MEAM 510 Final Project Report

TEAM 40: Nick Genovese, Evan Grant, Paul Owens, Richard Yeh

December 22, 2021

Dr. Mark Yim

Functionality

In creating our semi-autonomous robot for the evaluation and competition, we focused on developing a robust and reliable system which prioritizes simplicity and quality hardware. For instance, we planned our system such that only one ESP32 microcontroller is required - avoiding additional communications an additional ESP32 would introduce. To achieve this goal, we needed to keep IO requirements low via the following design choices: reducing the number of range sensors from our original design, using only two motors, designing a can gripper actuated by a single servo, implementing robot ID selection entirely in software, and more.

Our approach to performing well in the competition involved designing a bot with excellent mobility and grasping system such that our human driver could easily and efficiently collect all cans and the beacon on our team's side before initiating limited autonomous behavior.

For maneuverability, we selected a mobile base comprising 2 rubberized wheels attached directly to our two motors' shafts, and a single ball caster opposite them. This configuration was selected over a 4-wheel approach which would have introduced considerable turning friction which could overwhelm our relatively weak motors, as well as an omni-wheel design which would introduce unnecessary control complexity. As an added benefit, we can steer and control this base treating either direction as forward, and we got familiar driving with the gripper in both the "back" and "front" - ultimately reducing necessary turning time in the competition. In lab 4, our team constructed a mobile base circuit with a 5V source and motor drivers which could only effectively deliver 3V to each motor. To remain competitive in this game, we redesigned our motor driver circuit with a 6V source dedicated solely to powering the motors, and selected higher-quality MOSFET motor drivers with significantly lower voltage drop. This allowed us to dynamically set speed along a wide range to perform precision or fast-moving maneuvers.

For autonomous behaviors, we opted to implement the simplest solutions which we could get to work reliably. Wall following was performed using two ToF sensors mounted along the front and right side of the bot, enabling counter-clockwise movement around the perimeter of the field. Our initial plan for our beacon tracking system involved a scanning servo. However, after opting for a 2-wheel mobile base, we decided to scan by driving the bot to rotate about itself and move forward when a signal is detected. Finally, our autonomous movement to a specified X-Y location relied primarily on our dual-photodiode Vive circuit. With two sensors, we could simply find our position and orientation on the field then determine the direction to steer while applying a proportional gain to smoothly approach the target position.

During the competition, our robot performed largely as expected. We won our heats on day 1 by rapidly moving our cans into our scoring zone and then launching into autonomous behavior in any remaining time. This early success came from a maneuverable design with good control

and a grasping mechanism which could reliably grab and tow cans to the scoring zone. The control delay over video presented a challenge however, and we found that we needed to sacrifice our speed for precision control. For autonomous behaviors, we relied primarily on wall following to enter and remain in the enemy zone and attempted beacon tracking to locate and retrieve the enemy beacon. However, the presence of two beacons resulted in conflicted movement in our bot and we were unable to isolate the correct beacon and perform the steal. Despite this drawback, we were able to remain fully autonomous while attempting a steal in the enemy zone - and were awarded autonomous points for that time.

Mechanical Design

Our robot was intended to be an upgraded version of our mobile base from Lab 4. The robot chassis was laser cut using ⅛" acrylic. Mechanical components were purchased from various distributors and attached to the chassis. The intended function of our robot was to build a robust, easily maneuvered, and efficient robot.

The robustness of the robot would allow us to maintain functionality through contact with other robots and vibrations caused by the terrain. Having a robot that is easily maneuvered would allow us to control our functions and movements with ease. This would prove useful in the graded evaluation as well as the competition. An efficient robot would save battery power and grant us the ability to accomplish tasks with ease.

In order to attain these criteria in our robot, we implemented a few specific design constraints. The first being our steering mechanism. In our mobile base, we used two-wheel differential drive with a ball caster. Our wheel selection changed from our mobile base. We designed for larger wheels to gain better traction on the field. In selecting our motors, we investigated the performance of the yellow hobby motors from our mobile base. The performance of the mobile base was not as high as we wanted, so we needed an upgrade. However, we decided to keep the yellow motors, and upgrade our H-Bridge motor driver instead. The yellow motors were already properly interfaced with AdaFruit wheels, so we kept using them for ease of manufacturing.

The mechanism we designed to grab cans was a gate system that had an inset cavity in our robot for the cans. This was so our robot could fit within dimension constraints and would keep its maneuverability by being concise. The gate mechanism had two states; open and closed. The gate was activated via a hobby servo from Lab 3: Waldo. This servo has plastic gears, and could only support a certain amount of torque, so we needed a way to ensure the servo would not break or draw unnecessary current. When closed, the gate would lock into place up against the chassis and would not require any torque output from the servo even when towing a can.

