Ring Zero Final Project

Members:

Tyler Wartzok, Robert Swanson, Zachary Colbert, Sajan Gurung

Github Repository:

https://github.com/CSC615-2023-Spring/csc615-term-project-ttwartzok

Walk The Line

Description:

The objective of this project was to design and build a car that could follow a line and avoid obstacles using input from various sensors. The car will use the Raspberry Pi and its GPIO interface, and the code will be written in C.

Part List:

1 WaveShare Motor Driver HAT
2 HC-SR04 Ultrasonic Module
5 Line Sensor with TCRT5000 Reflective Optical Sensor
1 LS7366R Motor Encoder
2 DC Motors with Hall Sensor
1 Raspberrypi 4

Libraries and Software:

Pigpio Raspberry Pi GPIO library
LS7336R motor encoder driver library
WaveShare TB6612FND Motor Driver library

Pin Assignment:

Physical Pin	**GPIO Pin**	**Connected**
29	5	Line Sensor (front left)
31	6	Line Sensor (front center)
33	13	Line Sensor (front right)
16	23	Line Sensor (outer left)
18	24	Line Sensor (outer right)
32	12	Sonar Sensor - Front - Echo
36	16	Sonar Sensor - Front - Trigger
38	20	Sonar Sensor - Left - Echo
40	21	Sonar Sensor - Left - Trigger
21	9	Motor Encoder Board (MISO)
19	10	Motor Encoder Board (MOSI)

Approach / What We Did:

1. Hardware Selection: We researched and selected the necessary components for the project, we originally decided upon using three line sensors, hall effect motors with encoders, and a sonar sensor paired with 5 obstacle sensors split between the front and sides for obstacle detection. The initial design for line sensors involved three line sensors arranged on the front of the car. This layout was eventually revised in favor of a placement closer to the car's pivot point, as well as the addition of two additional sensors on the outer edges of the car to help disambiguate right-angled turns. We also decided to try using the lidar instead of the combination of sonar and obstacle sensors. After a long period of trial and error attempting to get the lidar operational, simplified our approach to obstacle avoidance and adopted a new strategy that could be accomplished with two sonar sensors on the front and the left-side of the car.

2. Circuit Design and Assembly: Since the selected motors and their quadrature encoder used the SPI protocol and had to be connected to specific GPIO pins, these were the first components that we connected to our system. From this point, the remaining components were connected to available GPIO pins and the LIDAR sensor was connected via USB. We used the power bus of a breadboard to provide a common connection point for all of the connected components.

3. Line Sensor Integration: The first system we approached was the line following algorithm. After mounting the line sensors, we iteratively developed a set of rules for following the lines. These rules were based on many small adjustments in reaction to the sensor inputs. If the car is following the line, then only the center sensor should be active. If either of the side sensors are active, then the car has drifted towards that side and directional corrections must be made until it is centered again.

4. Motor Control and Encoders: The initial plan was to use the sensors with their hall effect sensors to retrieve information about the motor's rotations so we could accurately derive our speed and distance. We spent quite a bit of time working with the hardware but were ultimately unsuccessful in our attempts to reliably obtain accurate readings. We still used these motors, but the adjustments ended up being based on time instead of rotational position.

5. Obstacle Detection and Avoidance: We originally decided upon using a combination of sonar sensors and obstacle sensors. The idea was the sonar sensor would be at the front of the car for early detection to know that we need to slow down and stop while approaching an obstacle. We would then start looking at the obstacle sensors and once in range turn the car to maneuver around the obstacle and then use more obstacle sensors along the sides of the car to know once we have passed. We decided to take a gamble and try out the lidar instead of this thinking the extra data the lidar could give us would make this easier than the original design. We spent

a significant amount of effort attempting to get the LIDAR working, but we ran into a number of problems with our implementation and ultimately opted to simplify our approach for the sake of time. We ended up using 2 sonar sensors instead of the lidar (more information on this in issues and resolutions) having one sonar sensor in the front to detect the obstacle as we approach, and the other sensor on the side so after turning we know when we have passed the obstacle. Looking back on our original idea for obstacle avoidance, there is no way that would have worked due to the range of the obstacle sensors being less than an inch. That would leave no room for error while stopping and executing our turn.

Issues and Resolutions:

1. Motor layout and mounting: The motors we selected were too large to be mounted side-by-side. Our initial solution was to zip tie the sensors to the back of the chassis plates. Since the rear edge of the plates was curved, the zip ties would eventually displace which caused our motors and wheels to fall out of alignment and come loose. The mounting brackets that were attached to the motors didn't fit in any of the convenient mounting locations, so we designed and 3D printed several different versions of mounting brackets until we ultimately settled on a one-piece bracket that mounted securely to the rear of the frame. The single piece design not only held the motors in alignment, but also provided some additional rigidity to prevent the car's frame from flexing under load.

2. Line sensor Calibration: The line sensors required calibration to accurately detect the line markings. Calibration was sometimes difficult because our testing track was taped onto a surface that was only slightly darker than the tape we were trying to follow. Sometimes the sensors would also move slightly out of position which altered the calibration. We designed and 3D printed a sensor mounting bracket which held the sensors steadily in position. The bracket was also height-adjustable, using a captive nut and some springs so we could advance or retract two mounting screws to raise or lower the entire array of sensors. By lowering the sensors so they were closer to the ground, the sensitivity adjustments were more effective and we saw fewer bad readings from the sensors.

3. Line Following: Our original layout was using 3 line sensors spaced about an inch apart in a row on the front of the car. The idea was the middle line sensor would be on the line at all times and if it veered to the right or left of the line we would reposition so it was back on the line. This solution seemed promising, but since we chose a three-wheeled design that pivoted about a point near the rear of the car, turns would cause the sensors to move drastically to a point where they would often miss the line entirely. This was mitigated by moving the sensors to the center of the car, closer to the pivot point. After relocating the sensors, we were able to accurately perform turns of a much smaller radius.

We were able to make most turns successfully, but we often ran into issues with 90 degree turns. When we approached from a less than perfect angle, all three sensors would activate and

it was impossible to disambiguate which way we should turn. We tried a number of software-based solutions before ultimately deciding that it would be simpler to solve the problem with additional hardware. We solved this by adding two more sensors at the far left and right of the car. If we reached an ambiguous situation where all three primary sensors were on, we looked to the two outer sensors to break the tie.

4. False Line Sensor Readings: We often ran into problems related to unreliable readings from the line sensors due to poor calibration, or due to false readings from the imperfect surface we were testing on. If a sensor picked up a dark spot on the ground, it would send our car off in a completely unexpected direction. We realized this was because we were reacting without hesitation to every change in the sensor state, and sometimes those sensor states were inaccurate. We solved this by implementing a confidence level for the sensor readings. Instead of blindly obeying every sensor reading, we required that the sensor must be reporting the same reading several times in a row before we trust that reading and act on the information. This almost entirely eliminated the erratic behavior in our line following algorithm, and only required some fine tuning of the confidence parameters.

5. Obstacle Detection Accuracy: We ran into multiple issues with the sonar sensors. The main issue was that they would sometimes get stuck and keep outputting the same reading forever. If the sensors were polled too frequently, or if they didn't receive a signal back, the echo pin would get pulled down and would never get pulled back up. We solved this by adding a timeout that would break the (previously infinite) loop. If the sensor waits for longer than the time of its maximum valid range, the reading is marked as invalid. We also limited our polling frequency to about 20hz, which reduced how often this error would occur.

Another issue we encountered was reliability of the sensor readings. The readings from these sensors fluctuated quite a bit, and would sometimes be completely invalid. We solved this with a two-stage approach. The first stage was to establish a confidence level for each reading. When reading the distance, a delta is calculated between the current and previous readings. If the delta is sufficiently small, the confidence is incremented and the reading is captured. If the delta is too large, the confidence is reduced by the delta times a weight factor. In the case of invalid readings, the confidence is reduced by the number of consecutive invalid readings. This confidence level helps to establish whether these readings are likely to be valid or not. The second stage is to decide whether or not the (already validated) readings from the sensors are being interpreted correctly. When navigating around an obstacle, we captured several readings to determine whether or not an obstacle was present. These two techniques combined helped to improve the reliability of our obstacle avoidance procedures and reduce errors due to inaccurate sensor data.

6. Lidar: We originally used a Lidar for obstacle avoidance but ended up substituting it for sonar sensors after running into many issues. Originally we were constantly checking all scans from the lidar but realized this was an error with how we handled obstacle avoidance. We would

check if there was an object within a certain viewing range in front of the car. The problem with our logic was that when we checked if we did not see anything within the range at the exact moment we checked, we would assume there was no obstacle present. That did not work because at the moment we checked, the lidar could be sending the results of a scan from a different angle so there still could be an object in front of us. After realizing this was an issue we decided to change to have the lidar only tell us the location of the closest object. That way we could store the location of this object and use that information if we were approaching it and had to maneuver around it. We then noticed the flaws with this plan. If we store the closest object how do we know when to replace it with something closer? We would compare the distance of the closest object with all new scans and if a new scan was closer we would update the closest object to be that scan. Then came the new issue of knowing when the closest scan was no longer valid. If we do another full scan in all directions after we have moved further from the closest object we have no way of knowing whether or not the stored closest scan was still the closest object. We chose to add an age to the closest scan and after a certain amount of time has passed decide that the scan was no longer valid but choosing the arbitrary amount of time in which the scan would expired gave us issues and with a deadline approaching we decided to pivot into use the sonar sensors because we knew we would be able to interpret the data from them with more accuracy with the amount of time we had to finish the project. After writing this I am now realizing many ways to overcome the issues we faced and only wish we had more time to complete the project with the sensor. Being in a time crunch led us to constantly try to put bandaids on the issues to try to overcome them, which only led to more issues, instead of removing the flawed logic completely and rebuilding from the ground up.

