1. INTRODUCTION

Our project is about a line tracking robot and obstacle avoiding robot which is equipped with sensors mounted on a four-wheeled platform. While the left and right IR sensors detect where the robot is on the line, the ultra sonic will continusley send ultra sonic waves to detect if there is an obstacle infront of it or not , the servo motor will help the ultra sonic to move 0 degree to 180 degree to take a decision to take a right or left line to take the side where there is no obstacle, the L298N driver is connected to 4 wheels which ensures precise control over the four-wheel motors with the help of the batteries it help to move the 4 wheels forward .

The main purpose of line-following robots is to achieve a specific goal or task by moving on a specific line or path. Some of the general purposes of line following robots are:

- **Educational Projects:** Line-following robots are often used in educational projects. Students and hobbyists can choose such projects to improve their robotics and programming skills.

- **Industrial automation:** In the field of industrial automation, line-following robots can be used to move materials or perform certain tasks on production lines.

- **Entertainment and Competitions:** Line-following robots can be used in robot competitions or recreational events. Competitors can try to get their robot to the finish line as quickly as possible by following the line on a specific race track.

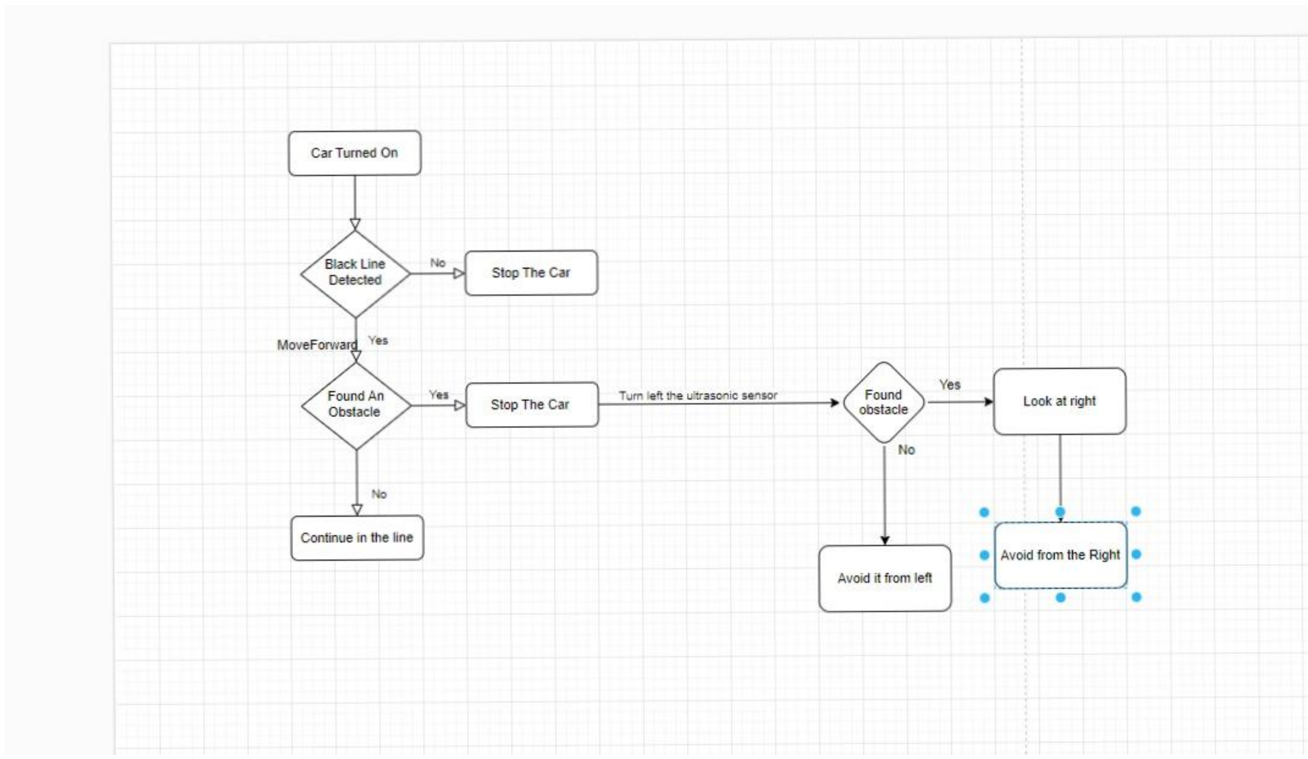
- **Magazine and Exhibition Shows:** Line-following robots can be used for demonstration purposes in store windows or at science and technology exhibitions. These types of robots can be used to engage visitors and promote technology.

