Principles of Integrated Engineering, Section 1

Mini-Project 3

Lilo Heinrich, Jadelin Kirkvold, Hazel Smith

Introduction

The goal of this project was to create a two wheeled robot capable of autonomously following a track demarcated by electrical tape on the ground. This was accomplished using the input from three infrared reflectance sensors mounted to the underside of the robot and a self-correcting navigation program that sought out and followed the dark line of the electrical tape. The robot was able to complete a full loop of the track in 57 seconds and had to relocate the track by driving in a circle three times during that run.

Mechanical System

The standard chassis functioned as the mounting point for all the components of the line following robot and the default motor mounts were used to affix the motors to the underside of the chassis. The arduino and motor shield stack and solderless breadboard were mounted to the chassis using double-sided tape. IR reflectance sensors were attached using 3D printed sensor mounts.

Sensor mounts were designed in Solidworks to attach to the front of the robot chassis and hold the sensors (some distance) away from the ground. This was done to achieve a cleaner reading than if the sensor was mounted directly to the chassis, where it's output could be distorted from being further away from what it was supposed to be sensing.

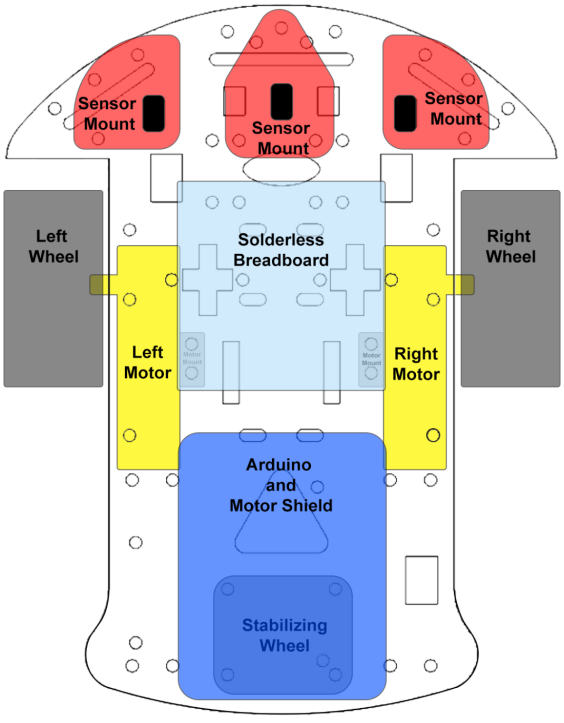

Figure 1: Final Robot Layout

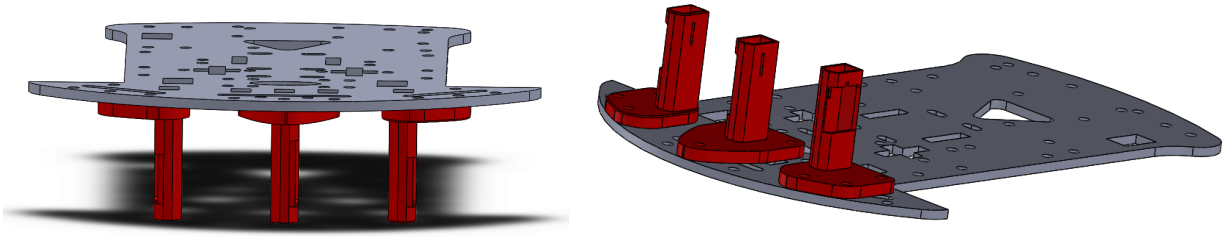

Figure 2(a, b): Render of Sensor Mounts, (a) Front View, (b) Angled View

Circuit Design

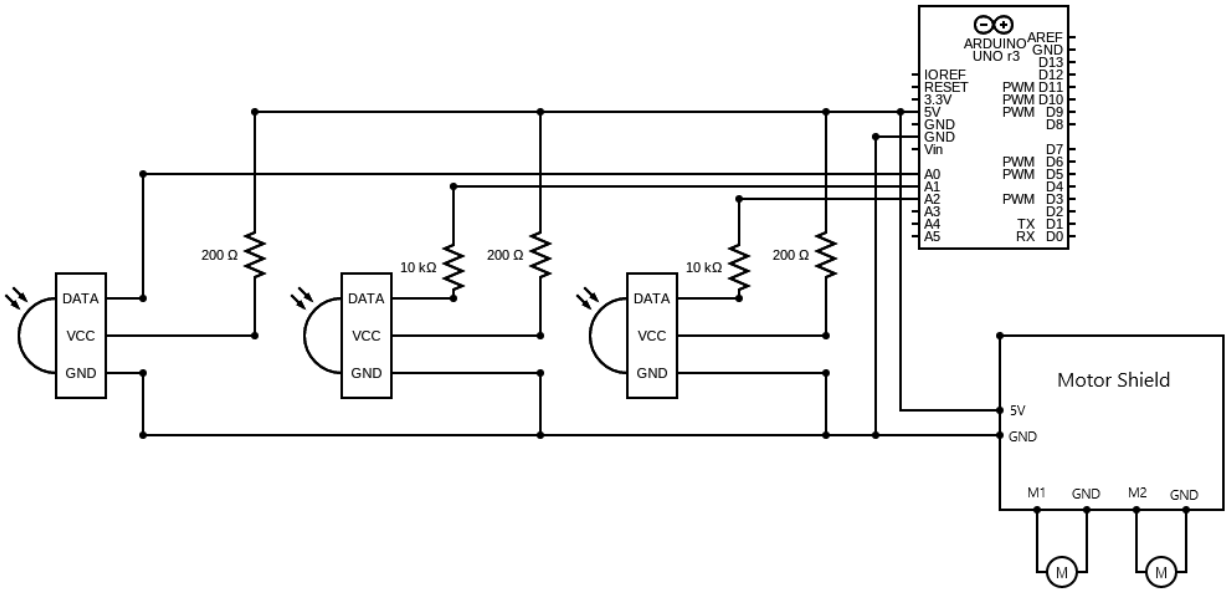

Figure 3: Final Circuit Diagram

We based this circuit off of the circuit diagram given in the Principles of Integrated Engineering Mini-Project 3 specifications, modifying it to add three IR sensors in parallel. Additionally, we used a resistance of 10kΩ in the voltage divider for sensing the transceiver output. This value was determined experimentally. When we used a 1kΩ resistor, we observed that the readings when off of the tape were quite close to the readings when on the tape, so we decided to increase the resistance value. First we tried a 5kΩ resistor which slightly increased the difference in reading, then we tried 10kΩ which gave a great enough difference to be satisfactory for our purpose of use.

Sensor Calibration

To calibrate our three sensors we took measurements both when it was on the tape and when it was off the tape. For each sensor we took sixty total measurements: thirty on the tape and thirty off. We plotted these readings using a boxplot to find the spread of values.