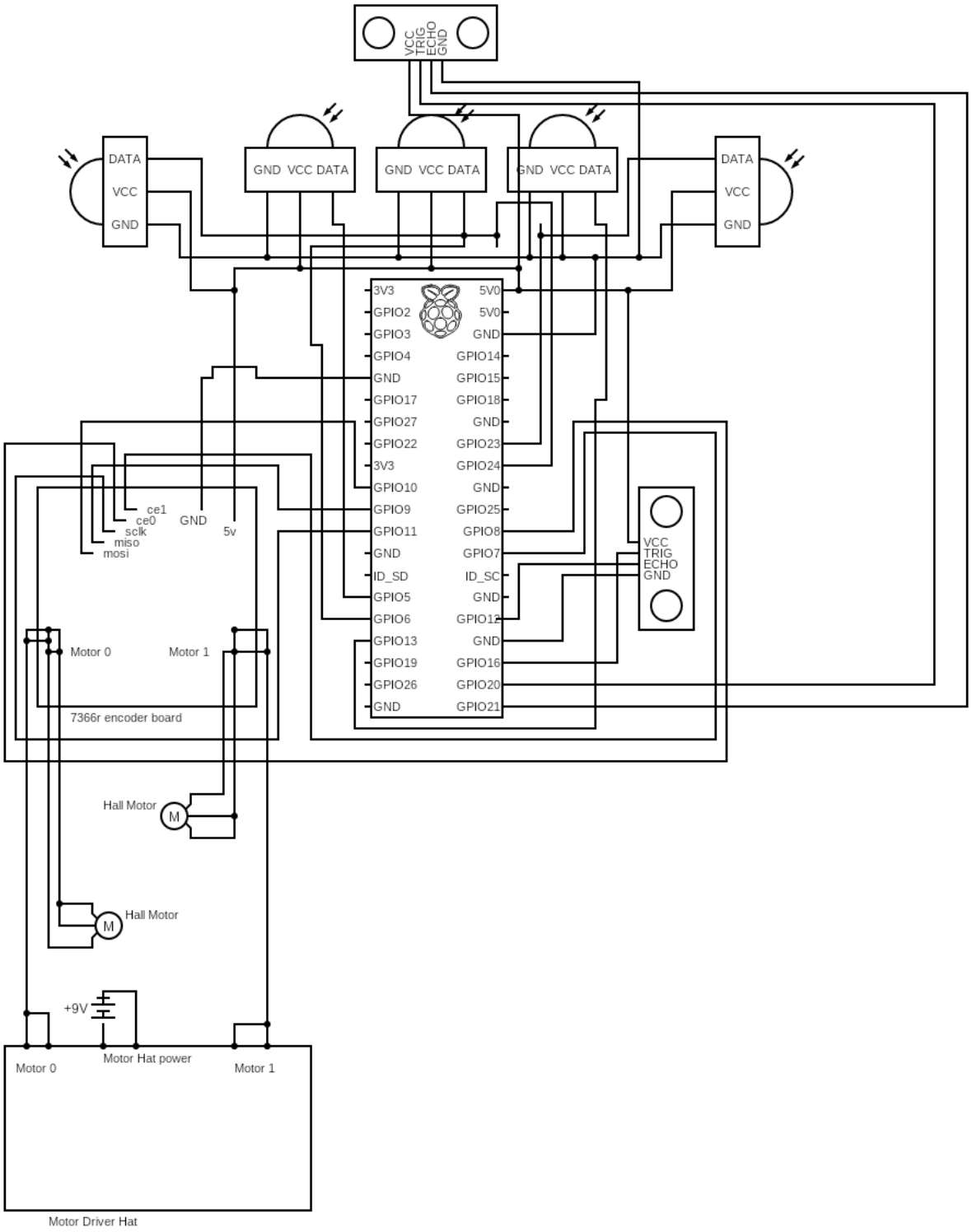
7. Ball bearing wheel: Our initial design used two motors in the rear, and a single ball-bearing wheel mounted near the center of the chassis. The ball-bearing would often get caught on small imperfections of the track surface including overlapping pieces of tape or bits of dirt on the floor. We swapped this for a caster wheel instead, which solved the problem of getting caught on surface imperfections but caused our line tracking to behave abnormally. We eventually noticed that the new wheel was causing interference with our line sensors. During a turn, the caster wheel would pivot to a position that was right next to the sensor and caused false readings. We solved this by moving the wheel closer to the front of the frame, further from the line sensors.

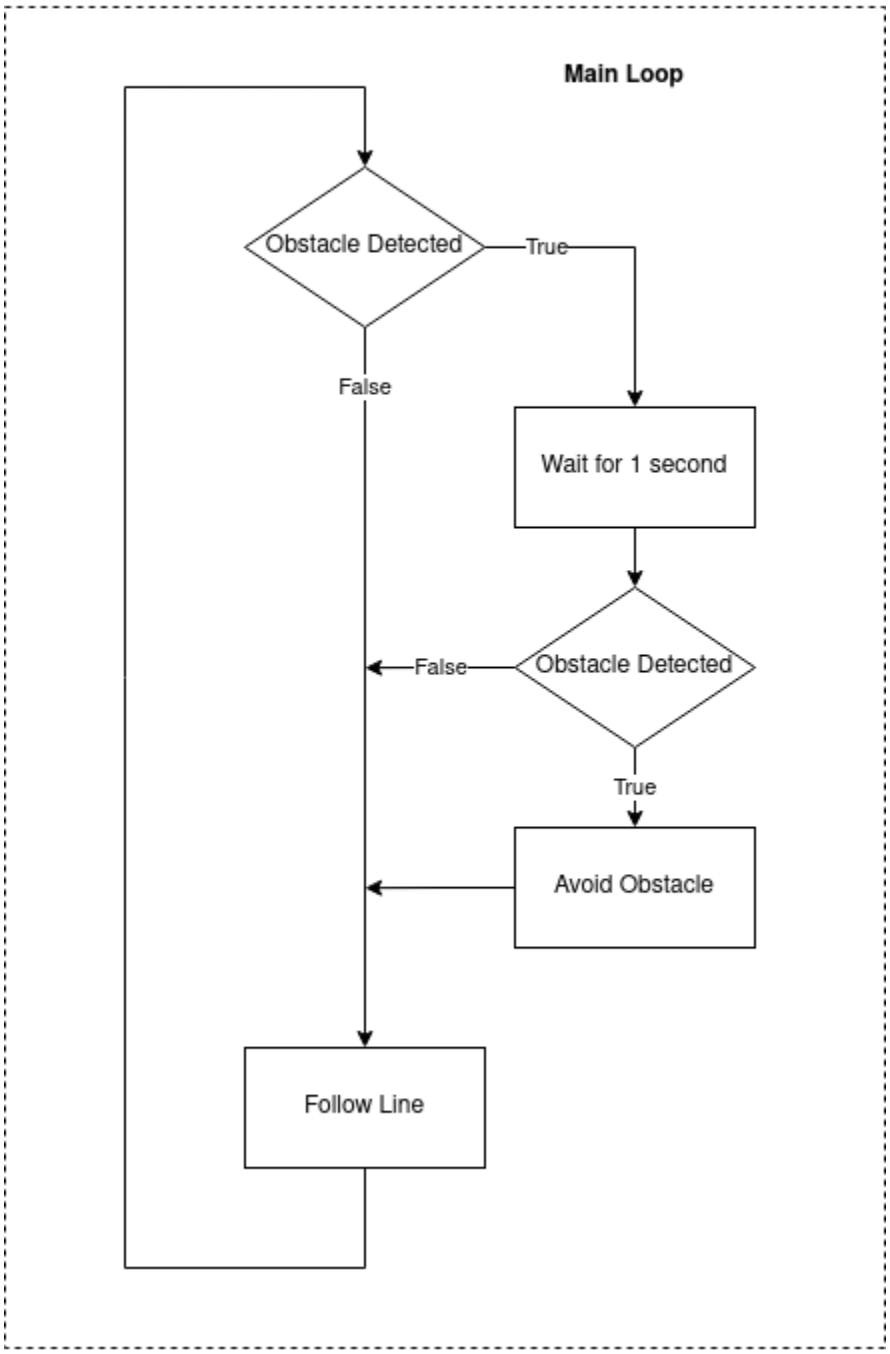
Analysis:

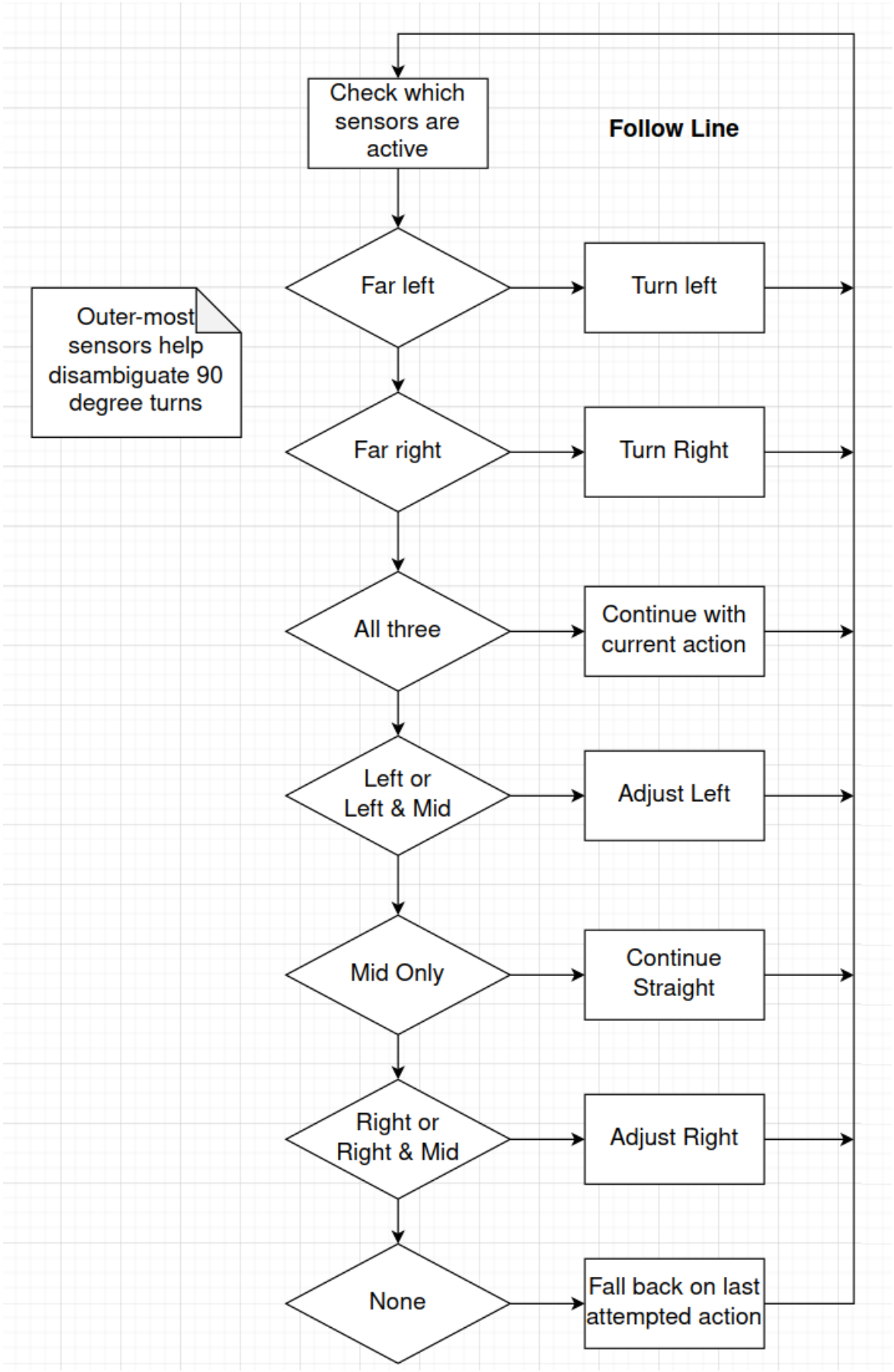
The goal of this project was to implement a car that could follow a line and navigate around an obstacle. Our initial assumptions were that we should have access to as much data as possible so that we would not be limited when implementing the logic of our system. What we realized as we developed the physical and logical systems of our project is that more complex components are not always desirable. While we did gain some additional useful information from the LIDAR and the motor counter boards, we also spent quite a bit of time in getting those components up and running when compared to the time it took to implement their more primitive counterparts. In hindsight, our initial design was built on a lot of poor assumptions and ultimately it was a simplification of our design that brought us success. For example, we initially planned to navigate an obstacle by following its perimeter, and assumed that the sonar sensors would not