- **Home Automation and Service Robots:** Line follower robots can be used as home automation or service robots. For example, a line-following robot can perform certain tasks around the home by traveling along a certain line, such as cleaning the house or carrying small objects.

- **Education and Awareness:** Line follower robots can be used for education and awareness raising on robotic technologies and automation. These robots can be used to provide information on these topics to students, teachers, and the general public.

These purposes are examples of the uses of line-following robots and obstacle avoiding .

- ■ We have designed a flowchart that explains the way that the car robot and the sensor works together .

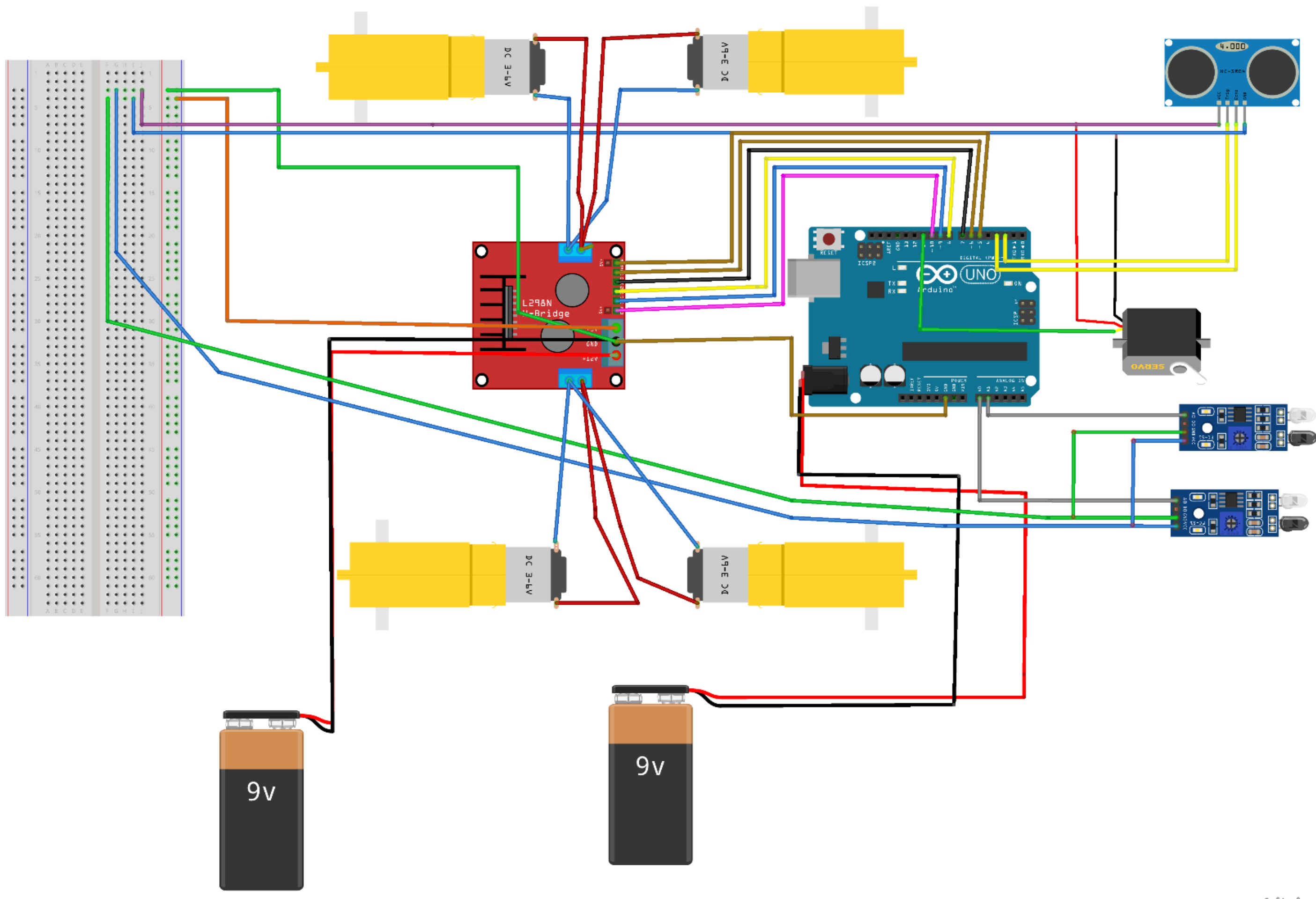


2. COMPONENTS

COMPONENTS	AMOUNT
• **Solderless Breadboard**	X 1
• **4-wheel Robot Car Kit**	X 1
• **Arduino UNO**	X 1
• **IR Sensor**	X 2
• **L298 Motor Driver**	X 1
• **Mini Servo Motor SG90**	X 1
• **Ultrasonic Sensor Holder**	X 1
• **Ultrasonic Sensor HC-SR04**	X 1
• **4Pcs Smart Robot Car Tyers Wheels**	X 1
• **Male to Female Jumper Wires**	X 1

• **Male to Male Jumper Wires**	X 1
• **On/Off Switch**	X 1
• **Battery Holder 4 Cell**	X 1
• **18650 battery Lithuim ion cell**	X 2
• **RGB**	X1
• **Buzzer**	X1
• **Resistiors**	X4

3. DESIGN



Components	Pins
buzzerPin	13
Echo	3
trigger	2
Servo	11
RGB	R= 12 , G = 1 , B = 4
IRL(Left IR)	A0
IRR(Right IR)	A1
Enable A (L298n Driver)	10
Enable B (L298n Driver)	5
In1	6

