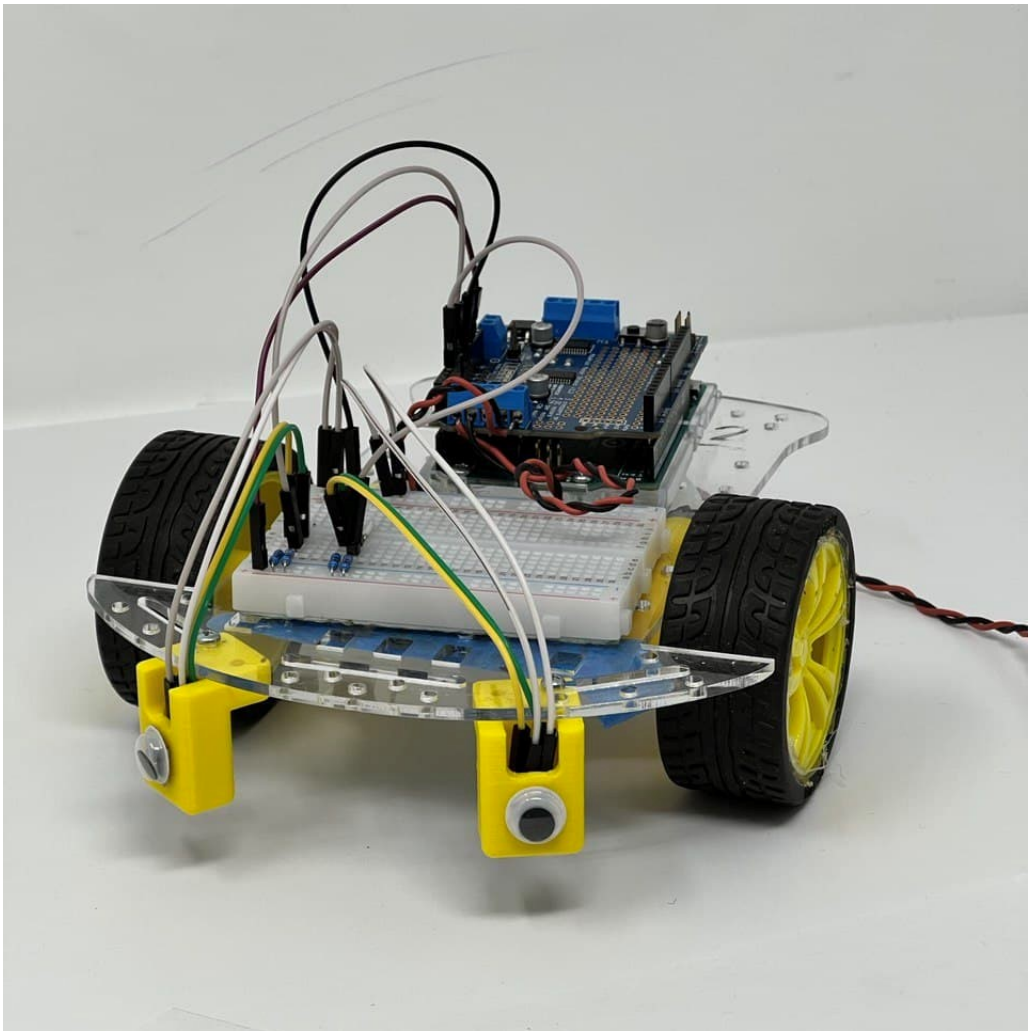
PIE Mini Project 3

Rishita Sarin, Prisha Sadhwani, S.C. 'Mack' McAneney – Section 1

October 16, 2021



1 Introduction

The goal of Mini-Project 3 was to design a robot that would follow the path of a line of tape. This robot was to be driven by DC motors, and maintain its position on the path based on input from IR reflective sensors.
An IR sensor works by emitting IR light and receiving a portion of that light reflected back. Due to the difference in reflectance of the black electrical tape and the shiny blue linoleum of the PIE room floor, we can determine whether a sensor is positioned above the floor or the tape. By positioning two of these sensors on a steerable robot, we can keep the line between the two sensors by steering the robot in response to sensor input. When a sensor detects the robot about to drive over the tape, it runs the wheels in opposite directions to turn the robot left or right, depending on which sensor detects the tape.