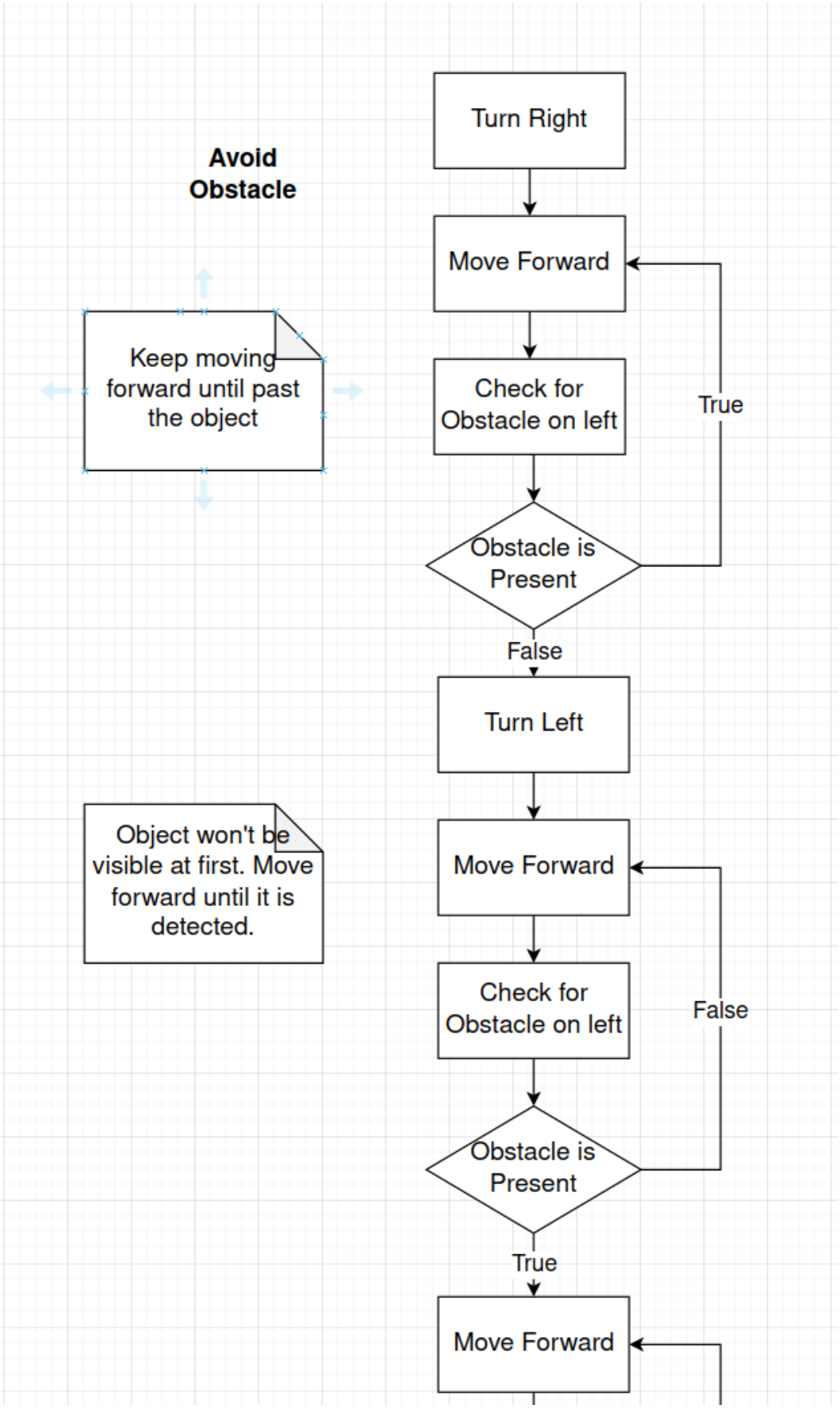
provide enough information to track the object, What we eventually decided was to just draw a bounding rectangle around the obstacle, which really only required knowledge of what was in front and to the left of us. If we had started with the simplest approach first, we would have had more time to fine tune these systems instead of trying to chase an optimal solution. We were also reminded repeatedly of the imperfections of the physical world. When developing software we are used to working in a tightly controlled environment where we can expect near-perfect reproducibility. Instead, we encountered many situations where external variables such as environmental noise, sensor positioning, battery power level, or any number of other factors would alter the behavior of our system. These factors combined have changed the way that we approach the design and development of systems and the assumptions we make about them.

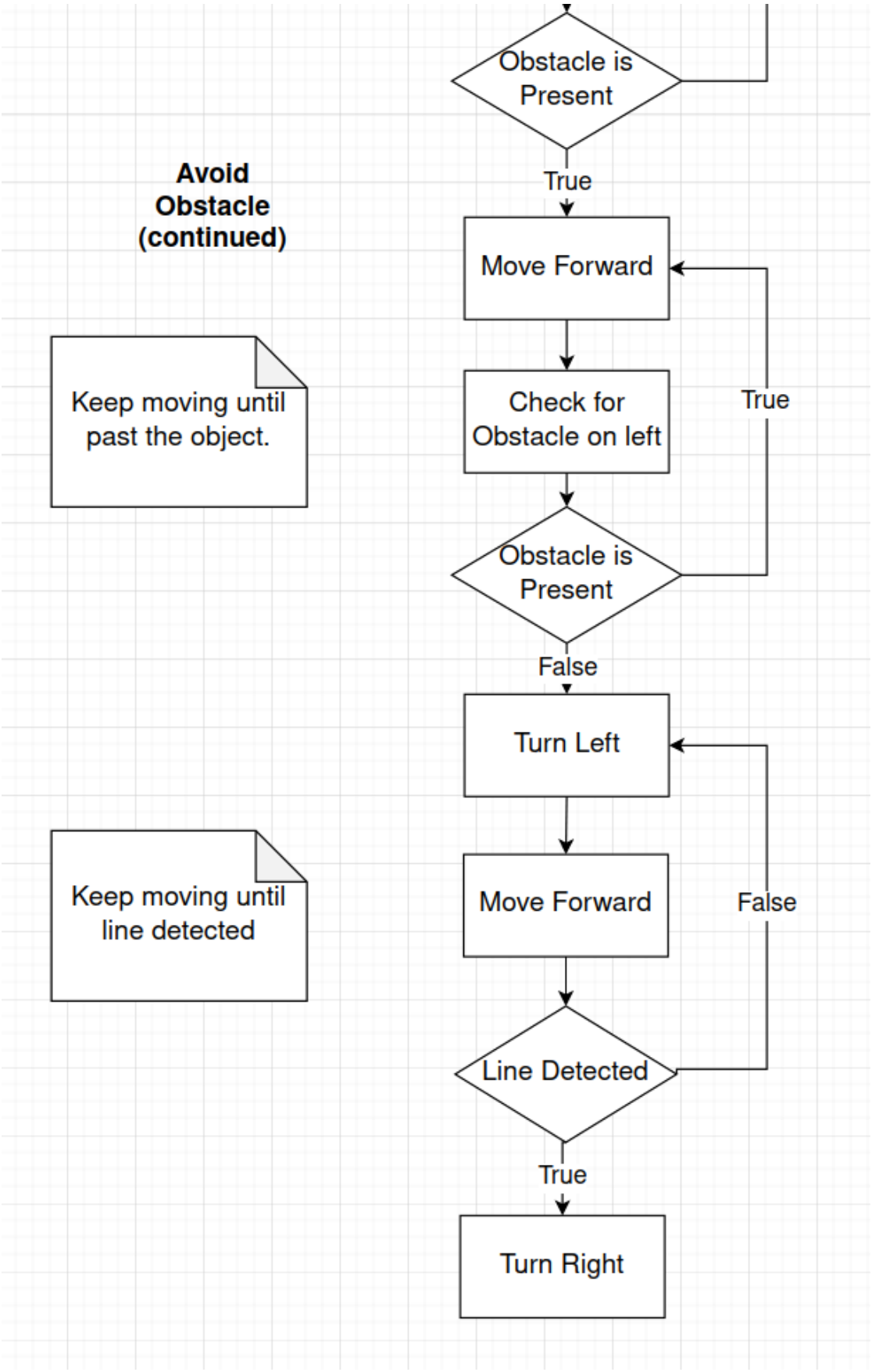
Wiring Diagram:



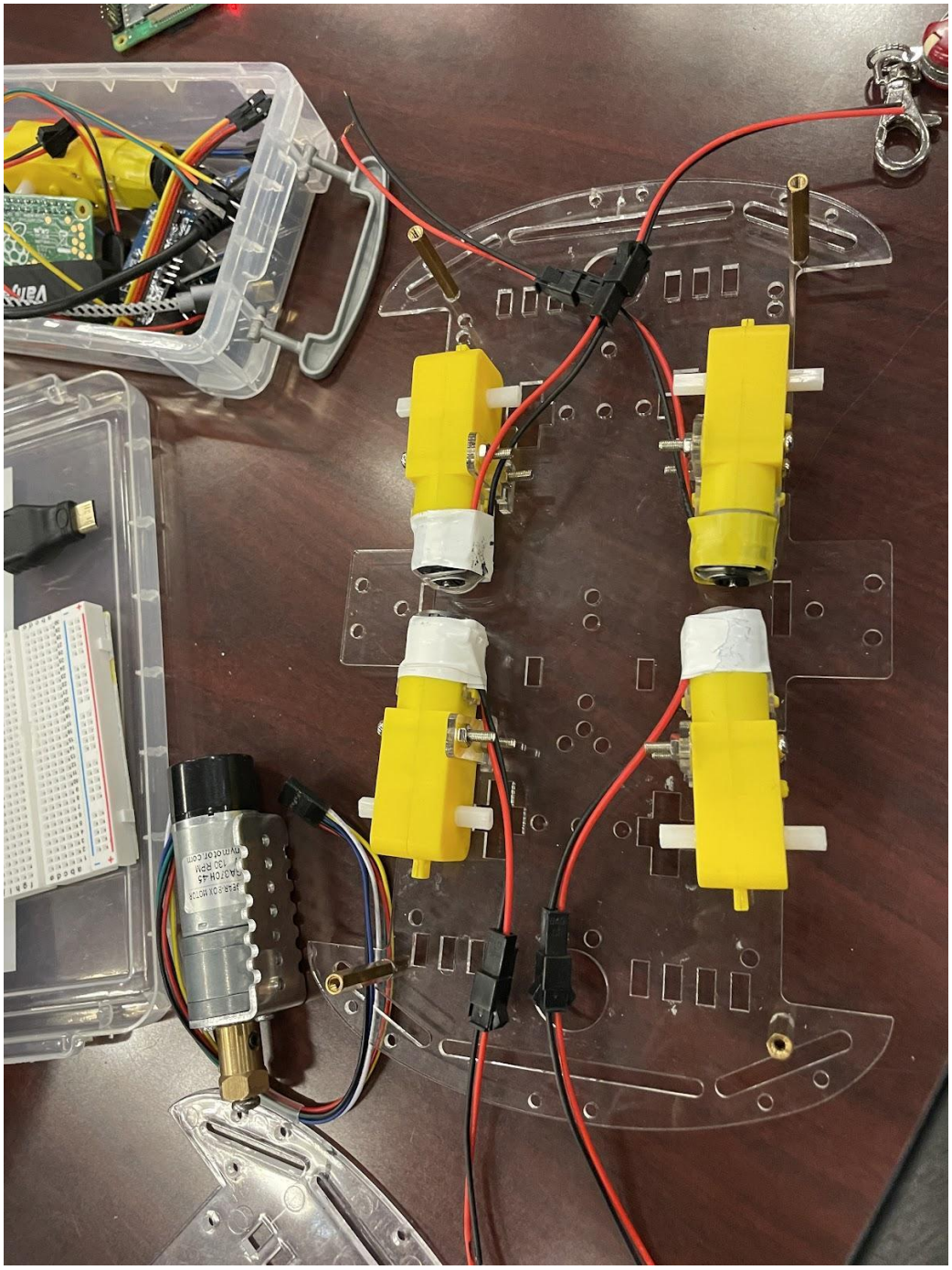

Main Loop
Obstacle Detected
True
False
Wait for 1 second
Obstacle Detected
False
True
Avoid Obstacle
Follow Line

