Assignment Term Project

Team members:

- Bryan Yan
- Natalie Yam
- Westley Cho
- Griffin Evans

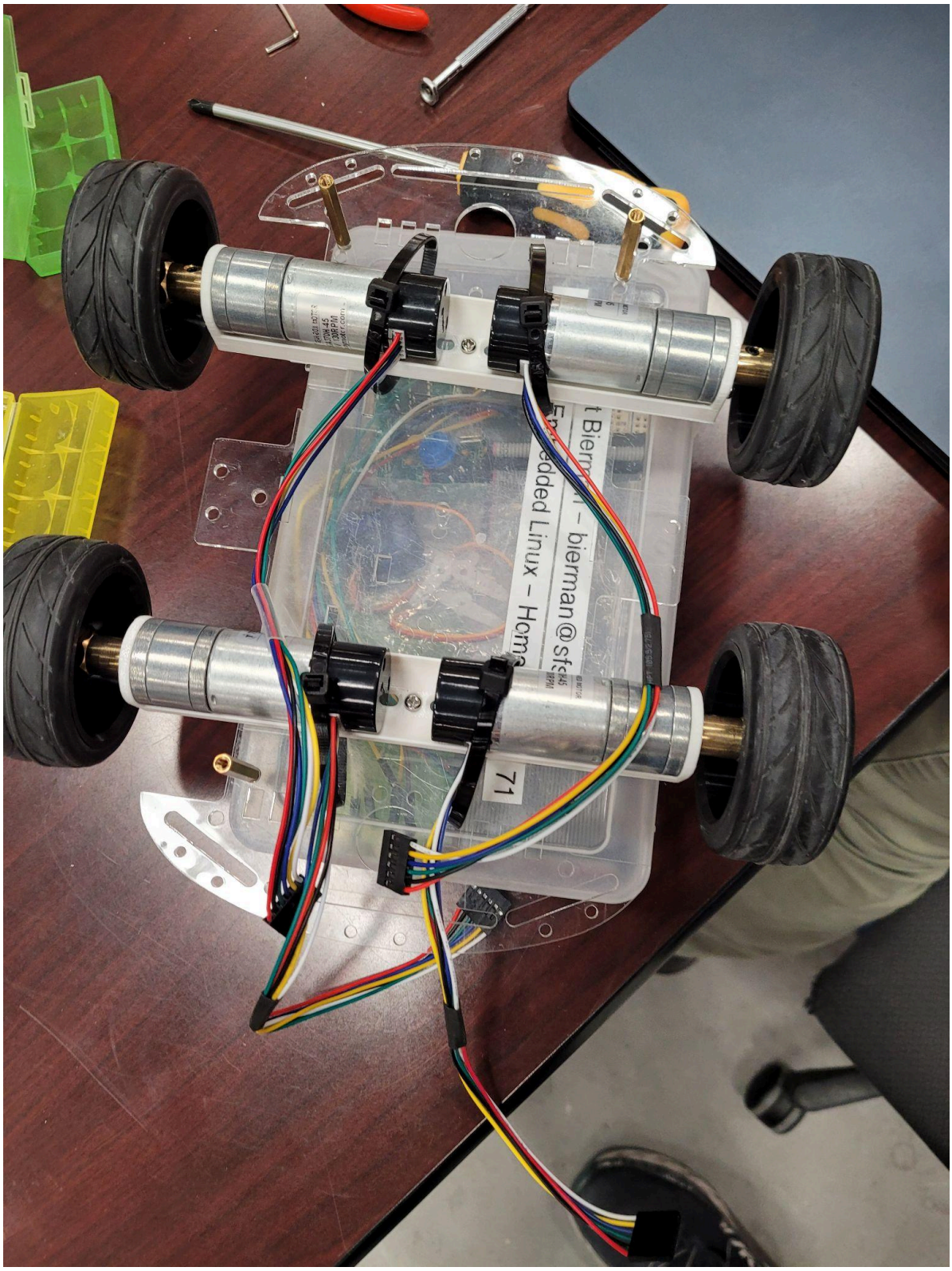
Primary Github Username: westleyc30

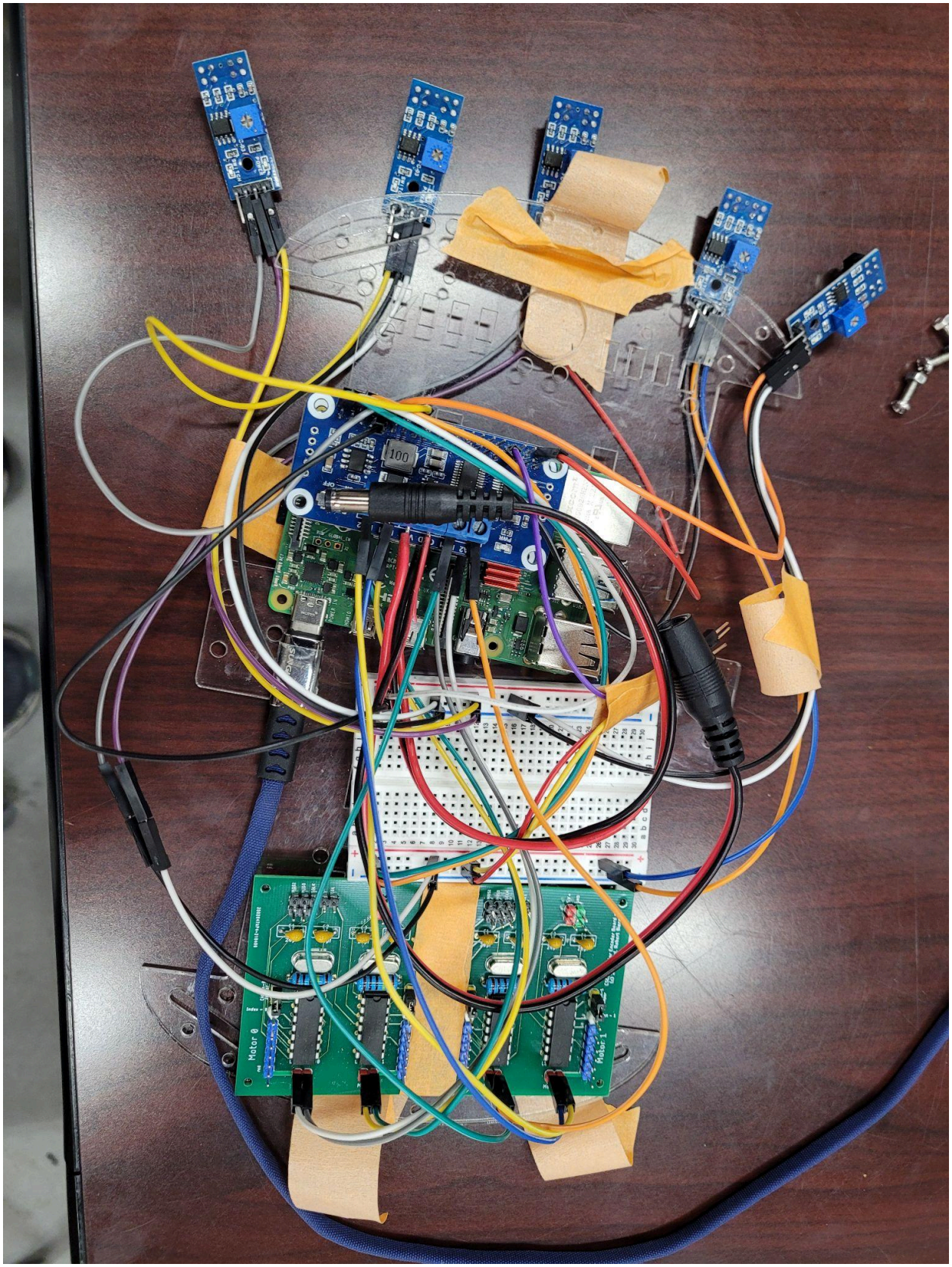
Github: https://github.com/CSC615-2024-Spring/csc615-term-project-westleyc30/

Task Description:

The task is to create a follow the line and obstacle avoiding bot. We are to select from a variety of line sensors and obstacle sensors to do this. Threading should be used to monitor the line sensors and obstacle sensors and control the vehicles movements according to the values that the various sensors read. There are many combinations of sensors and sensor placements and we have to choose the best combination to create a bot that follows the line in any pattern and maneuver around any obstacle on the line.