2 Procedure

2.1 Circuit

We assembled the following circuit, as shown below in our schematic, on a solderless breadboard.

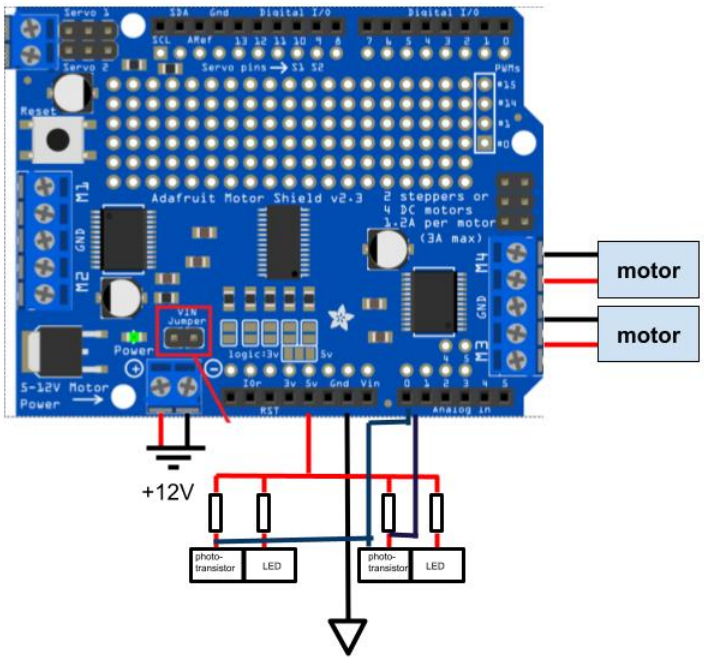


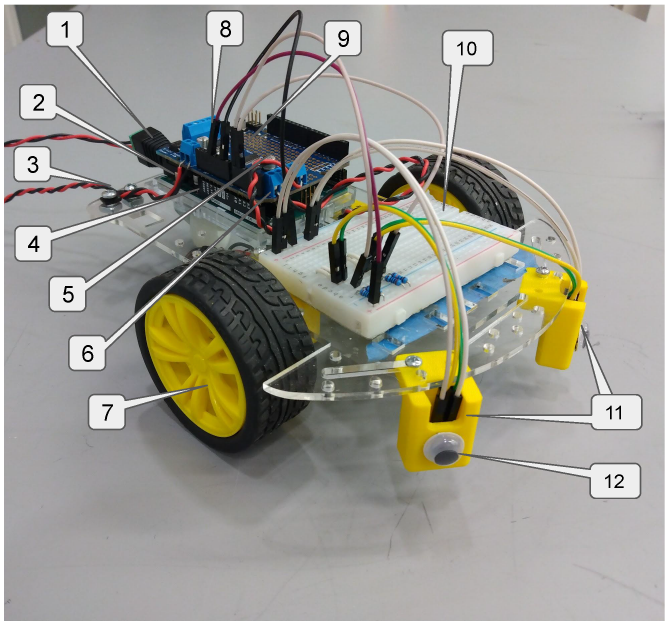
Figure 1: Circuit Diagram

The main components of this setup were the two servos connected to digital inputs on the Arduino, and the two IR sensors connected to analog inputs on the motor shield. We connected the breadboard to the motorshield, and the motorshield to the Arduino, using the 5V power and ground cables, and we made sure that current was flowing to our individual components.

The phototransistor and LED required two different resistor values - the LED resistor value was provided to us (200 Ω), but we had to calculate the resistor value for the phototransistor, which we did by using $V = IR$, with V being 12V and I being 1mA, to get a general value. It's important to note that from there, we had to use a purely experimental method of choosing our resistors, since the motors didn't operate at exactly 12V. As a result of this experimental method, we ended up choosing a 10K Ω resistor.

