

**Project Name : Line follower Robot**

**Author: Varun Singh J R**

**Date: 27/ 10 / 24**

**Institution : Sathyabama Institute of Science and Technology**

**Department: Bachelors in Electronics and Communication**

**Abstract**

This report presents the design, implementation, and testing of a line follower robot utilizing an Arduino microcontroller, two geared DC motors, and a 5-array infrared (IR) sensor module. The primary objective of the robot is to autonomously follow a predefined path by detecting line markings, with capabilities for both smooth and sharp turns. Key components in this project include the integration of IR sensors for real-time path detection, motor drivers for precise control of movement, and efficient embedded programming to coordinate sensor data and motor actions. The project explores foundational principles of sensor-based navigation, motor control, and embedded systems, making it a valuable learning experience in robotics. Testing was conducted on varied path patterns to evaluate the robot’s responsiveness and accuracy. Overall, the line follower robot successfully demonstrates effective autonomous navigation, providing insights into embedded design, hardware-software integration, and real-world robotics applications.

**Introduction**

Background:

A line follower robot is an autonomous robot designed to detect and follow a specific path marked by a line, usually painted on the ground. These robots use various sensors to identify the line's position and make real-time adjustments to their movement. Line follower robots are commonly used in industrial applications for automated material handling, in educational settings for teaching robotics and programming concepts, and in competitions where they must navigate predefined courses.

Project Objective:

The objective of this project is to design and build a line follower robot that effectively follows a predetermined path while demonstrating key principles of robotics, including sensor integration, motor control, and embedded programming. Through this project, I aim to gain hands-on experience in the practical application of theoretical concepts in electrical engineering and robotics.

Scope:

This project focuses on building a basic line follower robot using an Arduino microcontroller, two geared DC motors, and a 5-array IR sensor module. The design aims to achieve reliable line detection and smooth navigation, including the ability to make sharp turns when needed. However, specific limitations include the robot's speed, which is capped to ensure stability and accuracy in following the line. Additionally, the current design does not incorporate advanced features such as obstacle detection or adaptive speed control, which could enhance performance in more complex environments.

