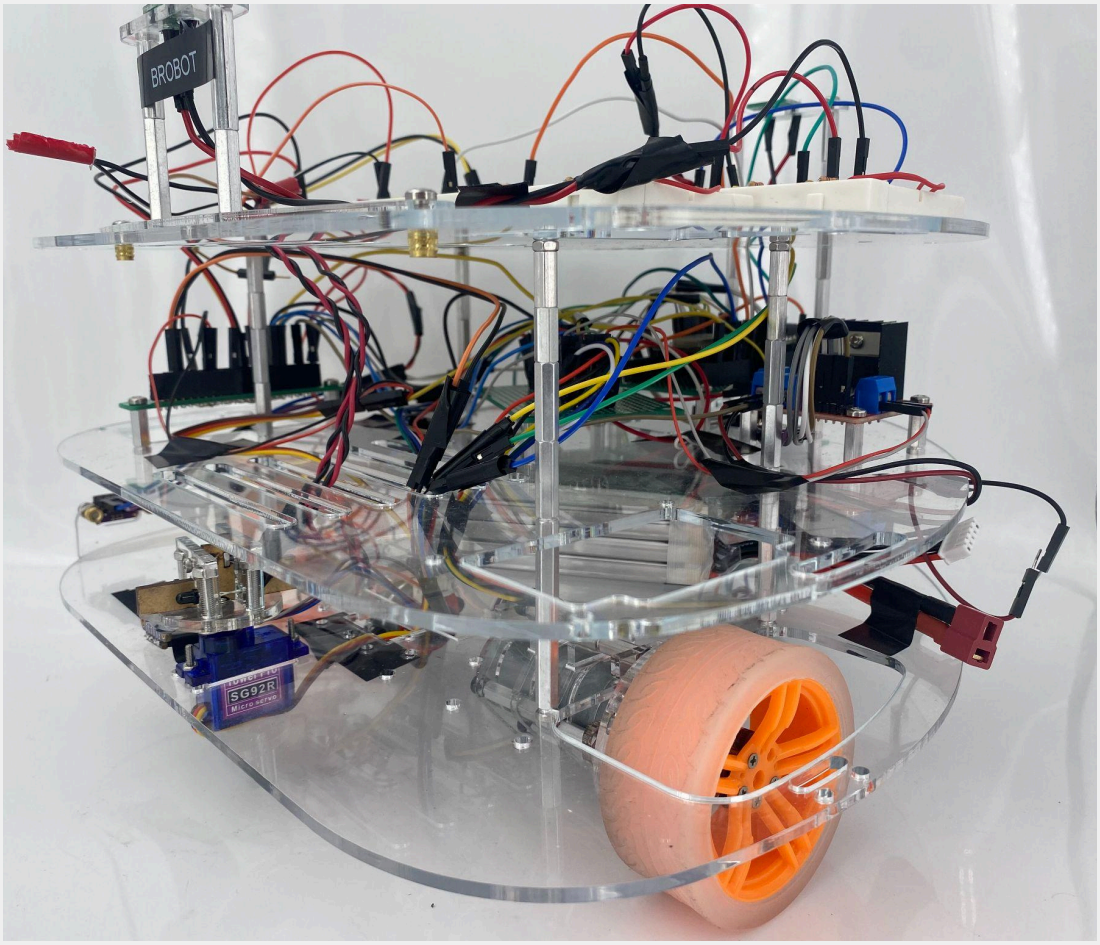


Figure 6: A photo of the finished robot, named “BroBot”.

