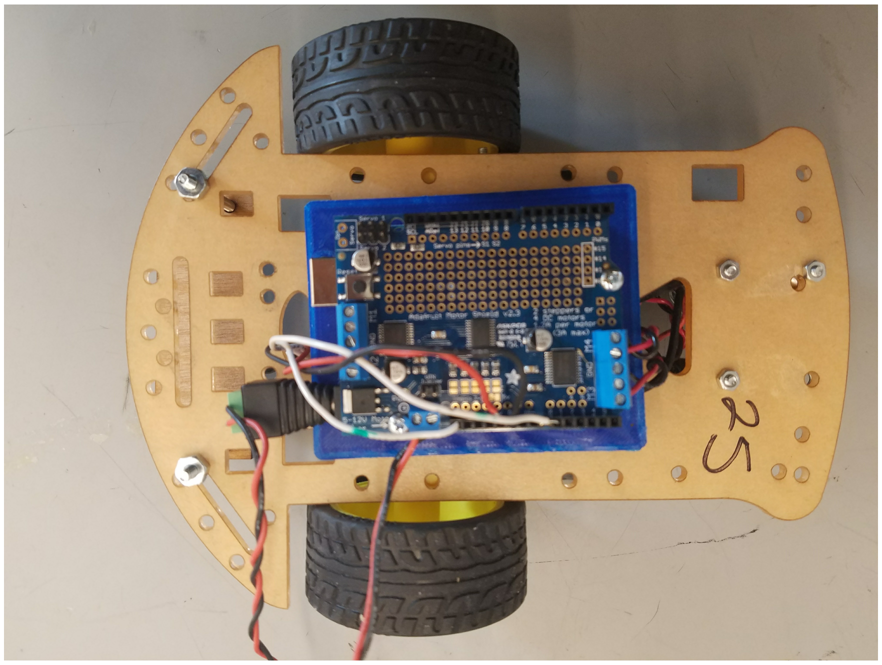

POE Lab 3: Line Following Robot

Katya Donovan, Victoria McDermott

October 13, 2017

Abstract

Using a motor shield, an Arduino, two motors, two reflective optical sensors, and a mechanical system, we created a line following robot that detected the presence and absence of the tape line. Our robot was able to complete the entire track.

1 Introduction

During this lab, we created a line following robot that could complete a course marked down by black tape. This technology highlights the self sufficiency of robots; we simply uploaded the code and connected the robot to power, and it was able to direct itself. Similar technology is used in some cars today that sense the lanes on the highway, and can automatically steer the car in the correct direction if the driver is drifting too heavily.

2 Electrical Configuration

One of the first decisions that we had to make when designing our robot was to determine which resistor we would have to use for the reflective optical sensor. We knew that the LED took a 200 Ohm resistor due to the lab requirements. At first, we chose our resistor value by dividing the voltage by the collector current of the sensor, which gave us a resistor value of 500 Ohms.

$$Resistor = 5Volts/.01Amps = 500Ohms \tag{1}$$

However we soon realized that this was a low resistor value so we raised the resistor values until we received values from the sensors that were very different for when the sensor was on the black line or off the black line. We eventually settled on a resistor value of 4.7 kOhms, which would give us values around 900 when the sensor was on the black line and 600 when the sensor was not on the black line.

We connected our two sensors to analog pins on the Arduino, which allowed us to read the output of the sensors. Using these outputs, we controlled the two motors using the motor shield.

Figure 1 shows the schematic of our electrical configuration, which we determined from the lab requirements.