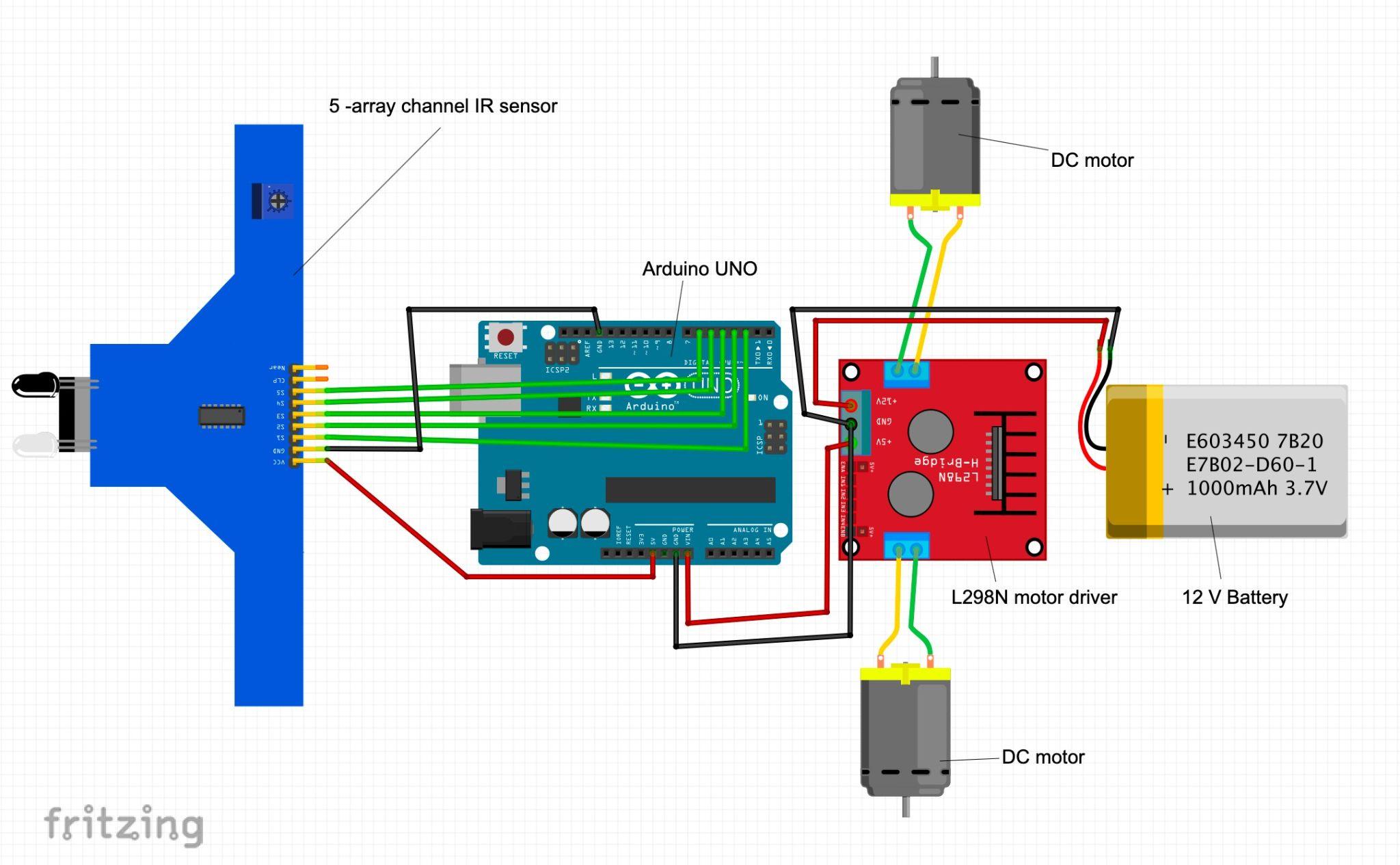
**Hardware Components**

| **Component** | **Quantity** | **Description** |
| --- | --- | --- |
| Arduino Uno | 1 | Microcontroller for processing sensor data and controlling motors |
| 5-Array IR sensor | 1 | Used for detecting the path (line) on the surface |
| DC Motors | 2 | Motors to drive the wheels |
| L298N Motor Driver | 1 | Motor driver for controlling the speed and direction of motors |
| 12V Battery | 1 | Power Source for the motors and Arduino |
| Wheels | 2 | Wheels for mobility |
| Chassis | 1 | Structure holding all components together |
| Jumper Wires | - | For circuit connections |

Additional Tools  
- Soldering Kit  
- Breadboard (for initial testing)  
- Screwdriver set

**Circuit Diagram & Schematic**

Circuit Diagram



**Software/ Coding**

The line-following robot is programmed using C/C++ in the Arduino IDE (Integrated Development Environment). The Arduino IDE provides a platform for writing, compiling, and uploading code to the Arduino board, allowing for real-time testing and debugging.

Code Overview:

**1.Pin Configuration**

The code begins by setting up pin configurations for the IR sensors and motor driver:

| int s1 = 2;  int s2 = 3;  int s3 = 4;  int s4 = 5;  int s5 = 6; |
| --- |

These variables represent each of the five IR sensors connected to pins 2-6 of the Arduino. The sensors detect the presence of a line underneath them, giving a binary value (0 or 1).

| int IN1 = 11;  int IN2 = 12;  int IN3 = 13;  int IN4 = 7;  int ENA = 9;  int ENB = 10; |
| --- |

These variables correspond to the motor driver pins that control the left and right motors of the robot. IN1 and IN2 control one motor, while IN3 and IN4 control the other. ENA and ENB are used for speed control with PWM signals.

**2. setup() Function**

| void setup() {  //Pin O/I (sensor)  pinMode(s1, INPUT);  pinMode(s2, INPUT);  pinMode(s3, INPUT);  pinMode(s4, INPUT);  pinMode(s5, INPUT);  //Pin O/I (Motor Driver )  pinMode(IN1, OUTPUT);  pinMode(IN2, OUTPUT);  pinMode(IN3, OUTPUT);  pinMode(IN4, OUTPUT);  pinMode(ENA, OUTPUT);  pinMode(ENB, OUTPUT);  Serial.begin(9600);  } |
| --- |

In the setup()function, each sensor pin is configured as an input. The motor control pins are set as outputs. Additionally, Serial.begin(9600); initializes serial communication for debugging purposes, allowing sensor values to be monitored in the serial monitor.

