Robot Programming Final Project
Line Tracer Robot
Portfolio

Course code: MEC2034-01

Teacher: 곽문규

Date: 19 June 2018

Made by: Alexandre Stelzig (2018130128)

Table of Contents

Introduction 3

Material Used 3

Robot design 3

- **Basic Explanation** 3
- **State Table** 5
- **State Diagram** 6
- **Arduino Wiring** 7
- **Code** 9
- **Robot Visual Design** 12
- **Testing methods (tracks)** 13
- **Problems Encountered during development** 14

Results of competition 14

Ways to improve the robot 15

Conclusion 15

Introduction

This project was made for the final project of the Robot programming course at Dongguk University. The goal of this project was to design and implement a line following robot using an Arduino as well as DC motors, IR sensors and other materials. At the end of the project, the goal was to complete a track in the shortest times.

Material Used

- Arduino Board x1
- Arduino prototype shield x1
- Arduino Motor Shield x1
- DC motors x2
- 4 Infrared Avoidance Module x1
- 9V Battery x1
- 9V Battery connector x1
- AA Battery x4
- 4 AA Battery connector x1
- 360 degrees' wheel x1
- Wheel x2
- Button x1
- Resistor x1
- Screws x30
- Bolts x10
- Plastic extensions x4
- 3D printed IR sensor holder x1
- Cables x24
- Electronic Tape

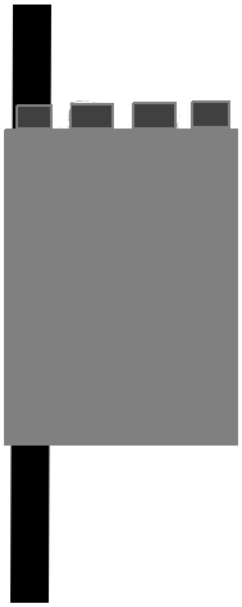
Robot design

Basic Explanation

To build a Line Tracer Robot, we first need to build the base using two DC motors, IC sensors, wheels, Arduino and more (see the Material Used section for all material used). Once the base is built, we need to figure out how to design the robot. The following is a basic explanation of the robot's logic:

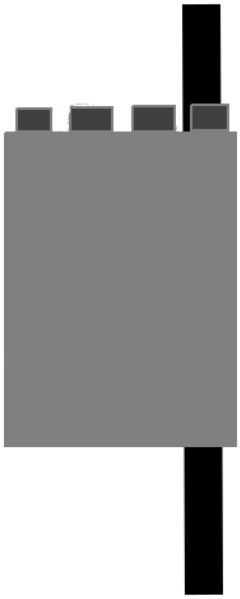
The goal of the robot is to stay in the center of the line, so, if the two middle sensors are Active, it means the the robot is centered and we can go forward.

The problem is when it is not centered, we need to move the robot back to it. Consider the following Diagram:



In this case, the move the robot back to the center of the line, we need to move the robot towards the LEFT.

Next, consider the following Diagram:



In this case, the move the robot back to the center of the line, we need to move the robot towards the RIGHT.

There are two other state that we need to consider too:

1. When all sensors are active: Stop the robot.
2. When all sensors are inactive: Repeat the previous action until we read the line.

Lastly, the must be a on/off switch to toggle to robot's power.

State Table

To have a better understand of the robot's state, lets consider the following state tables:

All States Results

IR4	IR3	IR2	IR1	RESULTS
0	0	0	0	PREVIOUS
0	0	0	1	LEFT
0	0	1	0	FORWARD
0	0	1	1	FORWARD
0	1	0	0	FORWARD
0	1	0	1	Impossible State - defaults to FORWARD
0	1	1	0	FORWARD
0	1	1	1	FORWARD
1	0	0	0	RIGHT
1	0	0	1	Impossible State - defaults to LEFT
1	0	1	0	Impossible State - defaults to FORWARD
1	0	1	1	Impossible State - defaults to FORWARD
1	1	0	0	FORWARD
1	1	0	1	Impossible State - defaults to FORWARD
1	1	1	0	FORWARD
1	1	1	1	STOP

