MAZE SOLVING ROBOT

By

T.RAGHAV KUMAR	15BCE1374
BHAGAVATULA AISWARYA	15BCE1235
DRISHTI JAIN	15BCE1014
Y.HEMANTH TEJA	15BCE1144
NATRAJ	16MIS1014

A project report submitted to

Dr. REKHA

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

In partial fulfilment of the requirements for the course of

CSE3011 – ROBOTICS AND ITS APPLICATIONS

in

B.Tech (Computer Science and Engineering)

VIT UNIVERSITY, CHENNAI

Vandalur – Kelambakkam Road

Chennai – 600127

INTRODUCTION

The field of robotics has advanced greatly in recent years, having a large field of research in finding solutions to everyday problems, such as problems solving mazes and test new designs. The objective of the project is in using an autonomous robot with ultrasonic sensors in solving mazes, using automation, robotics, artificial intelligence and computational logic. Verifying and analysing the logic used in the process of solving mazes, where there is only a beginning and an end, autonomously

COMPONENTS

- L293D Motor Shield
- Arduino Mega
- Ultrasonic Sensors x 3
- DC Motors
- 6V battery for DC Motors
- 9V battery for Arduino Mega
- Car Chassis with 2 wheels

CODE FOR MOTOR MOVEMENT

```
#include <AFMotor.h>
AF_DCMotor motor_l(2);
AF_DCMotor motor_r(4);

#define trigPin3 44
#define echoPin3 45
#define trigPin1 42
#define echoPin1 43
#define trigPin2 40
#define echoPin2 41

String path="";

long duration,distance,RightSensor,BackSensor,FrontSensor,LeftSensor;

void stopmove()
{
  motor_l.run(RELEASE);
  motor_r.run(RELEASE);
}

void initializeSpeed()
{
  motor_l.setSpeed(200);
  motor_r.setSpeed(200);
}
```

```
void turnRight()
{
  motor_l.run(FORWARD);
  motor_r.run(BACKWARD);
  delay(300);
  motor_l.run(RELEASE);
  motor_r.run(RELEASE);
}

void turnLeft()
{
  motor_l.run(BACKWARD);
  motor_r.run(FORWARD);
  delay(300);
  motor_l.run(RELEASE);
  motor_r.run(RELEASE);
}

void goStraight()
{
  motor_l.run(FORWARD);
  motor_r.run(FORWARD);
  delay(300);
  motor_l.run(RELEASE);
  motor_r.run(RELEASE);
}

void takeUturn()
{
  SonarSensor(trigPin1, echoPin1);
  FrontSensor = distance;
  while(FrontSensor < 10)
  {
    motor_l.run(FORWARD);
    motor_r.run(BACKWARD);
    delay(600);
    motor_l.run(RELEASE);
    motor_r.run(RELEASE);
  }
}


void SonarSensor(int trigPin,int echoPin)
{
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
//delayMicroseconds(2);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
```

```
distance = (duration/2) / 29.1;
}

void setup()
{
Serial.begin (9600);
pinMode(trigPin1, OUTPUT);
pinMode(echoPin1, INPUT);
pinMode(trigPin2, OUTPUT);
pinMode(echoPin2, INPUT);
pinMode(trigPin3, OUTPUT);
pinMode(echoPin3, INPUT);
}

void loop() {

SonarSensor(trigPin2, echoPin2);
RightSensor = distance;
SonarSensor(trigPin3, echoPin3);
LeftSensor = distance;
SonarSensor(trigPin1, echoPin1);
FrontSensor = distance;

Serial.print(LeftSensor);
Serial.print(" - ");
Serial.print(FrontSensor);
Serial.print(" - ");
Serial.println(RightSensor);

if ((LeftSensor>20) && (RightSensor>20) && (FrontSensor>20))
{
  Serial.println("Movement");
  initializeSpeed();
  goStraight();
}

else if ((LeftSensor>20) && (RightSensor>20) && (FrontSensor<20))
{
   //stopmove();
   Serial.println("Left Turn");
   initializeSpeed();
   turnLeft();
   path=path+"L";
   //path=path+"S";
}
else if ((LeftSensor>20) && (RightSensor<20) && (FrontSensor>20))
{
   //stopmove();
   Serial.println("Left Turn");
   initializeSpeed();
   turnLeft();
   path=path+"L";
```

