MP3: Line Following Robot

Cory Knox, Mari Kang, Efe Gulcu

October 2021; PIE F2021

1 Introduction

With mini project 3, there are many different objectives that we are focused on. Part of our goals is to integrate sense and control systems with a two-wheeled robotic platform. This project heavily integrates mechanical skills in assembly and electrical/coding aspects in incorporating IR sensors along with a serial system.

Our coding focus is to write a closed-loop controller that runs the Arduino, enabling the robot to follow the black line track made of electrical tape on the floor. From there we are focused on tuning our controller and robot to complete a full loop around an elaborate tape circle. Also, we use serial input to change the speed of the motor without recompiling and reloading the code.

2 Methodology

2.1 Exploration of the IR Reflectance Sensors

We start with exploring the IR sensors that we will use to sense the black line for the robot. To do this a basic circuit is made and tests different material surfaces and distances with the IR sensors. We also test different resistors and the effects they have on the range and sensitivity of the sensors. From here, we realize the greater the resistance, the greater the analog reading we get from our voltage divider. Because a high resistance valuye that gives high reading values while not being too high is needed to create meaningful data, a 49.9k resistor is used. When a very high resistance around 400k is used, the surfaces that we were sensing show no differences. This is critical element of our design, as it tells us:

1. Where the placement of the IR sensors should be in reference to the distance up off of the ground.

2. The sensitivity of the sensors that will be used. If the sensors are not sensitive enough, we won't be able to distinguish the light and dark surfaces. If they are too sensitive, they may mistake non-dark surfaces for potentially being dark surfaces thus misleading our robot.

3. The threshold that we then use in our code to stop and start the motors. If the IR sensor spots a black surface, the wheel it controls is instructed to cease, allowing the other wheel that is still in motion to rotate the body accordingly, staying along the black line. This is only possible if we understand the difference in light vs dark readings of the IR to then decide an appropriate stop-start threshold.

2.2 Electrical Components

The circuit has six main components; two IR sensors (as mentioned above), two gear motors, one Arduino and one Adafruit v2.3 Motor Shield. The IR reflectance sensors are used to detect which surface the robot is currently on by creating a voltage divider for each of them. The gear motors are connected to the motor shield and are in charge of the motion of the robot. The motor shield then connects the Arduino and gear motors and lets us use two gear motors at the same time with the intended behavior. Finally the Arduino governs the whole system by collecting analog data from IR sensors and communicating with the motor shield so that the gear motors are behaving as intended. The circuit diagram is given in figure 1.

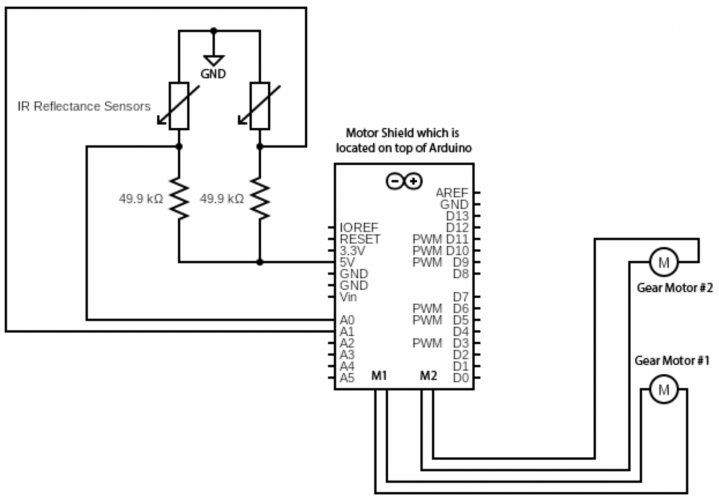

Figure 1: **Circuit diagram of our electrical components. Motor Shield and Arduino are powered by a 12V power supply.**

2.3 CAD and Mechanical Components

For the mechanical aspects of our rig we incorporate two specific pieces of mechanical design. One is the attachments to the main rig that would extend the IR sensors outward, reaching past the front wheel. Our team explicitly set a goal for ourselves to keep all 3 wheels as close to the line as possible. To do this, we make sure the system is taking in information about the line's position before the front wheel reached that point.