In our chassis design, we wanted to mitigate the risk of our robot becoming stuck on objects, walls, or other robots. To do this, we tapered our chassis from the nose to the wheels to prevent head-on collisions and also partially enclosed our wheels. In terms of board allocation, our electronics were located in tiers above the chassis top layer. Our electronics were wired so we would have quick access to each section; motor driver circuit, beacon tracking circuit, vive circuit, and logic circuit. The vive sensors and IR photodiodes were located approximately 11 inches from the ground. This was to ensure the vive sensors would maintain connection at all times without interference, and our diodes would accurately sense the beacons at a distance. We had two sources of power. A battery pack with four AA batteries was used to output a nominal 6V to the motors. A 5V power brick was used for the other circuits and servo control.

Our robot's mechanical design performed as intended. Our robot maneuvered easily while rarely getting caught. Our motors, wheels, and steering mechanism were efficient and robust. Our gate mechanism secured cans and beacons tightly. Additionally, our boards were easily accessible during testing.

This well-performing robot was not our first version. We designed multiple prototypes and continually redesigned them to improve function. An earlier design included exposed wheels, which we realized would occasionally get stuck on walls or other obstacles. Our grabbing mechanism was originally designed as a mechanical loop and lock mechanism, but proved unsuccessful at grabbing cans that were adjacent to the wall. This was addressed by having a gate system that could drive flat up the wall. Another early design change we made was to the forward facing direction of our robot. Originally, our grabbing mechanism was in the back, so that our caster could be in the front, which was easier to maneuver. However, this made it difficult to grab cans efficiently using controls. We flipped forward and backward to make the gate of our robot the front facing side. Our robot performed very well in terms of mechanical components. We tried and tested our robot consistently to ensure its success. Pictures and drawings can be found in the appendix.

Electrical Design

Beacon Tracking

The goal of our beacon tracking system was to use two IR phototransistors and a turning motion to sense the direction of a 700 Hz or 23 Hz beacon and drive straight towards that beacon. In code this was implemented on the Teensy in a loop checking if the beacon is visible to the two phototransistors. If a beacon is detected, a digital signal is sent to the ESP32 which will instruct the motors to gradually move forward. From an electrical perspective, our main goal was creating a circuit that amplifies the response of the IR phototransistors to the point where they could sense the beacon from across the field. To accomplish this we created a circuit that used an AC coupling capacitor as well as an inverting op amp with a voltage divider which centers the signal at 2.5V. The circuit diagram can be found in the appendix.

We decided to use the Teensy for our beacon tracking due to the fact that we already had functioning code that worked with the IR phototransistors from Lab 2. One issue with using the Teensy came from the fact that we had only used one input capture with the Teensy throughout the semester, and in order to accurately sense the beacon, we required two IR phototransistors, and two input captures. The Teensy has a second input capture in addition to port C7 - port D4. For the final beacon system, we duplicated the above mentioned circuit and connected one IR phototransistor to each input capture.

Our two IR phototransistors were affixed to the robot roughly ½" away from each other facing the same direction. This small distance between the sensors allowed for a small range directionally where both sensors could sense the beacon. This would be used as the "forward" direction for beacon tracking. We realized that these IR phototransistors not only needed to be perpendicular to the robot, but also at a high enough location where they could sense the beacon. Because we found these phototransistors could sense IR signals from almost any direction, it was necessary to completely cover them on all sides except the front. It took multiple rounds of trial and error to find a suitable height and covering that worked for the beacon tracking.

Overall, we found success in running our beacon tracking in practice and checkoffs. If we had additional time, we most likely would have added a second High/Low pin between the Teensy and ESP32 so that we could choose between and determine if we were sensing the 700 Hz or 23 Hz beacon individually. In its current state, we had limited success with using the beacon tracking in competition, as we had no way to distinguish between the two beacons, leaving our robot “bouncing” between both.

Motor Driver Circuit

As mentioned previously, we redesigned our motor driver circuit to be able to meet the upper specifications for our motors (the same small yellow ones provided in lab4). This required reliably providing 6V to each motor. We also intended, in an effort to avoid noise, to isolate the motor circuit wherever possible from the surrounding sensors and microcontrollers. Thus, our redesign involved a dedicated motor power supply with a 6V nominal voltage from Qty. 4 AA batteries.

The SN754410 H-Bridge used in lab4 proved inadequate for this goal. The voltage drop was too significant, and resulted in only 3V reaching each motor from a supply of 5V. Thus, we upgraded to the MOSFET driver LV8401V with significantly lower voltage drop and a slight increase in continuous current output rating (1.2A). After installing these drivers, we found that we could now supply about 6.15V to each motor.

