POE Lab 3 - A Line Following Robot

Nathan Yee, Carl Moser

October 15, 2016

Description

In this lab, we created a line following robot using an Arduino and various electrical components.

Video Link

https://www.youtube.com/watch?v=yVtvLZDIzOo

Process

Procedure

A line following robot requires interconnected circuits, computers, sensors, and software. We used an iterative process to consistently make progress towards a functioning and well line following robot. Enumerated, our process looked somewhat like this.

1. Calibrate IR sensors with resistors and Arduino.
2. Write code to make motors turn.
3. Interface with Arduino to slow and speed motors over serial.
4. Create dual IR sensor circuit on breadboard.
5. Create circuit diagram.
6. Make chassis mount.
7. CAD and render chassis mount.
8. Write code that interfaces both the motors and the dual IR sensors.
9. Robot followed line on first try (just kidding).
10. Troubleshooted code.
11. Played around with motor speeds.
12. Robot followed line.

Testing and calibrating the IR sensor

We started testing and calibrating the IR sensor by building a circuit consisting of a 200 and 52.3 Ohm resistor. The 200 Ohm resistor is in series with the LED and just needs to protect the LED. The 52.3 Ohm resistor is in series with the IR sensor and determines the range of two types of readings. The floor readings are the range of analog voltage reads of the floor. Tape readings are the range of readings we see when looking at the black tape. Here are some of our calibration results:

1. 52.3 Ω resistor. Tape 1022 - 1023. Floor 1007 - 1023
2. 200 Ω resistor. Tape 1022 - 1023. Floor 961 - 1023
3. 1K Ω resistor. Tape 975 - 1021. Floor 903 - 1007
4. 10K Ω resistor. Tape 650 - 960. Floor 53 - 1010
5. 100K Ω resistor. Tape 33 - 850. Floor 26 - 930

For the given resistor values above, we decided to go with the 10K resistor. Here's what we expect to happen. When we are over the floor, we will see a value between 50 - x, where x could be 650 - 960. If we are over the tape, we see a value between x - max. In the case of our robot, we choose to set a threshold value of 800 to determine when to move the left, right, or left and right motors.

How our controller works

Our controller works in a simple state diagram. If threshold is not passed on either of IR sensors, run both motors. If threshold is passed on right IR sensor, stop left motor. If threshold is passed on left IR sensor, stop right motor.

```
void loop() {
  // read irSensor values
  irSensorRead();

  if(leftValue > irThreshhold){
    // If the left sensor hits the tape, turn off the right motor
    m1->setSpeed(mainSpeed);
    m2->setSpeed(mainSpeed - diffSpeed);
    Serial.println(mainSpeed);
    Serial.println(mainSpeed - diffSpeed);
  }
  else if(rightValue > irThreshhold){
    // If the right sensor hits the tape, turn off the left motor
    m1->setSpeed(mainSpeed - diffSpeed);
    m2->setSpeed(mainSpeed);
    Serial.println(mainSpeed - diffSpeed);
    Serial.println(mainSpeed);
  }
  else{
    // Charge forward!
    m1->setSpeed(mainSpeed);
    m2->setSpeed(mainSpeed);
    Serial.println(mainSpeed);
    Serial.println(mainSpeed);
  }

}
```

Chassis Mount

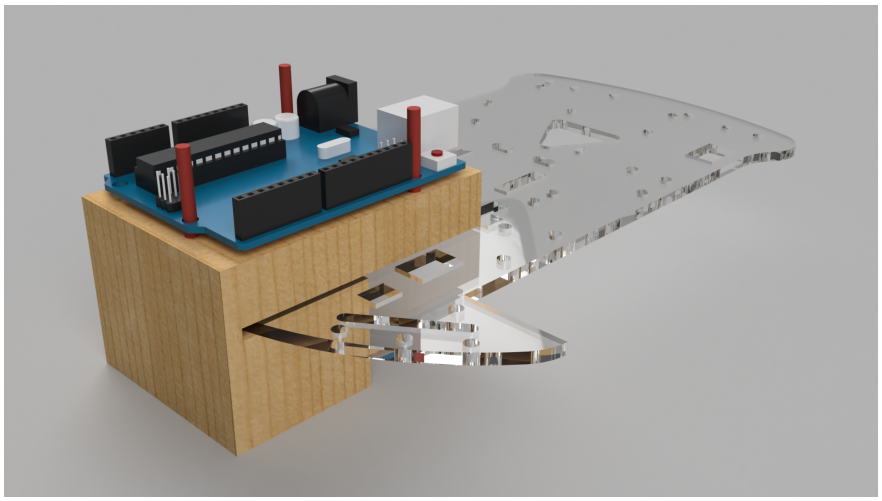

Figure 1: Render of chassis mount. Note that the breadboard is attached to the bottom