In2	**7**
In3	**8**
In4	**9**

4. CODE

```
/*
 * LINE FOLLOWER AND OBSTACLE AVOIDING CAR ROBOT
 */
 #include <Servo.h>
 #define buzzerPin 13
// Ultra Sonic Sensor
```

```
#define echo 3
#define  trigger 2
// Servo Motor
#define servo  11
#define black 1
#define white 0
const int R = 12, B=4, G=1;
Servo myservo;
void forward();
void turnLeft();
void turnRight();
void Stop();
long getObstacleDistance();
void avoidObstacle();
int checkblackline(int *s1, int *s2);
// Sensor pins
const int IRL = A0;   // Analog pin 0
const int IRR = A1;   // Analog pin 1
// Motor control pins
const int pwmA = 10;
const int MOT0 = 6;
const int MOT1 = 7;
const int MOT2 = 8;
const int MOT3 = 9;
const int pwmB = 5;
int S1, S2;
void setup() {
  pinMode(R,OUTPUT);
  pinMode(B,OUTPUT);
  pinMode(G,OUTPUT);
  pinMode(buzzerPin,OUTPUT);
  Serial.begin(9600);
  Serial.println("Start");
  //UltraSonic
pinMode(echo, INPUT );// declare ultrasonic sensor Echo pin as input
pinMode(trigger, OUTPUT); // declare ultrasonic sensor Trigger pin as Output
//servo motor
myservo.attach(servo);
myservo.write(70);
//Motor Setup
pinMode(MOT0, OUTPUT);
  pinMode(MOT1, OUTPUT);
  pinMode(MOT2, OUTPUT);
  pinMode(MOT3, OUTPUT);
  pinMode(IRL, INPUT);
  pinMode(IRR, INPUT);
  analogWrite(pwmA, 150);
  analogWrite(pwmB, 150);
```

```
}

void loop() {
  //for RGB
  for(int r = 0; r<2;r++){
    for(int g=0; g<2;g++){
      for (int b = 0;b<2;b++){
        digitalWrite(R,r);
        digitalWrite(G,g);
        digitalWrite(B,b);
        delay(150);
      }
    }
  }
  //-----
   S1 = digitalRead(IRL);
   S2 = digitalRead(IRR);
  // Conditions to drive line follower appropriately
  if (S1 == black &&  S2 == black) { // It's black wrongly written
    // Forward
    digitalWrite(MOT0, LOW);
    digitalWrite(MOT1, HIGH);
    digitalWrite(MOT2, LOW);
    digitalWrite(MOT3, HIGH);
  } else if (S1 == white && S2 == white) {
    // Stop
    digitalWrite(MOT0, LOW);
    digitalWrite(MOT1, LOW);
    digitalWrite(MOT2, LOW);
    digitalWrite(MOT3, LOW);
    checkblackline(&S1,&S2);
  } else if (S1 == black && S2 == white) {
    // Right
    digitalWrite(MOT0, LOW);
    digitalWrite(MOT1, LOW);
    digitalWrite(MOT2, LOW);
    digitalWrite(MOT3, HIGH);

  }
  else if (S1 == white && S2 == black) {
    // Left
    digitalWrite(MOT0, LOW);
    digitalWrite(MOT1, HIGH);
    digitalWrite(MOT2, LOW);
    digitalWrite(MOT3, LOW);

  }
  long obstacleDistance = getObstacleDistance();
if(obstacleDistance < 20)
```

```
{
  avoidObstacle();
}

  Serial.println("round");
}
long getObstacleDistance() {
  // Ultrasonic sensor distance measurement
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  long duration = pulseIn(echo, HIGH);
  long distance = ((duration/29)/2);
  Serial.println("Found Obstacle");
  return distance;
}
void avoidObstacle() {
  Stop();
  digitalWrite(buzzerPin,HIGH);