Our final circuit features a CD74HC04E inverter, with which we can use a single digital output from the ESP32 to specify motor direction. This works by directly feeding the pin output to one directional input on the LV8401V, and the inverted signal to the other. The only disadvantage of using this driver is that it offers only one motor channel and thus two drivers are required to power both wheels.

Note: while this new H-Bridge also offers braking, we did not utilize this feature because it would require two additional GPIO pins from the ESP32.

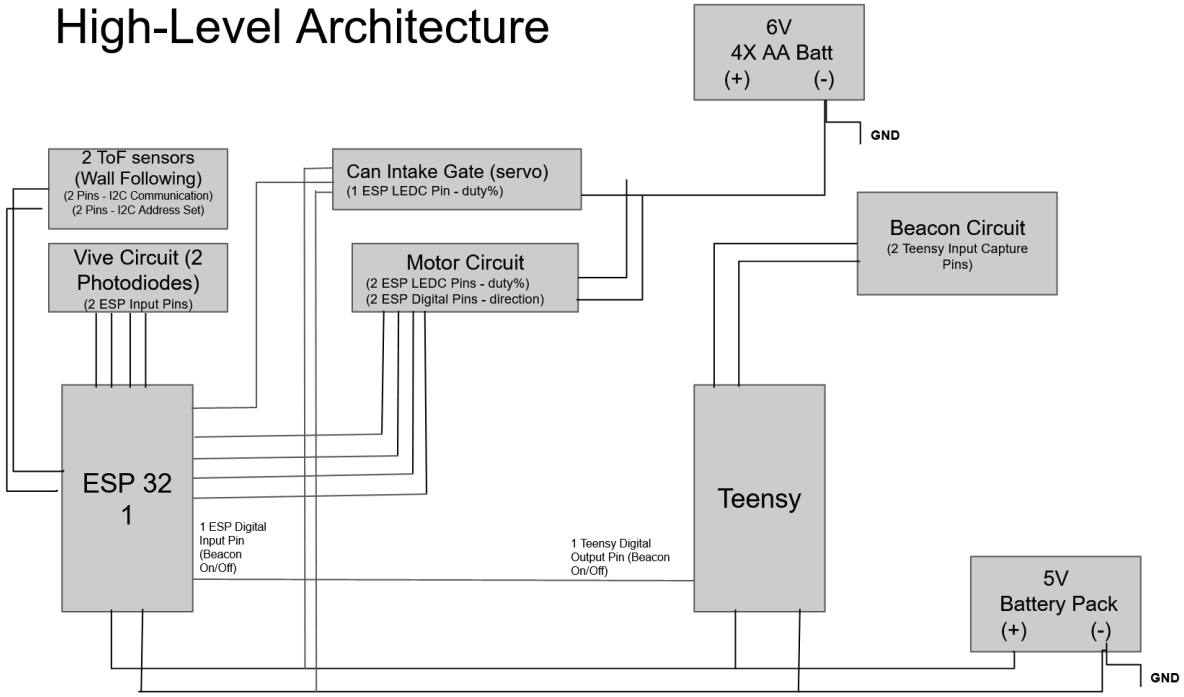
With this new circuit installed, the bot drove exactly as intended and was much more powerful than our lab4 solution. This prevented us from becoming stuck anywhere on the field and allowed us to implement speed control rather than setting maximum PWM duty cycle at all times.

Vive Circuit

The final Vive circuit remains largely unchanged from what was presented in lecture, except that two identical instances of it were soldered on the same board to support our 2-diode design. Additionally, some resistance values were changed for convenience while preserving approximate ratios for voltage dividers and gain settings.

Fortunately, this dual design worked as expected from the outset, and no further tweaking was required. However, we did often encounter an issue in which our x-y readings from the circuit would become “flipped,” often after entering the arena’s “dead zone.” It remains unclear if this behavior was the result of our code implementation or circuit design.

Processor and Code Architecture

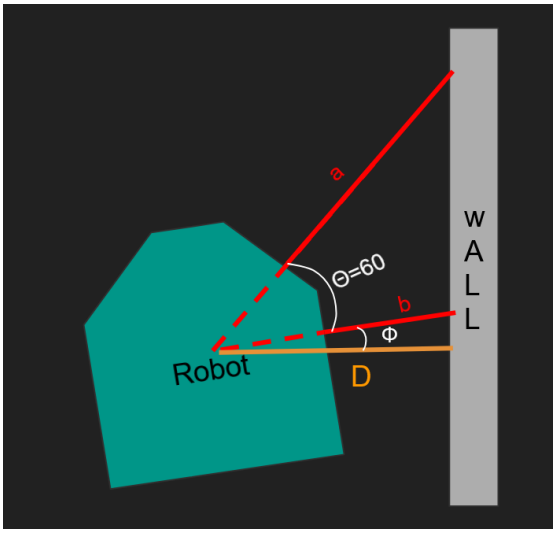


The above diagram is a high level architecture view of the signaling connections between subsystems on our robot.

