"Heaven's Light is Our Guide"

Rajshahi University of Engineering & Technology

Department of Electrical and Computer Engineering

Project Report

Course code: ECE 2200

Course title: Electronic Shop Practice

Project Name: LFR (Line Following Robot)

Date of submission: 18 February 2025

Submitted to Oishi Jyoti Assistant Professor Department of ECE, RUET & Md. Omaer Faruq Goni Assistant Professor Department of ECE, RUET	**Submitted by** Zanifa Islam Roll: 2110008 & Jannatul Ferdous Iqra Roll:2110030 Department of ECE, RUET

ACKNOWLEDGEMENT

We wish to express our deepest gratitude to Md. Omaer Faruq Goni, Assistant Professor, Department of Electronics and Computer Engineering (ECE), RUET, for his invaluable guidance, constant support, and insightful suggestions throughout the course of this project. His expertise and encouragement have been instrumental in the successful completion of our work. We are equally thankful to Oishi Jyoti, Assistant Professor, Department of Electronics and Computer Engineering (ECE), RUET, for her meticulous attention to detail, constructive feedback, and unwavering support. Her guidance has significantly contributed to the refinement and success of this project. Lastly, we would like to extend our heartfelt thanks to our friends and family for their continuous encouragement, support, and assistance in every possible way, which has been crucial in the successful completion of this project report. Thank you all for being an integral part of this journey.

Table of Contents

ABSTRACT

A Line Following Robot is a type of robot that can move on its own by following a black line on a surface. The surface has a different color than the line, making it easy for the robot to detect.To follow the line, the robot uses optical sensors (small devices that can "see" the line). These sensors help the robot stay on track. There are four sensors, which help the robot move smoothly and accurately.The robot moves using DC gear motors, which control its wheels. It also uses an Arduino Uno (a small computer) to run programs that adjust the motor speed, helping the robot move along the line without going off track.This project focuses on improving how the robot follows the line by fine-tuning the way it moves. Additionally, an LCD screen is added to show how far the robot has traveled.This robot can be used in factories, homes, museums, and other places where automatic movement is helpful.

List Of Figures

CHAPTER 1

1. Introduction

A Line Follower Robot (LFR) is an autonomous machine designed to follow a specific path, typically a visible line on a surface, using sensors to detect and track the line. By continuously correcting its movements based on feedback from these sensors, the robot can navigate the path accurately. This simple yet effective system has applications in various fields, including industrial automation, guidance systems, and educational projects, showcasing the integration of mechanical, electronic, and programming skills.

1.1. Background

In today's technology-driven world, it is essential not only to use technology but also to understand and create it. Engineers must have a broad knowledge base across various disciplines to foster innovation and creativity. This project bridges multiple fields, including mechanical, electronic, electrical, and programming, offering a comprehensive learning experience. It provides a visual understanding of math and science, builds logical thinking, encourages innovation, and enhances problem-solving skills.

1.2. Motivation

Inspired by the simplicity and efficiency of natural systems, such as ants following trails or vehicles adhering to road lanes, this project focuses on developing a Line Follower Robot (LFR) capable of tracking black lines and executing turns. The primary goal is to replicate these basic yet effective behaviors using a machine, demonstrating how technology can mimic natural processes to perform specific tasks. While this LFR is designed to follow black lines and navigate turns, it serves as a foundational step toward understanding automation and robotics, with potential applications in educational demonstrations, basic industrial tasks, and small-scale automation projects.

1.3. Problem Description

The primary challenge of this project was to design and build a basic Line Follower Robot (LFR) that could autonomously navigate a track marked with black lines. The robot needed to perform specific tasks such as making precise 45° and 90° turns, crossing dotted lines, and moving straight without deviating from the path. The project required integrating various components like sensors, motors, and a microcontroller to ensure smooth and accurate navigation. Additionally, the robot had to be cost-effective, easy to assemble, and reliable in its performance. The focus was on creating a system that could interpret sensor data and adjust motor speeds accordingly to maintain the desired path. This project aimed to address the fundamental aspects of robotics and automation, providing a hands-on learning experience in sensor integration and motor control.

1.4. Objectives

- Develop a functional LFR capable of following black lines and executing 45° and 90° turns with precision.
- Enable the robot to navigate and cross dotted lines without losing track or deviating from the path.
- Ensure the robot can move straight on continuous lines with minimal deviation, maintaining a steady course.
- Utilize affordable and readily available components such as Arduino Uno, L298N motor driver, N20 motors, and a 5-array IR sensor.
- Demonstrate the integration of hardware and software by writing and implementing code to control the robot's movements based on sensor feedback.
- Provide a foundational understanding of robotics and automation, showcasing how simple components can be combined to create an autonomous system.