CAD Drawings

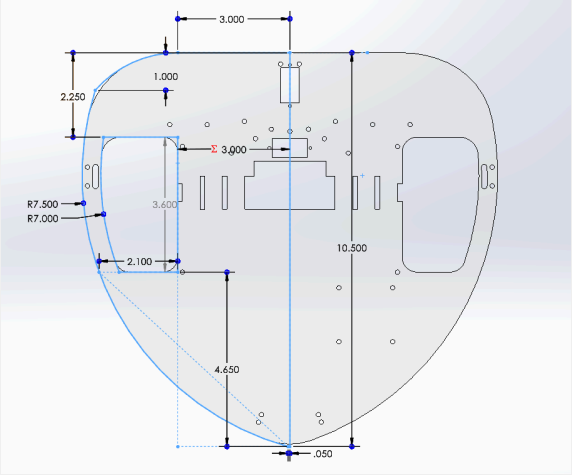


Figure 7: Base Plate

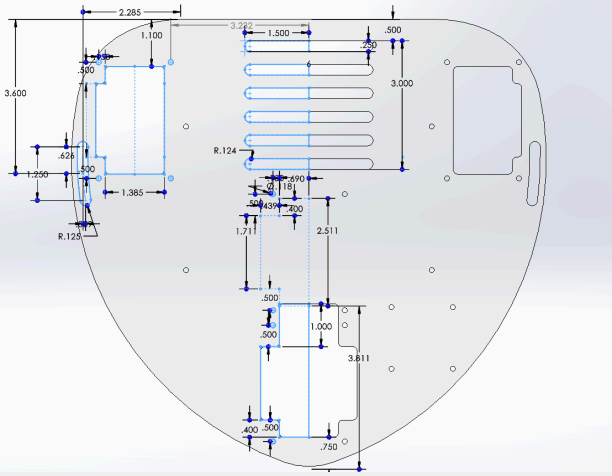


Figure 8: Top Plate

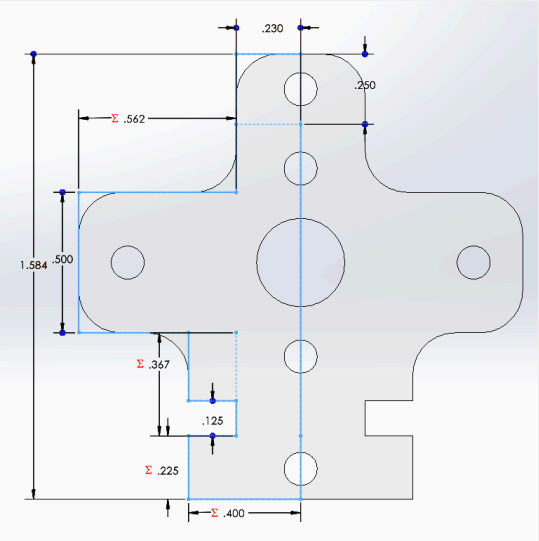


Figure 9: Motor encloser 1

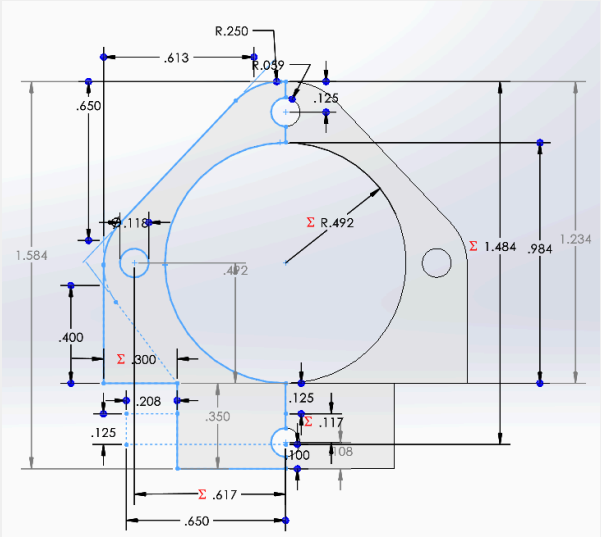


Figure 10: Motor encloser 2

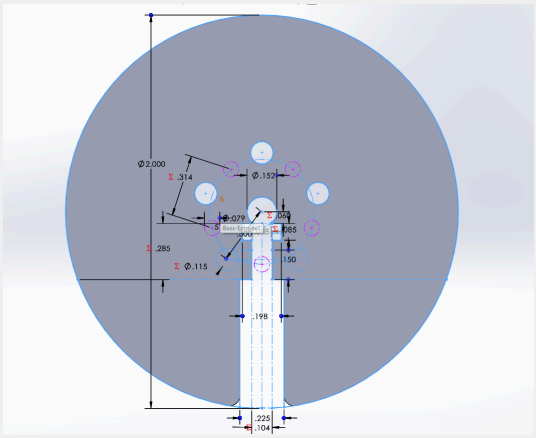


Figure 11: Wheel fastener

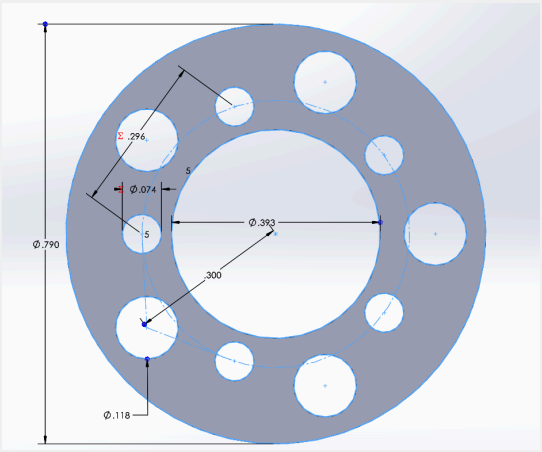


Figure 12: Wheel adapter

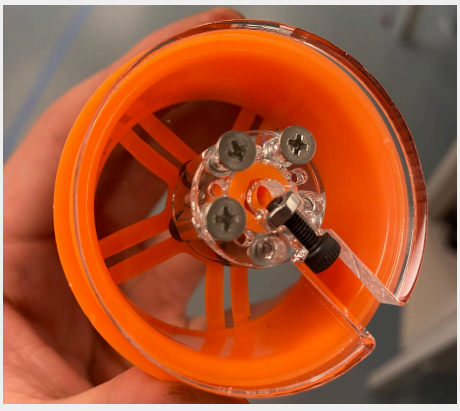


Figure 13: Custom wheel mount.