Building The Robot:



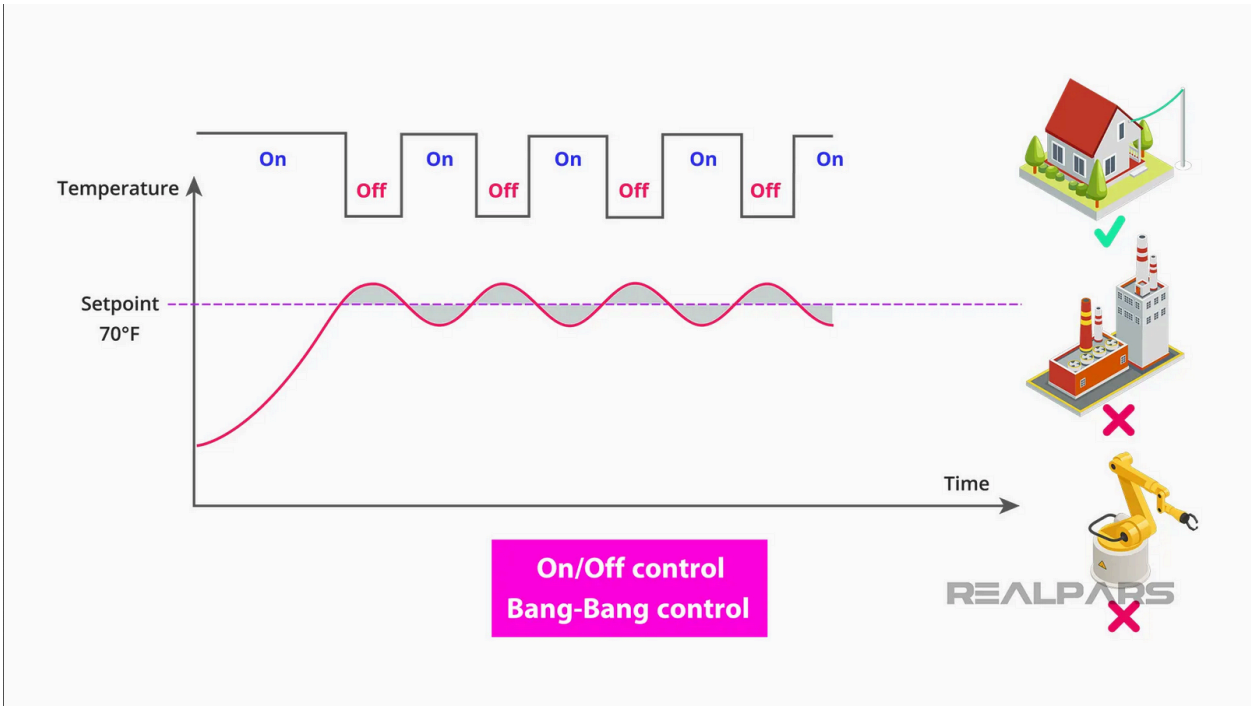

Motor 0
Motor 1

Parts / Sensors Used:

- Waveshare Motor Driver Hat
- 5x TCRT5000 Line Sensors
- 3x IR Infrared Sensors
- Large Rubber Tires
- TS25GA370-45 Engine
- 4x 3.7v Li-ion batteries

How the Bot was Built:

Our initial approach for our line follower was using a left and right sensor to figure out if we needed to make a right or left turn. They were spaced apart about the same width as the black tape on the ground and worked well only when the turns weren't sharp (I.E going from a straight line to a 90 degree turn). In order to account for sharper turns, we added two extra sensors on the far left and right of the car to let us know when a sharper turn needed to be made and used the inner to sensors to make smaller turns. With this method, we were able to clear the course but there were a few issues that were left to solve. What we had implemented is what is known as the "Bang-bang" controller which works but it only oscillates between the line and never truly centers itself fully.