2.2 Mechanical Components

While the robot was primarily assembled from the provided kit of the acrylic mounting plate, wheels, and DC motors, there were some elements which needed to be designed or created. One design choice we made was to position the two motor-driven wheels as the front of the robot (while the caster wheel was at the back). This allowed us to have the IR sensors above the wheels, so wheel adjustments made by the motors in response to an error signal from the IR sensors would be done roughly above where the error was detected. This would ideally prevent corrections from moving the sensors off the path.



Components
1. Arduino power source
2. Motor power source
3. Cable tension relief anchor
4. Arduino Uno
5. AdaFruit motor shield
6. DC motor signal output
7. DC motor driven wheel
8. Sensor power and ground
9. Sensor signal inputs
10. Breadboard
11. IR sensor in sensor holder
12. Googly eye

Figure 2: Annotated photo showing major components of the robot

One thing that needed to be designed for the circuit was a way to attach the IR sensor eyes in a spot where they could effectively read the tape on the floor. A component was designed and then 3D printed to snugly hold each eye approximately a centimeter above the ground.

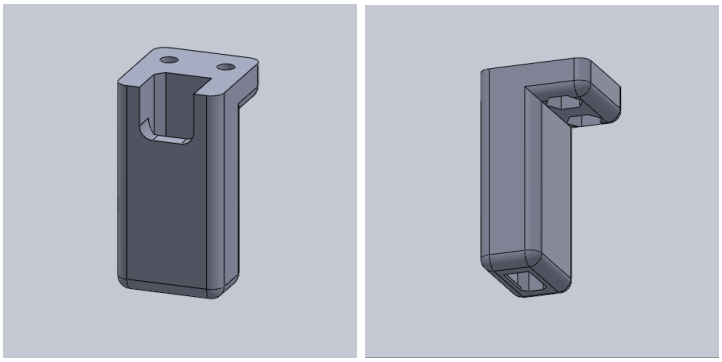


Figure 3: Two views of the IR sensor holder component

One other thing that needed to be considered was tension relief in the power cable. The robot needed an external source of power, meaning that it had to have a wire connecting it to a power source. To prevent the plug from coming disconnected as the robot maneuvers, the cable was clamped down with a piece of rubber that would stop tugging on the plug end.

2.3 Calibration

After constructing the circuit and assembling the robot, we needed to calibrate the sensors to detect when the robot was on or off the tape. The calibration was simply done by placing each sensor over the tape, noting the value, then doing the same with the sensor halfway over the tape and over the floor. We used these values to determine our thresholds for turning the wheels. The selected sensor values were as follows:

Position of sensor	Left IR Sensor	Right IR Sensor
On Tape Completely	498	500
Halfway On Tape and On Floor	204	197
On Floor Completely	52	50

2.4 Code

As described above in the "Mechanical Components" section, the robot was constructed in such a way that the sensors were on either side of the tape while driving. The signal output by these sensors varies based on light reflected back to it, creating distinct signals between the shiny floor and the black tape path. We started out by setting our threshold values (200), which meant that the robot was half on the tape and half off, and our low values (50), which meant that the robot was fully off the tape.

The sensor output was passed to a set of if statements which would determine the position of the robot relative to the line using the calibration values we collected in the previous section. If both sensors detected values below the threshold value of 200, the line was between the sensors, and both motors were set to equal forward speeds to drive straight.

If the left sensor output a value between the threshold and low values, this indicated that the robot was about to drive off the left side of the line. To correct this, the robot was turned right by turning the left wheel forward and the right wheel backward. It continued turning until the sensors were both below the threshold value and the robot could continue straight along the path. The same process was mirrored to turn the robot left when it started driving to the left of the path.

Under the circumstance that both the sensors were off the tape (only values under low were detected), the robot was programmed to spin slowly in a circle, with the hope that it would be able to catch itself before going too far off the tape. We never hit the stage where we needed to implement that code, but we included it anyway, as an edge case.

We also included a way to change the speed of the motors via the Arduino Serial Port by first checking to see if the port is available. If it is, we type a number from 0-255 into the command prompt, and after pressing send, that number becomes the new motor speed. We implemented this so that the speed of the motors could be changed without needed to re-upload the code every time we wanted to make a change.