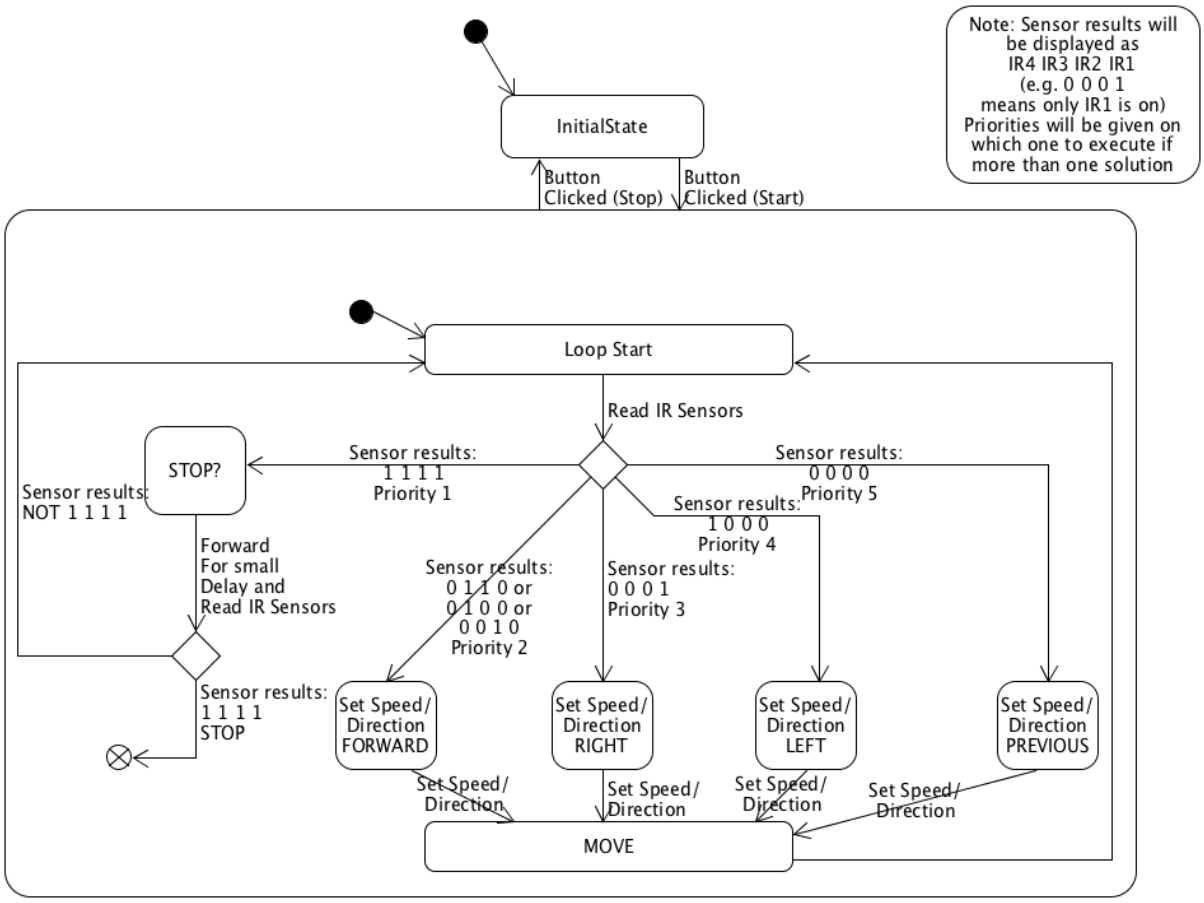
All States Results (Without Impossible States)

IR4	IR3	IR2	IR1	RESULTS
0	0	0	0	PREVIOUS
0	0	0	1	LEFT
0	0	1	0	FORWARD
0	0	1	1	FORWARD
0	1	0	0	FORWARD
0	1	1	0	FORWARD
0	1	1	1	FORWARD
1	0	0	0	RIGHT
1	1	0	0	FORWARD
1	1	1	0	FORWARD
1	1	1	1	STOP