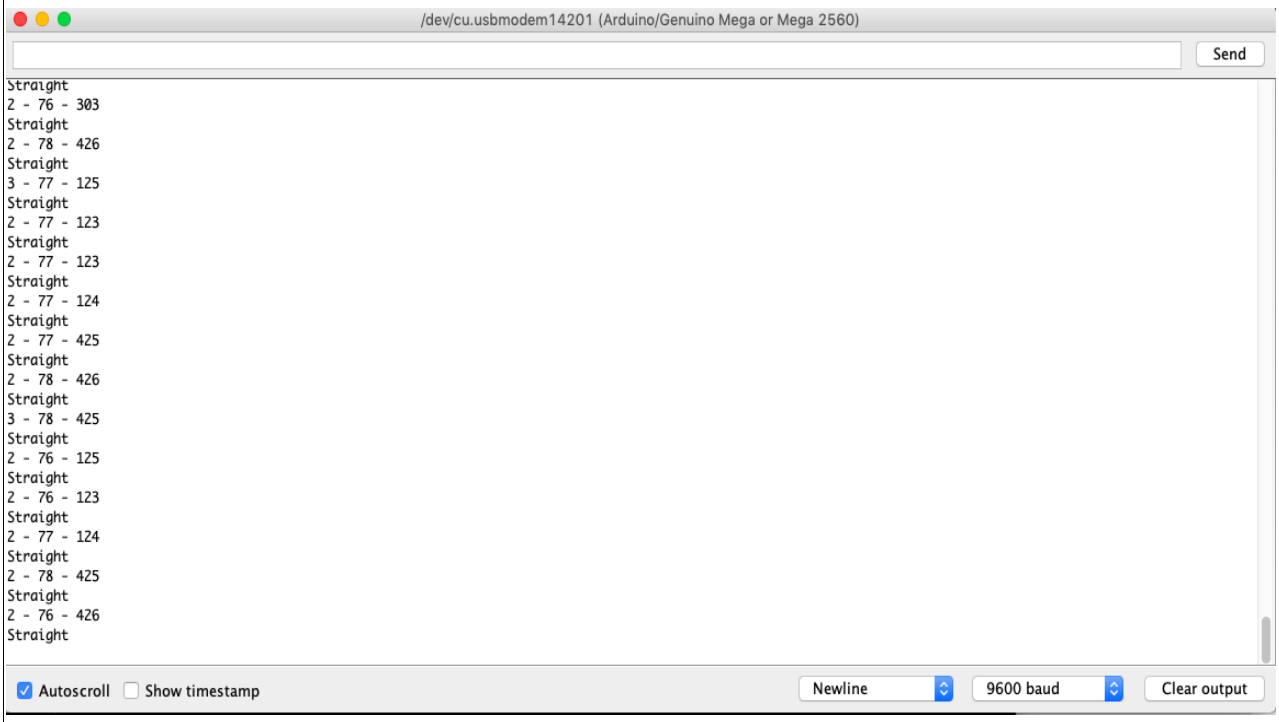
```
}
else if ((LeftSensor<20) && (RightSensor>20) && (FrontSensor>20))
{
  //stopmove();
  Serial.println("Straight");
  initializeSpeed();
  goStraight();
  path=path+"S";
}
else if ((LeftSensor<20) && (RightSensor<20) && (FrontSensor>20))
{
  //stopmove();
  Serial.println("Straight Turn");
  initializeSpeed();
  goStraight();
  path=path+"S";
}
else if ((LeftSensor<20) && (RightSensor>20) && (FrontSensor<20))
{
  //stopmove();
  Serial.println("Right Turn");
  initializeSpeed();
  turnRight();
  path=path+"R";
}
else if ((LeftSensor>20) && (RightSensor<20) && (FrontSensor<20))
{
  //stopmove();
  Serial.println("Left Turn");
  initializeSpeed();
  turnLeft();
  path=path+"L";
}
else
{
  //stopmove();
  Serial.println("U-Turn");
  initializeSpeed();
  takeUturn();
  path=path+"R";
}

//delay(200);

}
```