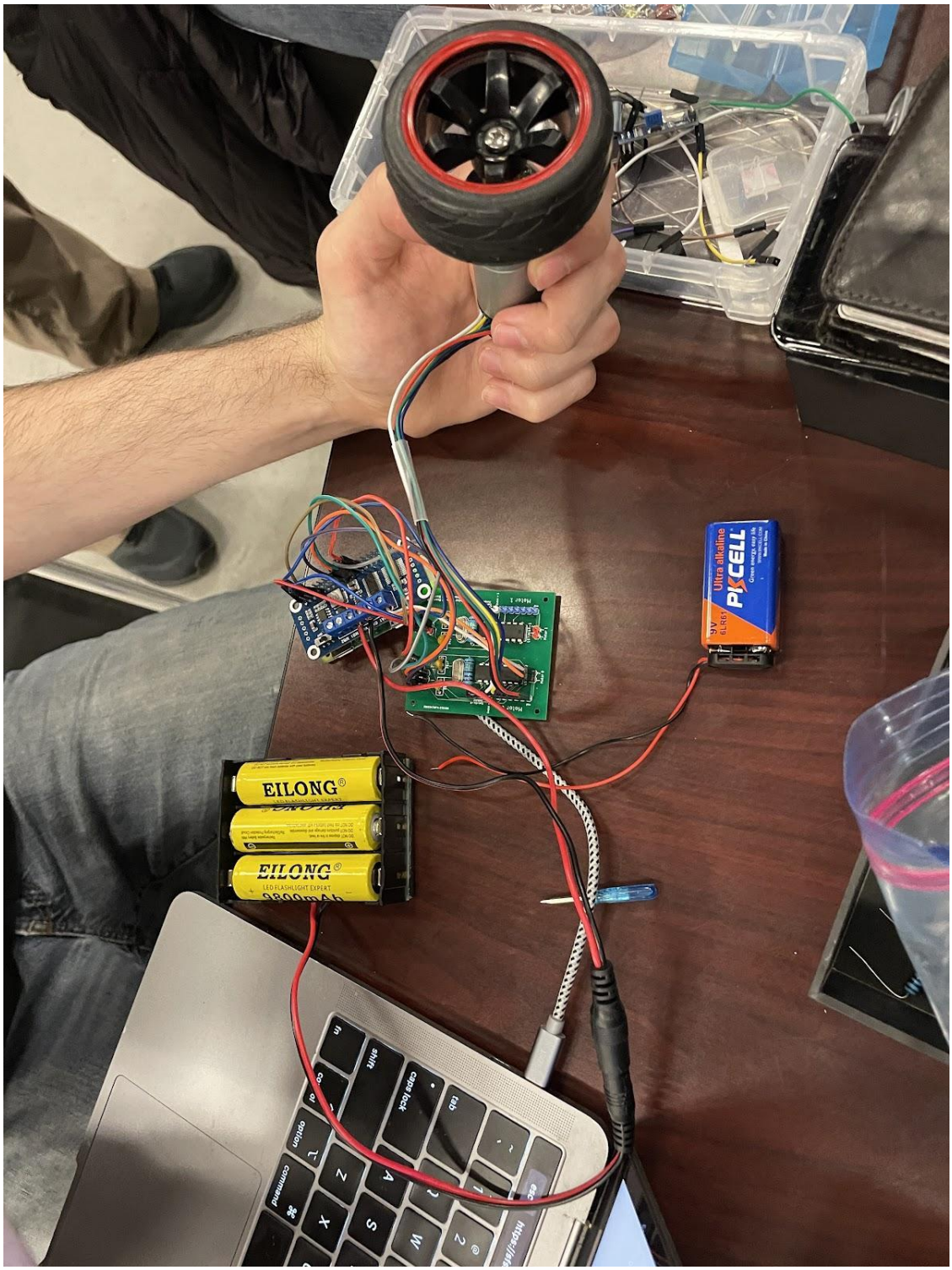

Follow Line
Check which sensors are active
Far left
Turn left
Outer-most sensors help disambiguate 90 degree turns
Far right
Turn Right
All three
Continue with current action
Left or Left & Mid
Adjust Left
Mid Only
Continue Straight
Right or Right & Mid
Adjust Right
None
Fall back on last attempted action

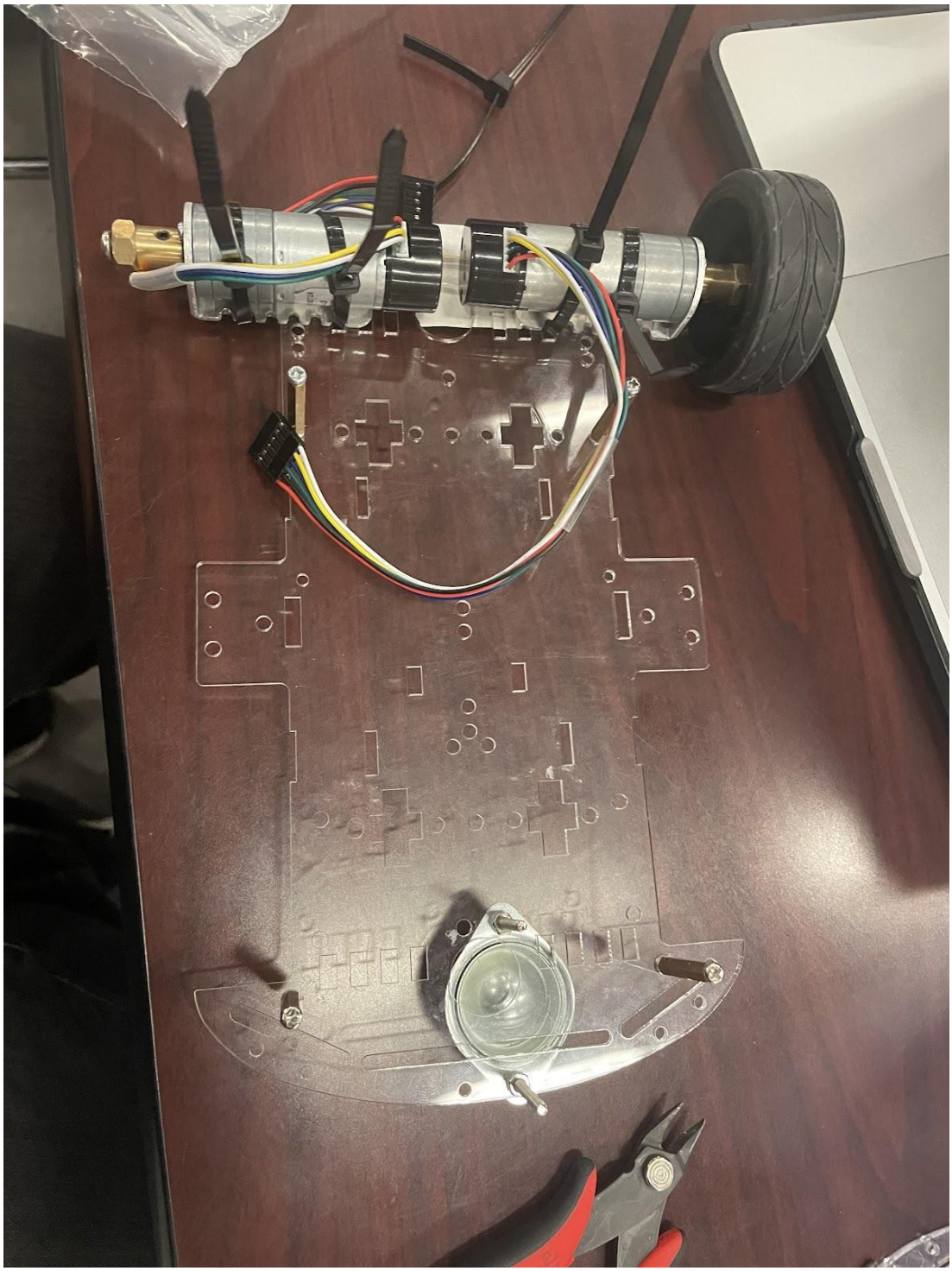

Avoid Obstacle
Turn Right
Move Forward
Check for Obstacle on left
True
Obstacle is Present
False
Turn Left
Move Forward
Check for Obstacle on left
False
Obstacle is Present
True
Move Forward
Keep moving forward until past the object
Object won't be visible at first. Move forward until it is detected.

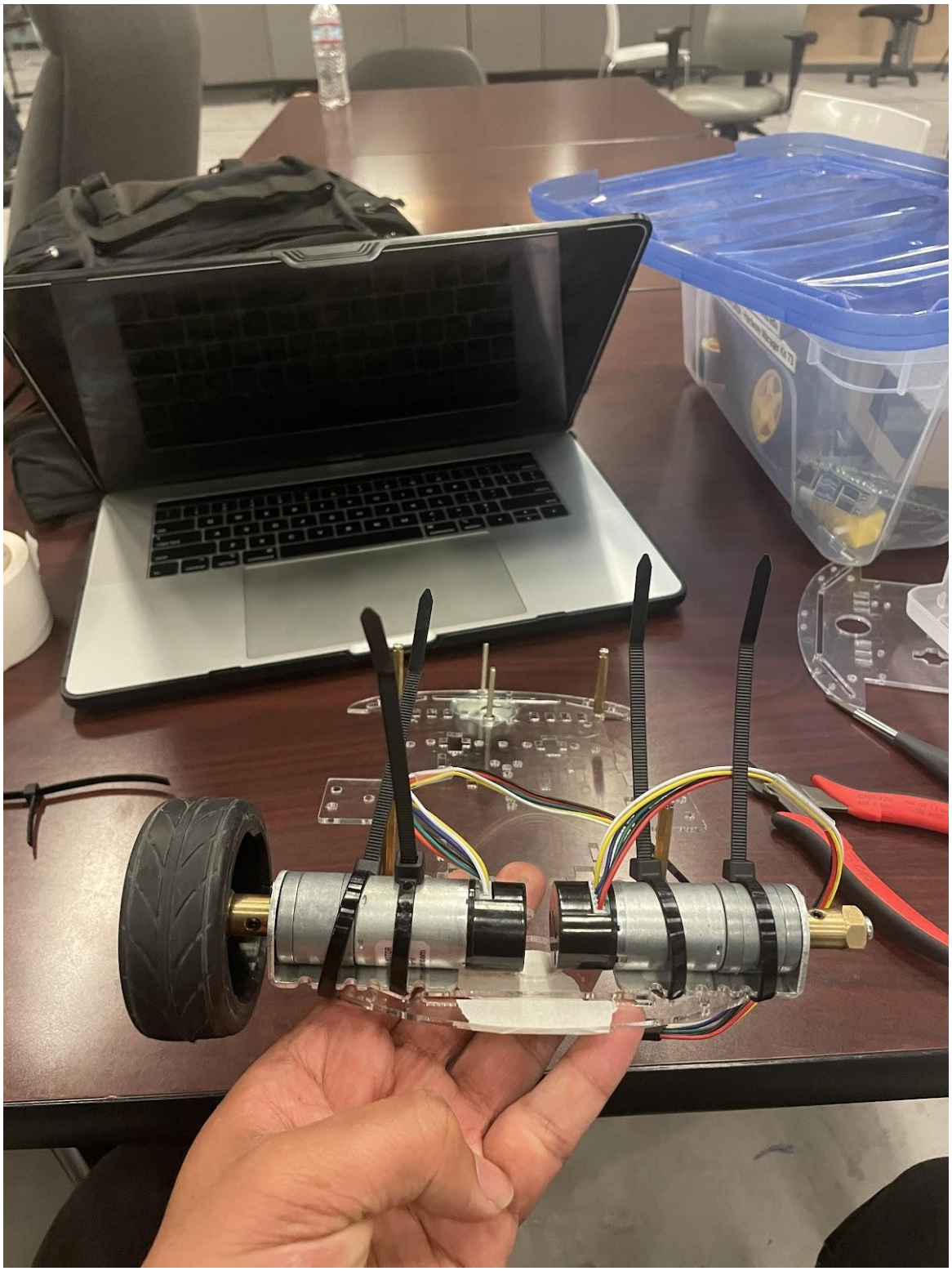

Avoid Obstacle (continued)
Keep moving until past the object.
Keep moving until line detected
Obstacle is Present
True
Move Forward
Check for Obstacle on left
True
Obstacle is Present
False
Turn Left
Move Forward
False
Line Detected
True
Turn Right