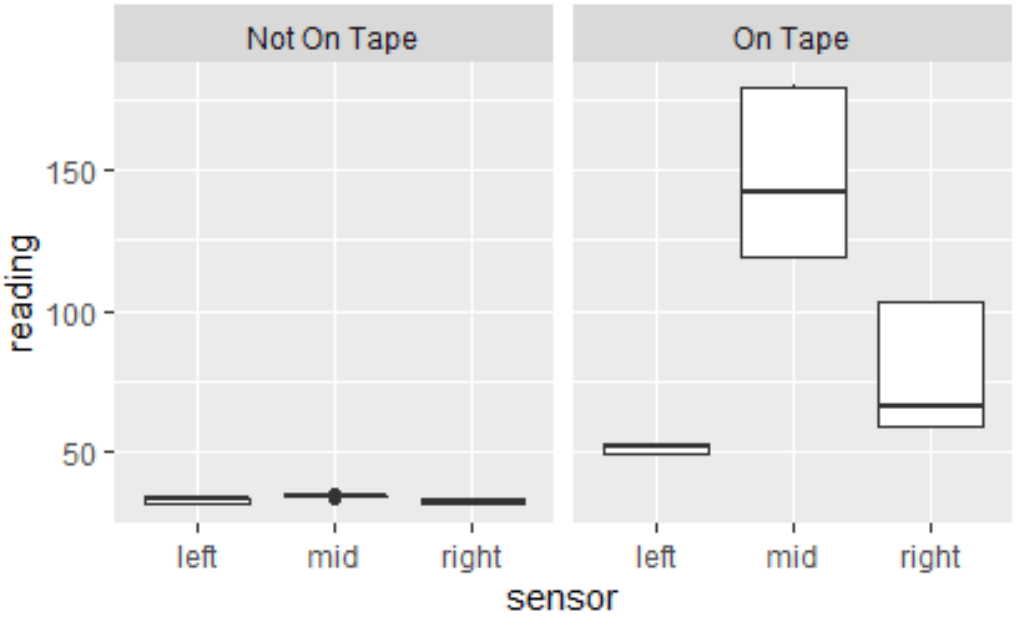

Figure 3(a, b): Boxplots of IR Reading by Sensor (a) Off Tape, (b) On Tape

The values received on the tape were much more variable than those received off the tape. However, there was a significant difference in the reading between the two conditions (on the tape and off the tape) for all three sensors, suggesting that they were working. As readings were always higher on the tape, we needed to pick cutoff values that were greater than the maximum values when off the tape, but less than the minimum values when on the tape.

Table 1: Table of IR Readings by Sensor

Sensor	Max. Value Not On Tape	Min. Value On Tape
Left	33	49
Mid	35	119
Right	33	59

Given the information in Table 1, cutoff values greater than thirty-five but less than forty-nine would work for all sensors. However, one thing we noticed is that sensor

readings values can vary depending on environmental conditions (i.e. lighting) so we implemented a serial input allowing us to alter each cutoff value individually as needed.

Control System

Check Sensors

To check whether an IR sensor was on the tape, we implemented a function called *checkSensor()* reproduced below. In *checkSensor()*, we return that the IR sensor is on the tape if the analog reading of the sensor exceeds a given threshold or cutoff value.

```
// checkSensor() returns true sensor if on the tape
bool checkSensor(int sensorPin, int cutoff_value){
  if (analogRead(sensorPin) > cutoff_value) {
    return true;
  }
  return false;
}
```

To set this cutoff value, we initially wrote static values into the code that were determined based off of the IR calibration data summarized in Table 1. We later implemented a function called *updateFromSerial()* which changes these threshold values based on user input on the serial console.

The function *updateFromSerial()* is rather long, so we will reproduce it in pseudocode directly below. The full function can be found starting on line 104 of the Arduino code in Appendix A. In *updateFromSerial()*, we first check if the Serial port has any bytes available to read. Then, we prompt the user to input a letter 'L' for Left, 'R' for Right, or 'M' for middle to signify which IR sensor they intend to change the threshold of. Until the user has input one of these values, the variable *serial_state* will not change from '-' (our "null" value). After accepting one of those recognized states ('L', 'R', or 'M') the program then prompts for a new IR threshold value, parses the next available data as an integer, and only accepts values within the range of 0 to 1023. It gives a print output when the threshold is successfully changed, and reverts back to the null *serial_state*.

```
// allow the user to update the three IR sensor threshold values
void updateFromSerial():
  if Serial.available():
    if serial_state != '-':
      int value = Serial.parseInt()
      if value is in valid range (value > 0 and value < 1023):
        set the appropriate IR sensor cutoff to value
        print success in setting that IR sensor cutoff
      else:
        print that the IR threshold given was invalid
```

```
      print out the current thresholds for all three IR sensors
      serial state = '-' // this resets the state, starting over the process

   else:
      char ch = Serial.read()
      serial state = '-'
      if ch == 'L' or 'M' or 'R':
        serial state = 'L' or 'M' or 'R'
        print that it set the IR threshold for 'L' or 'M' or 'R' to 'value'
```

Drive

To navigate along the line we implemented a simple function called *drive()*, reproduced below. In this function, if the middle IR sensor is found to be on the tape, the robot drives forwards. If the middle sensor is not on the tape but the left sensor is, the robot will turn left in order to re-center the middle sensor. Symmetrically, if the middle sensor is not on but the right sensor is, the robot will turn right to re-center the middle sensor.