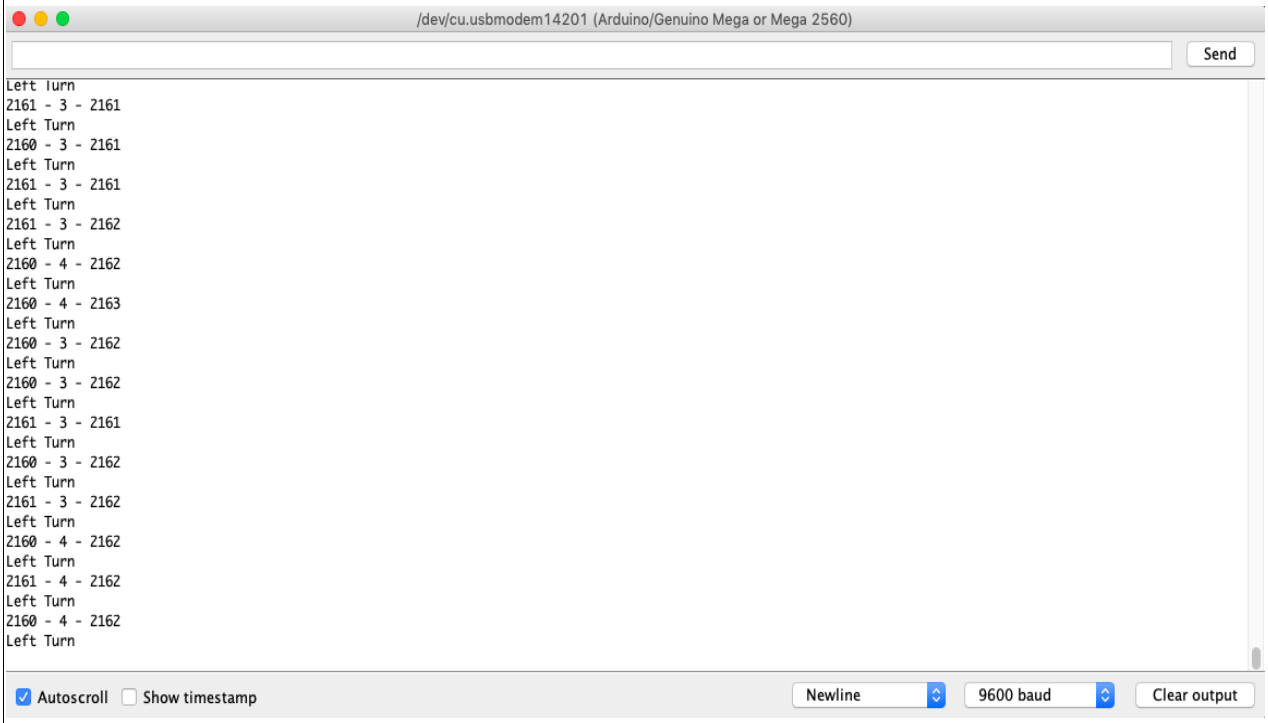
CODE FOR PATH OPTIMIZATION

```
String oldPath = "";
//String path = "LBLLLBSBLLBSLL";
String path = "SLBLSLBLSR";
String toFind[] = {"LBR","LBS", "LBL", "RBL", "SBL"};
```