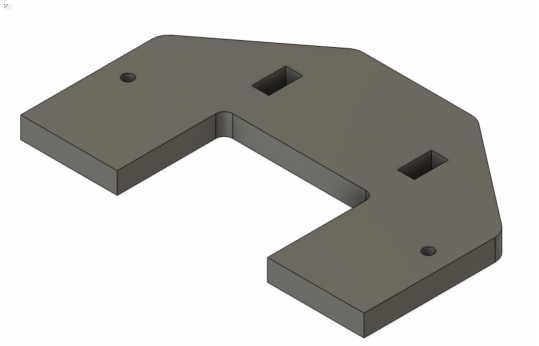

Figure 2: **CAD for front rig assembly**

Custom adaptors that help fasten the Arduino board to the rig are also incorporated in our design. It is important to maintain an organized setup, especially with all the wires used in this project. If the wires were to come loose or have little to nothing holding them down, they would easily become tangled in the motors. To solve this, the L-shaped adaptors lift the wires and protoboard, keeping them on the proper face of the robot. They reach over the protoboard board, pulling it down onto the rig, and the adaptors are screwed into position below the rig to prevent wobbling.

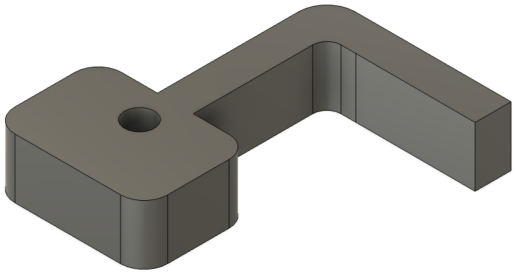

Figure 3: **CAD for hinge-like adaptors that hold Arduino board to the bot's mainframe.**

All pieces designed in this project are laser cut from MDF board. The motor

shield-arduino piece fit smoothly in already present indents of the rig, so we did not have to make any adaptors for that. The final rig can be seen below in figure 4.

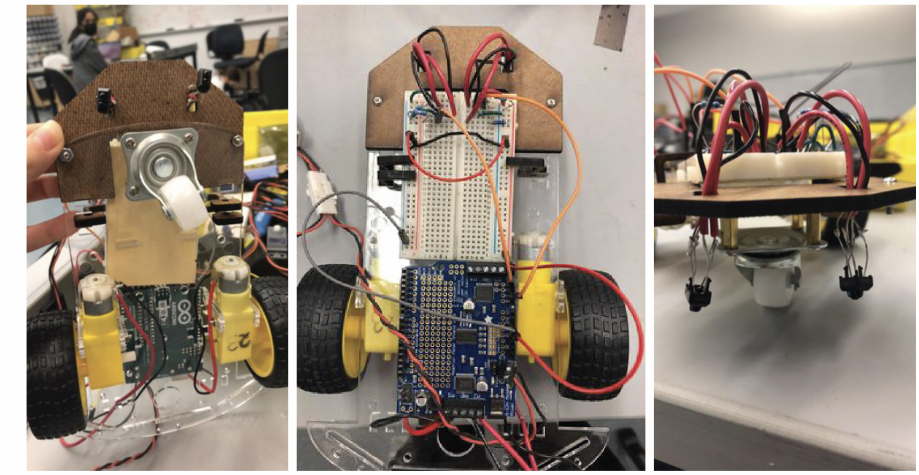

Figure 4: **Final assembly of the robot**

2.4 Software Components

We use two IR sensors to detect the color of the ground. Instead of detecting the black line, we put the two sensor on the left and right side of the robot and go straight if they both don't detect the black line. The IR voltage value increases when it detects the black line, so if the two sensor values are below the threshold of the black detection, it goes forward. Below is the code chunk that moves the robot forward if both sensor doesn't detect the black line.

```
1   //if both sensor doesn't detect black line, go forward
2   if(leftreading < leftthreshold && rightreading < rightthreshold){
3
4     rightMotor ->setSpeed(vspeed);
5     leftMotor ->setSpeed(vspeed);
6
7     leftMotor ->run(FORWARD);
8     rightMotor -> run(FORWARD);
9     Serial.print(vspeed);
10    Serial.print(" ");
11    Serial.print(vspeed);
```

```
12      Serial.print(" ");
13    }
```