STOP = IR1 && IR2 && IR3 && IR4
FORWARD = (IR3 || IR2) && !STOP
RIGHT = IR4 && !FORWARD
LEFT = IR1 && !FORWARD
PREVIOUS = !RIGHT && !LEFT

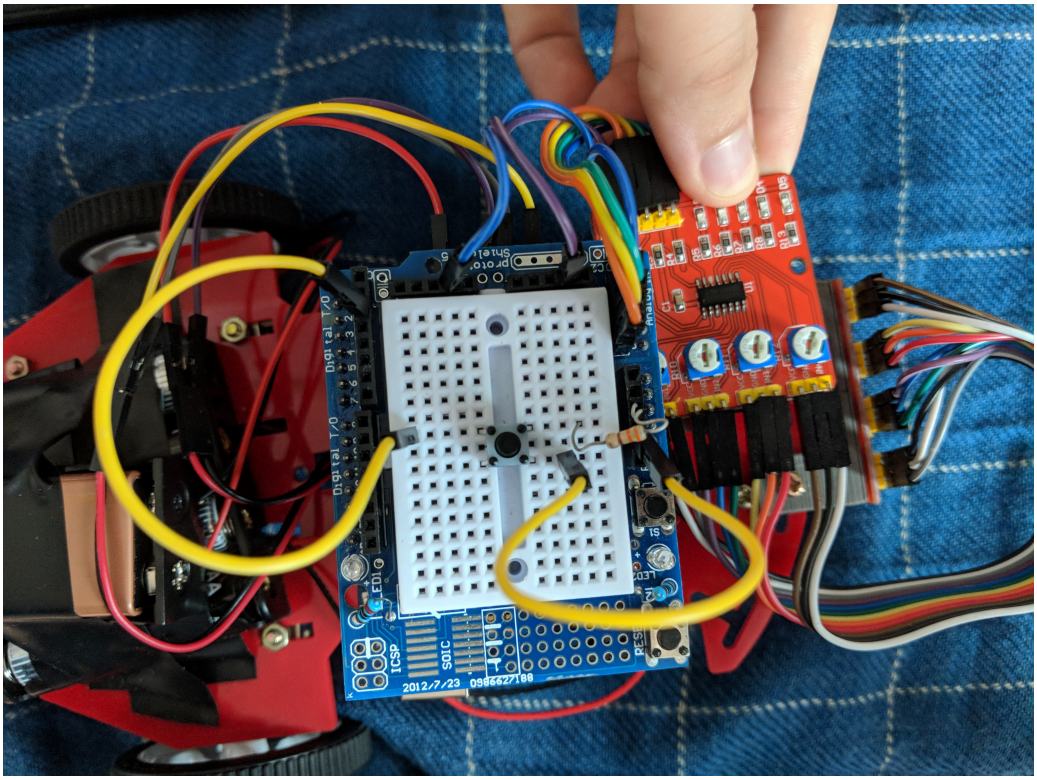
State Diagram

We can also visualize the states with a state diagram:



Arduino Wiring

The following image contains the wiring for the Arduino prototype shield:

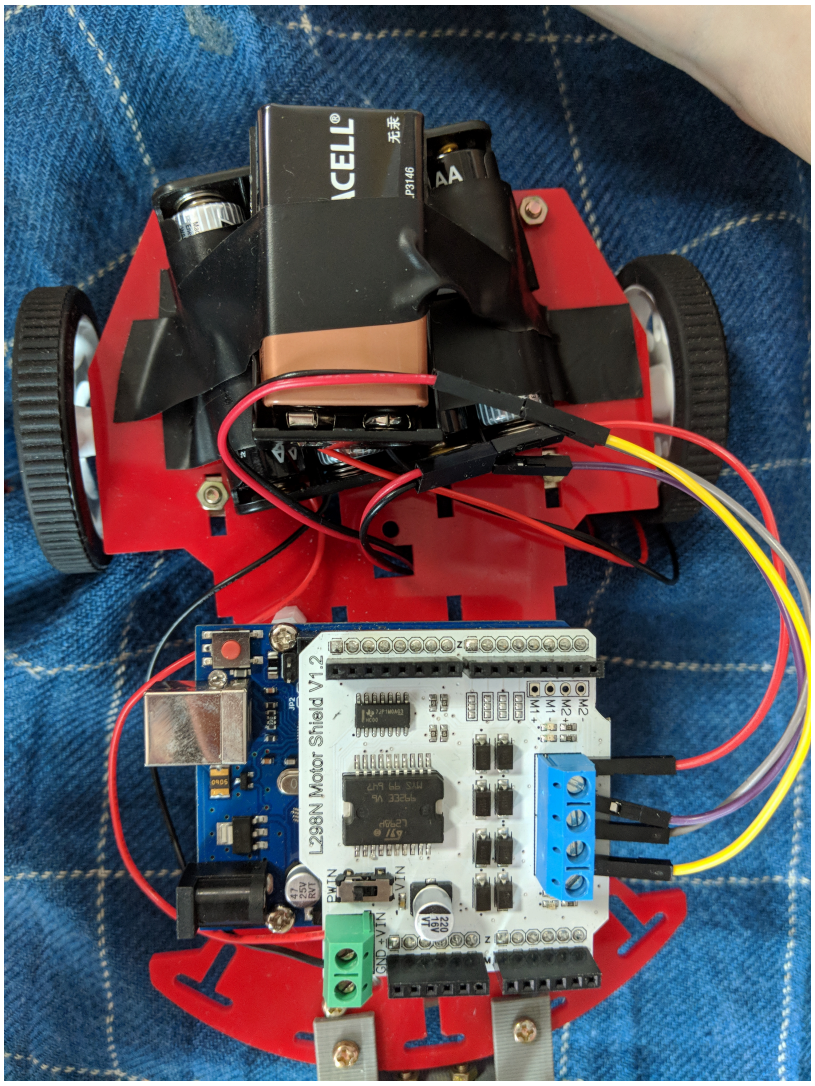


The pins are as follows:

- IR1: Analog 0
- IR2: Analog 1
- IR3: Analog 2
- IR4: Analog 3
- Button: Digital 2

When we need to use the 9V battery supply, we only need to connect the GND and 5V to the board. All the other pins not mentioned in the preceding list are GND or 5V connection.

The following image contains the wiring for the Arduino motor shield:



For the left wheel, I used E1 and M1 and for the Right wheel, I used E2 and M2. You can also see that the AA batteries supply is directly connected to the shield to power it.

Code

The following is the full code for my line tracer robot:

```
int IR1 = 11;
int IR2 = 10;
int IR3 = 9;
int IR4 = 8;

int E1 = 4;
int M1 = 5;
int M2 = 6;
int E2 = 7;

int IR1_v = 0;
int IR2_v = 0;
int IR3_v = 0;
int IR4_v = 0;

int threeshoold = 800;

int directionLeft = LOW;
int directionRight = LOW;
int previousDirectionLeft = LOW;
int previousDirectionRight = LOW;

int speedLeft = 0;
int speedRight = 0;
int previousSpeedLeft = 0;
int previousSpeedRight = 0;

int maxSpeed = 180;
int turnSpeed = 85;
int turnSpeed2 = 50;


bool stopTestExecuted = false;

const int buttonPin = 2;
bool isButtonPinPressed = false;
bool appRunning = false ;
int buttonPinState = 0;

void setup() {
  Serial.begin(9600);
}
void loop() {
  buttonPinState = digitalRead(buttonPin);

  Serial.print("buttonPinState ");
  Serial.println(buttonPinState);

  if (buttonPinState == HIGH) {
    if (!isButtonPinPressed) {
      isButtonPinPressed = true;
      appRunning = !appRunning;

      if (appRunning) {
        forward();
        Move();
        delay(500);
      }
    }
```