  delay(200);
  digitalWrite(buzzerPin, LOW);
  Serial.println("Found Obstacle-1");

  // Rotate servo to look left
  myservo.write(0);
  delay(500);
  long leftDistance = getObstacleDistance();
  delay(500);

  // Rotate servo to look right
  myservo.write(140);
  delay(500);
  long rightDistance = getObstacleDistance();
  delay(500);

  // Bring servo to the center
  myservo.write(70);
int S1 = digitalRead(IRL);
  int S2 = digitalRead(IRR);

  // Decide which way is clear
  if (leftDistance > rightDistance) {
    // Turn left
    turnLeft();
    delay(500);
    forward();
    delay(600);
    turnRight();
    delay(400);
    forward();
    delay(600);
    turnRight();
    delay(400);
    forward();
    delay(600);
    turnLeft();
    delay(500);
    checkblackline(&S1,&S2);
    Serial.println("Turn Left");
  } else {
    // Turn right
    turnRight();
    delay(500);
    forward();
    delay(600);
    turnLeft();
    delay(400);
    forward();
    delay(600);
    turnLeft();
    delay(400);
    forward();
    delay(500);
    turnRight();
    delay(400);
    checkblackline(&S1,&S2);
      Serial.println("Turn Right");
  }
}

void forward() {
  analogWrite(pwmA, 150);
  analogWrite(pwmB, 150);
  digitalWrite(MOT0, LOW);
  digitalWrite(MOT1, HIGH);
  digitalWrite(MOT2, LOW);
  digitalWrite(MOT3, HIGH);
}

void turnRight()
{
  analogWrite(pwmA, 150);
  analogWrite(pwmB, 150);
digitalWrite(MOT0, LOW);
    digitalWrite(MOT1, LOW);
    digitalWrite(MOT2, LOW);
    digitalWrite(MOT3, HIGH);
}
void turnLeft()
{
  analogWrite(pwmA, 150);
  analogWrite(pwmB, 150);
  digitalWrite(MOT0, LOW);
    digitalWrite(MOT1, HIGH);
    digitalWrite(MOT2, LOW);
    digitalWrite(MOT3, LOW);
}

void Stop()
{
    digitalWrite(MOT0, LOW);
    digitalWrite(MOT1, LOW);
    digitalWrite(MOT2, LOW);
    digitalWrite(MOT3, LOW);
}

int checkblackline(int *s1, int *s2) {
  // Right
  analogWrite(pwmA, 150);
  analogWrite(pwmB, 150);
  digitalWrite(MOT0, LOW);
  digitalWrite(MOT1, LOW);
  digitalWrite(MOT2, LOW);
  digitalWrite(MOT3, HIGH);
  delay(600);  // Adjust this delay based on your robot's turn speed
  (*s1) = digitalRead(IRL);
  (*s2) = digitalRead(IRR);// Check for black line
  if ((*s1) == black && (*s2) == black) {
    return 0;  // Black line detected, no need to turn left
  }
  // Left
  analogWrite(pwmA, 150);
  analogWrite(pwmB, 150);
  digitalWrite(MOT0, LOW);
  digitalWrite(MOT1, HIGH);
  digitalWrite(MOT2, LOW);
  digitalWrite(MOT3, LOW);
  delay(600);  // Adjust this delay based on your robot's turn speed
  (*s1) = digitalRead(IRL);
  (*s2) = digitalRead(IRR);
  Serial.println("turned Right and Left");
  // Check for black line
  if ((*s1) == black && (*s2) == black) {
    return 0;  // Black line detected, turn successful
  }
  // No black line detected after turning right and left
  return 1;
}*/
```