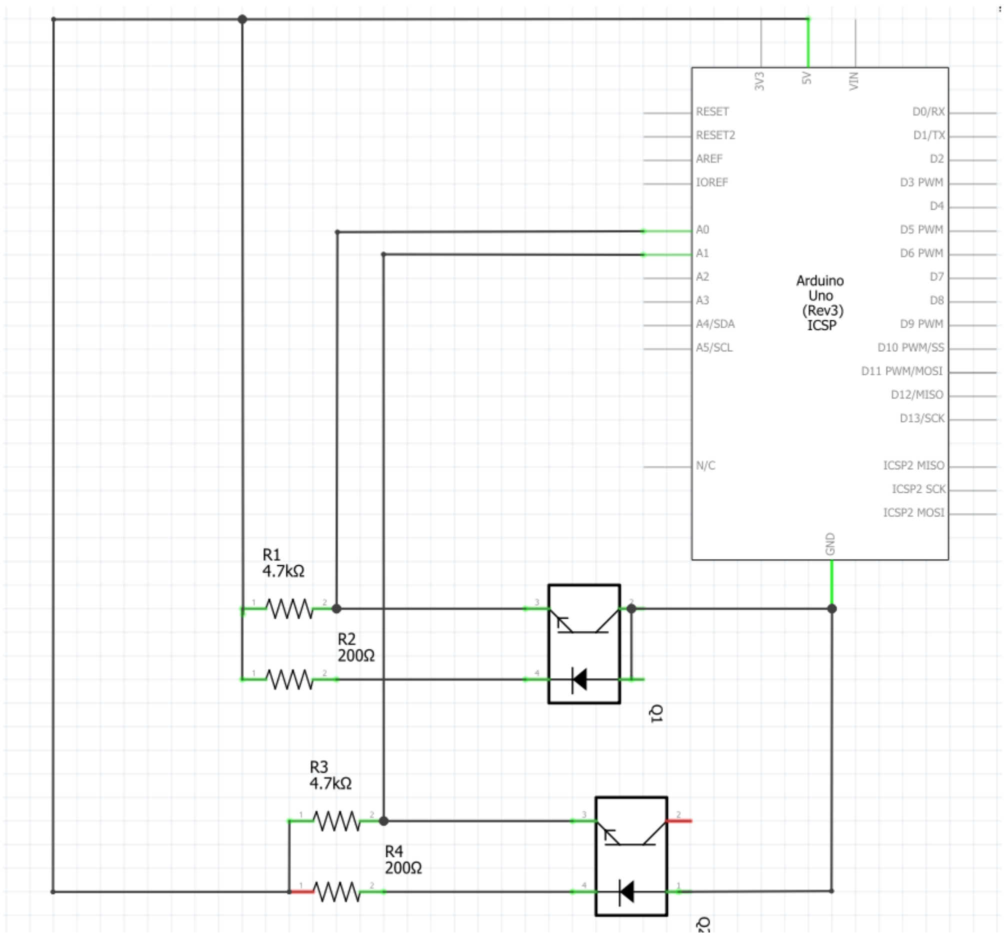

Figure 1: Schematic diagram

3 Calibrating the IR Sensor

In order to calibrate our two sensors, we mounted them onto the chassis and stored the outputs of the sensors. We tested the sensors at different heights and set the height where the output was consistent to determine at what height the sensors should be mounted. If the sensors were two low to the ground, the values were very high, and if they were too far away, the sensor values were also too high. Once we determined the correct height, we tested the sensors on two surfaces, one with the black tape on the ground and one with black tape on the white foam board. Figure 2 and Figure 3 show the calibration graphs of the two sensors on two different surfaces. The lower values are when the sensors were over the ground or white foam board without the black tape, and the higher values are when the sensors were over the black tape.

We determined the maximum of the sensor values when they were above the ground and white board and the minimum value of when they were on top of the black tape in order to find the threshold of values between the two options. We chose a value that was between both ranges for both sensors. We decided to chose 850 as our threshold value.

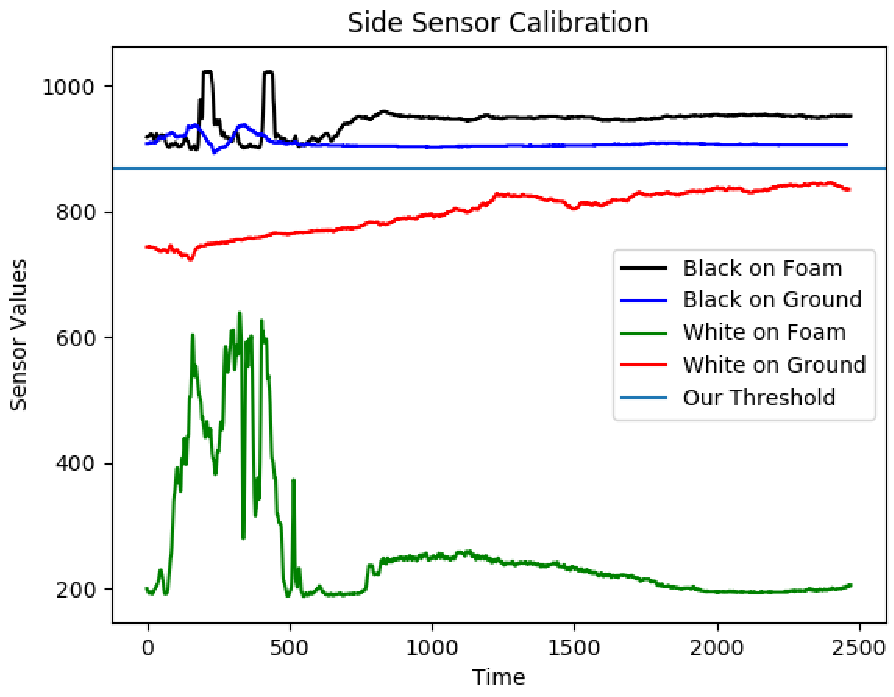

Figure 2: Calibration for our side IR Sensor with our threshold value of 850.