Real Life

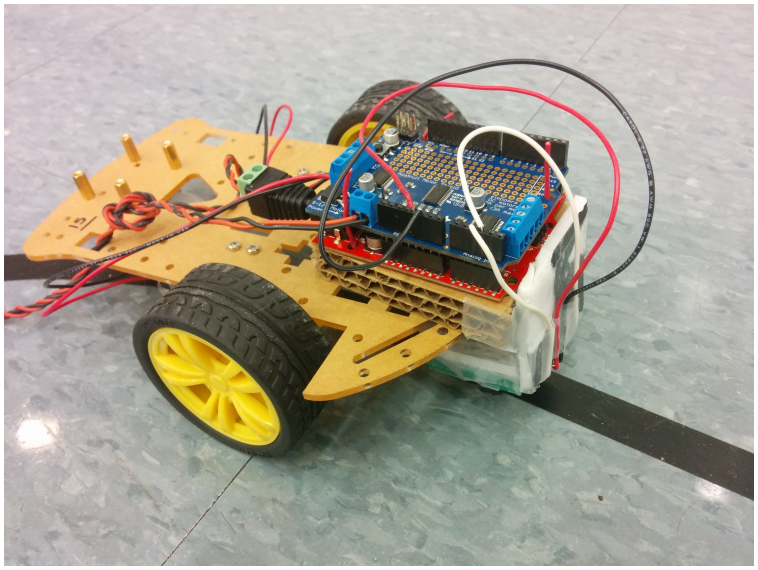

Figure 2: Photo of our line following robot

Figure 2 shows a photo of our line following robot. This chassis mount was ultimately constructed out of cardboard, foam board, wires, scotch tape, and hot glue for fastest possible construction. The mount is attached to the chassis with clamping action. The top of the mount has three wires which bend through and around the Arduino holes. The bottom of the mount holds our breadboard and IR sensors.

Circuit Diagram

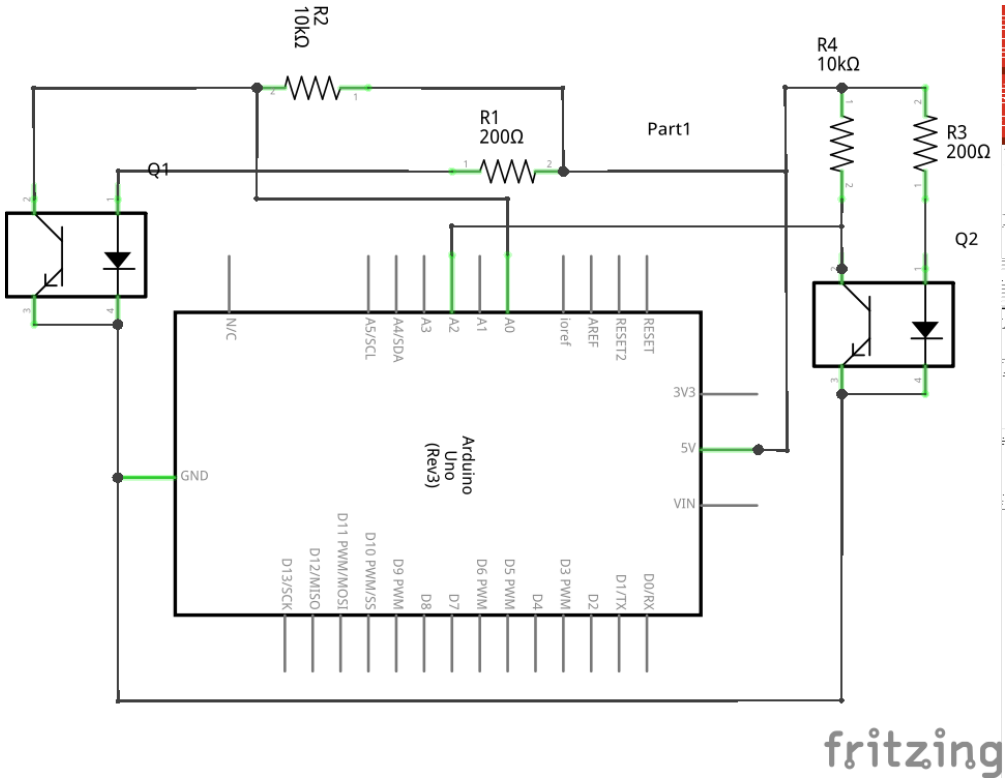

Figure 3: Schematic of the dual IR sensor and Arduino circuit

Figure 3 shows our circuit diagram used in the line following robot. Notice that both IR sensors contain identical 200Ω and a $10k\Omega$ resistors allowing us to use the an identical threshold value (the number that decides what motor is on and off) for each sensor and motor.