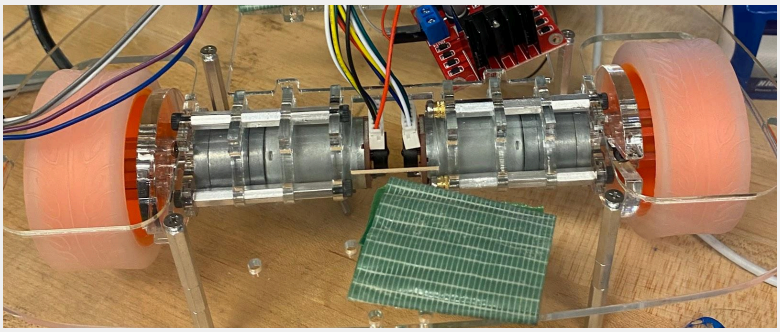


Figure 14: Custom motor mount.

Datasheets

I. DC Motors - Specifications in Description
I. TOF Sensors
II. Motor Driver

Image of Webpage

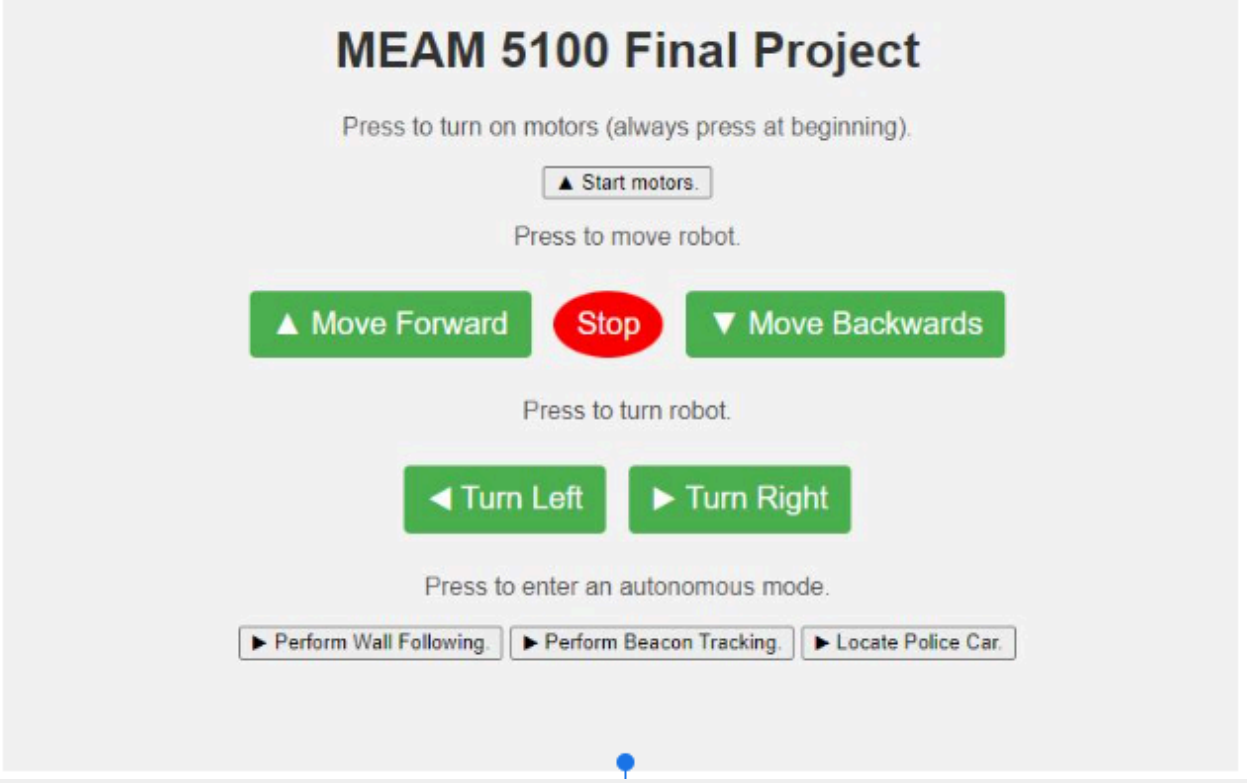


Figure 13: Custom webpage used to control the robot.

Videos of Functionality

I. Wall-Following
    A. Wall Following.MOV
II. Locating Police Car
    A. Police Car Track.MOV
III. Custom Wheel Mount Testing
    A. Wheel Mount Testing
IV. Bloopers
    A. Blooper 1.MOV
    B. Blooper 2.MOV