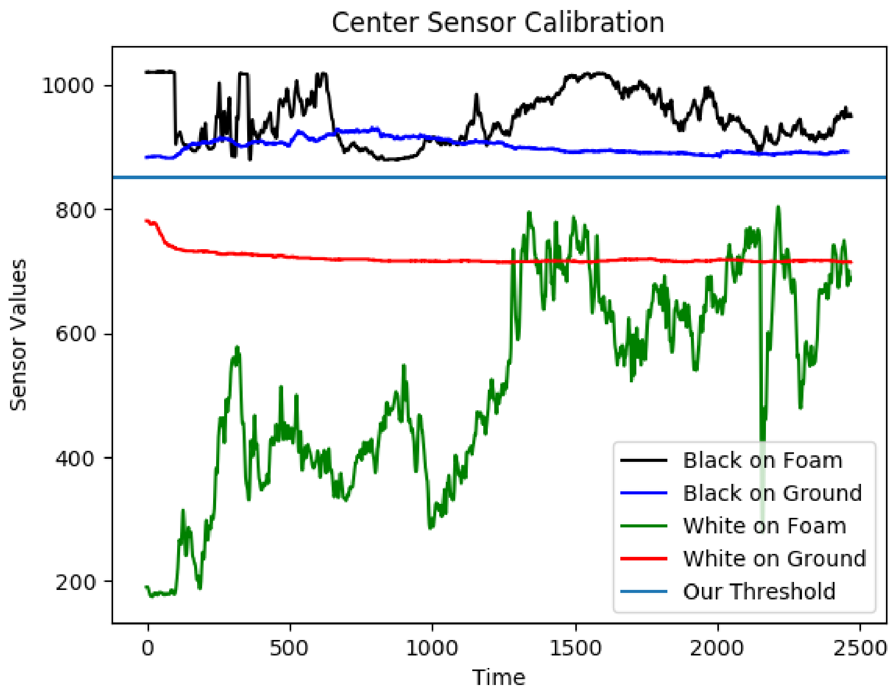

Figure 3: Calibration for our center IR Sensor with our threshold value of 850.

4 Mechanical Design and Integration

The biggest priorities for our mechanical design were easily removable and adjustable mounting. We decided to put the sensors at the front of the chassis, because we thought that our robot would react more quickly to the line if the front end of the chassis could observe the changes in the ground. If the sensors were on the back, then the front of the car would not follow the line nearly as much.

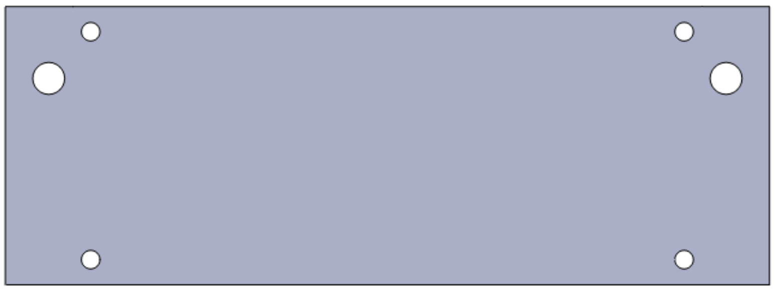

Figure 4: Mount for the printed circuit board.

We placed a sensor directly above the black tape and one to the left of the black tape. Since we did not know what height our sensors should be at originally, we wanted to mount the breadboard that the sensors were soldered to in a way that we could configure the height easily. Thus we decided to follow a turn buckle configuration, with two bolts connected to the mounting of the printed circuit board and two hex nuts that clamp on either side of the chassis. With this configuration, we only had to screw the hex nuts up or down in order to adjust the height.

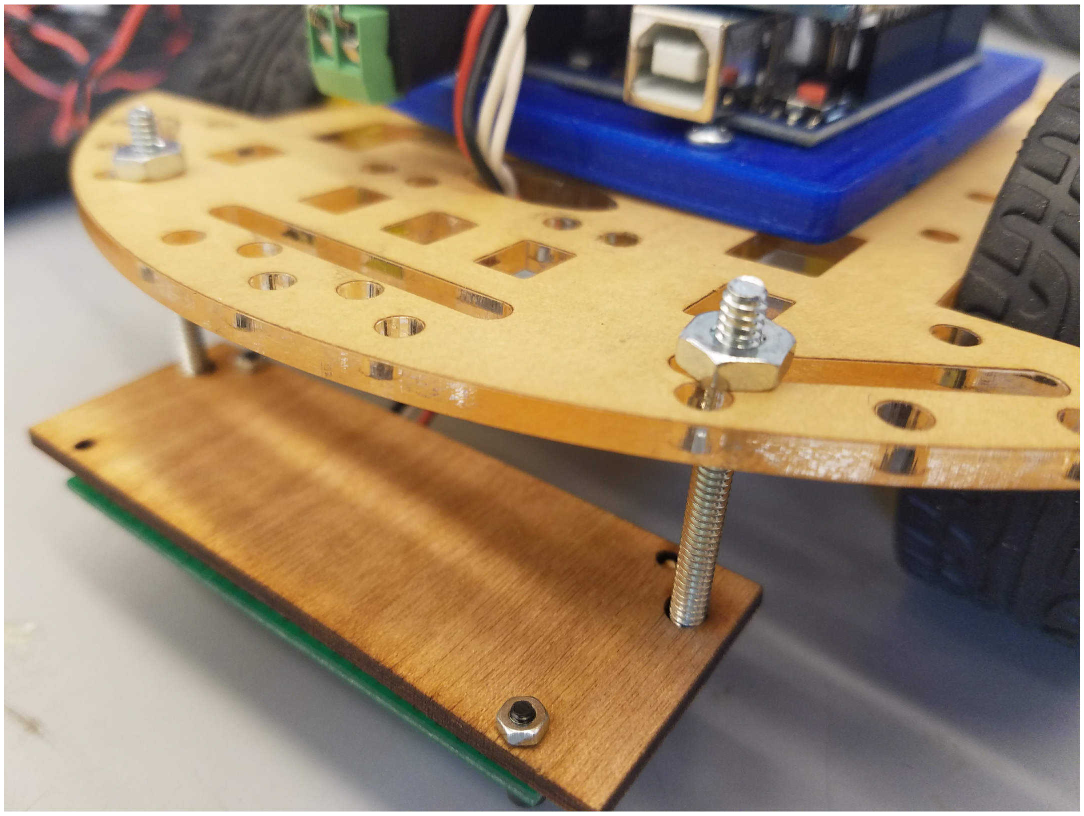
Figure 5: Mount for the printed circuit board.