5. DEMO

We have attached a demo video in the file attached , which shows the movement of the car robot .

5.1. Initial State

The line-following robot used in our project initially stands in a certain position. The robot's motors are initially stopped, and its line tracking sensors are ready for interaction. The robot's starting coordinates are: x=0, y=0. This specifies the robot's starting position. Line tracking sensors are initially active to detect color changes in their surroundings. Our first task is for the robot to detect the line around it and start moving. In the initial state, the robot waits at a certain position on the line. Its sensors are ready to detect the line.

5.2. Entrance

This section is important to launch the project and explain to the audience how to use it or how the project works. The first step to start the line follower robot is to connect the robot's 18650 lithuim ion Battery . Next, the necessary connections are made to activate the Arduino board and motor driver. The basic materials we need to realize this project are; Arduino board, motor driver, line tracking sensors, etc. Before connecting these materials, we must make sure that they are in the correct position. We must install the program containing the line tracing algorithm on the Arduino board. This step is important to ensure that the robot detects the line and there’s is no obstacle infront it and strat moves. By calibrating the line tracking sensors, we can enable the robot to follow the line around it more precisely. This step is important to ensure correct operation of the sensors. After connecting the robot's power supply, the line tracking sensors begin to detect color changes in its surroundings and the robot is ready to move. We can use a start button (switch) to start the movement of the robot.