From the ESP32, 3 PWM pins are used to control the two drive motors as well as a servo connected to the gate that opens/closes the can intake. An additional two digital output pins are connected to the h-bridges to control the directionality of the drive motors. 2 ESP input pins are used to read data from the 2 vive photodiode circuits on the robot, and 1 more input pin is used to interface between the Teensy and ESP32 which indicates when the beacon circuit detects either a 23Hz or 700 Hz signal. 4 pins are allocated to the two time-of-flight sensors on board, 2 for the data and clock lines for I2C, and 2 digital output pins for configuring the I2C addresses of each sensor during the setup phase.

The software approach was built upon the manual control code we had from lab 4. Additional UI toggles were added to the webpage that enabled or disabled each of the three autonomous behaviors on the robot (wall following, vive XY tracking, beacon tracking). A textbox was also added to input the robot ID used for UDP transmission messages as well as a toggle for the can intake gate. When any of the autonomous behavior toggles are enabled, a corresponding boolean for the behavior is set to True, and an if statement in the main loop accordingly runs the algorithm for the enabled autonomous mode. The algorithms for each of the three autonomous behaviors are described below and videos for each are available in the appendix.

Wall Following



The two time of flight sensors on the robot are used to calculate the distance D, between the robot and the wall, as well as the heading, Φ of the robot, using the below formulas.

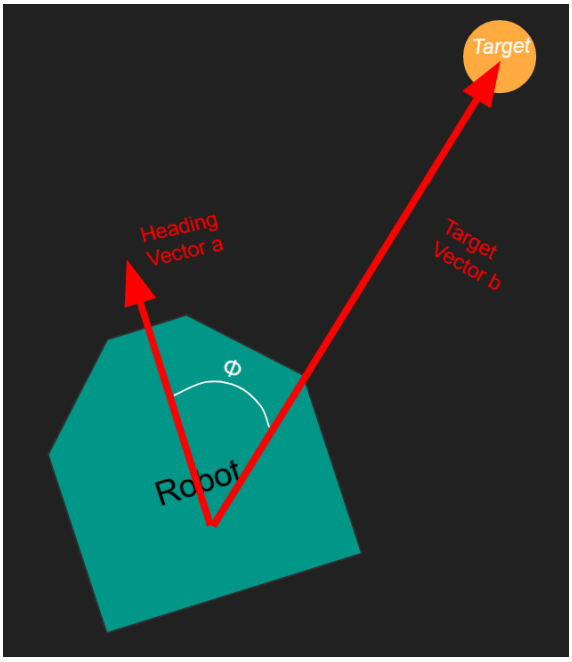
$$\phi = \tan^{-1}\left(\frac{a\cos(\theta) - b}{a\sin(\theta)}\right)$$

$$D = b\cos(\phi)$$

Together, these can be used to extrapolate the distance of the robot from the wall at a future time (lookahead distance). This extrapolated distance is then subtracted from the desired distance we wish to remain from the wall (currently set to 30 cm) to obtain an error value, which we then use as input to our proportional controller. When the error value is negative, we know that the robot is too far from the wall and needs to turn right, and when the error value is positive, we know that the robot needs to turn left.

This algorithm works for corners as well. As the time of flight data from sensor A (at 60 degrees) senses the approaching wall ahead (smaller values than adjacent wall), the algorithm knows to turn the robot left, as the extrapolated position of the robot results in a value below the desired distance from the wall.

Vive XY Tracking



To track a given XY coordinate, we calculate two vectors representing the heading of the robot and target location from the current position. The heading vector is obtained from the two vive circuits located at the front and back of the robot, by subtracting the x,y coordinates of the front circuit from those of the back circuit. Meanwhile, the target vector is obtained by subtracting the x,y coordinates of the target from those of the back circuit. The angle between these two vectors can be calculated using the dot product, represented by Φ. To determine whether the robot should turn right or left by the angle Φ requires the sign of the cross product between the two vectors.

With the angle Φ that the robot needs to turn at and the direction of the turn, a proportional controller is designed using Φ as the error term and sent as PWM signals to the left and right motors.