In order to connect the Arduino and Motor Shield to the chassis, we decided that we should 3-d print a mounting. The mounting had a pocket for the pins of the Arduino to rest. The Arduino was screwed directly into the mounting to ensure that no metal was on the bottom of the mounting. We attached the mechanical mount to the chassis with two bolts. Since we decided that the Arduino would most likely not need to move, we did not think it was necessary to make the design adjustable.

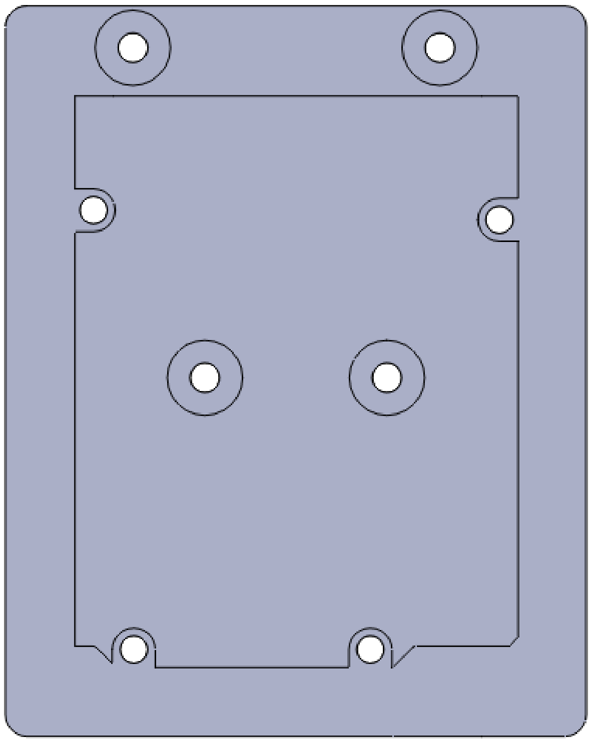

Figure 6: Mounting system for the Arduino and Motor Shield.

Our entire mechanical system included one 3-d printed Arduino mount and one laser cut bread board mount as shown in the figure below.