Note that the spacing of our IR sensors is wider than the tape itself, meaning that no two IR sensors can be active at the same time. Additionally, if the robot is positioned in such a way that the tape falls in between for example the right and middle sensors, the robot will not register the location of the tape on any of the sensors. If none of the IR sensors are on, the robot will default to searching by continuously turning left until it relocates the tape.

```
// drive() drive track going forward if centered, otherwise turning right/left
void drive() {
  if (checkSensor(middleSensorPin, cutoff_mid)) {
    forward();
  } else if (checkSensor(leftSensorPin, cutoff_left)){
    turn(true); // turn left
  } else if (checkSensor(rightSensorPin, cutoff_right)){
    turn(false); // turn right
  } else if (!checkSensor(middleSensorPin, cutoff_mid) and
             !checkSensor(leftSensorPin, cutoff_left) and
             !checkSensor(rightSensorPin, cutoff_right)){
    turn(true); // turn left
  }
}
```

When driving forward, the robot moves with a constant speed. When turning, the motor on the side of the direction to turn is set to a speed of zero, while the other motor is set to continue driving forwards. We chose to use constant motor speeds both because we did not have enough time to implement a more complex control system, and also because the robot does not travel in a straight line when set to equal motor speeds. We calibrated an offset for the left motor to keep the robot travelling straight at one particular speed, but

this offset is not generalizable for other speeds. It would require additional characterization of our system to understand this effect at multiple speeds.