**3. loop() Function**

The loop() function is the main part of the code that continuously reads the sensor values, processes them, and controls the robot’s movement based on the detected line position.

| // Reading inverted sensor value  v1 = !digitalRead(s1);  v2 = !digitalRead(s2);  v3 = !digitalRead(s3);  v4 = !digitalRead(s4);  v5 = !digitalRead(s5); |
| --- |

Here, the IR sensor values are read and inverted (!digitalRead) so that a high value (1) indicates the presence of a line (assuming the sensors give a low reading when detecting a dark line).

The sensor values are printed to the serial monitor for debugging:

| Serial.print(v1);  Serial.print(v2);  Serial.print(v3);  Serial.print(v4);  Serial.print(v5);  Serial.println(); |
| --- |

**Decision Logic**

The code contains if-else conditions to control the robot's direction:

-Sharp Right Turn: If the line is detected only on the far-right sensor **(v5 == 1)**, the robot makes a sharp right turn.

- Sharp Left Turn: If the line is detected only on the far-left sensor **(v1 == 1)**, the robot makes a sharp left turn.

- Right Turn: If the line is detected on the two rightmost sensors **(v4 == 1, v5 == 1)** or adjacent right positions, the robot turns right.

- Left Turn: If the line is detected on the two leftmost sensors **(v1 == 1, v2 == 1)** or adjacent left positions, the robot turns left.

- Straight : If the middle sensor **(v3 == 1)** detects the line with no sensors on either side, the robot moves straight.

**4. Motor Control Functions**

Each movement command corresponds to a function that controls the motors by setting the appropriate pins high or low.

**sharp\_left\_turn():**

| void sharp\_left\_turn() {  digitalWrite(IN1, HIGH);  digitalWrite(IN2, LOW);  digitalWrite(IN3, LOW);  digitalWrite(IN4, HIGH);    analogWrite(ENA, 255);  analogWrite(ENB, 100);  } |
| --- |

This function makes the robot turn sharply to the left by setting different speeds for the left and right motors.

**sharp\_right\_turn():**

| void sharp\_right\_turn() {  digitalWrite(IN1, LOW);  digitalWrite(IN2, HIGH);  digitalWrite(IN3, HIGH);  digitalWrite(IN4, LOW);    analogWrite(ENA, 100);  analogWrite(ENB, 255);  } |
| --- |

Similarly, this function makes a sharp right turn.

**right\_turn() and left\_turn():**

The right\_turn() and left\_turn() functions make gentler turns by stopping one motor while running the other at full speed.

**stop():**

| void stop() {  digitalWrite(IN1, LOW);  digitalWrite(IN2, LOW);  digitalWrite(IN3, LOW);  digitalWrite(IN4, LOW);    analogWrite(ENA, 0);  analogWrite(ENB, 0);  } |
| --- |

The stop() function halts both motors by setting all control pins low and setting the speed to zero.

**straight():**

| void straight() {  digitalWrite(IN1, HIGH);  digitalWrite(IN2, LOW);  digitalWrite(IN3, HIGH);  digitalWrite(IN4, LOW);    analogWrite(ENA, 255);  analogWrite(ENB, 255);  } |
| --- |

The straight() function makes the robot move forward at full speed by setting both motors to rotate forward.

Full code is given on github page.

**Working Principle of a Line Follower Robot**

Sensor Detection: The robot has an array of IR sensors that emit infrared light and measure the reflection. A low reflection indicates the sensor is over the black line, while a high reflection indicates it’s over the white surface.

Data Processing: The sensor signals are sent to a microcontroller (like an Arduino), which processes the data to determine the robot's position relative to the line. Depending on the readings:

* If the center sensor detects the line, the robot moves forward.
* If the left sensor detects the line, the robot turns right.
* If the right sensor detects the line, the robot turns left.

Motor Control: The microcontroller sends signals to motor drivers, controlling the speed and direction of the DC motors to execute the necessary movements.

Feedback Loop: The robot continuously monitors sensor readings, allowing it to make real-time adjustments and stay aligned with the line, even when navigating curves.

**Challenges Faced**

-Optimizing Motor Control Logic: Initial motor control logic was either too fast or slow for turns, making sharp turns tricky. Fine-tuning the speed and delay values for each turn direction, especially with sharp turns, was essential to improve response time.