Variables for the IR sensors

Variables for the DC motors

Variables for the IR sensors results

Variables for the current and previous direction of the motor

Variables for the current and previous speed of the motor

Variables for the forward speed and turning speed

Variable to start/stop the app on a button click

Logic for the start/stop on button pressed

```
  } else {
    isButtonPinPressed = false;
  }

  if (appRunning) {
    ReadIR();

    if (IR1_v == 1 && IR2_v == 1 && IR3_v == 1 && IR4_v == 1) {

      if (!stopTestExecuted) {
        previous();
        Move();
        delay(50);
        ReadIR();
      }

      if (IR1_v == 1 && IR2_v == 1 && IR3_v == 1 && IR4_v == 1) {
        stopMoving();
        stopTestExecuted = true;
      } else {
        stopTestExecuted = false;
      }

    }
    else if (IR2_v == 1 || IR3_v == 1) {
      forward();
    }
    else if (IR1_v == 1) {

      left();
    }


    else if (IR4_v == 1) {
      right();
    } else {

      previous();
    }

    Move();



    Serial.print(" ");
    Serial.print(IR1_v);
    Serial.print(" ");
    Serial.print(IR2_v);
    Serial.print(" ");
    Serial.print(IR3_v);
    Serial.print(" ");
    Serial.println(IR4_v);
    //    delay(5);
  } else {
    stopMoving();
    Move();
  }
}
```

If all IR are active, stop

Ensure that it's a real stop and not an error by moving forward for 50 ms. If it's a real stop, stop executing the application

If one of the two middle sensors are active, go forward

If the right sensor is active, go left

If the left sensor is active, go right

If none of the sensors are active, repeat the previous action

Set the speed to 0 if the app is not running (button was not pressed)

```
void Move() {

  digitalWrite(E1, directionLeft);
  digitalWrite(E2, directionRight);
  analogWrite(M1, speedLeft);
  analogWrite(M2, speedRight);

  previousDirectionLeft = directionLeft;
  previousDirectionRight = directionRight;
  previousSpeedLeft = speedLeft;
  previousSpeedRight = speedRight;

}
```

Move the robot by applying the direction and speed for each wheel which was assigned in the main loop

```
void stopMoving() {
  Serial.println("Stop");
  directionLeft = LOW;
  directionRight = LOW;
  speedLeft = 0;
  speedRight = 0;
}
```

Stop moving by setting a speed of 0

```
void right() {
  Serial.println("right");
  directionLeft = LOW;
  directionRight = HIGH;
  speedLeft = turnSpeed;
  speedRight = turnSpeed2;
}
```

Go right by putting the right wheel reversed.
* I also used a turn speed which is different from the forward speed to have a better control of the turns

```
void left() {
  Serial.println("left");
  directionLeft = HIGH;
  directionRight = LOW;
  speedLeft = turnSpeed2;
  speedRight = turnSpeed;
}
```

Go left by putting the left reverse

```
void forward() {
  Serial.println("forward");
  directionLeft = LOW;
  directionRight = LOW;
  speedLeft = maxSpeed;
  speedRight = maxSpeed;

}
```

Go forward using the max speed

```
void previous() {
  Serial.println("Previous");
  directionLeft = previousDirectionLeft;
  directionRight = previousDirectionRight;
  speedLeft = previousSpeedLeft;
  speedRight = previousSpeedRight;
}
```

Set the current speed and direction to the previous one by using temp variables to store the previous values