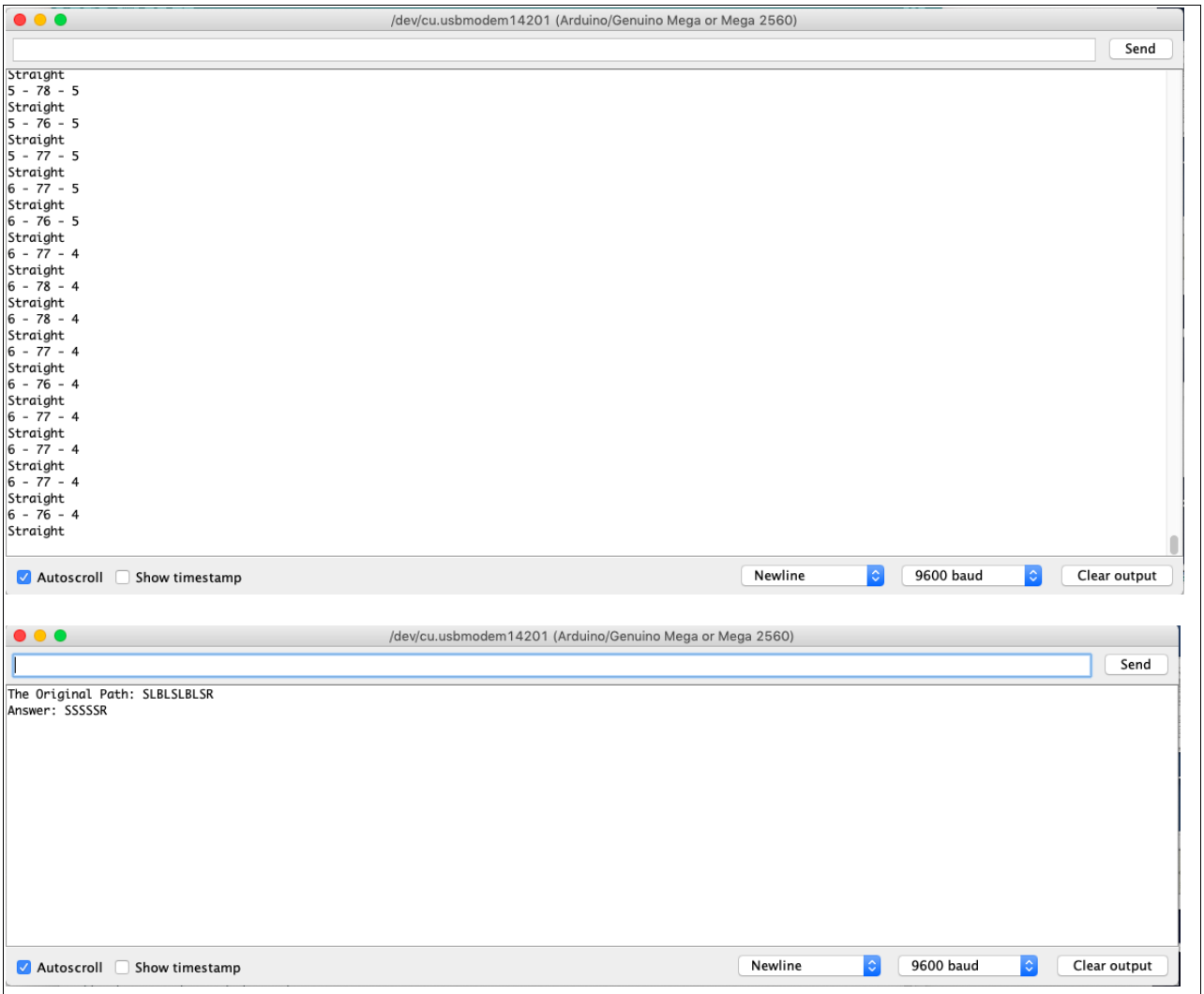
```
String toReplace[] = {"B","R","S","B","R"};
int possibilities = sizeof(toFind)/sizeof(String);
int counter = 0,i;

void optimizePath()
{
  while(true)
  {
    oldPath = path;
    for(i=0; i<possibilities; i++)
      path.replace(toFind[i], toReplace[i]);
    //Serial.println(path);
    if(oldPath == path)
      break;
    counter++;
  }
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  optimizePath();
  Serial.print("Answer: ");
  Serial.println(path);
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

SCREENSHOTS:




/dev/cu.usbmodem14201 (Arduino/Genuino Mega or Mega 2560)
Send
Left Turn
2161 - 3 - 2161
Left Turn
2160 - 3 - 2161
Left Turn
2161 - 3 - 2161
Left Turn
2161 - 3 - 2162
Left Turn
2160 - 4 - 2162
Left Turn
2160 - 4 - 2163
Left Turn
2160 - 3 - 2162
Left Turn
2160 - 3 - 2162
Left Turn
2161 - 3 - 2161
Left Turn
2160 - 3 - 2162
Left Turn
2161 - 3 - 2162
Left Turn
2160 - 4 - 2162
Left Turn
2161 - 4 - 2162
Left Turn
2160 - 4 - 2162
Left Turn
Autoscroll
Show timestamp
Newline
9600 baud
Clear output


/dev/cu.usbmodem14201 (Arduino/Genuino Mega or Mega 2560)
Send
Right Turn
6 - 4 - 427
Right Turn
6 - 4 - 428
Right Turn
6 - 4 - 426
Right Turn
6 - 4 - 444
Right Turn
7 - 4 - 428
Right Turn
6 - 4 - 429
Right Turn
6 - 4 - 427
Right Turn
6 - 4 - 426
Right Turn
6 - 4 - 427
Right Turn
6 - 4 - 425
Right Turn
6 - 4 - 426
Right Turn
6 - 4 - 426
Right Turn
6 - 4 - 426
Right Turn
6 - 5 - 426
Right Turn
Autoscroll
Show timestamp
Newline
9600 baud
Clear output


/dev/cu.usbmodem14201 (Arduino/Genuino Mega or Mega 2560)
Send
Straight
5 - 78 - 5
Straight
5 - 76 - 5
Straight
5 - 77 - 5
Straight
6 - 77 - 5
Straight
6 - 76 - 5
Straight
6 - 77 - 4
Straight
6 - 78 - 4
Straight
6 - 78 - 4
Straight
6 - 77 - 4
Straight
6 - 76 - 4
Straight
6 - 77 - 4
Straight
6 - 77 - 4
Straight
6 - 77 - 4
Straight
6 - 77 - 4
Straight
6 - 76 - 4
Straight
Autoscroll
Show timestamp
Newline
9600 baud
Clear output

/dev/cu.usbmodem14201 (Arduino/Genuino Mega or Mega 2560)
Send
The Original Path: SLBLSLBLSR
Answer: SSSSSR
Autoscroll
Show timestamp
Newline
9600 baud
Clear output

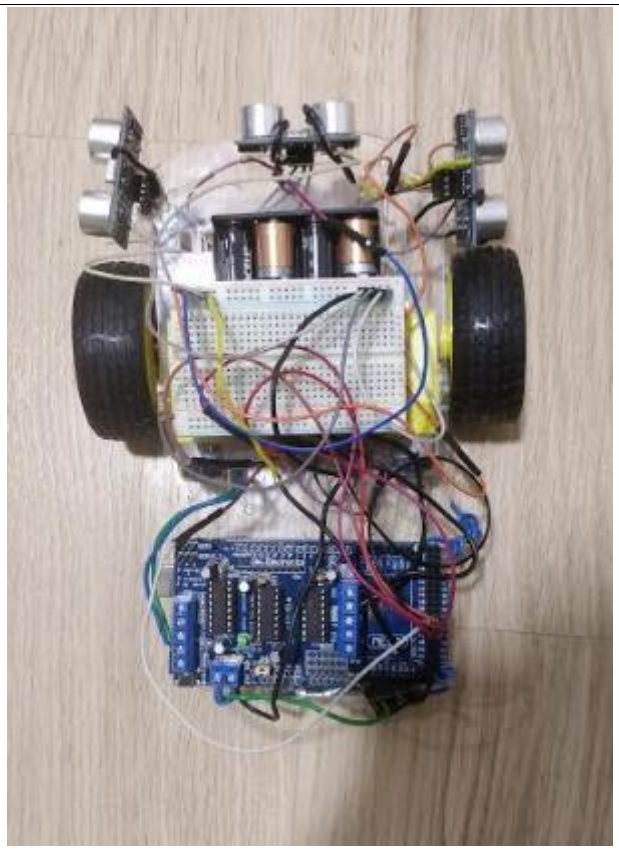


IMAGE OF THE BOT

ABSTRACT:

The project involves verifying and analyzing the logic used in the process of solving mazes, where there is only a beginning and an end, autonomously, without human intervention. We used prototyping platform electronics free hardware , the ARDUINO , which through its programming language, the Wiring, a derivative of C/C++, you can write the program with the necessary logic for the development of automation around robot, where there are three ultrasonic sensors for measuring and monitoring the distances from the walls of the maze.
At the base that supports are three wheels, with two individual motors controlled by an integrated circuit connected to Arduino manipulating the speeds of these motors and a wheel without a motor in order to get around. A real model was introduced to solve this kind of problem and was analyzed both the theoretical and practical as this solution was made

MOTOR SHIELD:

The shield contains two L293D motor drivers and one 74HC595 shift register. The shift register expands 3 pins of the Arduino to 8 pins to control the direction for the motor drivers. The output enable of the L293D is directly connected to PWM outputs of the Arduino. To increase the maximum current, the L293D allows extra chips with "piggyback". Piggyback is soldering one, two, or three extra L293D drivers on top of the L293D drivers on the board to increase the maximum current. The L293D allows parallel operation.The Motor Shield is able to drive 2 servo motors, and has 8 half-bridge outputs for 2 stepper motors or 4 full H-bridge motor outputs or 8 half-bridge drivers, or a combination.

ULTRASONIC:

The Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object.
The trigPin and EchoPin are connected to the arduino's digital pins, which are PWM enabled (here we have considered pins 9 and 10) through which the data is collected. The voltage and ground connections are made with the Arduino.

ARDUINO MEGA:

The Mega is a microcontroller that has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

DC MOTOR:

A **DC motor** is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic; to periodically change the direction of current flow in part of the motor.

INDIVIDUAL CONTRIBUTION

RAGHAV:

My contribution for the project was to code the Arduino Mega in such a way that the Robot Car would move in different direction when the Ultrasonic Sensors attached to the Arduino receives a value lesser than the threshold, which means, there is an obstacle present nearby and the robot should find an alternative path to reach the destination. The Arduino would then make the DC Motor move by giving the suitable commands (i.e., to move Left, Right or Straight). This way, the robot will avoid the obstacle and move from point of origin to the destination.

HEMANTH:

My contribution involves connecting the Arduino Motor Shield, which allows to easily control motor direction and speed using an Arduino. The pins of the official Arduino motor shield will only align with Arduino Uno Rev.In order to make it work with older versions of the Arduino, I had to trim a few pins off the motor shield. Then, I Inserted the motor shield pins into the socket of the Arduino Uno. By addressing these pins I selected a motor channel to initiate, specified the motor direction (polarity), set motor speed (PWM), stop and started the motor, and monitored the current absorption of each channel. Then, I plugged the Arduino into the computer's USB port and worked on debugging the Arduino development environment. I also helped integrate the bot's ultrasonic sensors and helped manage the structural layout of the components on the map. I also helped in create and design a maze for the hardware implementation.

NATRAJ

I with my team members collected the ultrasonic sensors and the Arduino board. I then identified the pins required for the board. I then connected the wires from the board and from the breadboard. Then based on the requirement I connected the ultrasonic sensors in robot and then connected to the breadboard and driver. I then placed the ultrasonic sensors based on positions and requirement needed for the robot. In addition, the then helped the team to code for the ultrasonic sensors for its function in the program of the robot. I helped createthe map required for the maze-solving robot with obstacles in the map