If the right sensor only detects the black line, the right sensor value will be higher than the threshold but the left sensor value will be lower than the threshold. The robot should turn right, so we are moving the left motor forward and stopping the right motor. Line 8 in the code below is the line that stops the motor.

```
1   //if right sensor detect black line, turn right (move left motor)
2   if(leftreading < leftthreshold && rightreading >= rightthreshold){
3
4     rightMotor ->setSpeed(vspeed);
5     leftMotor ->setSpeed(vspeed);
6
7     leftMotor ->run(FORWARD);
8     rightMotor -> run(RELEASE);
9
10    Serial.print(vspeed);
11    Serial.print(" ");
12    Serial.print(0);
13    Serial.print(" ");
14
15  }
```

For the threshold, we set the two threshold differently because the value of the two sensors are different despite the same resistors we are using. We choose the two values based on the calibration.

```
1   //Set the threshold of left and right IR sensor value
2   uint16_t leftthreshold = 800;
3   uint16_t rightthreshold = 600;
```

We include the feature of updating the behavior of the control code using serial connection without recompiling the code when it's connected to the computer. Line 2 in the code below checks if there's a new serial input. If there's a new input, get the value and change the speed of the motor if the value is not 0.

```
1   //Get the serial input and set as the new speed
2   if (Serial.available() > 0) {
3     int new_speed = Serial.parseInt();
```

```
4       if (new_speed != 0){
5         vspeed = new_speed;
6       }
7     }
```

3 Results

3.1 Video of the Robot

Here's the link to the video of our robot successfully completing the track:
Link to the Video[Click Here]

3.2 Video of the Serial Input Controller

We also use serial input to tune the control loop and motor speed settings without recompiling the code if the robot is connected to the computer. Here's the link to the video of the change in motor speed based on the serial input we put:
Link to the Video[Click Here]

As shown in the video, the motor speed changes based on the input the user gives without recompiling the file.

3.3 Plot of superimposing sensor data

The plot below shows the IR sensor value and the corresponding motor speed. The threshold for left IR output value is 800 and the threshold for right IR output values is 600. In the graph, when the blue line (IR Left) is over threshold (800) and the yellow line (IR Right) is lower than the threshold(600), the green line (left motor) decreases to zero and the red line (right motor) stays in speed of 25. This makes sense because the left sensor detected the black line and should turn left, so only the right motor should move. And when the yellow line goes higher than the threshold and the blue line goes below the threshold, it should turn right and the left motor should be moving and the right motor should stop, which is also shown in the graph.

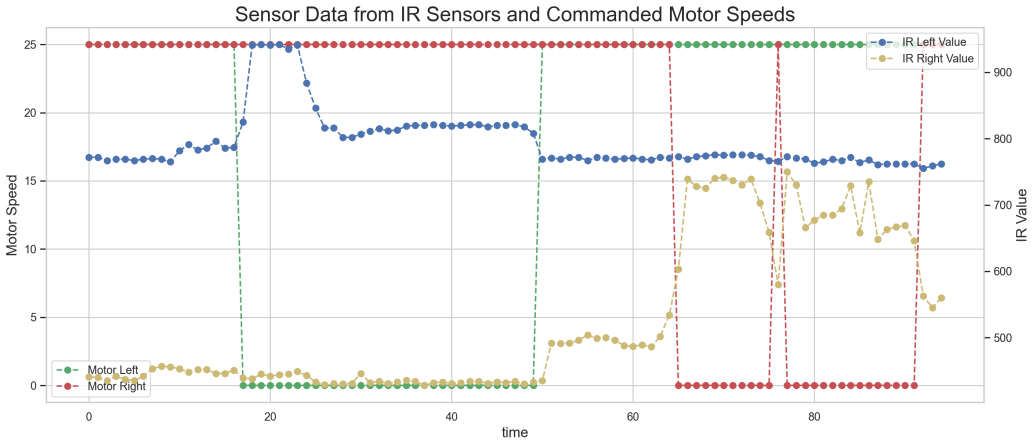

Figure 5: **Superimposing data of IR Sensor Output and Motor Speed**

4 Reflection

4.1 Teaming

This project was an opportunity for us to explore individual skills that we either are already very strong in to then try new things or to push ourselves in areas that we aren't the most familiar with. Our team tested our skills in 3D modeling and manufacturing, different circuitry and incorporation of a sensor none of us were comfortable with, and code that pushed previously known knowledge. Generally our team worked very well together, all able to explore and contribute to the teaming goal. We communicated well and made sure we were all on the same page about commitment to the project. It felt as though our team was consistently on the same page for both project ideas and the time we wanted to spend on this.