2.4.1 Plot

We ran our robot on a shorter path to generate a plot of IR sensor data and motor values, as shown below. The motor values shift from +35 to -35, since those are the only two speeds our motors ran at, and the big clusters of IR values were either below 50, when the robot was driving straight, or somewhere between the low and threshold values, when the robot was turning.

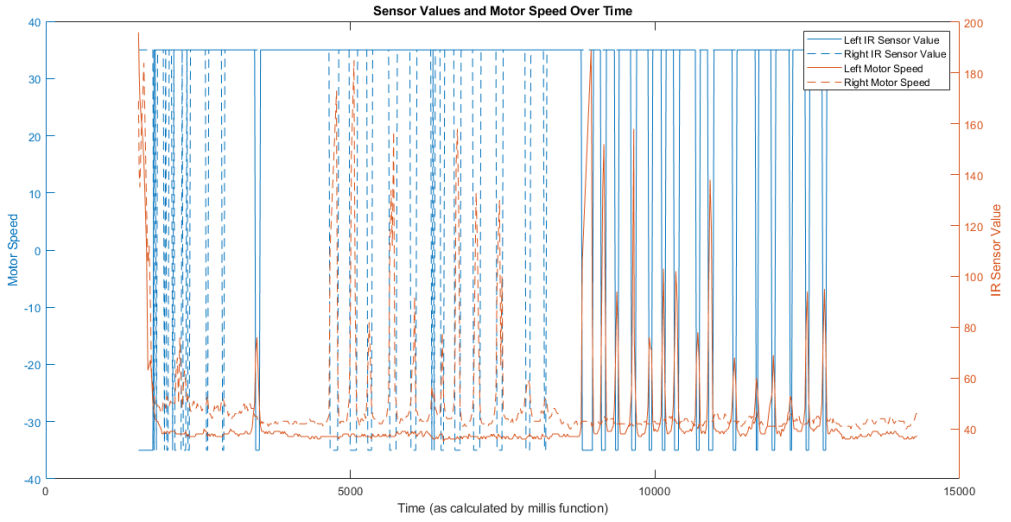


Figure 4: Superimposed Plot

3 Troubleshooting

One thing that we had some issues with was getting the robot to navigate the sharp 90 degree turns on the path, which we were able to work around by positioning the sensors farther apart, giving us a wider range for correction before driving off the path. This solution was generally a better choice, as this margin for error meant the robot spent more time driving straight, which is faster than turning and meant that the laps were done faster than with the first sensor configuration.

4 Conclusion

Overall, this project was a success. We managed to tune our robot to navigate the path in less than one minute. One area of expansion for this project would be to implement PID controlling in our code, which could further speed up the time our robot needs to complete a lap. A future iteration of the project could also adjust speed of the motors during turns, rather than simply turning the wheels, which is what we ended up doing. This would also help with speed optimization in general, which would be another area to improve.

5 Reflection

We were a really excited to start this project - our team consisted of an ECE, an E:C, and an ME, so we genuinely felt prepared to tackle this project. It also really helped that we had completely different interests going in, from mechanical design to motion control, so we were able to work while also keeping everyone happy. We learned how to iterate on our code, and genuinely improved our ability to debug (with a mixture of pair programming, explaining code to each other, and going to a lot of CA hours). Once we progressed along the project and had our initial code going, we were able to have fun with the different programming methods. Overall, we had a great, valuable time with this lab and are really excited to tackle the final project!

6 Appendix

Link to repository containing code, images, charts, and video of the robot completing the track: https://github.com/Prisha247/DC-Motor-Control