-Sensor Calibration Issues: The IR sensors required careful calibration to detect the line accurately on various surfaces. Adjusting the sensor thresholds for different lighting conditions and floor colors was challenging, as variations could lead to incorrect path detection.

-Uneven Surface Navigation: The robot faced difficulties in maintaining stability on uneven surfaces, which caused deviations from the path. Adjusting the chassis and optimizing wheel alignment were necessary to improve performance on these surfaces.

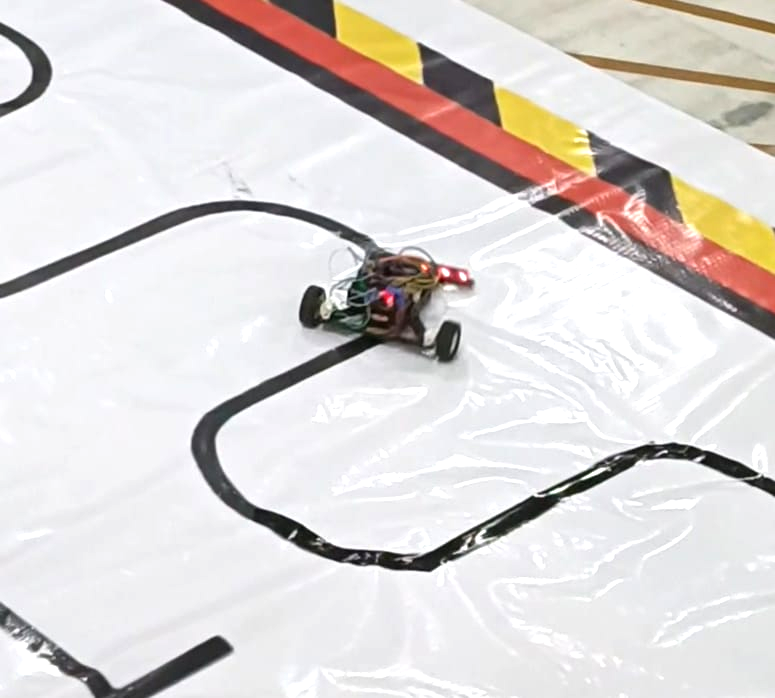
**Results**

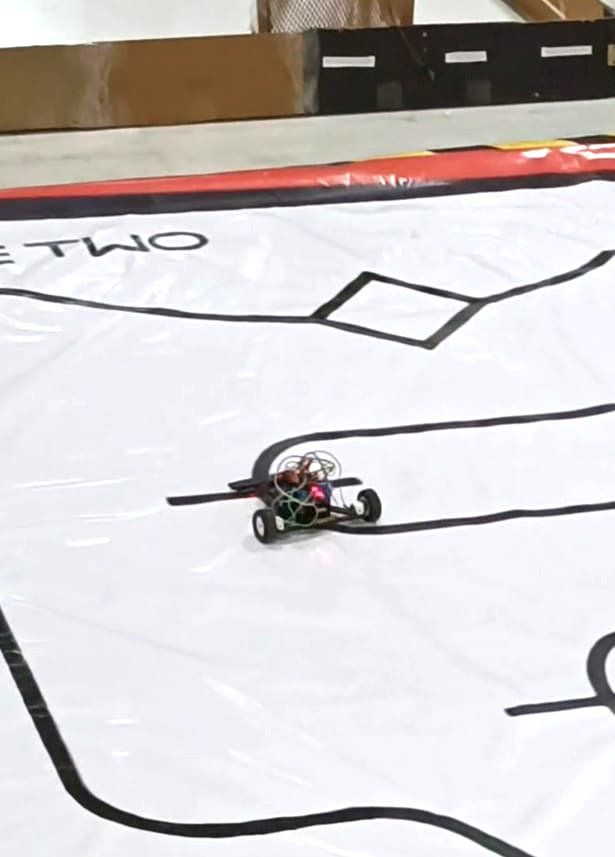
The line follower robot was able to:

- Successfully follow straight paths and standard turns.

- Navigate sharp turns with updated code logic.

- Maintain speed and stability across different surfaces.





**Future Improvements**

1.New light weight customized chassis.

2. Speed Optimization: Implement PID control for smoother turns and speed control.

3. Enhanced Power Management: Use a more efficient battery setup.

4. High rpm motors : increase the speed of the robot .

5. Use of PCB: combine motor driver , arduino nano , buck convertor in one PCB .