5.3. Basis Function

The robot detects color changes in the surrounding ground through line tracking sensors and successfully follows the line. The motors of the line-following robot are controlled according to data from sensors. Engines are adjusted in speed and direction based on position on the line. If the robot goes out of line while following the line, a programmed response ensures that the robot returns to the correct path. The robot has the ability to turn on the line. For example, when it reaches a turning point, it rotates by following the line. Thanks to the speed settings programmed in the project, the speed at which the robot moves while following the line can be adjusted. The robot uses ultrasonic distance sensors to detect obstacles in its environment. Controls engines appropriately to avoid detected obstacles. In a specific situation or depending on the user's request, you can issue commands that affect the robot's movement. This makes the project more interactive. The robot completes its task by following a certain distance along the line or reaching a certain target. Also the robot is designed to avoid any obstacle infront of it with the help of the ultra sonic sensor and with the help of the servo motor which is attached to it gives the ultra sonic ability to rotate left and right in case found an obstacle infront of it to avoid the front obstacle.

5.4. Usage Scenarios

Straight Line Tracking: The robot can follow a certain distance, moving successfully in a straight line.

Following Curved Paths: Thanks to line tracking sensors, the robot can follow curved paths. This demonstrates the robot's ability to flexibly follow various paths.

Following Zigzag Lines: The robot can successfully follow zigzag-shaped lines. This demonstrates the ability of the line-following algorithm to adapt to various patterns.

Obstacle Avoidance Scenario: Thanks to ultrasonic distance sensors, the robot detects obstacles around it and continues to follow its line by avoiding the obstacle.

Quick Turns: When it reaches a certain point, the robot quickly turns and follows its line. This demonstrates the robot's ability to react quickly to changes in its environment.

Stop and Start: We can control the line-following process by giving commands to stop and restart the robot in certain situations.

Special Functions and Tasks: We can customize the robot's behavior with special functions and tasks programmed into the project. For example, we can add a function to sound an alarm when it reaches a certain goal.

User Controlled Scenarios: Users can use tools such as buttons or a remote control to control the movement of the robot when it reaches a certain point or in a certain situation.

5.5. Parameter Settings

Speed Parameters: This parameter allows us to control the line tracing speed.

Sensitivity Setting: By adjusting the sensitivity of the line tracking sensors, it allows us to customize the level of response to color changes on the line.

Turn Sensitivity: This parameter allows us to achieve more controlled turns during line tracking.

Sensor Operating Range: Setting the operating range of the line tracking sensors determines how far the sensors detect when the robot follows the line.

Ultrasonic Distance Sensor Sensitivity: By determining the sensitivity of ultrasonic distance sensors, we see how effective the obstacle avoidance function is.

5.6. Errors and Solutions

Solutions to possible errors are:

Problem of not being able to follow the line: If the robot has difficulty following the line, we can check the line tracking sensors. We need to make sure that the sensors are clean and in the correct position. Additionally, we can optimize the sensor response by controlling the sensitivity settings

Sliding Problem in Turns: If the robot experiences slipping in turns, we should check the condition of the wheels. If the wheels are slipping, we must make sure that the wheels are clean and working properly. Additionally, we can fix this issue by checking the rotation sensitivity settings."

Obstacle Avoidance Problem: If the robot crashes into obstacles without detecting them, we must ensure that the ultrasonic distance sensors are working correctly and that the sensors accurately detect whether there are obstacles around them. "We can solve this problem by checking the calibration of the sensors."

Sensor Malfunctions: When line tracking sensors or ultrasonic sensors fail, we must check the connections of the sensors and replace the sensors if necessary. "We also need to make sure the sensors are calibrated correctly."

Program Error: If the robot behaves unexpectedly, we should check the Arduino program. We should review the program for syntax errors or logic errors. "If necessary, we can solve the problem by reinstalling the program."

Power Problems: In case the robot shuts down unexpectedly, we must check the power supply. We can troubleshoot power issues by checking battery levels and connections."

Speed Problems: If there are problems with the speed of the robot, we should check the motor driver settings. We can fix this problem by adjusting the speed parameters or checking the motor driver connections."