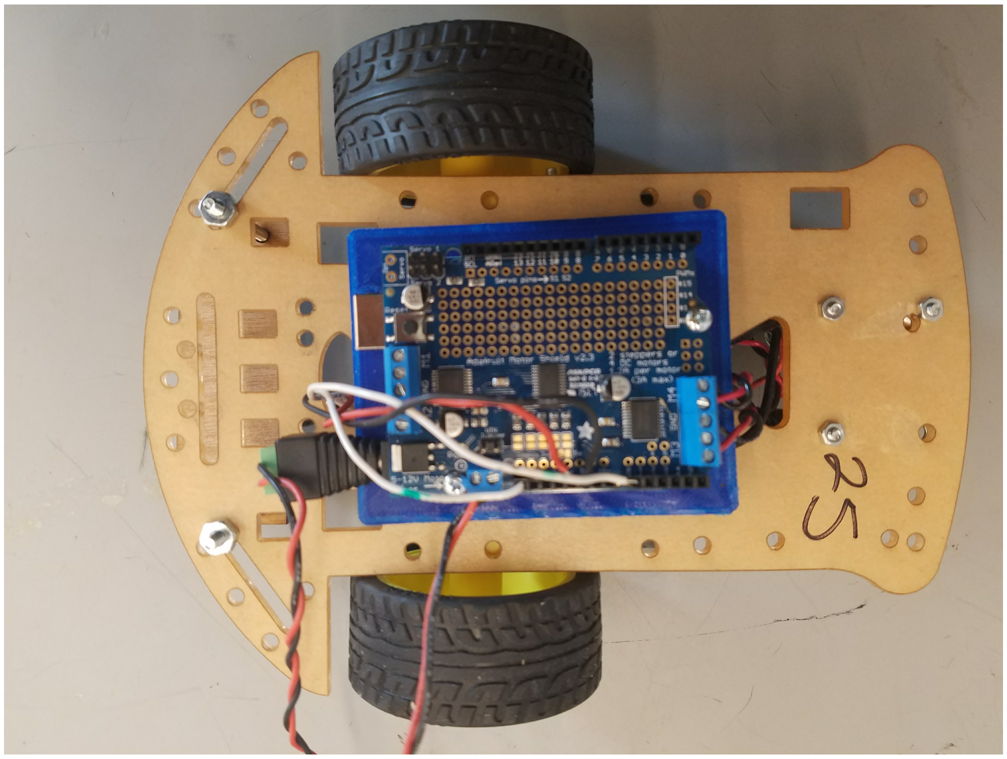

Figure 7: Mounting system for the Arduino and Motor Shield.

5 Control System

For our control system we decided to simply use the sensor values to map to different motor speeds for turning our robot. We knew that when the center sensor detected black and the side sensor detected white, our robot could just continue moving forward at a constant speed because in this case it was going in a straight line. If our robot hit a right turn both of our sensors would fall off the black line and detect only the floor. In this case, we would tell our robot to turn right by instructing one of our motors to move at a faster speed than the other.

```
void turnRight(int mySpeed){
        motor1->setSpeed(mySpeed);
        if(mySpeed>20){
           motor2->setSpeed(mySpeed-20);
        }
        else{
           motor2->setSpeed(0);
        }
        motor1->run(FORWARD);
        motor2->run(FORWARD);
      }
```

If both of our sensors detected black we would tell our robot to turn left in a similar manner. Similarly, if the side sensor detected black and the center sensor detected white, we would know that our robot needed to turn left.

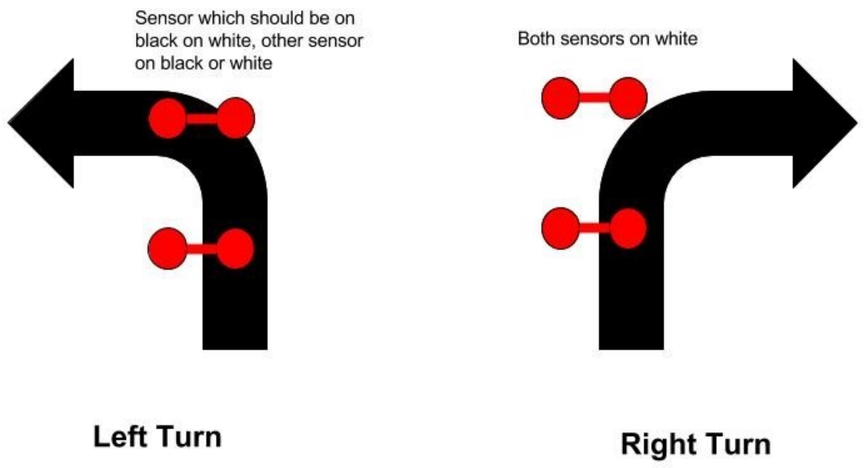
Sensor which should be on black on white, other sensor on black or white
Both sensors on white
Left Turn
Right Turn

Figure 8: Image which demonstrates the conditions we used to determine which way to turn based on our input sensor values.

When we tested it on the track for the first time, we realized that the robot would sometimes over correct itself because it kept turning for however long the delay was. We decided to implement an if function that told the robot to turn left until a certain time period or until the middle sensor reached black and the side sensor reached the ground. If the sensors were in this position, then the robot would break out of the if statement and continue moving forward. The following code shows this implementation.

```
else if(b>bthreshold && a<athreshold){
 previousTime = millis();
 while(a < athreshold){
   turnLeft(speed);
   currentTime =millis();
   if (currentTime - previousTime >= interval) {
     previousTime = currentTime;
     break;
```

6 Sensor Values and Motor Speeds

As seen in the following Figure 9, Figure 10 and Figure 11, the motor speeds change when the sensor values change. When the middle sensor value is on the line and the side sensor value is on the ground, both the wheels continue to move at speed of 30. This is clearly shown in Figure 11 that displays motor speeds and sensor values over time. When the sensor values switch, so that the middle sensor is above the ground and the side sensor is above the black line, then the robot will turn left. Thus, the left motor slows down and the right motor stays the same, which is visible because the red line graphs to 10, which means that the robot turns left.