Beacon Tracking

IR detection of the beacons was handled by the Teensy, which used code adapted from Lab 2. The Teensy outputs a digital signal to the ESP32 indicating HIGH when either 23 Hz or 700 Hz is detected, or LOW when neither are detected. From this digital signal, we implement a simple control strategy, where the robot spins until an IR signal is detected (indicated by HIGH), and moves forward until that IR signal is lost.

During this process, we also keep track of the direction the robot spins. Each time the robot gains and then loses detection of the beacon, the robot will spin in the opposite direction of its last spin, allowing the robot to avoid spinning 360 degrees before finding the beacon signal again. With this strategy, our robot was able to consistently and efficiently detect a beacon from across the competition arena and effectively “lock on” as it traveled forward.

Retrospective

PAUL: I learned many things over the course of the semester. One of the most important things that I learned was the prototyping of electrical circuits. I have been a very mechanically focused engineering student, and have good prototyping skills for mechanical systems, but learning how to build circuits and basic electronic manufacturing was integral to my education. I thought the best parts of the class were learning new coding languages. I do not think the depth that we learned was too challenging for graduate students. I certainly had the most trouble with the non-tangible aspects of the electronics like timers and storage. This class was great and I would recommend it to other students interested in electronics.

NICK: I thought the best part of the class was iterating through designs and working through problems throughout the entirety of the final project. There were so many small challenges with incorporating all of the different features into one cohesive robot, and I enjoyed brainstorming mechanical approaches to the chassis and other mechanisms to enhance the electrical systems. I definitely had the most trouble with math related operations within C. For some reason I had multiple labs where using a float vs double vs long gave me real problems with outputting the correct mathematics, and I spent many hours with TAs attempting to figure these problems out. Other than that I thought that using C was really interesting and I learned a lot in a new coding language. The most important things I learned were probably electrical circuits, as it was something I was unfamiliar with at the start. I would recommend the class to anyone interested in mechatronics, and I got a lot out of it.

EVAN: Some of the most important learning outcomes of this course for me were likely integration, microcontroller programming, and electrical circuit design. As a mechanical engineering undergrad, I had startlingly little exposure to electronics, and this course provided an excellent opportunity to get hands-on experience designing and building circuits (and learning to read confusing datasheets!). Additionally, learning good practices for programming microcontrollers (including non-Arduino IDE) was huge, as I had only done really hacky solutions in the past. The best part of the class came from the opportunity to be creative and design solutions to the labs and final project. I think I struggled most with programming, especially when it came to the wifi requirements - I would have liked to have a better understanding of those methods but felt I didn't have the time. Perhaps the thing I wish was improved in this course was access to components in the GM lab. Often there was a scarcity of necessary components (IR phototransistors, H-Bridges, etc.) during a given lab, and I feel that in future classes the Qty in stock should already be enough to cover all students. Apart from that, this course was perhaps my favorite this semester. Time consuming, sure, but exactly what I hoped to become involved in during my Master's (integration of mechanical, electrical, and programming). I look forward to taking an advanced mechanics course next!

RICHARD: The most important thing I learned from this class were the challenges of integration of several subsystems and electrical circuit design for sensors and small signals. When working on the labs and especially for the final project, I discovered that even small sources of noise or interference from different subsystems when integrated together can greatly affect the overall performance of the entire system. I also learned about preferred ways for programming microcontrollers like event-based programming to help increase the efficiency of my code and make debugging easier. The best part of the class was seeing the culmination of what we learned being realized in an actual robot that we could control and compete with.