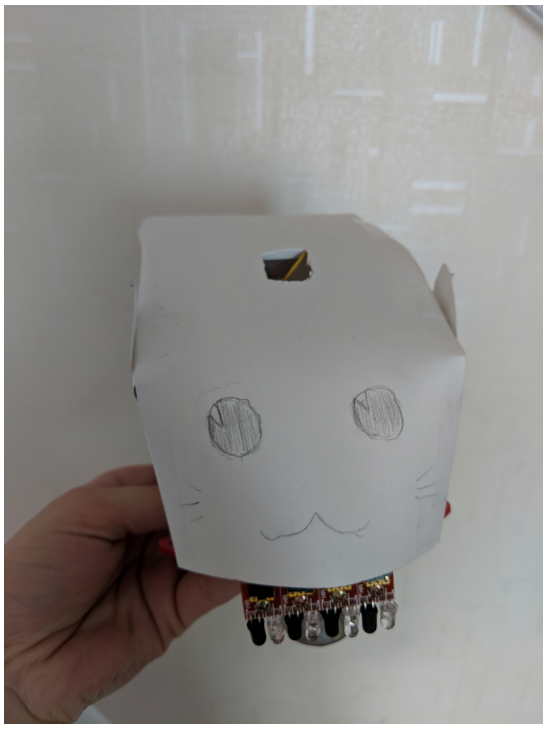
```
void ReadIR() {
  IR1_v = analogRead(IR1);
  IR2_v = analogRead(IR2);
  IR3_v = analogRead(IR3);
  IR4_v = analogRead(IR4);

  if (IR1_v > threeshoold)
    IR1_v = 1;
  else
    IR1_v = 0;
  if (IR2_v > threeshoold)
    IR2_v = 1;
  else
    IR2_v = 0;
  if (IR3_v > threeshoold)
    IR3_v = 1;
  else
    IR3_v = 0;
  if (IR4_v > threeshoold)
    IR4_v = 1;
  else
    IR4_v = 0;
}
```

Logic used to read the IR sensors
I used analogRead because I felt like it gave better results

Robot Visual Design

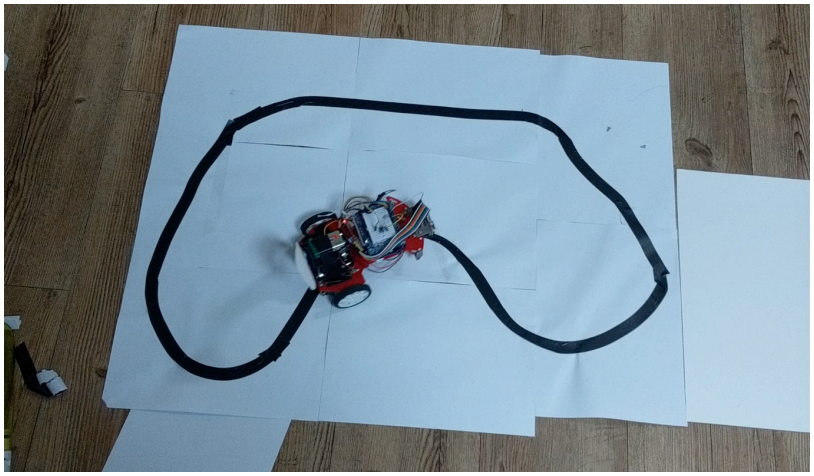
For the visual design of the cat, I wanted to create a cat with a tail controlled by a servo. Unfortunately, once I implemented the tail with the servo, the robot wasn't able to operate at all because it was taking too much power (but it was working fine when connected to a bigger power supply such as my computer). I designed the cat using cardboards, a pencil and tape.
Here is the final design:

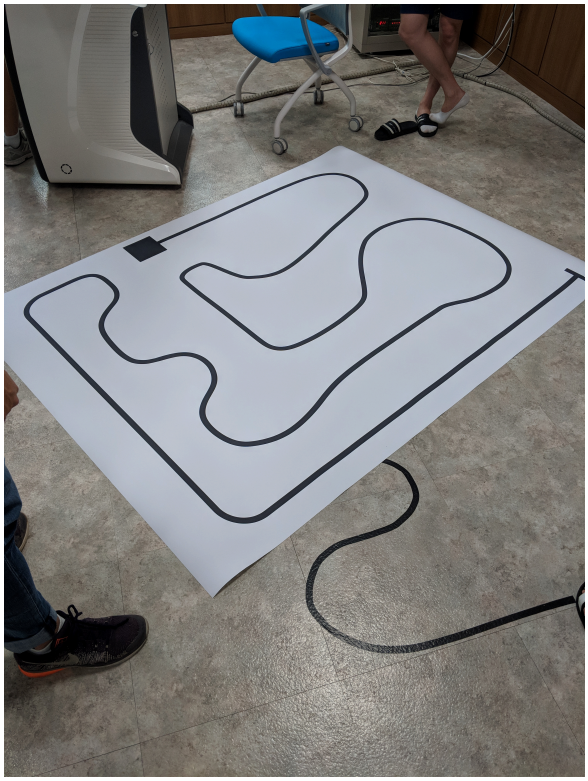


(I also added a hole on the top to be able to press the on/off button)

Testing methods (tracks)

To test the robot, I used multiple different tracks. At first because we didn't know how the competition track looks like, I usually tested very difficult turn angles by putting down electronic black tape on the ground. But a few days before the competition, the track was revealed and there were not too many difficult turns. Because I was already testing difficult turns, my robot was already able to complete the track without problems. The only thing that I needed to do now is to make it go faster. To do so, I created a small track (see image below) that would allow me to test different speeds and try to be faster.





(Competition Track)

Problems Encountered during development

During the development, I encountered multiple problems. Most of my issues were related with Hardware. One of the major issues at the beginning of the development was that the wheels weren't moving at all if I put a speed of more than 100 when connected to the 9V power supply. This issue was fixed after a while after I learned that the Arduino Motor Shield switch must be set to Pwin and not Vin. Once this was fixed, I was able to work more efficiently.

Another issue was that my IR sensors weren't changing states when going over a black line. This issue was mostly because of the potentiometer behind the IR sensors weren't calibrated correctly. The issue also occurred when some wires weren't connected properly which happened after moving the robot a lot.

As for the code related issue, I only have one major one which was the the robot was sometimes fully stopping when turning a corner. This was happening because during some turns, all the sensors were active which, at the time, meant that it needs to stop. To fix this issue, once all sensors are active, I make the robot move forward for a few milliseconds and revalidate the state. If they are still all active, it means that it was not a corner and stop but if they were not, the robot continues as normal.

Results of competition

Round 1

Results: Success, track completion after around 35 seconds.

Round 2

Results: Failure after the second turn.

Explanation: After succeeding at the first round, I decided to up the max speed of bother the normal speed and the turning speed. There might have been an issue where the speed was too fast for the sensors to keep up and didn't make the turn.

Round 3

Results: Failure after the first turn.

Explanation: After the second round, I went to do more testing on which speed would be better for better results but in-between the rounds, the 9V battery supply started to run out and the robot wasn't working efficiently. By the time I figured out that it was a problem with my batteries, I only had 1 more minute before the start of the round. I switch my batteries as fast as I could but I didn't have time to re-qualibrate the speeds and the robot went straight through the first turn.

Ways to improve the robot

This section will describe about lessons learned as well as better ways that the line tracer can be improved to perform better. Even though I was able to complete the round, there is still a lot of improvement that can be made to it.

One of the first lessons learned was after the 3rd Round. Before the start of the competition, it would have been better to change the batteries to ensure that the robot performs at his best abilities.

Also, while the code is decent, while researching how to design a better line tracer robot, I found out about line tracer robot based using PID controllers. Basically, this design is made to calculate the error and correction for the speed of the wheels depending on which sensors are active. This seemed to be a better way to do it because the robot consistently adapts his speed to stay at the center of the line. Unfortunately, I wasn't able to adapt this method for various reasons with the main one being that we have an even number of sensors instead of odd. Personally, even without using the PID controller method, I think that having 5 sensors would greatly improve performance because we could detect the true middle as well as having more guidance for the direction of which the robot should move.

Conclusion

In conclusion, even though I was able to finish one round on three, there are still improvements that can be made to my design. This project was very enjoyable because it allowed me to fully design and implement an Arduino project from scratch and it was a challenge to make it efficient.