4.2 Technical Parts

If we were to go back and redo the system, there were a few points that we saw for potential improvement. For one our mechanical system and modeling took a couple of tries to create our adaptors via laser cutting. After seeing many of our classmates adaptations of the line reading robot, we would want to go back and try swapping the front and the back of the system. While we led with the singular, non-motorized wheel, many led with the two motorized wheels as the front. This appeared to leave a lot more workable space for fasteners and would have required less CAD work. It also made the driving of the robot more stable with less wiggling, while the non-motorized wheel caused rotation and we had

to maintain a constant wheel balance from the motors to keep the robot from wandering.

Another mechanical aspect of this project that we would want to improve is incorporating an adaptor that would hold the IR sensors at constant and equal heights. We did not incorporate this in our first iterations of the robot. This created calibration difficulties for us, as we would constantly have to go back and check the IR sensors to readjust the threshold used. Incorporating an IR height adaptor would have made this a lot easier.

While the electrical part of the project was straightforward, it was a great experience to learn that the circuits should be as tidy as possible. We encountered several problems because the breadboard itself was too crowded and wires were tangled. We even fried one Arduino in the process because we had some disconnected grounds and stranded wires but we were unable to detect them because the circuit was visually too confusing. Prototyping the circuit from the start would make the process more safe and easy.

The software part of the project had several trials and errors. One of the biggest challenges was that the value of the IR output kept changing depending on the location of the sensor. We had to set a threshold for left and right IR sensor output detecting black or not, but the value varied every time we tested due to the distance from the ground or the distance between two sensors. This could have been avoided if we created a stable design of the IR sensors so that the values are consistent.

If we had more time, we would have worked on the experiment on the speed of the motor more thoroughly to make it go faster. Currently, the motor is moving at a maximum speed of 25 in order to move along the loop. Increasing the speed of the motor when turning left or right for a sharp turn might enable our robot to make a full loop faster.

Generally, our team was hardworking and efficient with clear communication. We were all able to test new skills and help each other improve. There was a lot of teaching amongst the team and we are excited to work on the final project together.