Motors and Sensors

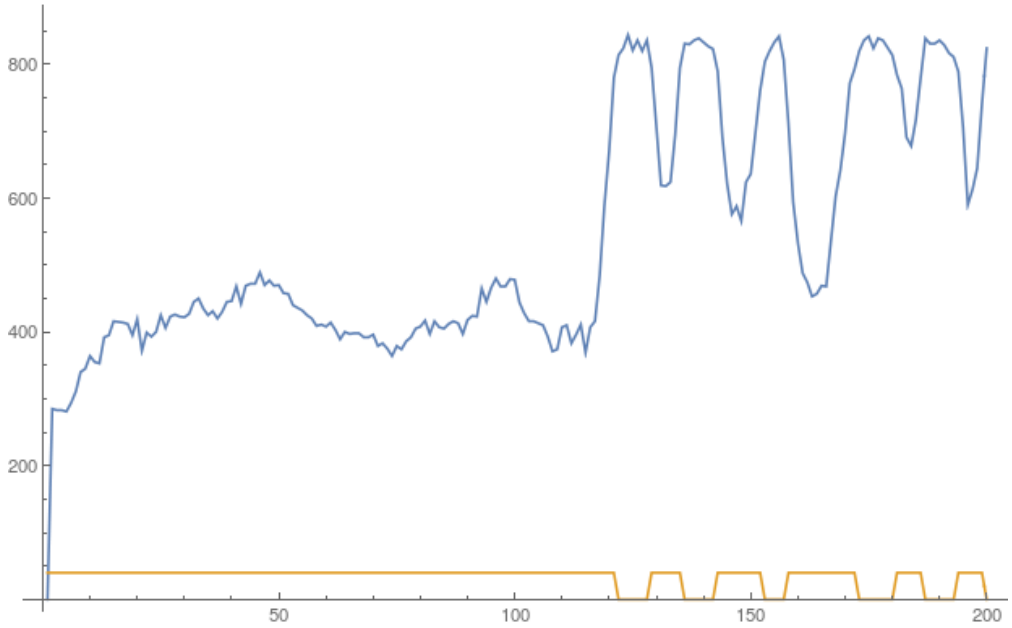

Figure 4: Plot of the left motor and the right sensor. Blue line is sensor, yellow is motor speed.

Figure 4 shows the plot of the left motor and the right sensor. We see that the left motor is only on when it is above the sensor is above 800.

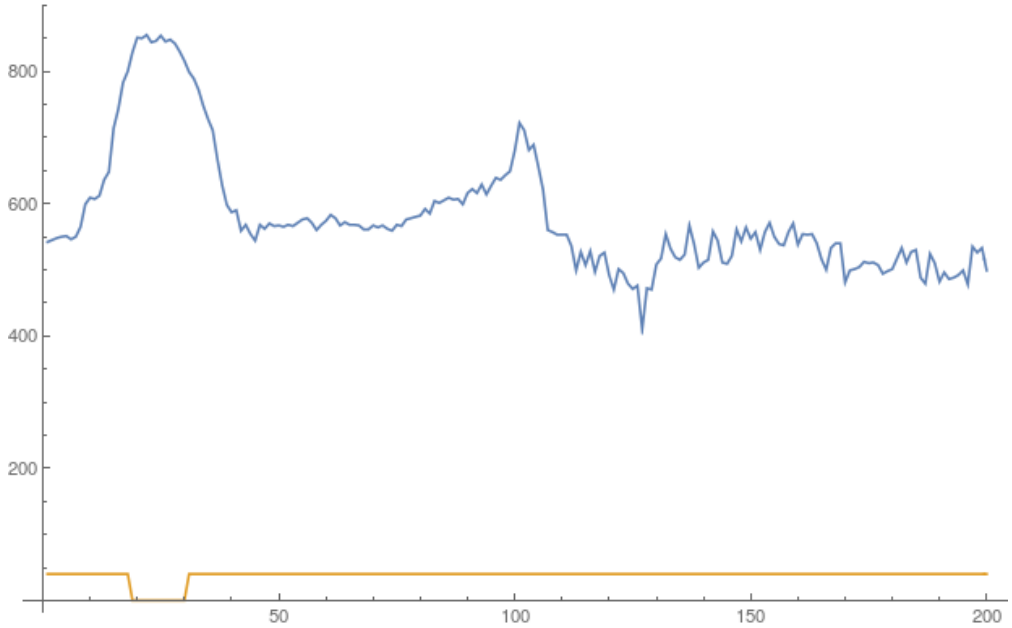

Figure 5: Plot of the right motor and the left sensor. Blue line is sensor, yellow is motor speed.