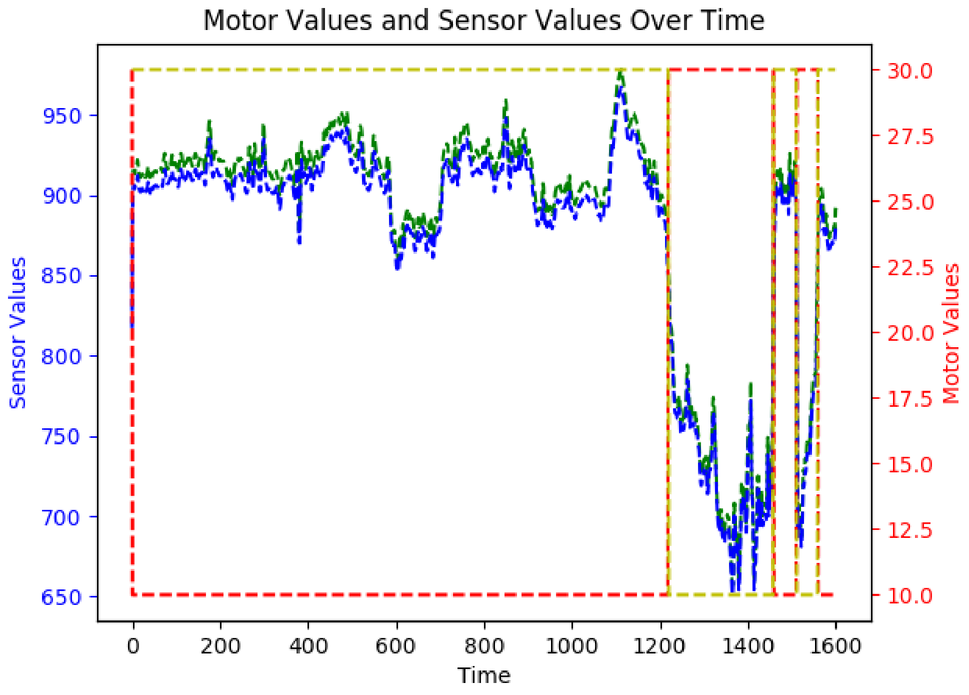

Figure 9: Graph of Motor Speeds and Sensor Values over time.

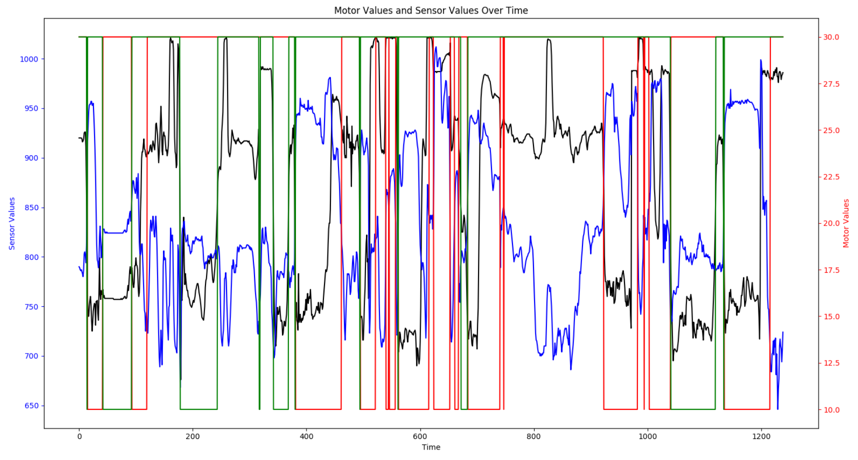

Figure 10: Graph of Motor Speeds and Sensor Values over time.

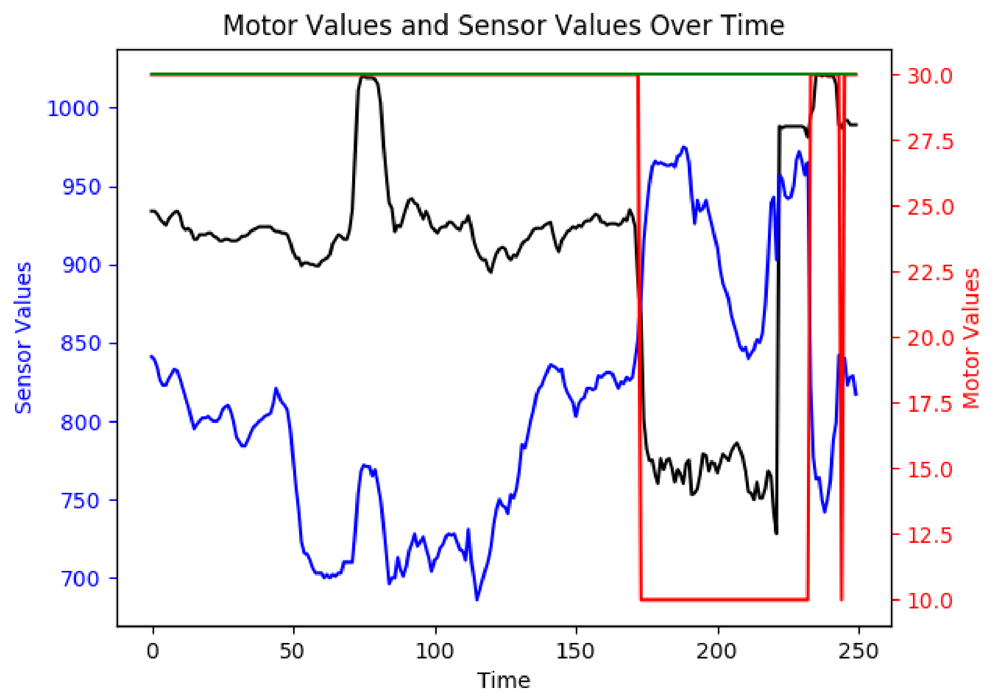

Figure 11: Graph of Motor Speeds and Sensor Values over time.