1.5 Methodology

The project began with the selection and assembly of hardware components, including the Arduino Uno microcontroller, L298N motor driver, 5-array IR sensor, N20 motors, and wheels. The 5-array IR sensor was used to detect the black line and provide real-time feedback to the Arduino, which processed the data and controlled the motors via the L298N driver. The code was written to interpret the sensor inputs and adjust the motor speeds for smooth navigation, including making turns and crossing dotted lines. Calibration was performed to ensure the robot responded accurately to the track's layout. Multiple test runs were conducted to refine the robot's performance, ensuring it could handle various track configurations. The methodology emphasized a systematic approach, combining hardware assembly, programming, and iterative testing to achieve the desired functionality.

1.6 Limitations

- The robot's performance is affected by highly reflective or uneven surfaces, as the IR sensors may struggle to detect the line accurately.
- Sharp turns or complex paths with frequent changes in direction can cause delays in the robot's response time, leading to minor deviations.
- The battery life is limited, requiring frequent recharging or replacement, which can interrupt continuous operation.
- The robot is designed for basic tracks and may not perform well on highly intricate or dynamic paths with varying line thicknesses or colors.
- Environmental factors such as lighting conditions can impact the sensor's accuracy, affecting the robot's ability to follow the line consistently.

1.7 Summary

This project successfully demonstrates the design, construction, and programming of a basic

Line Follower Robot capable of navigating black lines, making 45° and 90° turns, and crossing dotted lines. Using components like Arduino Uno, L298N motor driver, N20 motors, and a 5-array IR sensor, the robot showcases the integration of hardware and software for autonomous navigation. The project highlights the importance of sensor feedback, motor control, and iterative testing in achieving reliable performance. While the robot has some limitations, such as sensitivity to surface conditions and limited battery life, it serves as an excellent foundation for understanding robotics and automation. This project not only provides practical insights into building autonomous systems but also encourages further exploration and innovation in the field of robotics.

CHAPTER 2

2. Technology & Literature Survey

A basic Line Follower Robot (LFR) is an autonomous robot designed to follow a predefined path, typically a black line on a white surface. It uses sensors, such as infrared (IR) sensors, to detect the line and adjust its movement accordingly. The robot follows the line by continuously reading sensor data and making real-time corrections to stay on track. The primary goal of a basic LFR is to demonstrate fundamental concepts of robotics, including sensor integration, motor control, and autonomous navigation.

2.1. Basic Operation

The basic operations of the line follower is given below-

- The 5-array IR sensor detects the black line on a white surface by emitting infrared light and measuring the reflected light intensity.
- The sensors provide real-time feedback to the Arduino Uno about the robot's position relative to the line.
- The Arduino Uno processes the sensor data to determine whether the robot is on the line, deviating to the left, or deviating to the right.
- Based on the sensor input, the Arduino calculates the necessary adjustments to keep the robot on track.
- The Arduino sends signals to the L298N motor driver to control the speed and direction of the two N20 motors.
- The robot moves straight when all sensors detect the line evenly. It makes 45° or 90° turns when the sensors detect a curve or intersection in the line. The robot can cross dotted lines by briefly losing and reacquiring the line without stopping.
- Two batteries provide power to the Arduino, sensors, and motors via the L298N motor driver. While the battery holder ensures a stable and secure connection for uninterrupted operation.

2.1.1 Block Diagram

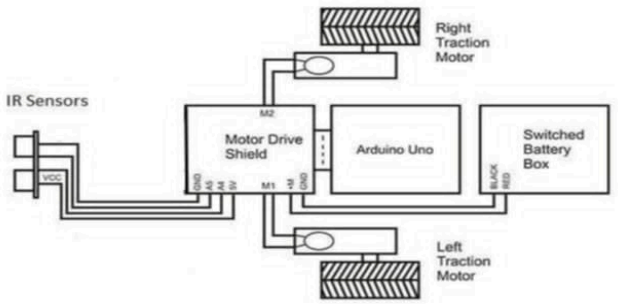

Fig 2.1.1.1 Block Diagram Of Line Following Robot

2.2 Hardware Required

Following hardwares are required to make the basic LFR (Line Following Robot)

2.2.1 Input System

2.2.1.1 TCRT5000 5-Channel IR Sensor

- Acts as the primary input system.
- Detects the black line on a white surface by emitting infrared light and measuring the reflected light intensity.

Sends real-time data (analog or digital signals) to the Arduino Uno about the robot's position relative to the line.

Fig: 2.2.1.2.1 A 5-Array IR Sensor

2.2.1.1.1 Arrangement Of The Sensors

The five sensors in the TCRT5000 5-channel IR sensor are arranged in a linear array with a gap of approximately 1-1.5 cm between each sensor. The sensors are named as follows:

- Leftmost Sensor: Positioned at the far left.
- Left Sensor: Positioned to the left of the center.
- Center Sensor: Positioned at the middle of the array.
- Right Sensor: Positioned to the right of the center.
- Rightmost Sensor: Positioned at the far right.