5 Appendix

5.1 Code for Line Following Robot

```
#include <Adafruit_MotorShield.h>

// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
```

```
// Select the ports for both motors (M1 and M2)
Adafruit_DCMotor *leftMotor = AFMS.getMotor(1);
Adafruit_DCMotor *rightMotor = AFMS.getMotor(2);

int vspeed = 25;
int leftreading;
int rightreading;

int IrLeft = A3;
int IrRight = A1;

//Set the threshold of left and right IR sensor value
uint16_t leftthreshold = 800;
uint16_t rightthreshold = 600;


void setup() {

  Serial.begin(9600);

  AFMS.begin();
  leftMotor ->setSpeed(vspeed);
  leftMotor ->run(FORWARD);
  rightMotor ->setSpeed(vspeed);
  rightMotor ->run(FORWARD);

  // put your setup code here, to run once:
  pinMode(IrLeft, INPUT);
  pinMode(IrRight, INPUT);

   // turn on motor
  leftMotor->run(RELEASE);
  rightMotor ->run(RELEASE);


}

void loop() {

  //Get the serial input and set as the new speed
  if (Serial.available() > 0) {
    int new_speed = Serial.parseInt();
    if (new_speed != 0){
      vspeed = new_speed;
```

```
  }
}


leftreading = analogRead(IrLeft);
rightreading = analogRead(IrRight);


Serial.print(leftreading);
Serial.print(" ");

Serial.print(rightreading);
Serial.print(" ");

//if both sensor doesn't detect black line, go forward
if(leftreading < leftthreshold && rightreading < rightthreshold){

  rightMotor ->setSpeed(vspeed);
  leftMotor ->setSpeed(vspeed);

  leftMotor ->run(FORWARD);
  rightMotor -> run(FORWARD);
  Serial.print(vspeed);
  Serial.print(" ");
  Serial.print(vspeed);
  Serial.print(" ");
}

//if both sensor detect black line, still go forward
if(leftreading >= leftthreshold && rightreading >= rightthreshold){

  rightMotor ->setSpeed(vspeed);
  leftMotor ->setSpeed(vspeed);
  leftMotor ->run(FORWARD);
  rightMotor -> run(FORWARD);

  Serial.print(vspeed);
  Serial.print(" ");
  Serial.print(vspeed);
  Serial.print(" ");
}

//if right sensor detect black line, turn right (move left motor)
if(leftreading < leftthreshold && rightreading >= rightthreshold){

  rightMotor ->setSpeed(vspeed);
```

```
    leftMotor ->setSpeed(vspeed);

    leftMotor ->run(FORWARD);
    rightMotor -> run(RELEASE);

    Serial.print(vspeed);
    Serial.print(" ");
    Serial.print(0);
    Serial.print(" ");

  }

  //if left sensor detect black line, turn left (move right motor)
  if(leftreading >= leftthreshold && rightreading < rightthreshold){

    rightMotor ->setSpeed(vspeed);
    leftMotor ->setSpeed(vspeed);

    leftMotor ->run(RELEASE);
    rightMotor -> run(FORWARD);

    Serial.print(0);
    Serial.print(" ");
    Serial.print(vspeed);

  }
  Serial.println();
  delay(500);

}
```

5.2 Code for Collecting Data

```
import serial
import csv
import time

arduino_com_port = "/dev/cu.usbmodem142301"

baud_rate = 9600

serial_port = serial.Serial(arduino_com_port, baud_rate, timeout=1)

with open("line_follower_data.csv", "w") as outfile:
    outfile.write("IR_left,IR_right, Motor_left, Motor_right\n")
```

```
while True:
    time.sleep(1)
    data = serial_port.readline().decode()

    file = open("line_follower_data.csv")
    reader = csv.reader(file)
    lines= len(list(reader))

    if len(data):
        ir_left, ir_right, motor_left, motor_right = [float(x) for x in data.split()]


    with open("line_follower_data.csv", "a") as outfile:
        outfile.write(f"{ir_left},{ir_right},{motor_left}, {motor_right}\n")
```

5.3 Code for Plotting the Data

```
import pandas as pd
import csv

import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")


time = []
ir_left = []
motor_left = []

ir_right = []
motor_right = []

#Get the data from the csv
with open('line_follower_data.csv','r') as csvfile:
    lines = csv.reader(csvfile, delimiter=',')
    headings = next(lines)
    timestamp = 0
    for row in lines:
        time.append(timestamp)
        ir_left.append(float(row[0]))
        ir_right.append(float(row[1]))
        motor_left.append(float(row[2]))
        motor_right.append(float(row[3]))
        timestamp+=1
```

```
# create figure and axis objects with subplots()
fig,ax = plt.subplots()
# make a plot

ax.plot(time, motor_left, color="g", linestyle = 'dashed', marker = 'o', label="Motor Left")
ax.plot(time, motor_right, color="r", linestyle = 'dashed', marker = 'o', label="Motor Right")

# set x-axis label
ax.set_xlabel("time",fontsize=14)
# set y-axis label
ax.set_ylabel("Motor Speed",fontsize=14)

# twin object for two different y-axis on the sample plot
ax2=ax.twinx()

# make a plot with different y-axis using second axis object
ax2.plot(time, ir_left,color="b", linestyle = 'dashed', marker = 'o',label = "IR Left Value")
ax2.plot(time, ir_right,color="y", linestyle = 'dashed', marker = 'o',label = "IR Right Value")
ax2.set_ylabel("IR Value",fontsize=14)

leg = ax.legend()

plt.xticks(rotation = 90)
plt.title('Sensor Data from IR Sensors and Commanded Motor Speeds', fontsize = 20)
plt.grid()
plt.legend()
plt.show()
```