Appendix

Bill of Materials

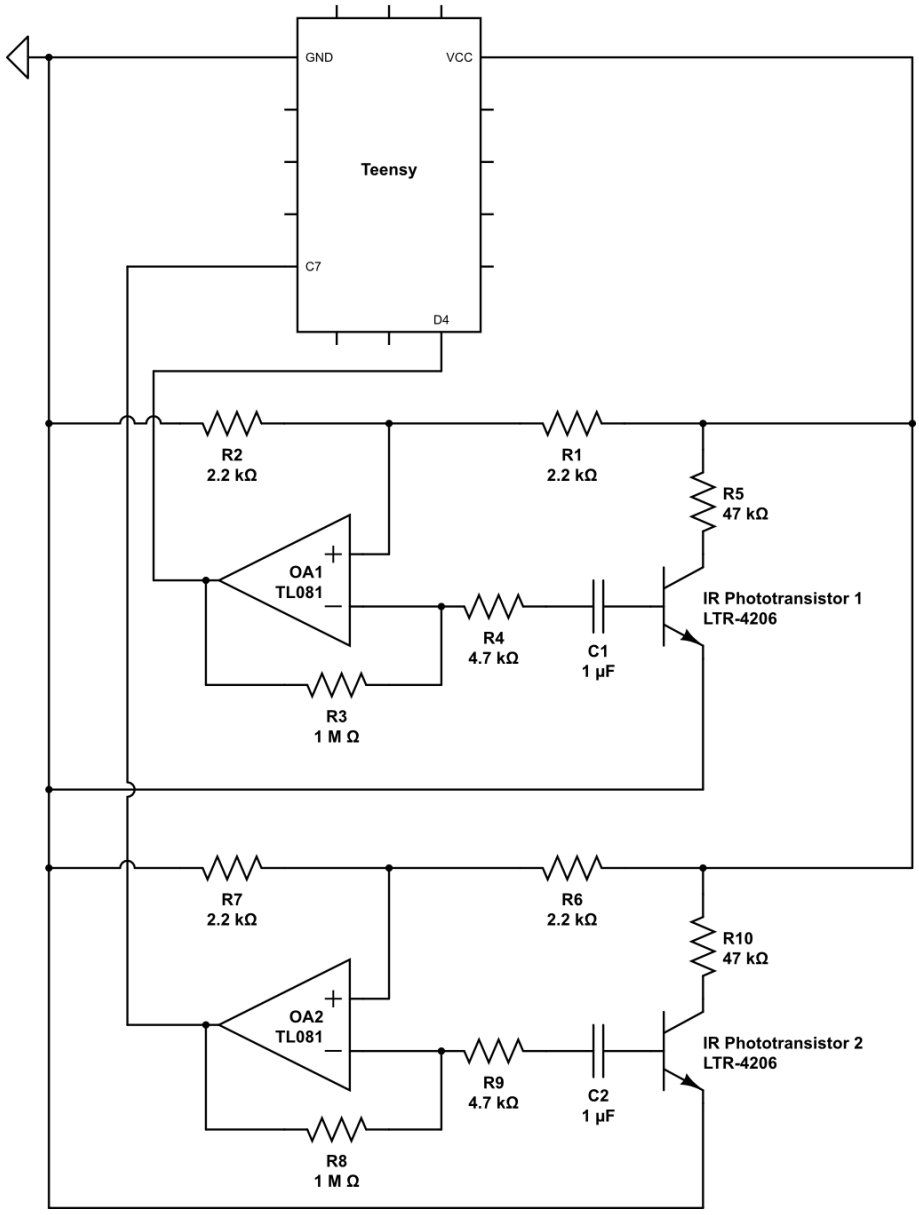
Item #	Description	Quantity	Reference #	Company	URL
1	Orange Wheel	2	3766	AdaFruit	3766
2	Perfboard w Rails	2	1609	AdaFruit	1609
3	TOF Sensor	3	3317	AdaFruit	3317
4	TT DC Motor	2	3777	Sourced from M.Yim	TT DC Motor
5	Servo	1	SG90	Sourced from M.Yim	SG90
6	H-Bridge	1	LV8401V	Sourced from M.Yim	LV8401V
7	ESP32 Pico Kit	1	ESP32-PicoKit	Sourced from M.Yim	ESP-32-PICO-KIT
8	Teensy2.0	1	Teensy2.0	Sourced from M.Yim	Teensy2.0
9	Vive Photodiode	2	PD70-01C	Sourced from Ministore	PD70-01C
10	Switch	1	RA11131121	Sourced from Ministore	RA11131121
11	Green Perfboard	2	N/A	Sourced from Ministore	N/A
12	IR Photodiode	2	LTR-4206E	Sourced from Ministore	LTR-4206E
13	OpAmp	3	TLV272	Sourced from Ministore	TLV272
14	Hex Inverter	1	CD74HC04E	Sourced from Ministore	CD74HC04E
15	Battery Pack	1	BH24AAW	Sourced from Ministore	BH24AAW
16	Power Bank	1	N/A	N/A	N/A
17	Acrylic Chassis	1	N/A	Sourced from RPL	N/A
18	Fasteners	N/A	N/A	Sourced from Ministore	N/A
19	Supplemental Electronics	N/A	N/A	Sourced from Ministore	N/A