We can imagine the setpoint in this photo as the line we are trying to follow and the "ON" "OFF" to be our "LEFT" "RIGHT" turns

This oscillation can be minimized by hard setting specific speeds for left and right turns but this is a temporary solution that only accounts for one specific type of turn and doesn't account for all possible types of turns that lie in between a straight and a sharp turn (I.E All turns that lie in between 0-90 degrees) and we end up oscillating or undershooting turns again.

This leads to our current algorithm, the PID controller. The PID controller stands for Proportional, Integral, Derivative and it's a popular method for maintaining a certain state (In our case, staying on the line).

- $u(t) = K_p e(t) + K_i \int e(t)dt + k_p \frac{de}{dt}$

The first part of the equation is Proportional, second part is the Integral, and third part is the Derivative

- Here Kp and Ki are constants that give the error its weight in regards to what you are trying to accomplish(For example, if I needed to do a sharp Proportional turn and I got an error of 4 and I have no Kp, an addition and subtraction of 4% to the power of the motors isn't going give me the turn I want)
- e(t) is the error at a given moment

PID varies depending on where it's being applied but in the case of our car, we use P to add and subtract power from our left and right motors proportionally to how much error we have(how far left/right we are away from the line)

- I is used to get rid of any steady-state error we have from D
- D is used to predict what our future error is going to be based on how fast our error has been changing when trying to follow the line.
- In summary, "P" does the main bulk of the turning, "I" solves steady-state error when P stops adjusting as it gets closer to the targeted goal, and "D" predicts future error to avoid constant corrections.

With this control system, our car makes precise adjustments for all types of turning angles.

For obstacle avoidance, we went with using IR Obstacle sensors to detect if an object is in front and to detect if we've passed an object while making a turn.
When the front two obstacle sensors trigger, the car will adjust itself to be parallel to the obstacle and the side obstacle sensor takes over.

While the side obstacle sensor detects that an object is in range, the car will move forward and when the side obstacle sensor detects no obstacle, it will turn.