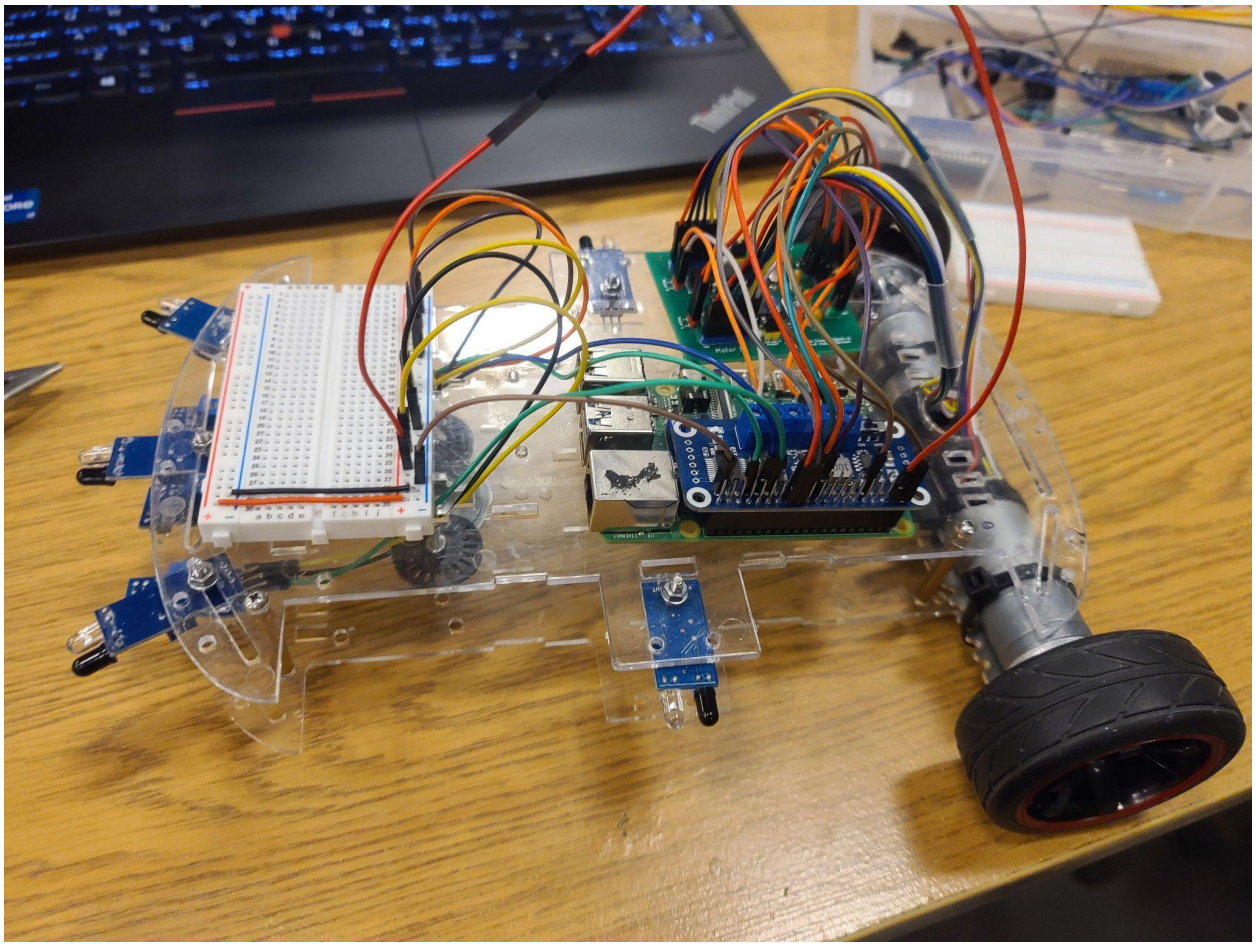
Photos of the progress of the project

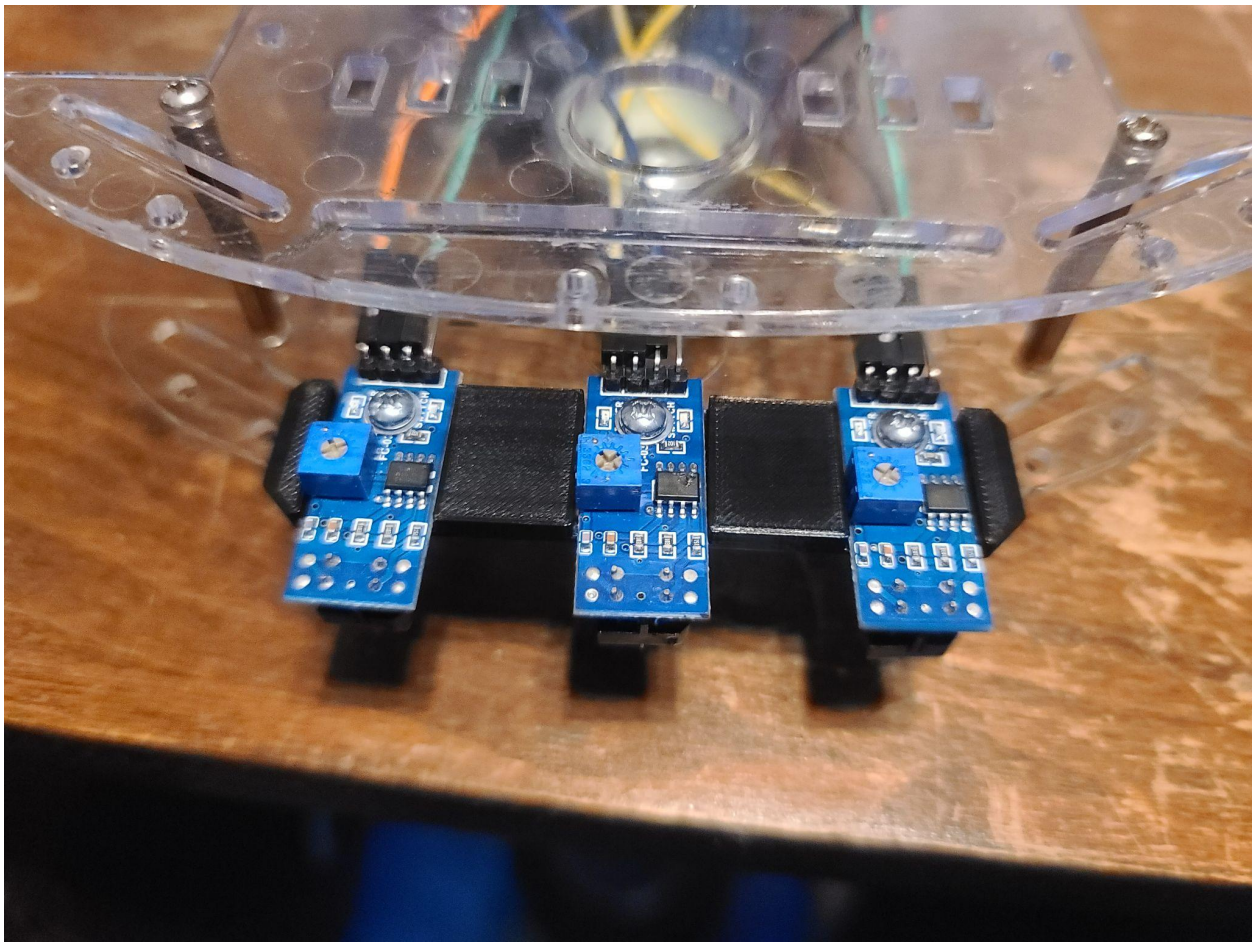


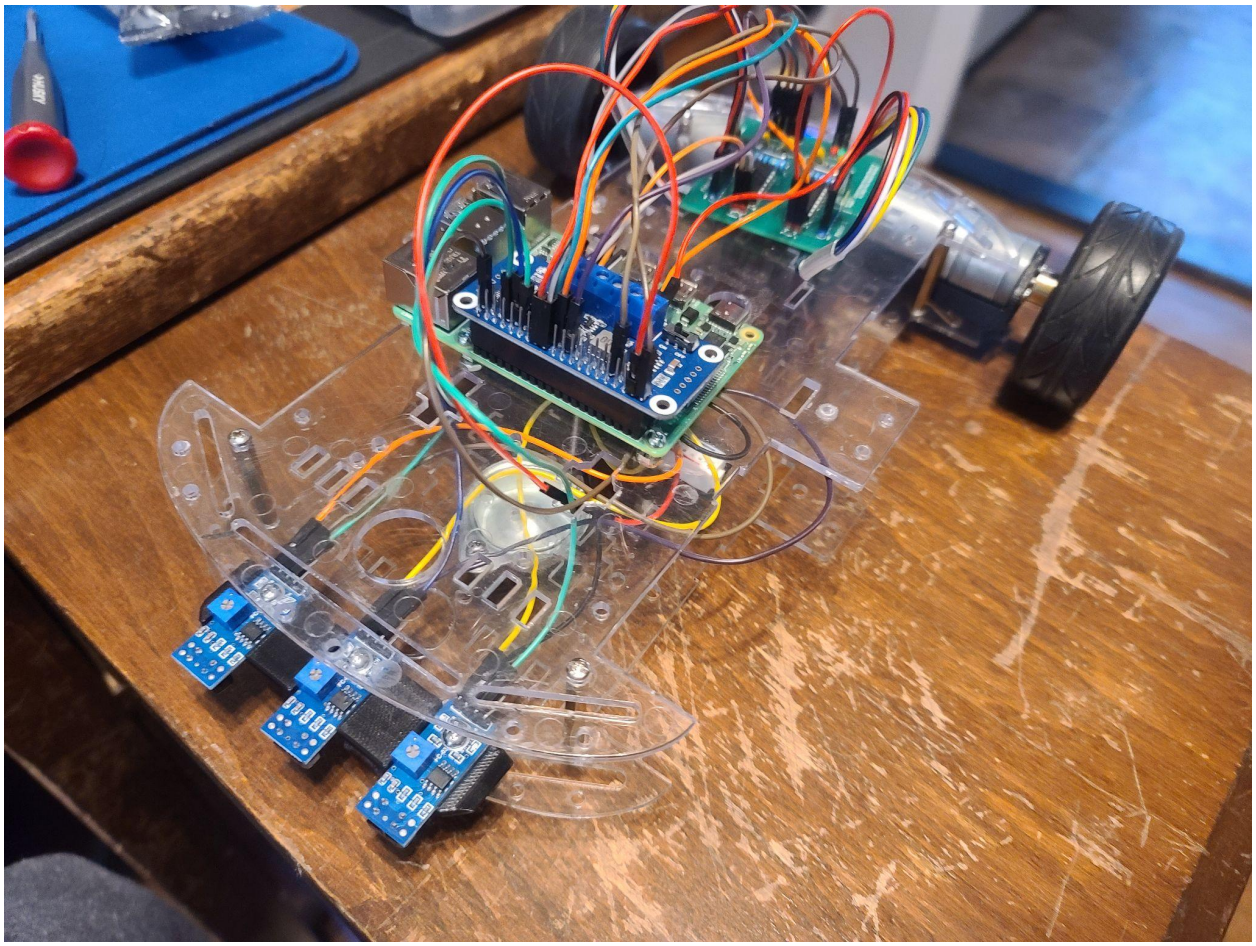

EILONG
EILONG