This arrangement allows the robot to detect the position of the black line relative to its center and make precise adjustments to stay on track.

Five sensors provide a good balance between precision and complexity.The center sensor helps the robot stay on the line, while the left and right sensors detect deviations. The leftmost and rightmost sensors provide additional information for sharper turns and

corrections. With five sensors, the robot can detect and navigate 45° and 90° turns more effectively compared to fewer sensors.

2.2.1.2 Switch

- Used as a manual input to turn the robot ON or OFF.
- Controls the power supply to the entire system.

Fig : 2.2.1.2.1 Switch

2.2.1.3 Battery Power Supply

- The two Li-ion 3.7V batteries provide the necessary voltage and current to power the Arduino Uno, motors, and sensors.
- The battery holder ensures a stable connection for the power supply.

Fig: 2.2.1.3.1 Rechargeable Li-Ion Battery

2.2.2 Output System

2.2.2.1 N20 Motors

- Act as the primary output system.
- Receive control signals from the L298N motor driver to rotate the wheels.
- Control the movement of the robot (forward, backward, left, right).

Fig: 2.2.2.1.1 N20 Motor

2.2.2.2 Wheels

- Connected to the N20 motors.
- Translate the motor's rotational motion into linear motion, allowing the robot to move along the path.

Fig 2.2.2.2.1: Rubber Wheel For N20 Motor

2.2.2.3 L298N Motor Driver

- Receives control signals from the Arduino Uno.
- Drives the N20 motors by providing the required voltage and current.
- Controls the speed and direction of the motors based on the Arduino's command.

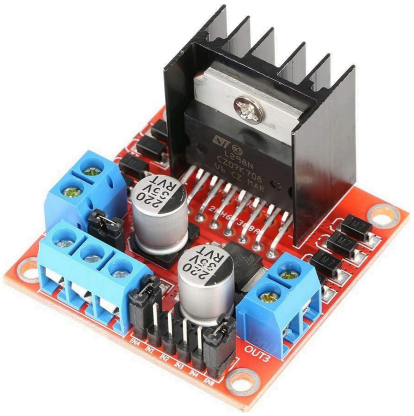

Fig 2.2.2.3.1 L298N Motor Driver

2.2.2.3.1 Features

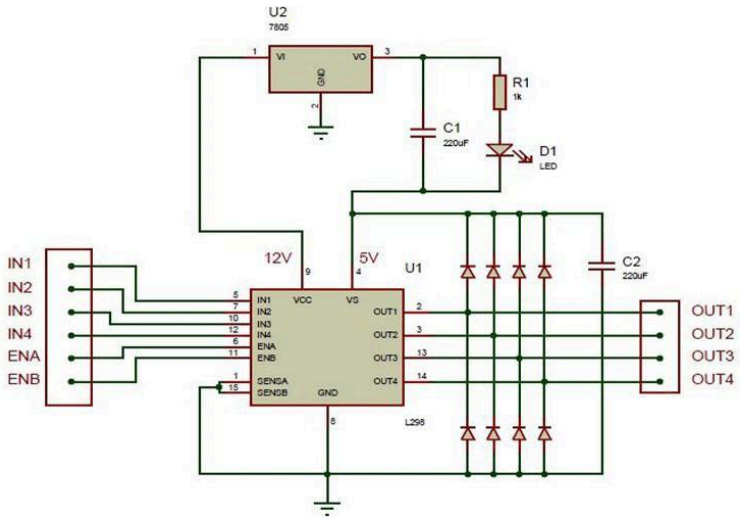

Fig 2.2.2.3.2 L298N Motor Driver Pin Diagram

2.2.2.4 Ball Caster

- Acts as a support system for the robot.
- Provides stability and balance, ensuring smooth movement.

Fig 2.2.2.4.1 Ball Caster

2.2.2.5 Arduino Uno (Output Control)

- Processes input data from the 5-array IR sensor.
- Sends output signals to the L298N motor driver to control the motors.
- Acts as the brain of the system, coordinating input and output operations.