Figure 5 shows the plot of the right motor and the left sensor. We see that the right motor is only on when it is above the sensor is above 800.

Reflection

One of the interesting things about this lab is the order in which things were done. In particular, making something work and real life, and then documenting it. For example, we built the circuit, and then made a circuit diagram. We also made the chassis mount, and then CAD'ed and rendered it. In the future, I would like to be able to use computer aided design / documentation to assist me in creating their corresponding real life components.

Another big takeaway from this lab is to make a "what is wrong checklist". In other words, we spent quite a bit of time troubleshooting our system. The most time consuming problems were particular Arduino pins and order of powering robot. Both of these now need to be documented so we avoid them in the future.

Appendix

Source Code

```
1  // POE Lab 3 - A Line Following Robot
2
3  #include <Wire.h>
4  #include <Adafruit_MotorShield.h>
5  #include "utility/Adafruit_MS_PWMServoDriver.h"
6
7  #define leftSensor A2 // Left IR sensor
8  #define rightSensor A3 // Right IR sensor
9
10 byte input[256]; // Serial input buffer
11 volatile int leftValue = 0; // Left IR Value
12 volatile int rightValue = 0; // Right IR Value
13
14 Adafruit_MotorShield ms = Adafruit_MotorShield();
15 Adafruit_DCMotor *m1 = ms.getMotor(1); // leftMotor
16 Adafruit_DCMotor *m2 = ms.getMotor(2); // rightMotor
17 volatile int mainSpeed = 40; // The main speed for the motors
18 volatile int diffSpeed = 40; // The difference in speed based on the
       irThreshhold
19 volatile int irThreshhold = 800; // The threshhold for the floor vs
      the black tape
20
21 void setup(){
22   ms.begin(); // Begins the motorshield
23   m1->setSpeed(mainSpeed); // Sets the main speed for both of the
       motors
24   m2->setSpeed(mainSpeed);
25   m1->run(FORWARD); // Sets both of the motors to go forward
26   m2->run(FORWARD);
27   Serial.begin(9600); // Begins serial communication
28 }
```

```
29
30 void loop() {
31   // read irSensor values
32   irSensorRead();
33
34   if(leftValue > irThreshhold){
35     // If the left sensor hits the tape, turn off the right motor
36     m1->setSpeed(mainSpeed);
37     m2->setSpeed(mainSpeed - diffSpeed);
38     Serial.println(mainSpeed);
39     Serial.println(mainSpeed - diffSpeed);
40   }
41   else if(rightValue > irThreshhold){
42     // If the right sensor hits the tape, turn off the left motor
43     m1->setSpeed(mainSpeed - diffSpeed);
44     m2->setSpeed(mainSpeed);
45     Serial.println(mainSpeed - diffSpeed);
46     Serial.println(mainSpeed);
47   }
48   else{
49     // Charge forward!
50     m1->setSpeed(mainSpeed);
51     m2->setSpeed(mainSpeed);
52     Serial.println(mainSpeed);
53     Serial.println(mainSpeed);
54   }
55
56 }
57
58 void irSensorRead()
59 {
60   /*
61    * irSensorRead reads the sensor 5 times and returns the average
62    */
63
64   // Reset the values
65   leftValue = 0;
66   rightValue = 0;
67
68   // read from the irSensors 5 times
69   for (int i = 0; i < 5; i++)
70   {
71     leftValue += analogRead(leftSensor);
72     rightValue += analogRead(rightSensor);
73   }
74
75   // divide the value of irValue by 5 to find the average
76   leftValue = leftValue/5;
```

```
77    rightValue = rightValue/5;
78
79    Serial.println(leftValue);
80    Serial.println(rightValue);
81  }
82
83  void serialEvent(){
84    /*
85     * This function reads bytes from serial at the end of each
86     * loop if there are bytes to read
87     */
88    Serial.readBytes(input, 4); // read 4 bytes from serial
89    mainSpeed = int(input[0]); // The values for the left motor should
          be between 0 - 255 (size of byte)
90    diffSpeed = int(input[1]); // The values for the right motor
         should be between 0 - 255
91    irThreshhold = int(input[2]*256+input[3]); // The values for the
         threshhold can be between 0 - 1023 so there need to be 2 bytes
92  }
```