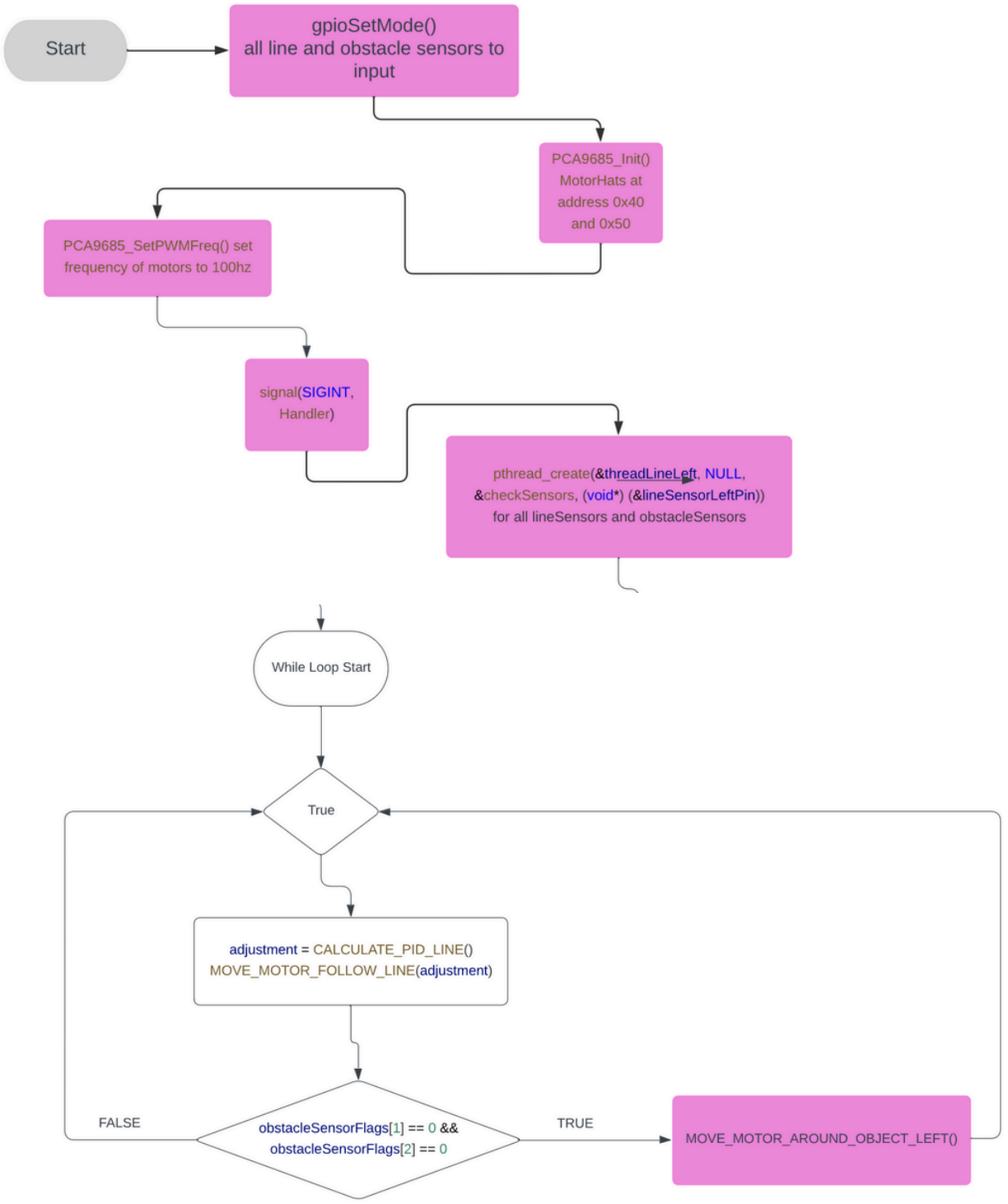
This process will continue until the car detects the track once more, at which point the obstacle sensors will transfer control back to the line sensors.

In summary, our obstacle avoidance method outlines the obstacle until the car finds the track again.

What libraries/software did you use in your code (include full reference):

We used the waveshare motor hat driver with pigpio under the hood.

Flowchart of your code:



Pin assignments you used:
Line Sensors Pin numberings:
LINE_LEFT 14

LINE_RIGHT 15

LINE_MIDDLE_LEFT 23

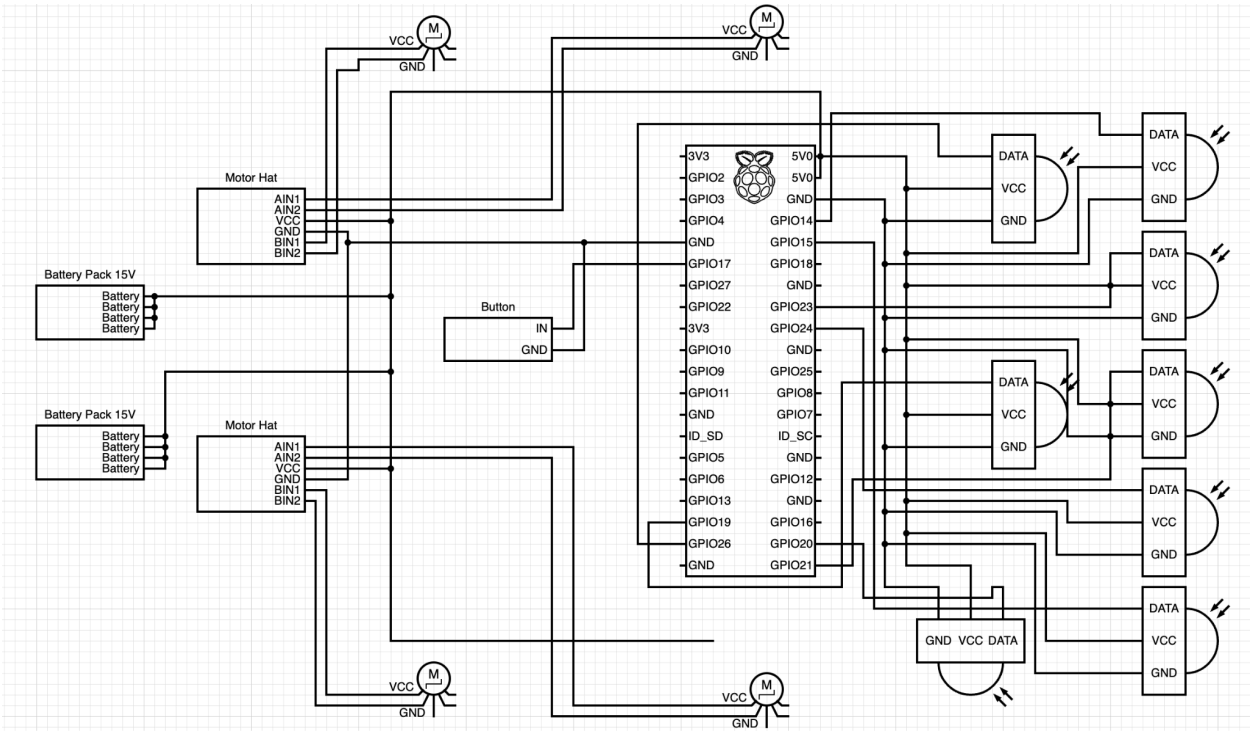
LINE_MIDDLE_RIGHT 24

LINE_MIDDLE 21

Obstacle Sensors Pin Numberings:
OBSTACLE_FRONT_LEFT 26
OBSTACLE_FRONT_RIGHT 19

OBSTACLE_SIDE_RIGHT 20

Hardware Diagram:



What worked well:

- When we first began designing the line following feature for the bot we chose to use two line sensors. We then realized this form of line following called, bang-bang controller method,
- Using IR sensors provided us with consistency due to the IR sensors always triggering at a set distance and trigger almost instantly. This allowed us to always have enough clearing when making turns and instantly make adjustments.
- Rubber wheels allowed us to get better traction and higher speeds

What were issues:

- One of our biggest issues early on was power consumption. We had to upgrade the batteries several times, and we ended up changing out the omniwheels for rubber tires for more speed.
- Yesterday, we ran into a problem where if we tried to add the servo+ultrasonic sensor for obstacle avoidance, our system would shut off due to a lack of power. On top of that, our line sensors were too high up off the ground so they needed to be reattached under the chassis. I thought this would be a good opportunity to try out the setup where we take out the front motors and add a roller wheel for the potential weight reduction and power reduction benefits.
- After the modifications, our power issue was resolved but created two new issues. Firstly, due to our cars speed and the lack of traction from our Omni wheels, our car would go past turns because it couldn't turn sharply enough to clear sharp turns and secondly, our car was just out right not moving as fast with four motors.

With this discovery, I think it's a good idea to implement the four motors again, change the wheels to rubber wheels, and just add an extra battery set for the second motor hat even if our car is a bit heavier. I think the extra power from the motor+batteries outweighs the negatives extra weight brings, especially since the course is probably going to take place on a white tarp where we will have even less traction than we already do with a regular floor. If anything, the extra weight will help us in this scenario.

- Our initial approach for obstacle avoidance was to use an ultrasonic sensor + servo combo to read the distance between the car and the obstacle and make turns accordingly. We would try to maintain a certain distance away from the obstacle and use PID to make precise adjustments. After extensive testing with the servo + ultrasonic sensor combo, we decided to go with the IR Obstacle sensors instead due to the obstacle sensor's reliability. Both our ultrasonic sensors gave inconsistent distance readings which made our turning adjustments sporadic and inconsistent as well. Once we switched to using three IR obstacle sensors, our turns were more consistent due to the readings being more consistent as well.