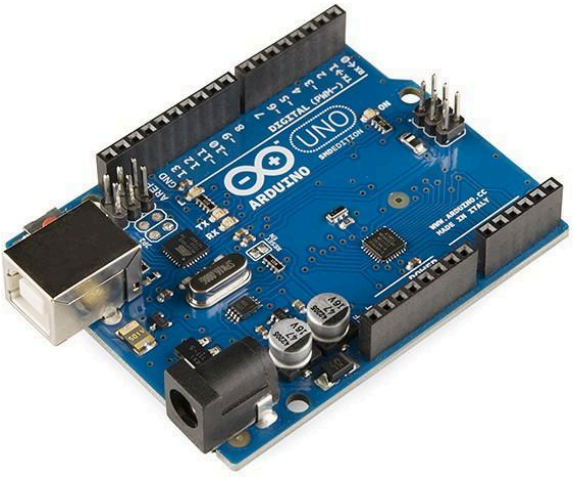

Fig 2.2.2.5.1 Arduino UNO

2.2.2.5.1 Features

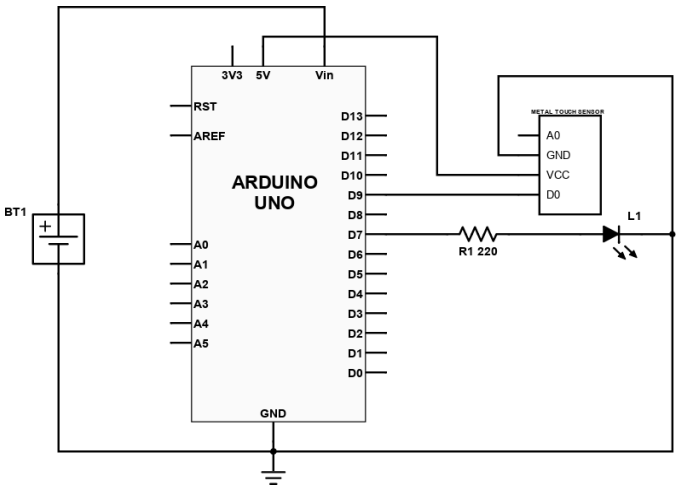

Fig 2.2.2.5.2 Arduino UNO Schematic

1. Microcontroller:

The Arduino Uno is based on the ATmega328P microcontroller, which operates at 16 MHz and has 32 KB of flash memory for storing code.

2. Input/Output Pins:

- 14 Digital I/O pins can be used for both input and output operations. Some pins have special functions:
  i) 6 PWM Pins: Pins 3, 5, 6, 9, 10, and 11 support Pulse Width Modulation (PWM) for controlling devices like motors or LEDs.
  ii) Serial Communication: Pins 0 (RX) and 1 (TX) are used for serial communication.

- 6 Analog Input Pins A0 to A5 can read analog signals (e.g., from sensors) with a resolution of 10 bits (0–1023).

3. Power Supply:

- Operating Voltage: 5V (logic level).
- Input Voltage can be powered via USB (5V) or an external power supply (7–12V recommended).
- Power Pins:
  i) 5V Pin: Provides 5V output for powering components.
  ii) 3.3V Pin: Provides 3.3V output for low-power devices.
  iii) GND Pins: Ground pins for completing circuits.
  iv) Vin Pin: Can be used to supply external voltage to the board.

4. Memory:

- Flash Memory: 32 KB (for storing the program code).
- SRAM: 2 KB (for storing variables and data during program execution).
- EEPROM: 1 KB (for storing data that persists even after power-off).

5. Communication Interfaces:

- Serial Communication: Built-in UART (via pins 0 and 1) for serial communication with other devices.
- I2C: Supported via the A4 (SDA) and A5 (SCL) pins.
- SPI: Supported via pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK).

6. USB Connectivity:

- USB Type-B Port: Used for connecting the Arduino Uno to a computer for programming and power supply.
- USB-to-Serial Converter: The board uses an ATmega16U2 or CH340 chip for USB-to-serial communication.

7. Reset Button:

A reset button is provided to restart the program or reset the microcontroller.

8. Onboard LEDs:

- Power LED: Indicates when the board is powered.
- TX/RX LEDs: Indicate data transmission and reception during serial communication.
- Pin 13 LED: Connected to digital pin 13, useful for testing and debugging.

9. Clock Speed:

16 MHz Crystal Oscillator provides the clock signal for the microcontroller, ensuring stable operation.

2.3 Software Required

The software which was used here is Arduino IDE. Arduino IDE is a software used to write, compile, and upload code to Arduino boards. It supports C/C++ programming and provides a simple interface for debugging and serial communication.

2.4 Summary

Chapter 2 provides a detailed overview of the key components used in constructing the Line Follower Robot (LFR). The central component is the Arduino Uno, a versatile microcontroller that processes sensor data and controls the robot's movements. The N20 motors, paired with 2 wheels, drive the robot, providing the necessary motion for navigation. To control these motors, the L298N motor driver is used, which interfaces with the Arduino to regulate speed and direction. Power is supplied by 2 rechargeable Li-Ion batteries, housed in a battery holder, ensuring a stable and portable energy source for the robot. The TCRT5000 5-channel IR sensor array is employed for line detection, enabling the robot to follow black lines and make precise turns. Additionally, the chapter highlights the importance of each component's role in the overall functionality of the LFR, emphasizing their integration to achieve autonomous navigation. This chapter lays the foundation for understanding how these components work together to create a functional and efficient line-following robot.