DRISHTI

I worked on the software part of the project. The algorithm to be implemented had very little documentation hence it was to be coded from scratch. Firstly, I analysed what is to be done at each combination of ultrasonic sensor values and assign the command of turning or going straight accordingly. After coding and testing the robot movements, I also adjusted the motor speed to make the wheels move in sync and reduce the drift effect. Then we coded the algorithm of using the left hand rule to solve the maze. Finally, the code optimization was done by replacing redundant moves and breaking loops

AISWARYA

My contribution to this project was mainly to work on the algorithm for optimising the path for the robot. I worked on coding the mechanism for the robot to sense the path to travel in real time using ultrasound sensors, which is taken as the input string for the algorithm. The optimization is done to find the best path by traversing the path, noticing and eliminating loops wherever possible. This is done in the least number of lines of code and the most efficient manner. I also helped with the hardware connections and motor movement whenever necessary and provided my share of solving problems faced during implementation and integration of the project.

CONSTRAINTS:

Some of the reasons that robots don’t drive straight

- Variations in the friction or other internal factors affecting performance of the motors.
- Carpet pile direction.
- Centre of balance / symmetry. Loss of symmetry in terms of construction, weight, material, wheels, axis of rotation, and balance.
- Robot construction. Uneven wheels axis and frame, and that the beams and other wires may not be firmly connected to each other.
- Caster wheel not aligned, sticking.
- Uneven pressure on the start button, the robot may not start straight.
- Motors have a small amount of internal gear slop (or backlash) between the encoders and the drive shaft that may affect initial trajectory.
- Wheels or gears catching or rubbing more on one side than the other.
- Different wheels might have slightly different circumferences.
- Dust/dirt causing a loss of traction.
- Multithreading is not possible in Arduino Mega since it runs only on one processor. It can only process one task at a time and hence cannot multitask at a given point of time. This came as a disadvantage to our machine, as multitasking was required in the mentioned

setup. The Ultrasonic sensors need to continuously sense the environment in one thread as well as the motor should be properly rotate at the same time in the other thread.

- If the Ultrasonic detected any obstacle, the thread running the Ultrasonic sensing would Interrupt the other thread making the wheels rotation and make them rotate in a way (i.e, left, right or straight) that would avoid the obstacle. This is not possible in the Arduino Mega. Therefore, a single thread runs both the processes due to which the one of the process either stops or continues performing the older action till the other process is completed. This can lead to inaccuracies in the movement of the robot which could lead to undesirable results.