Orange Wheel URL: https://www.adafruit.com/product/3766

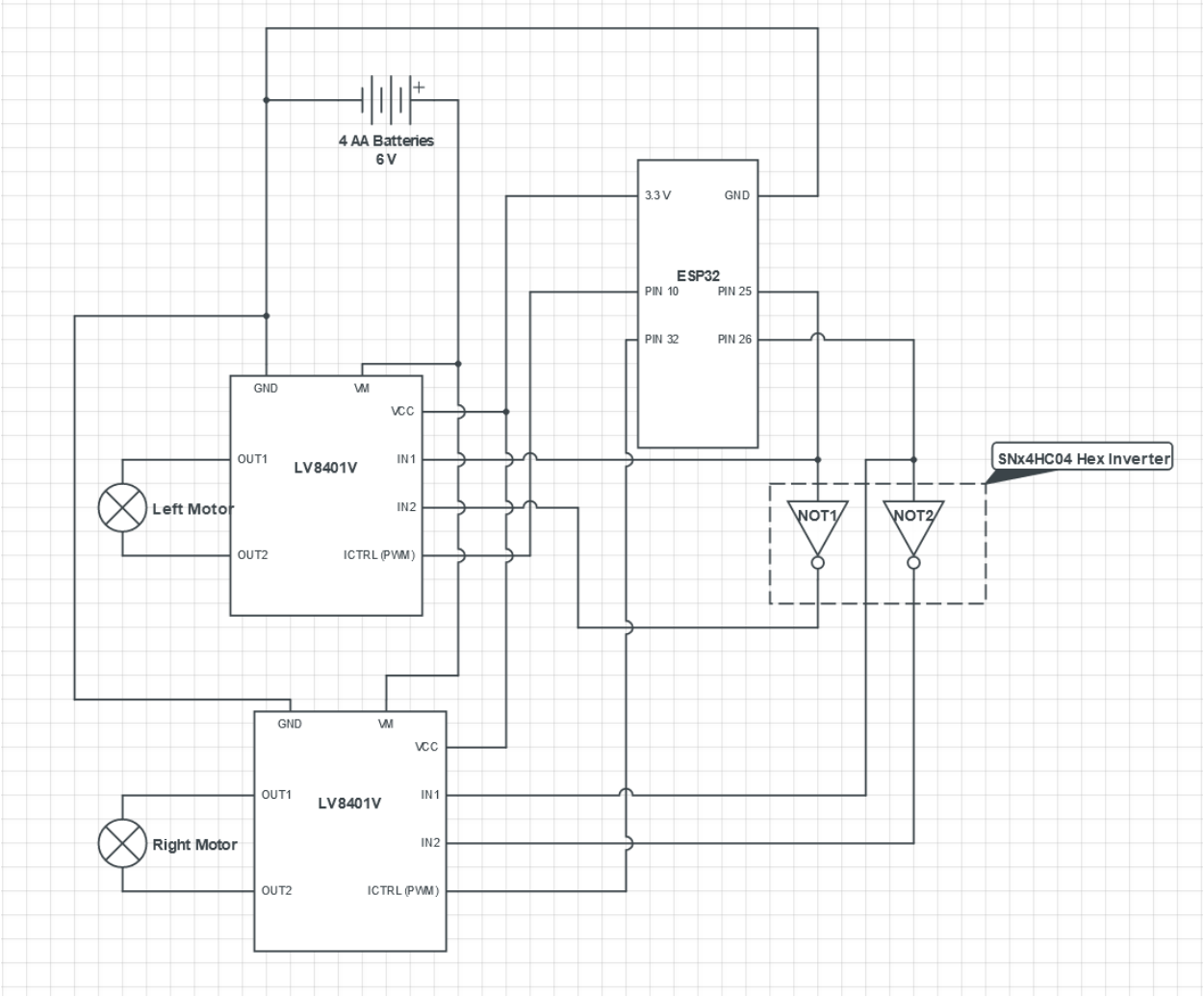
Perfboard w Rails URL: https://www.adafruit.com/product/1609