CHAPTER 3

3. Design & Implementation

3.1. Schematic

The schematic of the “Line Following Robot (LFR) ” is shown in the figure. The main component is the Arduino UNO.

The main features incorporated into the hardware are given below-

- Arduino Uno (Microcontroller)
- TCRT5000 5-Channel IR Sensor Array (Line Detection)
- L298N Motor Driver (Motor Control)
- N20 Motors (2 Motors for Movement)
- 2 Wheels (Attached to N20 Motors)
- 2 3.7V Li-Ion Batteries (Power Supply)
- Battery Holder (Holds the Batteries)
- Jumper Wires (Connections)
- Chassis (Robot Body)

Each of the hardware is dissected and was designed separately for their functionality and later incorporated as one whole application. This helped in the debugging processes. The schematic circuit is given below-

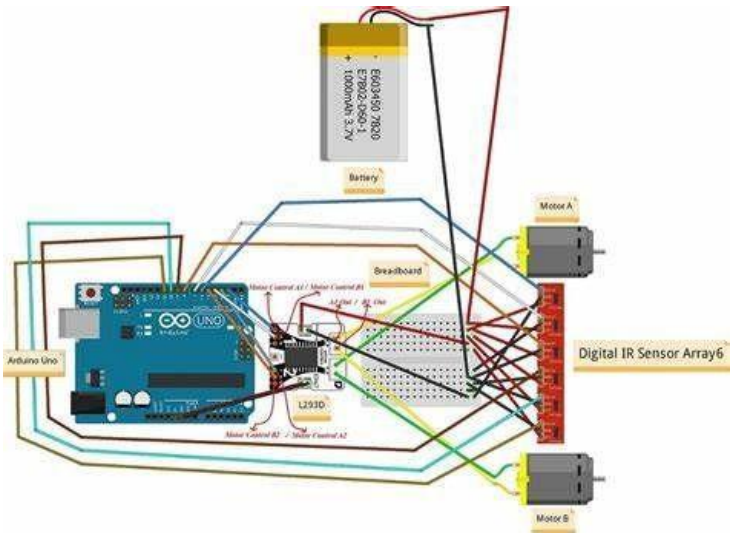

Fig 3.1.1 Connections of Line Following Robot

3.2 Arduino Working Logic

The Arduino processes input from five IR sensors to determine the position of the line and adjust the motor speeds accordingly. Based on the detected sensor values, the system calculates an error, which is then fed into a PID controller to generate corrective motor speeds. The robot moves straight when the line is centered, turns when deviating, and rotates in place if the track is lost until it is detected again. This ensures smooth and efficient line following.

Sensor Input (IR Sensors)	Error Value	PID Correction	Motor Speed (PWM)	Action
10000 (Leftmost detected)	-2	Negative	Left slower, Right faster	Turn Right
11000 (Left two detected)	-1	Slight Negative	Left slightly slower, Right slightly faster	Slight turn right
11100 (Center-left detected)	0	0	Both Equal	Move straight
01110 (Only center detected)	0	0	Both Equal	Move straight
00111 (Center-right detected)	0	0	Both Equal	Move straight
00011 (Right two detected)	1	Slight Positive	Right slightly slower, Left slightly faster	Slight turn left
00001 (Rightmost detected)	2	Positive	Right Slower, Left Faster	Turn Left
00000 (No sensor detected)	Last Known Error	Rotate In Place	Rotate left or right based on last error	Circular motion until track detected

3.3 Summary

The Arduino-based Line Follower Robot (LFR) operates using five infrared (IR) sensors that detect a black line on a white surface and provide input signals to the microcontroller. The robot reads these sensor values and calculates an error based on deviation from the center of the line using a weighted sum method. A PID (Proportional-Integral-Derivative) control algorithm processes this error to generate corrective motor speed adjustments via PWM signals. Depending on the error value, the robot decides whether to move straight, turn left, turn right, or make sharp turns to stay on track. If all sensors lose the line, the robot moves in a circular motion until it detects the path again.

Chapter-04

Result and Analysis

DC motors are widely utilized due to their simplicity in control and operation. A DC motor functions using two electrical signals, enabling its rotational direction to be altered by reversing the polarity of the power supply. Additionally, its speed can be regulated by adjusting the voltage applied across the motor terminals—an increase in voltage results in higher rotational speed, whereas a decrease leads to a reduction in speed.

Despite their advantages, DC motors inherently lack sufficient torque to directly drive high-load applications, such as robotic systems. This limitation arises because standard DC motors prioritize rotational speed over torque. To address this constraint, a DC gear motor is employed. A gear motor integrates a DC motor with a gear reduction system, which effectively decreases the motor's speed while significantly enhancing its torque output. This trade-off ensures that the motor generates adequate force to drive mechanical systems efficiently, albeit at a lower speed. Consequently, DC gear motors are preferred in applications where torque is prioritized over speed to ensure optimal performance.