Ultra alkaline
PKCELL

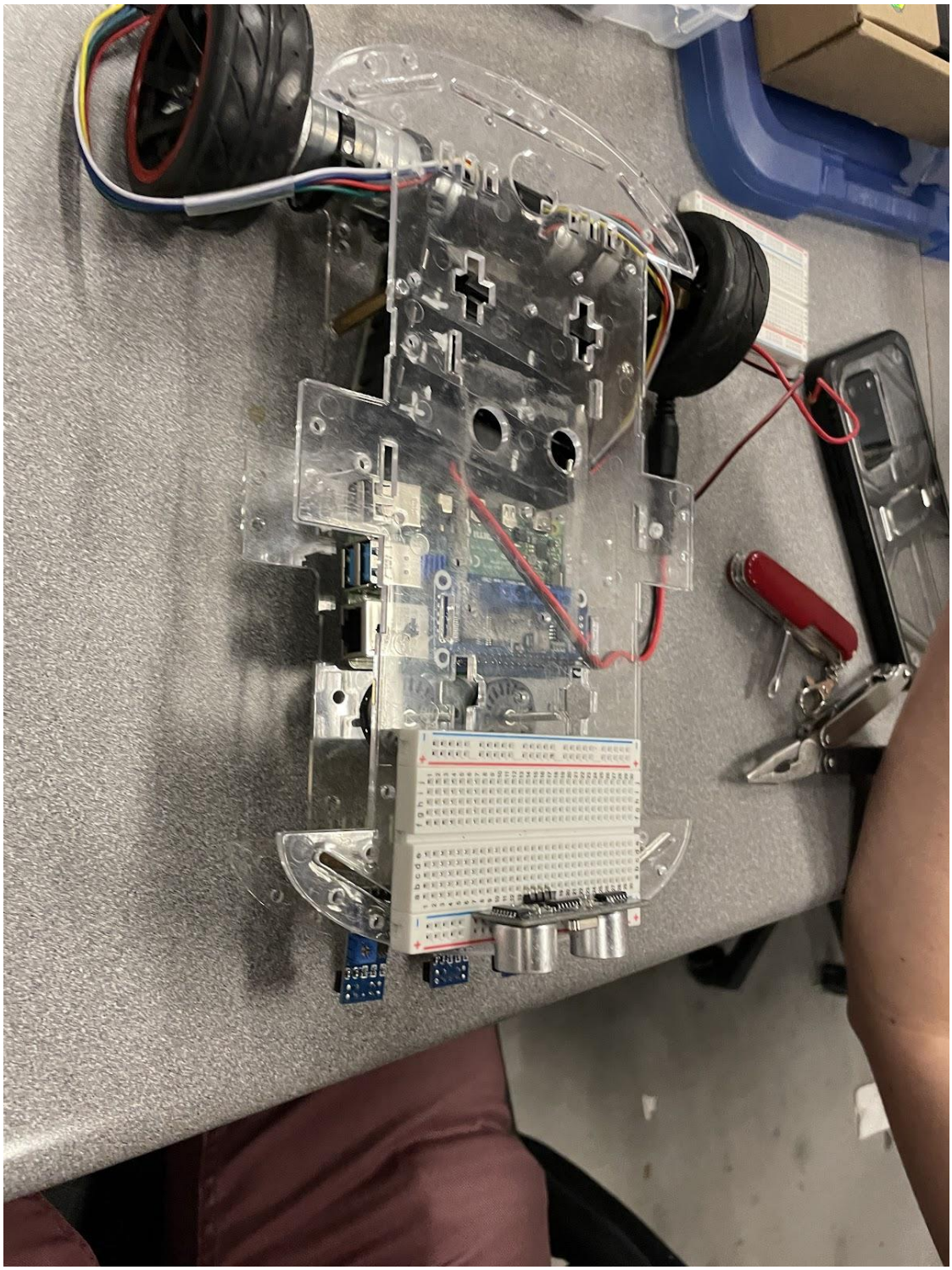


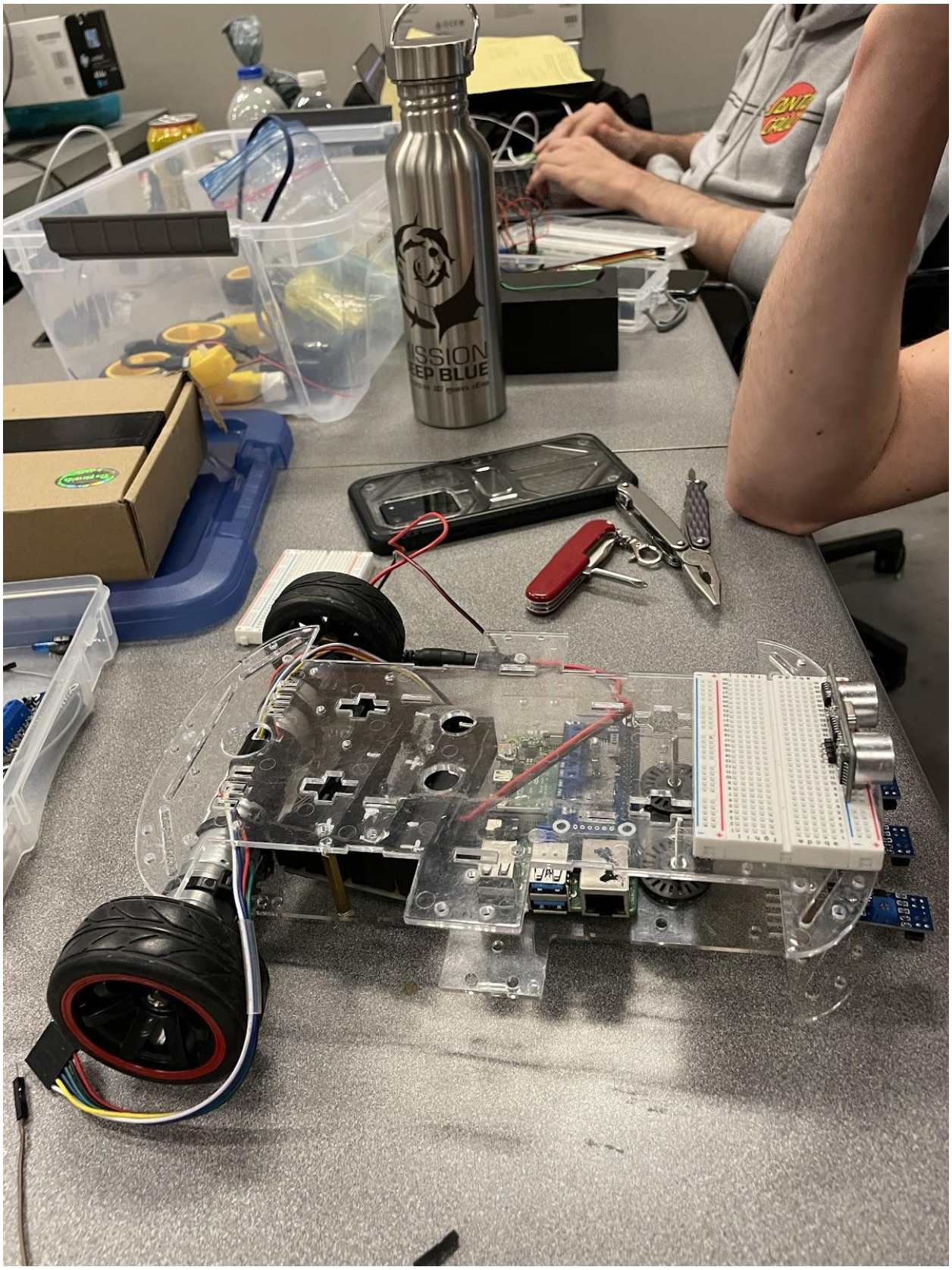


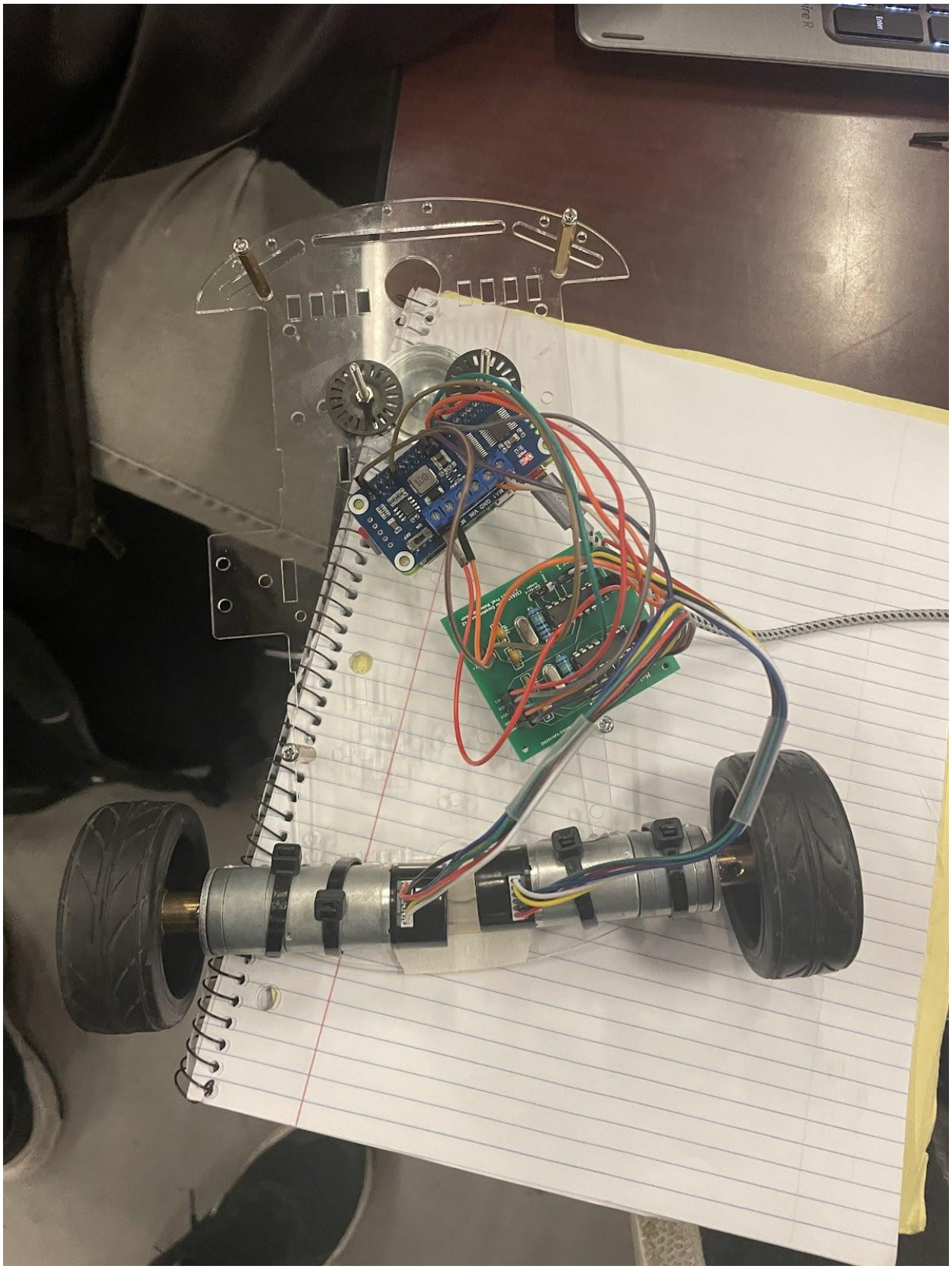


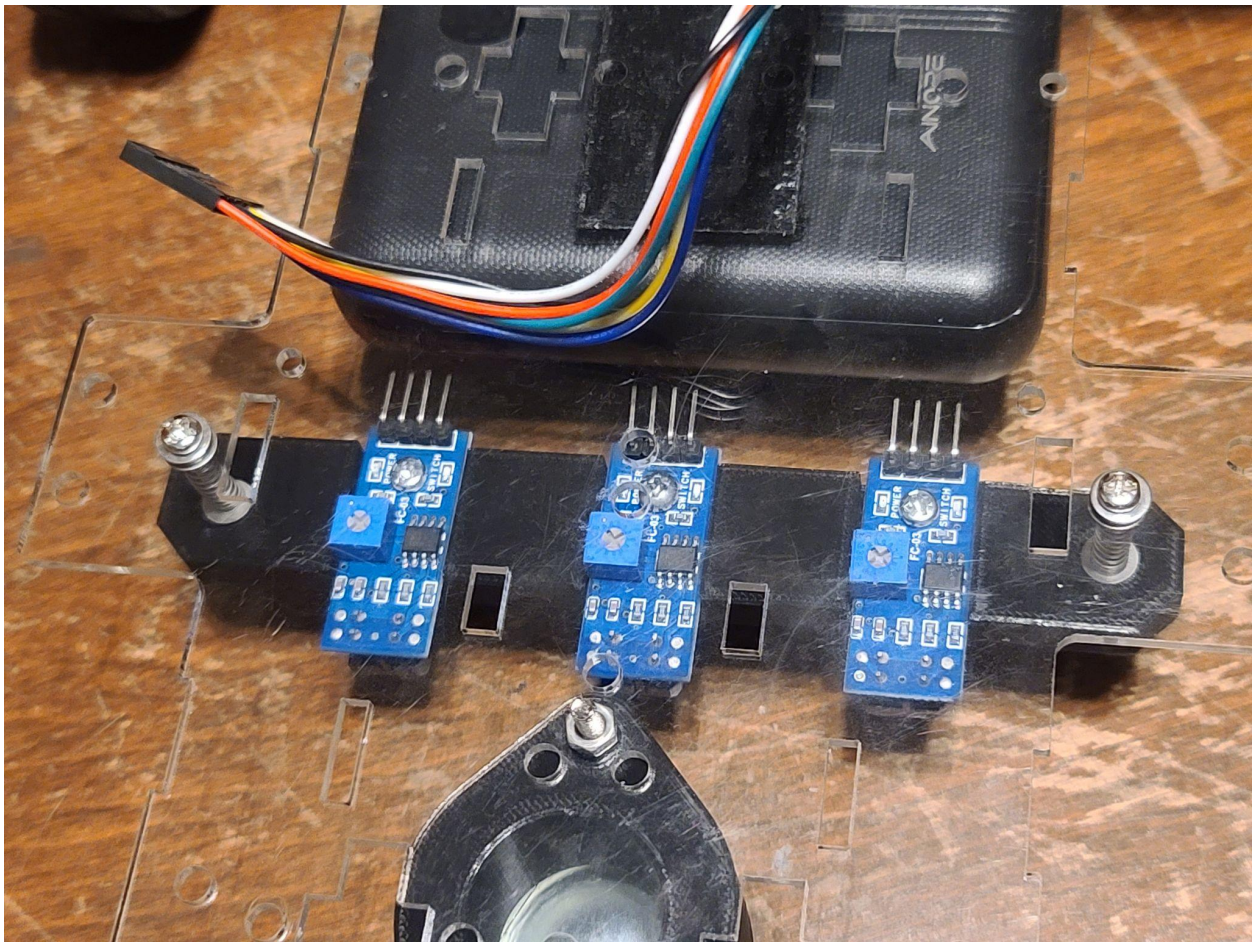