7 Completing the Track

Our line following robot completed the track in 76 seconds. The video of our robot completing the track can be found here (https://www.youtube.com/watch?v=Zpd-80-eS0Qfeature=youtu.be).

8 Obstacles and Solutions

Throughout this lab there were many obstacles that slowed down the progress of the line following robot. One of the biggest problems that we faced was not being able to print to the serial monitor. The serial monitor would only read the first two characters of the first print statement before stopping its print statements. After testing many different possibilities as to why this might be the case, we tried using another computer and testing the Arduino code on a separate operating system which seemed to work well.

We also had some difficulties with our electrical connections where our motors were not getting power even when we sent the example motor code to them. We were able to solve this problem by re crimping our wires and ensuring that all of our electrical connections were in working order.

9 Appendix

Listing 1: IR Sensor Calibration Code

```
int cmd_id;

```

```
void setup()

{

Serial.begin(9600);

pinMode(4,OUTPUT);
digitalWrite(4,HIGH);



}

void loop(){
//Read incoming data from serial
if(Serial.available() >= 0) {
    cmd_id = Serial.read();
  }
  else{
    cmd_id = 0;
  }

//Start reading sensor values and send over serial (to Python)
switch(cmd_id){
  case 1:

    delayMicroseconds(500);

    int a=analogRead(A0);

    Serial.println(a);

   break;
}

}
```

Listing 2: Python Code to Graph Calibration Data

```
import json
import matplotlib.pyplot as plt

with open('side_sensor_black_on_foam.csv', 'r') as f:
    black_on_foam = json.load(f)

with open('side_sensor_black_on_ground.csv', 'r') as f:
    black_on_ground = json.load(f)

with open('side_sensor_white_on_foam.csv', 'r') as f:
    white_on_foam = json.load(f)
```

```

with open('side_sensor_white_on_ground.csv', 'r') as f:
    white_on_ground = json.load(f)

plt.title('Side Sensor Calibration')
values = plt.plot(black_on_foam, 'black', label='Black on Foam')
values2 = plt.plot(black_on_ground, 'b', label='Black on Ground')
values3 = plt.plot(white_on_foam, 'g', label='White on Foam')
values4 = plt.plot(white_on_ground, 'r', label ='White on Ground')
ourline = plt.axhline(y=870, label='Our Threshold')
plt.legend()

plt.xlabel('Time')
plt.ylabel('Sensor Values')

plt.show()
```

Listing 3: Line Following Arduino Code

```
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"

// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
unsigned long previousTime = 0;
unsigned long currentTime = 0;
const long interval = 500; // the time interval

// set up the two motor variables
Adafruit_DCMotor *motor1 = AFMS.getMotor(3);
Adafruit_DCMotor *motor2 = AFMS.getMotor(4);
int athreshold = 850;
int bthreshold = 850;
int stop = 0;
int speed = 30;
int cat = 0;
int integerValue = 0;
int incomingByte = 0;
int cmd_id;

void setup()
{
  Serial.begin(9600);           // set up Serial library at 9600 bps


  AFMS.begin();  // create with the default frequency 1.6KHz
  // Set the speed to start, from 0 (off) to 255 (max speed)
  motor1->setSpeed(0);
  motor2->setSpeed(0);
  // make sure the motor is stopped to start
  motor1->run(RELEASE);
  motor2->run(RELEASE);
}

void loop(){

    // control motor speed over serial connection
   if (Serial.available() > 0) {    // something came across serial
        integerValue = 0;           // throw away previous integerValue
        while(1) {              // force into a loop until 'n' is received
          incomingByte = Serial.read();
          if (incomingByte == '\n') break;    // exit the while(1),
                                              // we're done receiving
          if (incomingByte == -1) continue;   // if no characters are in
                                             // the buffer read() returns -1
          integerValue *= 10;  // shift left 1 decimal place
          // convert ASCII to integer, add, and shift left 1 decimal place
          integerValue = ((incomingByte - 48) + integerValue);
        }
      speed = integerValue;
    }

    Serial.println(speed);

    delayMicroseconds(500);
    int a=analogRead(A0);
    int b=analogRead(A1);

    if (b<bthreshold && a>athreshold){
      // if center is on black and side on white, move forward
      moveForward(speed);
    }
    else if(b<bthreshold && a<athreshold){
      // if both are on white then turn right
      turnRight(speed);
    }
    else if(b>bthreshold && a>athreshold){
      // if both are on black then turn left
      turnLeft(speed);
    }
    else if(b>bthreshold && a<athreshold){
      previousTime = millis();
      // continue turning left until the center motor sees black
      while(a < athreshold){
        turnLeft(speed);
        currentTime =millis();
```

```
          // break out of the loop if 500 ms has passed
          if (currentTime - previousTime >= interval) {
            previousTime = currentTime;
            break;
          }

        }
      }
}
}

void stopAllMotors(){
        motor1->setSpeed(0);
        motor2->setSpeed(0);
        motor1->run(RELEASE);
        motor2->run(RELEASE);
}

void moveForward(int mySpeed){
        motor1->setSpeed(mySpeed);
        motor2->setSpeed(mySpeed);
        motor1->run(FORWARD);
        motor2->run(FORWARD);
}

void turnRight(int mySpeed){
          motor1->setSpeed(mySpeed);
          if(mySpeed>20){
            motor2->setSpeed(mySpeed-20);
          }
          else{
            motor2->setSpeed(0);
          }
          motor1->run(FORWARD);
          motor2->run(FORWARD);
         }

void turnLeft(int mySpeed){
          motor2->setSpeed(mySpeed);
          if(mySpeed>20){
            motor1->setSpeed(mySpeed-20);
          }
          else{
            motor1->setSpeed(0);
          }
          motor1->run(FORWARD);
          motor2->run(FORWARD);
}
```

```

//function to turn right less sharply
void turnLessRight(int mySpeed){
        motor1->setSpeed(mySpeed);
        if(mySpeed>10){
          motor2->setSpeed(mySpeed-10);
        }
        else{
          motor2->setSpeed(0);
        }
        motor1->run(FORWARD);
        motor2->run(FORWARD);
}

//function to turn left less sharply
void turnLessLeft(int mySpeed){
        motor2->setSpeed(mySpeed);
        if(mySpeed>10){
          motor1->setSpeed(mySpeed-10);
        }
        else{
          motor1->setSpeed(0);
        }
        motor1->run(FORWARD);
        motor2->run(FORWARD);
}
```