4.1 PWM Speed Control

Pulse Width Modulation (PWM) is a widely employed technique for controlling the speed of DC motors by modulating the effective voltage supplied to them. The **duty cycle (ρ)** of a PWM signal represents the fraction of time within a single cycle during which the signal remains in the **high (ON) state**. By varying this duty cycle, the average voltage delivered to the motor is effectively altered, thereby regulating its rotational speed.

A **higher duty cycle** corresponds to a greater proportion of time in which the signal remains active, resulting in an increased effective voltage and, consequently, a higher motor speed. Conversely, a **lower duty cycle** reduces the effective voltage, leading to a decrease in motor speed. This method provides an efficient means of speed control without necessitating changes to the power supply voltage, thereby enhancing energy efficiency while minimizing power dissipation.

The generation of the **PWM signal** is facilitated by the **built-in hardware of the Arduino microcontroller**, ensuring precise and stable pulse generation. This approach is widely utilized in embedded systems and motor control applications due to its reliability, accuracy, and effectiveness in regulating motor performance.

4.2 Digital RPM Measurement System

A Digital RPM (Revolutions Per Minute) counter is an essential instrument used to accurately measure the rotational speed of a machine. It operates on a principle similar to that of a tachometer, a widely used device in industrial and laboratory applications. This

The measuring tool plays a significant role in electrical engineering, ensuring precise monitoring of machine performance.

At the core of this system lies an Infrared (IR) sensor pair, which detects and counts the number of rotations. The sensor works by identifying a marked black spot on the rotating wheel, generating a digital pulse each time the spot passes through the sensor. This digital signal is then transmitted to an Arduino or microcontroller for processing and display. Such systems are crucial for monitoring industrial equipment, laboratory experiments, and automotive applications, providing real-time data for analysis and optimization.

CHAPTER 5

The Line Following Robot

The mechanics of the Line Follower Robot (LFR) involve a differential drive system consisting of two independently controlled DC motors and a caster wheel for stability. The robot's chassis holds the motors, IR sensors, motor driver, and battery. The IR sensors, positioned at the front, detect the black line on a contrasting surface, while the motor driver regulates the power supplied to the motors based on the control signals from the Arduino. By adjusting the speed and direction of the left and right motors, the robot can move forward, turn left or right, and correct its path using PID control. The placement of components ensures balance, smooth movement, and efficient line tracking.

5.1 Mechanical Design

The chassis of the robot is made up of the plywood since it can carry more load and be lighter in weight.

The detailed orthographic projection is given below-

5.1.1 Design of the Track

The patterns of the track include:

1. **Curved Path** – A smooth, wavy section that tests the robot's ability to make gradual turns.
2. **Sharp Turns** – A zigzag pattern that requires quick left and right adjustments.
3. **Intersection** – A diamond-shaped crossing where the robot must determine the correct path.
4. **Dashed Line Section** – A broken-line segment that might simulate an interrupted path, testing how the robot handles missing track portions.
5. **A Bridge -** A bridge is included at the second half of the track.