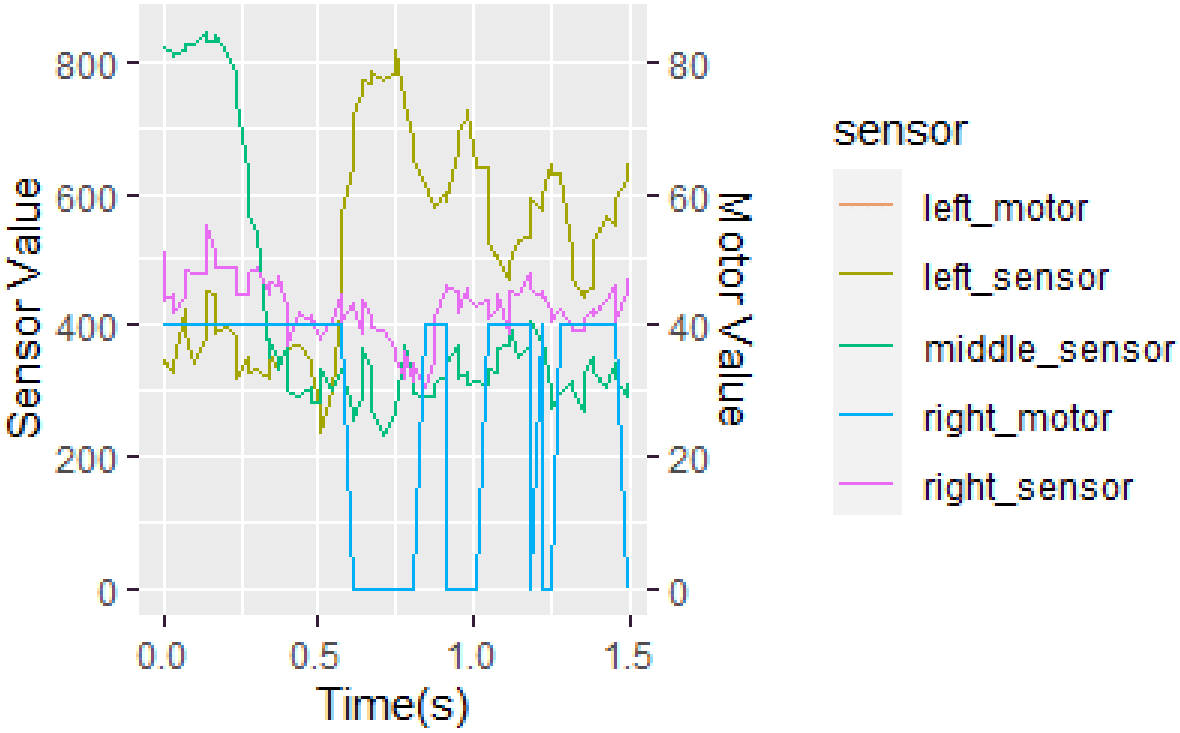

Figure 4: Superimposed Motor Powers and IR Readings over Time

Here is a plot comparing our motor speeds to our observed sensor values over 1.5 second. (Note: the left motor does not turn at all during this window.) In this 1.5 second window, we can see that the left sensor is reading a higher value, indicating it is on the tape, while the middle and right sensors are not. Therefore, the right motor is given a power of 40 while the left motor has a power of 0. This turns the robot to the left to realign the robot back onto the middle of the line.

Results

Our robot was able to successfully complete the track in just under one minute. It had to relocate the tape by searching in a circle three times. Here is the video of successful completion.

Reflection

Hazel: This project went really well for me. Three people instead of two definitely made the workload more manageable. I enjoyed the fact that the teaching team told us to honor fall break and take time off, but I wish that we had had more time at the beginning of the project to get planning and going. Most of our major problems came from a multitude of issues with our power cable and figuring out how to connect the sensors. The

former issues (the ones with the power cables) were almost unavoidable, but the latter might have been solved if we had actually aimed for a MVP at first instead of trying to design our stretch goal first. Going into a later project I will definitely be advocating for a MVP first before moving into stretch goals.

Jadelin: In contrast to the previous two mini-projects, I was able to work on a team with non-mechanical engineers. This meant that I was able to focus on the mechanical design and CAD instead of having to work on the code with other people who also don't fully understand what is going on. I had hoped to get the CAD done and components printed sooner but a combination of unclear communication and my own busy schedule meant that didn't happen. While I wish I had been able to allocate more time and effort to this project, I recognize that I did the best I could given all the things on my plate.

Lilo: Making a line-following robot was fun, and it was good to get to work with two new project partners. Like Hazel said above, it made workload division better. We made a successful line-following robot in the end (which is awesome!) but I feel that we accidentally got off to a rough start on this project. We spent the first class period figuring out why our power cable didn't work and how to connect the IRs. Next class, we tried soldering our circuit and instead decided to use a mini solderless breadboard, which we spent a week trying to obtain and later had to improvise by cutting on the bandsaw. These impedances reduced the time we had to work on the code, so we never got to implement PID control, something I was looking forward to. Going forward, I will advocate for getting the mechanical and electrical work done sooner to allow more time to work on code.