Listing 4: Line Following Python Code

```
#!/usr/local/bin/python

from serial import Serial, SerialException
from tkinter import *
import json
import time

cxn = Serial('COM7', baudrate=9600)
data = 0

def send_data():
    var = int(textbox.get())
    if var > 255 or var < 0:
        res.configure(text = "Please enter a valid number(between 0 and 255).")
    else:
        res.configure(text = "Sending value to serial...")
        print(int(var))
        cxn.write([int(var)])
        cxn.write([int(255)])
        time.sleep(6)
        res.configure(text = cxn.readline())

master = Tk()

Label(master, text="Enter␣a␣motor␣speed:␣").grid(row=0)
res = Label(master)
res.grid(row=1)

textbox = Entry(master)
textbox.grid(row=0, column=1)

Button(master, text='Quit', command=master.quit).grid(row=3, column=0,
            sticky=W, pady=4)
Button(master, text='Enter', command=send_data).grid(row=3, column=1,
            sticky=W, pady=4)

mainloop( )
```

Listing 5: Python Code to Collect Motor and Sensor Values

```
#!/usr/local/bin/python

from serial import Serial, SerialException
import json
import time

cxn = Serial('/dev/ttyACM0', baudrate=9600)
results_list = []
first = True
cmd_id = 0

while(True):
    try:
        if cmd_id != 1:
            cmd_id = int(input("Please␣enter␣a␣command␣ID␣(1␣-␣collect␣values:␣"))
            prevtime = time.clock()
            print(time.clock())
        if int(cmd_id) > 3 or int(cmd_id) < 1:
            print ("Values␣other␣than␣1␣2␣or␣3␣are␣ignored.")
        else:
            cxn.write([int(cmd_id)])
            while cxn.inWaiting() < 1:
                pass
            if cmd_id == 1:
                result = cxn.readline();
                result = str(result)
                result = result[2:]
                result = result.strip("\\r\\n'")
                results_list.append(result)
```

```
                    first = False
                if time.clock()-prevtime > 30:
                    cmd_id = 2
                    with open('motor_speeds2.csv', 'w') as f:
                        json.dump(results_list, f)
                    first = True
                print (result)
        except ValueError:
            print ("You must enter an integer value between 1 and 3.")
```

Listing 6: Python Code to Graph Sensor Values and Motor Values

```
import json
import matplotlib.pyplot as plt

with open('motor_speeds.csv', 'r') as f:
    results_list = json.load(f)

sensor1vals = (results_list[0::4])
sensor2vals = (results_list[1::4])
motor1vals = (results_list[2::4])
motor2vals = (results_list[3::4])

fig, ax1 = plt.subplots()

plt.title('Motor Values and Sensor Values Over Time')
ax1.plot(sensor1vals, 'g--', sensor2vals, 'b--')
ax1.set_xlabel('Time')
ax1.set_ylabel('Sensor Values', color='b')
ax1.tick_params('y', colors='b')

ax2 = ax1.twinx()
ax2.plot(motor1vals, 'r--', motor2vals, 'y--')
ax2.set_ylabel('Motor Values', color='r')
ax2.tick_params('y', colors='r')

plt.show()
```