5. SIMULATION

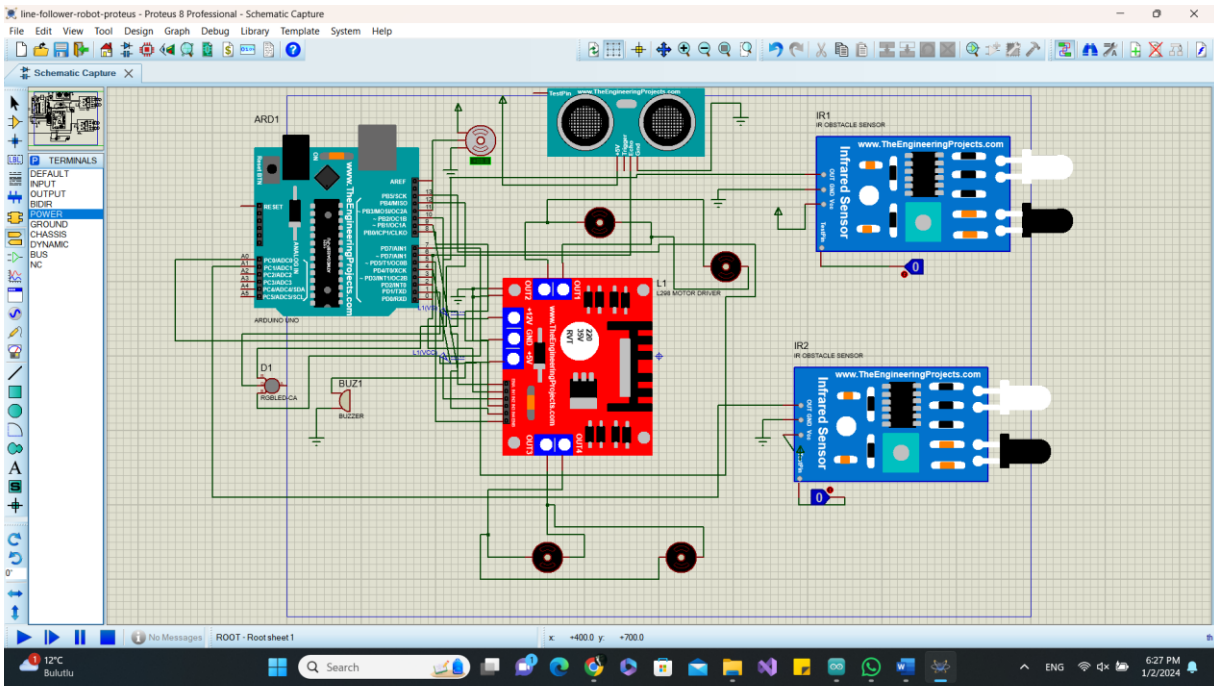


Figure: simulation of line follower and Obstacle avoiding robot circuit in proteus

- In the Proteus simulation for a 4-wheel line-following and obstacle-avoiding car with Arduino Uno:

- The car is equipped with two infrared (IR) sensors (IRL and IRR) for line following, connected to analog pins A0 and A1 on the Arduino Uno.
- Motor control is achieved through two H-bridge motor driver modules with pins MOT0, MOT1, MOT2, MOT3, pwmA, and pwmB.
- An ultrasonic sensor is employed for obstacle detection with trigger and echo pins connected to digital pins 2 and 3, respectively.
- A servo motor (servo) is used for rotating an ultrasonic sensor for effective obstacle detection.
- The Arduino Uno is connected to a buzzer (buzzerPin 13) for audible feedback during certain events.
- The simulation integrates line-following logic, where the car adjusts its direction based on input from the IR sensors.
- The ultrasonic sensor is utilized for obstacle avoidance, causing the car to stop, rotate the sensor to detect obstacles on both sides, and then resume normal operation.
- The simulation involves PWM-controlled motors to achieve variable speed control for precise movement.

- Overall, the simulation combines line following and obstacle avoidance functionalities, ensuring the car can navigate a path while adapting to environmental obstacles.