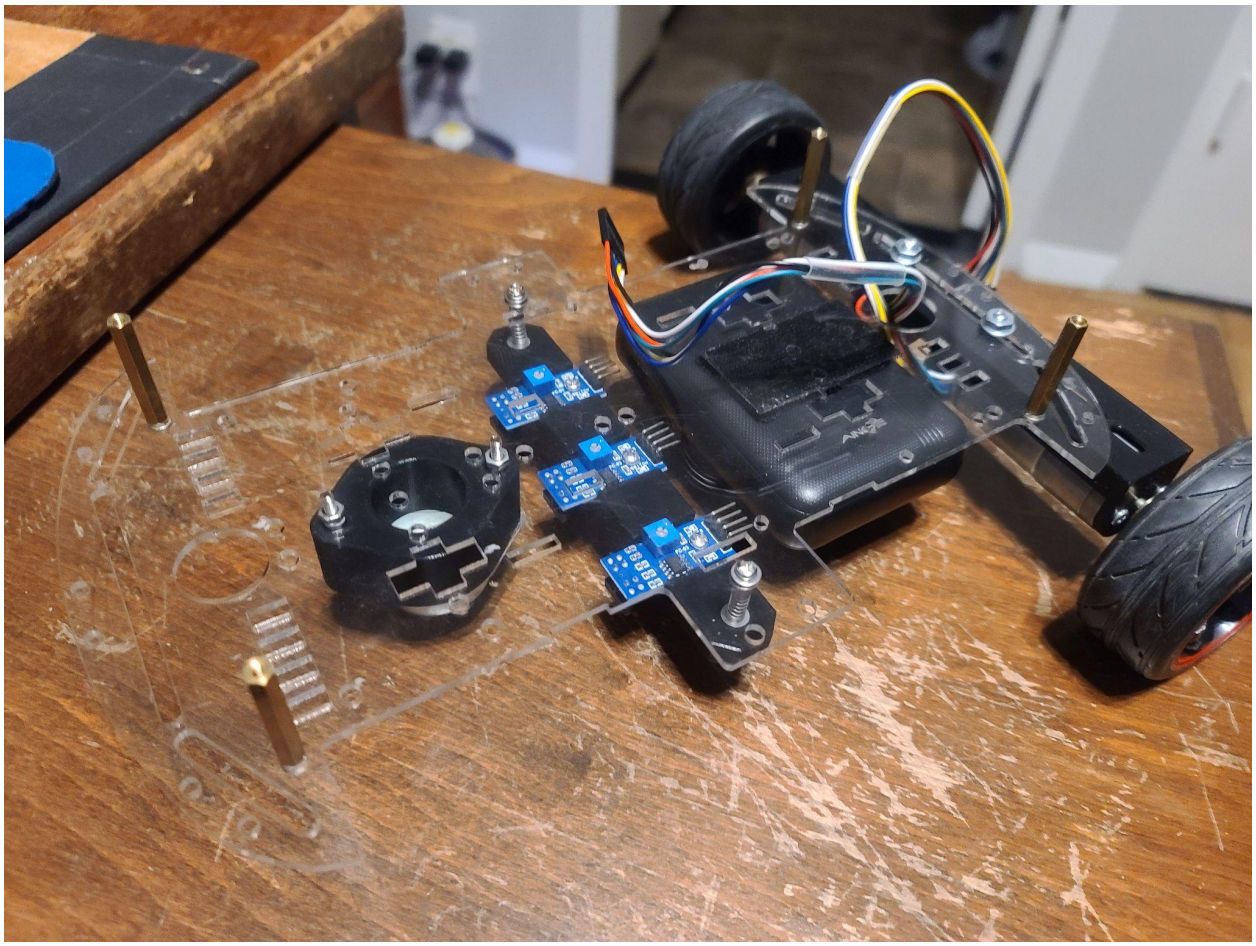




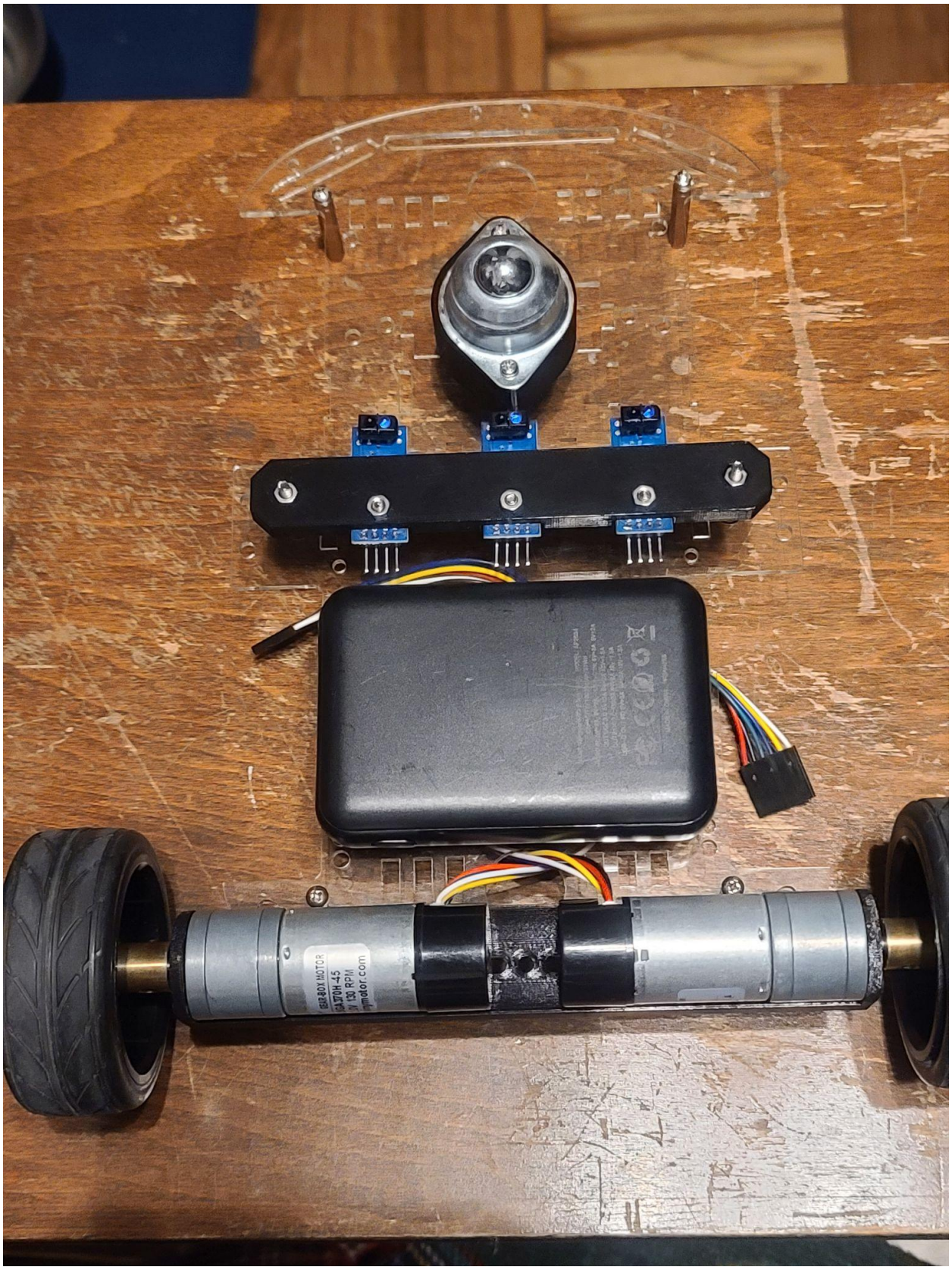

SSION
EEP BLUE

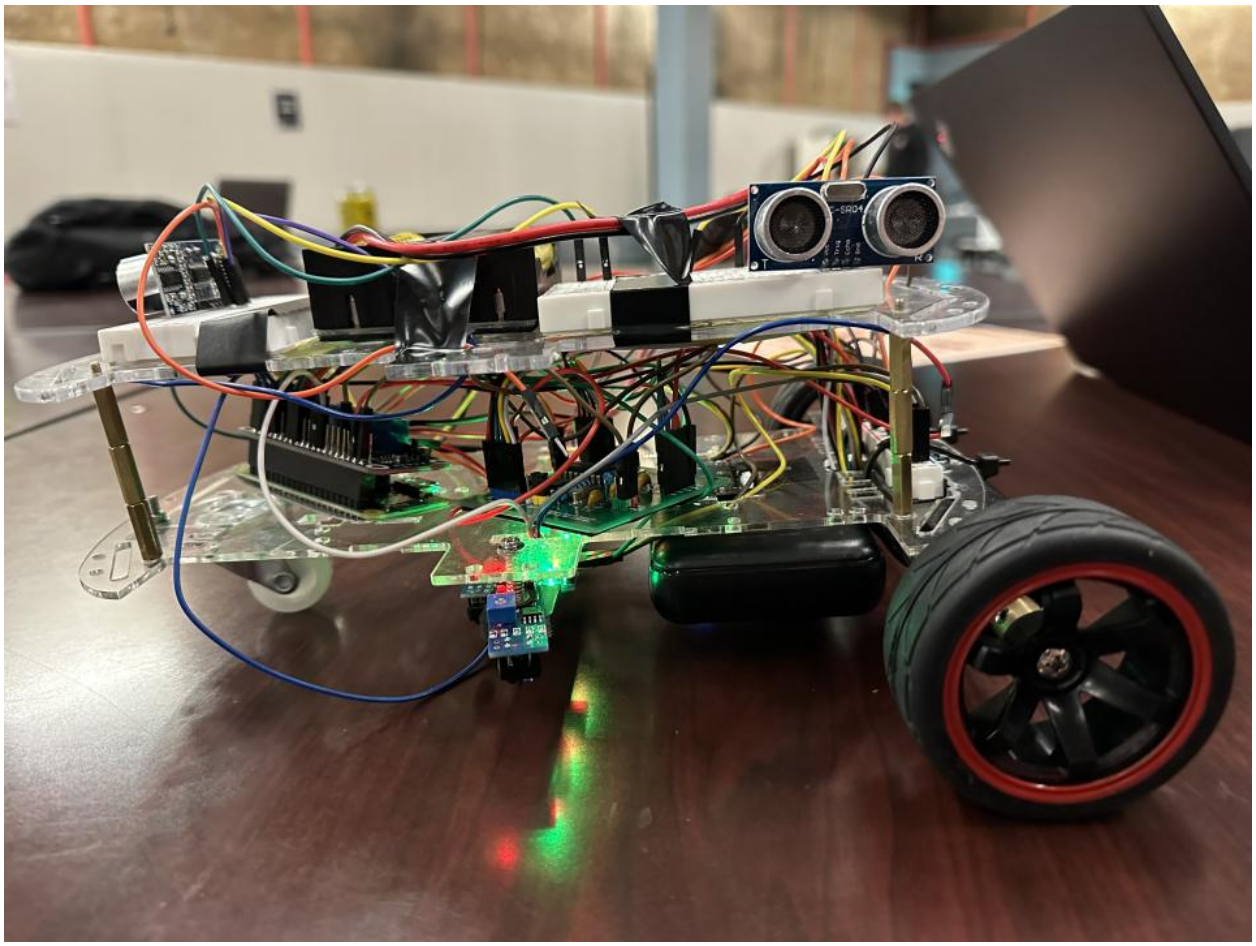


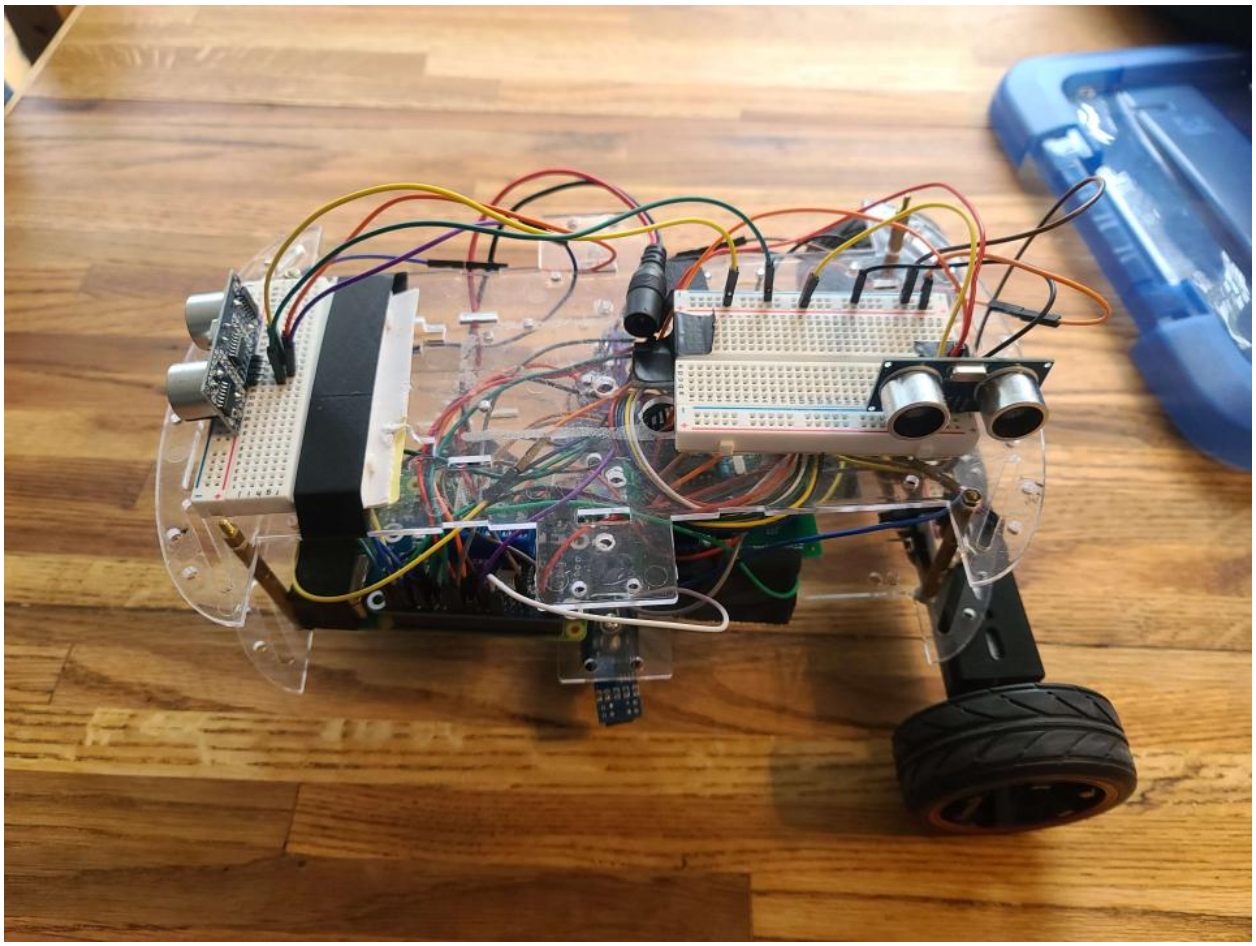