Arduino Movement Code:

```
//including the libraries
//Include requisite libraries
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"

//defining pins and variables

Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_DCMotor *motor1 = AFMS.getMotor(3);
Adafruit_DCMotor *motor2 = AFMS.getMotor(4);

#define lefts A1
#define rights A0

//motor speeds
int leftMotorSpeed = 35;
int rightMotorSpeed = 35;
int run_time;

//The point at which the sensor knows it is LEAVING the line
int leftSensorThreshold = 200;
int rightSensorThreshold = 200;

//The point at which the sensor knows it HAS LEFT the line
int leftSensorLow = 50;
int rightSensorLow = 50;

void setup() {
  //setting the speed of motors
  AFMS.begin();
  motor1->setSpeed(leftMotorSpeed);
  motor2->setSpeed(rightMotorSpeed);
  //declaring pin types
  pinMode(lefts,INPUT);
  pinMode(rights,INPUT);

  //begin serial communication
  Serial.begin(9600);
}

void loop(){
```

```
43
44  // SERIAL PROTOCOL:
45  // Format -> [num_value];
46  // Serial input changes the speed of the motors
47  run_time = millis();
48
49  //  if (Serial.available()>0)
50  //    {
51  //      int speed = Serial.parseInt();
52  //      Serial.println(speed);
53  //      if (speed >= 0 && speed <= 255)
54  //      {
55  //        motor1->setSpeed(speed);
56  //        motor2->setSpeed(speed);
57  //      }
58  //    }
59
60
61     //Read IR sensors
62    int l = analogRead(lefts);
63    int r = analogRead(rights);
64
65    //line detected by both
66  //  if(l > leftSensorThreshold && r > rightSensorThreshold){
67  //    //spin
68  //    motor1->run(FORWARD);
69  //    motor2->run(FORWARD);
70  //  //printing values of the sensors to the serial monitor
71  //    Serial.print(analogRead(lefts));
72  //    Serial.print(",");
73  //    Serial.print(analogRead(rights));
74  //    Serial.print(leftMotorSpeed);
75  //    Serial.print(",");
76  //    Serial.print(rightMotorSpeed);
77  //    Serial.print(",");
78  //    Serial.println(run_time);
79  //  }
80
81    //line detected by left sensor
82    if(r < rightSensorThreshold && l > leftSensorLow && l < leftSensorThreshold){
83      //turn left
84      motor1->run(BACKWARD);
85      motor2->run(FORWARD);
86    //printing values of the sensors to the serial monitor
87      Serial.print(analogRead(lefts));
88      Serial.print(",");
89      Serial.print(analogRead(rights));
90      Serial.print(",");
91      Serial.print(-1*leftMotorSpeed);
92      Serial.print(",");
93      Serial.print(rightMotorSpeed);
```

```
94      Serial.print(",");
95      Serial.println(run_time);
96    }
97
98    //line detected by right sensor
99    else if(l < leftSensorThreshold && r > rightSensorLow && r < rightSensorThreshold){
100     //turn right
101     motor1->run(FORWARD);
102     motor2->run(BACKWARD);
103   //printing values of the sensors to the serial monitor
104     Serial.print(analogRead(lefts));
105     Serial.print(",");
106     Serial.print(analogRead(rights));
107     Serial.print(",");
108     Serial.print(leftMotorSpeed);
109     Serial.print(",");
110     Serial.print(-1*rightMotorSpeed);
111     Serial.print(",");
112     Serial.println(run_time);
113   }
114
115   //line detected by none
116   else if(l < leftSensorThreshold && r < rightSensorThreshold){
117     //stop
118     motor1->run(FORWARD);
119     motor2->run(FORWARD);
120   //printing values of the sensors to the serial monitor
121     Serial.print(analogRead(lefts));
122     Serial.print(",");
123     Serial.print(analogRead(rights));
124     Serial.print(",");
125     Serial.print(leftMotorSpeed);
126     Serial.print(",");
127     Serial.print(rightMotorSpeed);
128     Serial.print(",");
129     Serial.println(run_time);
130   }
131 }
```

Matlab Plotting Code:

```
Array=csvread('raw_data.csv');
l_ir = Array(:, 1); %left IR sensor
r_ir = Array(:, 2); %right IR sensor
l_motorspeed = Array(:, 3); %left motor speed
r_motorspeed = Array(:, 4); %right motor speed
time = Array(:, 5); %time


yyaxis left
plot(time, l_motorspeed)
hold on;
plot(time, r_motorspeed)
title('Sensor Values and Motor Speed Over Time')
xlabel('Time (as calculated by millis function)')
ylabel('Motor Speed')

yyaxis right
hold on;
plot(time, l_ir)
hold on;
plot(time, r_ir)
ylabel('IR Sensor Value')
legend('Left IR Sensor Value', 'Right IR Sensor Value', 'Left Motor Speed', 'Right Motor Speed')

hold off
```