TOF Sensor URL: https://www.adafruit.com/product/3317

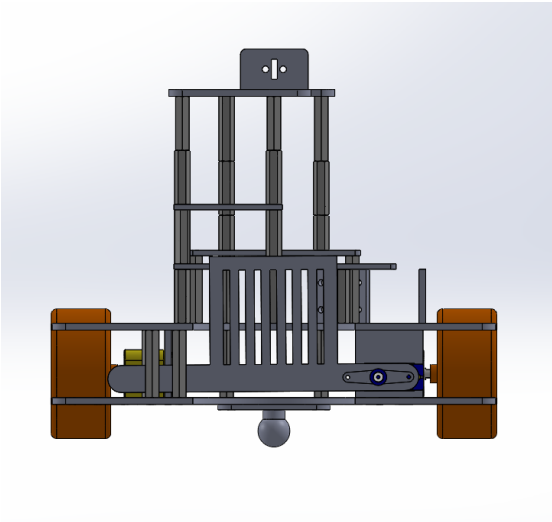
Beacon Electrical Circuit

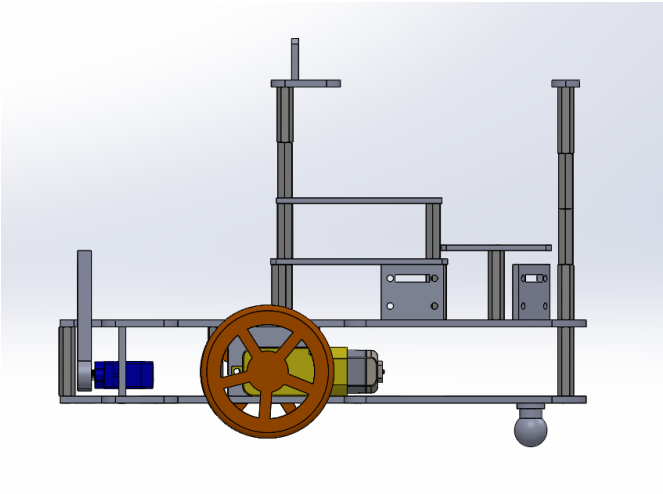


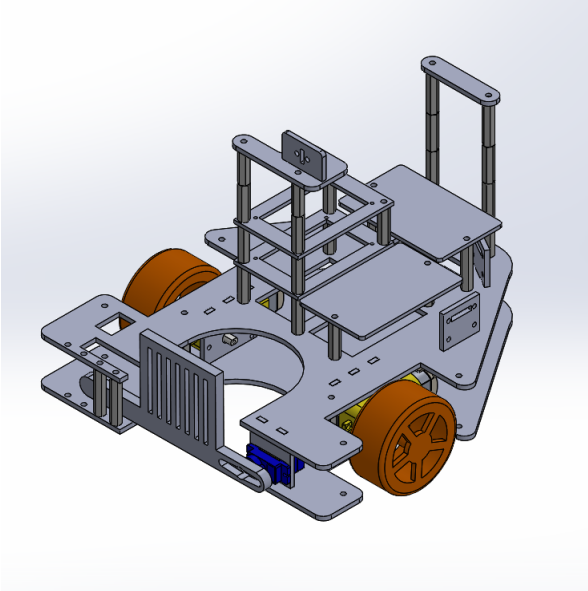
Motor Circuit



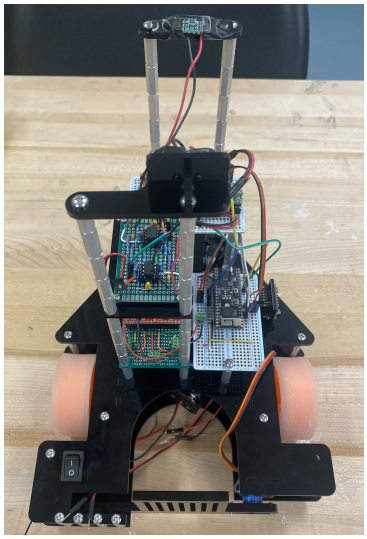
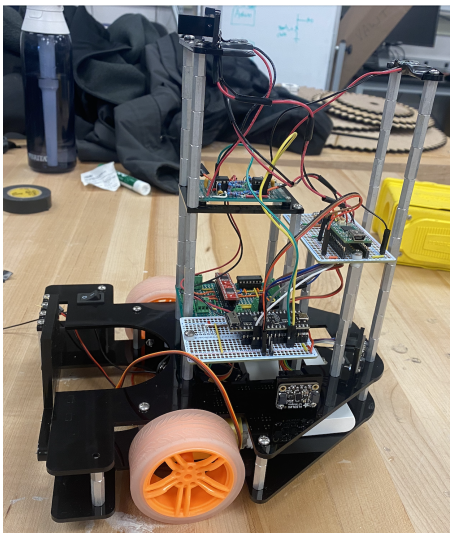
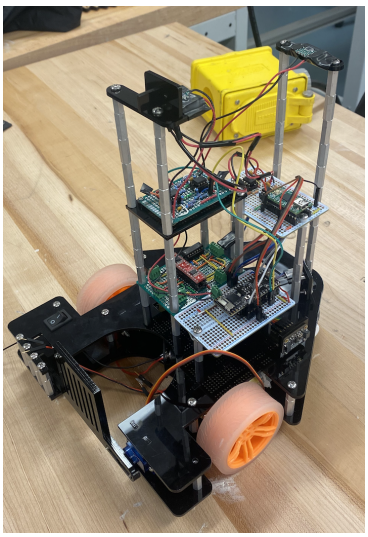
CAD Images







Final Design Images

Working Video Links

Competition Group Stage

Can/Beacon Moving During Competition

XY Moving to Can

Beacon Tracking

Wall Following