Appendix A: Arduino Code

Link: github.com/Winterbl00m/PIE_MiniProject3

```
#include <Adafruit_MotorShield.h>

const int leftSensorPin = A2; // 880-940: use 910 as cutoff
const int middleSensorPin = A1; // 830-900: use 870 as cutoff
const int rightSensorPin = A0; // 790-870: use 840 as cutoff

int cutoff_left = 920;
int cutoff_mid = 850;
int cutoff_right = 800;

Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_DCMotor *leftMotor = AFMS.getMotor(1);
Adafruit_DCMotor *rightMotor = AFMS.getMotor(2);

int left_offset = 6;
int default_speed = 50;
char serial_state = '-';

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Adafruit Motorshield v2 - DC Motor test!");
  if (!AFMS.begin()) {         // create with the default frequency 1.6KHz
    Serial.println("Could not find Motor Shield. Check wiring.");
    while (1);
  }
  Serial.println("Motor Shield found.");

//  start up the serial console input system
  Serial.println("Type letter \"L\", \"M\", or \"R\" to set new IR threshold value in range (0,1023)");
  Serial.print("LeftIR = ");
  Serial.print(cutoff_left);
  Serial.print(" | MiddleIR = ");
  Serial.print(cutoff_mid);
  Serial.print(" | RightIR = ");
  Serial.println(cutoff_right);
  Serial.println();

  // Set the speed to start, from 0 (off) to 255 (max speed)
  stop_driving();
}

// checkSensor() returns true sensor if on the tape
bool checkSensor(int sensorPin, int cutoff_value){
  if (analogRead(sensorPin) > cutoff_value) {
    return true;
  }
  return false;
}

void turn(bool leftDir){
```

```
51   int sensorPin = rightSensorPin;
52   if (leftDir) {
53     sensorPin = leftSensorPin;
54   }
55   if (leftDir) {
56     leftMotor->setSpeed(0);
57     rightMotor->setSpeed(default_speed);
58   } else {
59     leftMotor->setSpeed(default_speed-left_offset);
60     rightMotor->setSpeed(0);
61   }
62   leftMotor->run(FORWARD);
63   rightMotor->run(FORWARD);
64 }
65
66 void forward(){
67     leftMotor->setSpeed(default_speed-left_offset);
68     rightMotor->setSpeed(default_speed);
69     leftMotor->run(FORWARD);
70     rightMotor->run(FORWARD);
71 }
72
73 void stop_driving(){
74     leftMotor->setSpeed(0);
75     rightMotor->setSpeed(0);
76     leftMotor->run(FORWARD);
77     rightMotor->run(FORWARD);
78 }
79
80 void loop() {
81   // get updated IR cutoff/threshold values from serial console input menu
82   updateFromSerial();
83
84   // drive track going forward if centered, otherwise turning right/left
85   drive();
86 }
87
88 // drive() drive track going forward if centered, otherwise turning right/left
89 void drive() {
90   if (checkSensor(middleSensorPin, cutoff_mid)) {
91     forward();
92   } else if (checkSensor(leftSensorPin, cutoff_left)){
93     turn(true); // turn left
94   } else if (checkSensor(rightSensorPin, cutoff_right)){
95     turn(false); // turn right
96   } else if (!checkSensor(middleSensorPin, cutoff_mid) and
97              !checkSensor(leftSensorPin, cutoff_left) and
98              !checkSensor(rightSensorPin, cutoff_right)){
99     turn(true); // turn left
100   }
101 }
102
103 // allows the user to update the three IR sensor threshold/cutoff values
104 void updateFromSerial(){
105   if (Serial.available()) {
106 //    serial_state not being '-' (null) means that the program knows which
sensor to change the threshold for
```

```
107 //    therefore, the next value it expects as input is an integer on (0,1023)
108     if ((serial state != '-')) {
109       int value = Serial.parseInt();
110       Serial.println(value);
111 
112       if (value > 0 and value < 1023){
113         if (serial state == 'L') {
114           cutoff left = value;
115           Serial.println("Left IR threshold set!");
116         } else if (serial state == 'M') {
117           cutoff mid = value;
118           Serial.println("Middle IR threshold set!");
119         } else if (serial state == 'R') {
120           cutoff right = value;
121           Serial.println("Right IR threshold set!");
122         }
123 
124 // the program does not know which sensor to change the threshold for.
125 // therefore it's waiting for a letter L M or R to specify Left, Mid, or Right
126       } else {
127         Serial.println("Invalid IR threshold value given.");
128       }
129       Serial.print("LeftIR = ");
130       Serial.print(cutoff left);
131       Serial.print(" | MiddleIR = ");
132       Serial.print(cutoff mid);
133       Serial.print(" | RightIR = ");
134       Serial.println(cutoff right);
135       Serial.println();
136 
137       serial state = '-';
138       Serial.read();
139     } else {
140       char ch = Serial.read();
141       serial state = '-';
142 
143       if (ch == 'L') {
144         serial state = 'L';
145         Serial.print("Set Left IR threshold: ");
146       } else if (ch == 'M') {
147         serial state = 'M';
148         Serial.print("Set Middle IR threshold: ");
149       } else if (ch == 'R') {
150         serial state = 'R';
151         Serial.print("Set Right IR threshold: ");
152       }
153       Serial.parseInt();
154     }
155   }
156 }
```