ANCHE
POWER
SWITCH

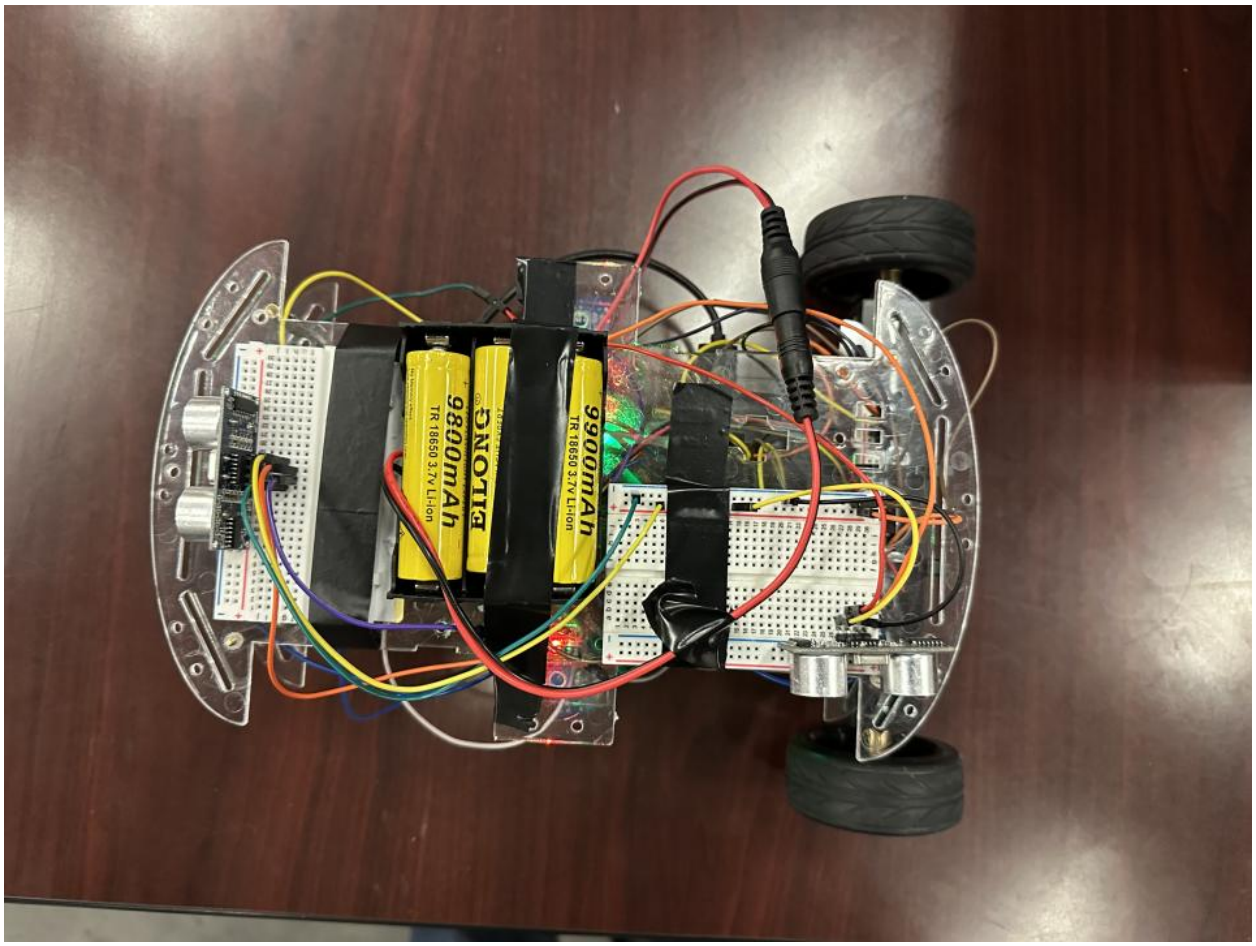


-


GEARBOX MOTOR
GM370H-45
130 RPM
motor.com






9800mAh
TR 18650 3.7v Li-ion
9900mAh
TR 18650 3.7v Li-ion


EXIT