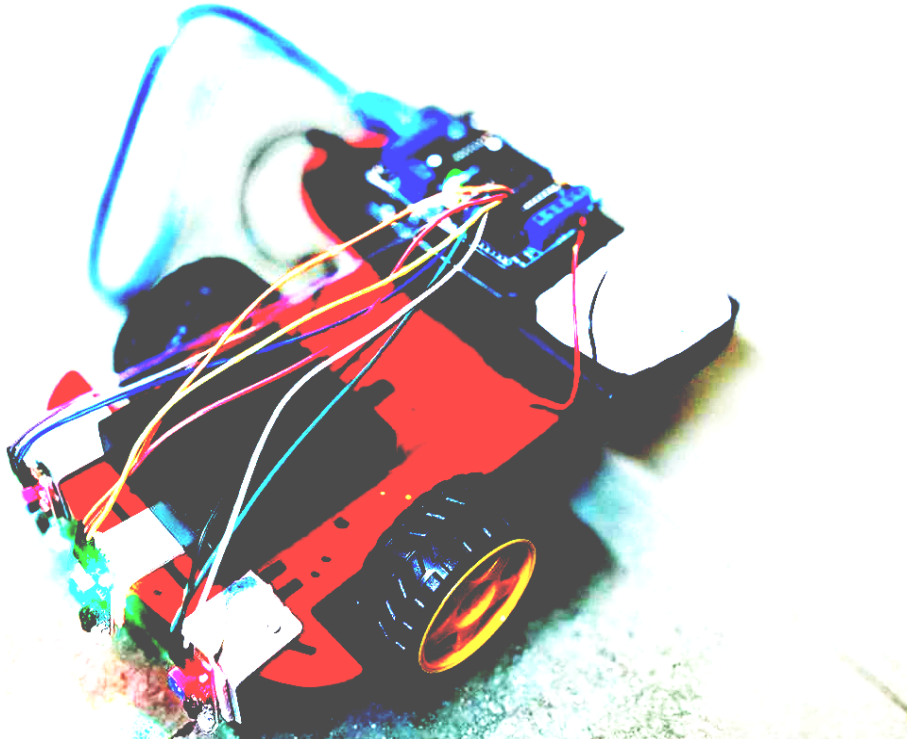
Course Project: Robotics

Line Following Mobile Robot Design

Group Info.

Arnav Singh **2021019**
Yash Khatri **2020267**



Project Description

This project focused on developing a mobile robot capable of navigating predefined test tracks, using IR sensors, microcontrollers and L293D motor shields to detect and follow black tape curves on A0 sized tracks while demonstrating flexible traversal behavior.

Objectives:-

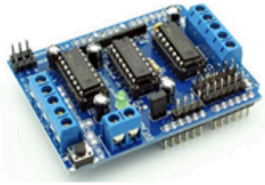
- **Assembling the Robot**
  Purchase a DIY Robotics kit, tape and screw the motors on the chassis, attach power sources, screw the unicycle and castor wheels, attach the motor L293D shield with and the UNO R3 Compatible, connect the UNO and R3 Compatible with IR Sensors using 9 F-F wires.

- **Coding the Feedback Controller**
  Develop an algorithm capable of detecting and correctly following the test tracks regardless of the initial orientation.

- **Tuning the Sensors**

  Tune the IR Sensors using a screwdriver to correctly discern between black and white surfaces

Robot Components

			
UNO R3-Compatible	USB Cable	IR Sensors	L293D Motor Shield
			
2WD Car Chassis	Battery Holder	F-F Wires 10pcs	
			
	Black Tape	L-Clamp X2	

In addition to the listed components, I used double sided tape to stick the L-Clamps to the chassis, and 8 Phillips head screws with nuts to hold the wheels and the motors.

How does it work?

The robot has 3 IR Sensors to detect potential paths, in the right, left and center positions, which feed binary data to the UNO R3 through the L293D motor shield, with a “1” representing a dark surface, and “0” otherwise. The UNO R3 then processes this data using the loop algorithm fed to it, and passes instructions to the 4 DC motors, which then rotate the wheels through EMI, thereby moving the entire body.

The robot is powered by a Xiaomi 20,000 mAh power bank (through the USB connection), and 4 Duracell AA Cells in a Battery Holder (through the power jack). We chose to include the power bank since, (a) it incremented the input, and (b) it pushed the center of mass to the back, which (based on our observations of other models) helped to ensure smooth-er motion (in addition to the castor wheel), despite us using a bang-bang controller.

Finally, we implemented an unused function in the setup code, which delegates the bot to move forward till it finds a black path to traverse, which had been suggested as possible initialisation by the course TA, during a team trial in October.

The Code

The algorithm is coded in the C++, using only the AFMotor Library, Three global variables, lefts, center and rights are declared, followed by the instantiation of the motors, and the speed.

```
// Can successfully traverse sinusoidal, circular and straight line path at a constant speed.
#include <AFMotor.h>
// Reference :- https://quadstore.in/wp-content/uploads/2020/07/LINE_FOLLOWING_CAR.zip
// Pin definitions
#define lefts A1
#define center A2
#define rights A0

// Motors
//Right
AF_DCMotor motor1(1,MOTOR12_1KHZ);
AF_DCMotor motor2(2,MOTOR12_1KHZ);
//Left
AF_DCMotor motor3(3,MOTOR34_1KHZ);
AF_DCMotor motor4(4,MOTOR34_1KHZ);

// Motor speed
const int speed=255;
```

In the setup, the IR sensors are set to input mode, and the motors' speed is set. Further, a while loop is created which forces the robot to move forward till it detects a surface.

```
void setup() {
  pinMode(lefts,INPUT);
  pinMode(center,INPUT);
  pinMode(rights,INPUT);
  motor1.setSpeed(speed);
  motor2.setSpeed(speed);
  motor3.setSpeed(speed);
  motor4.setSpeed(speed);
  while(digitalRead(center) == 0) {
    motor1.run(FORWARD);
    motor2.run(FORWARD);
    motor3.run(FORWARD);
    motor4.run(FORWARD);
  }
  haltbot();
}
```

In the loop function, functions are called depending on the digital input received from the IR sensors. The robot's supposed to stop once it crosses the track.

```
void loop() {
  int leftVal=digitalRead(lefts);
  int centerVal=digitalRead(center);
  int rightVal=digitalRead(rights);
  if (centerVal == 1) {
    moveforward();
  }
  else if (leftVal == 1) {
    turnleft();
  }
  else if (rightVal == 1) {
    turnright();
  }
  else {
    haltbot();
  }
}
```

The definitions for these functions are attached below:-

```
void moveforward() {
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}

void turnleft() {
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}

void turnright() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}

void haltbot() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}
```

Note: I haven’t gone into greater detail w.r.t the electronic equipment since they weren’t part of the course curriculum. (Also, I am not an ECE major)

Results

The robots' performed rather well on all the tracks, but especially on the third track, where it was able to somehow improve its performance iteratively, after being given an unorthodox start. We'd like to believe that our team's performance was the best among all the participating groups, but given the overall performance, that accolade isn't much to celebrate.

All in all, our robot's performance improved throughout the semester, becoming more and more adaptive to the different tracks it was supposed to follow, and aided our understanding of robot dynamics, and modelling, from a practical standpoint.

We wished to have a chance to implement the PID algorithm, but given the shift to virtual classrooms in the latter half of November, we weren't left with many opportunities to tune the control gains to adequately track the given paths. Nonetheless, our performance with the bang-bang controller is still laudable, and we hope the instructors see it that way as well.