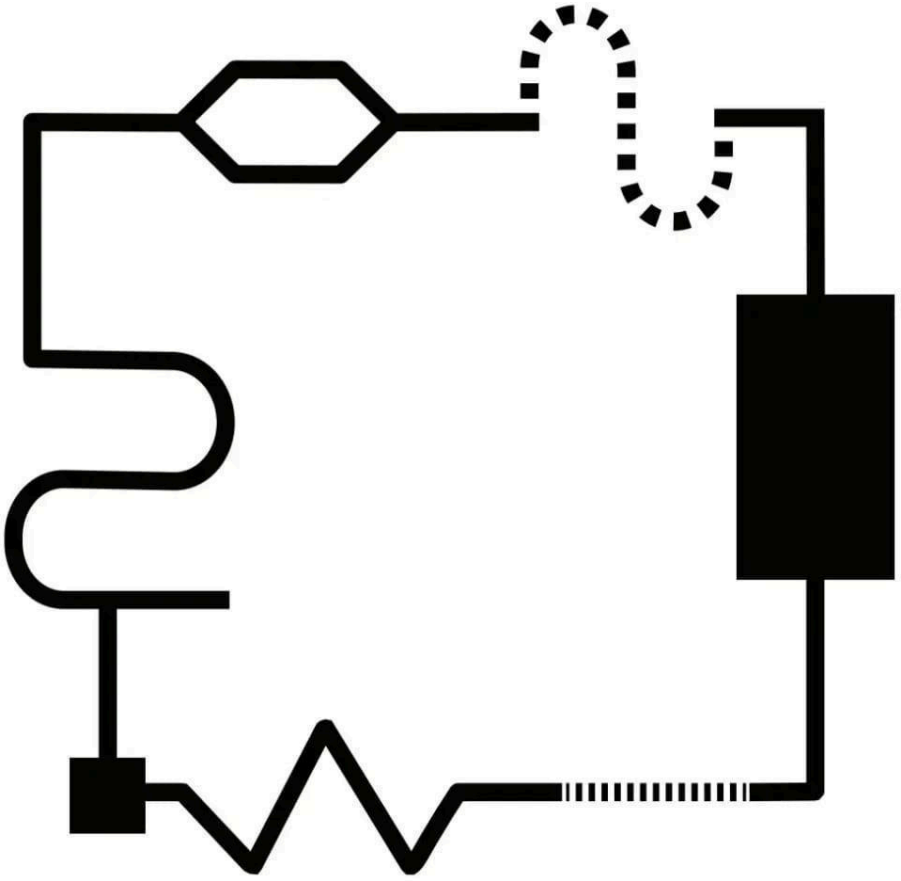

Fig 5.1.1.1 Track of the Line Following Robot

5.1.2 Picture of the Line Following Robot

The picture of the Line Following Robot from different angles are given below-

5.1.2.1 Top View

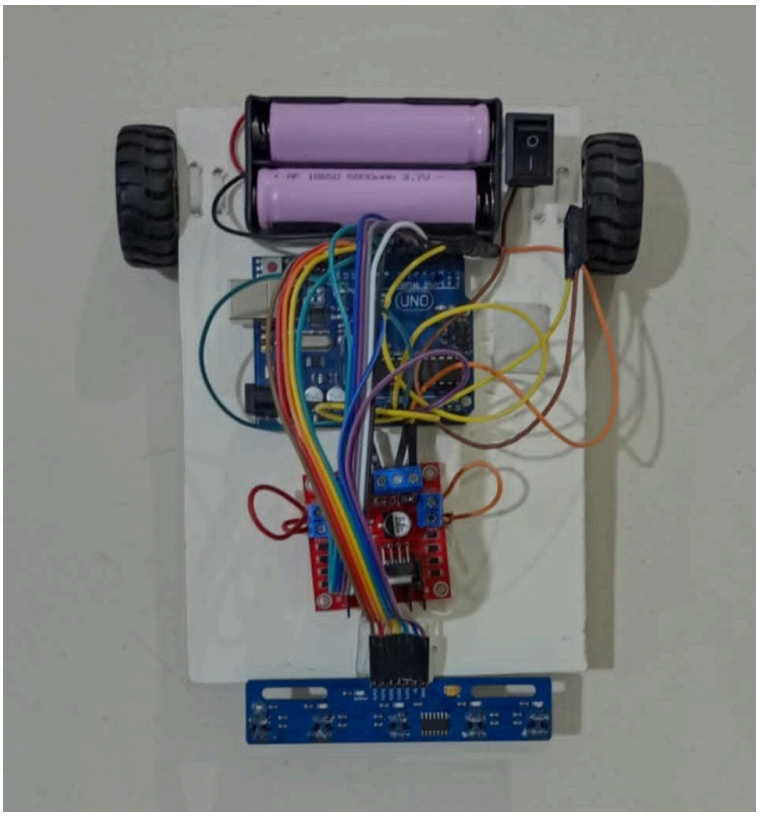

Fig 5.1.2.1 Top View Of LFR

5.1.2.2 Front View

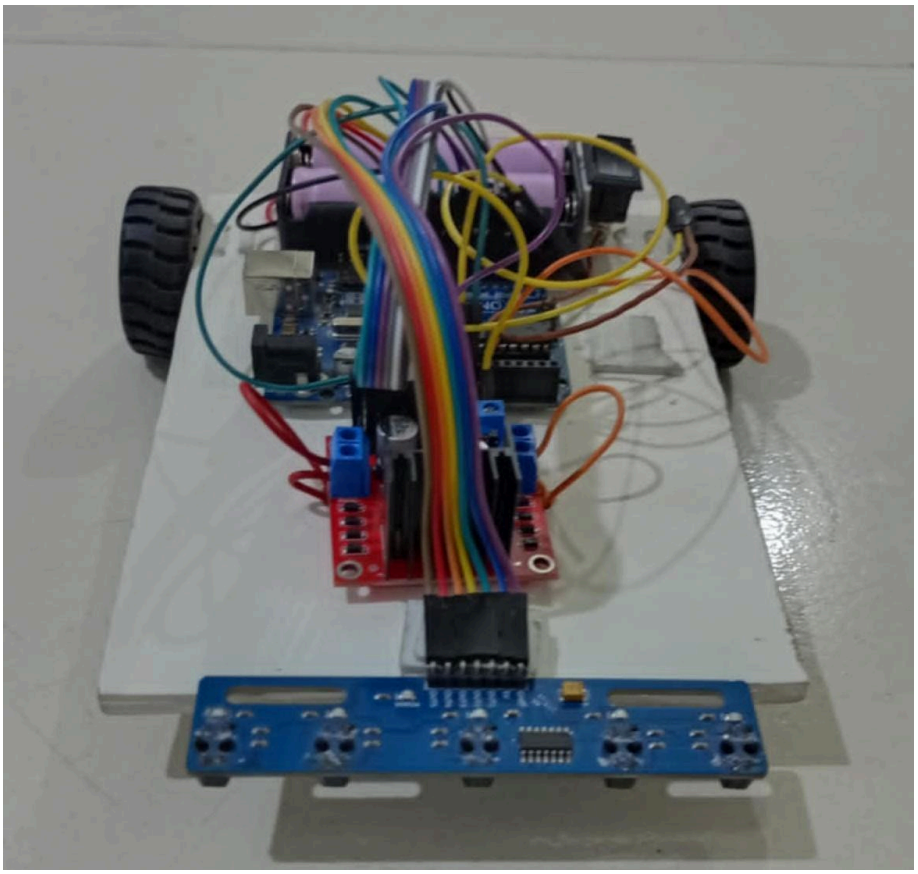

Fig 5.1.2.2 Front View of LFR

5.1.2.3 Side View

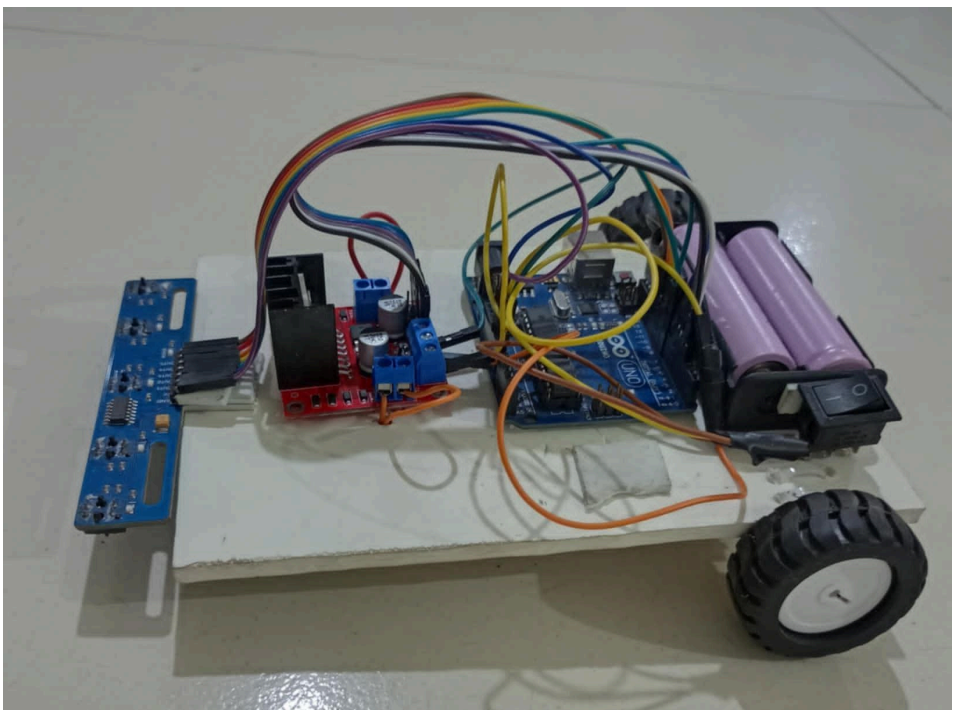

Fig 5.1.2.3 Side View of LFR

CHAPTER 6

6.1 Conclusion

This project successfully demonstrated the design, construction, and implementation of a Line Following Robot (LFR) using an Arduino Uno, a 5-channel IR sensor array, L298N motor driver, and DC gear motors. The robot was able to accurately follow a predefined black line, perform smooth turns, cross dashed paths, and maintain stability through effective mechanical and electronic integration.

The use of PID-based motor control and PWM speed regulation significantly improved the robot's tracking accuracy and responsiveness. This project provided practical experience in sensor interfacing, motor control, embedded programming, and system debugging, strengthening the understanding of core robotics and automation concepts.

Although the robot performs reliably on standard tracks, its performance is influenced by surface conditions, lighting variations, and battery limitations. These constraints highlight opportunities for future improvement, such as adaptive control algorithms, wireless monitoring, obstacle detection, and enhanced power management.

Overall, this project serves as a strong foundation for further exploration in robotics and autonomous systems, bridging theoretical knowledge with real-world application.

Appendices

Code

```
// Motor Pins
#define enA 3     // Right Motor Enable Pin
#define in1 4     // Right Motor in1
#define in2 5     // Right Motor in2
#define enB 9     // Left Motor Enable Pin
#define in3 6    // Left Motor in3
#define in4 7    // Left Motor in4

// Sensor Pins (connected to digital pins)
#define ir1 13     // Leftmost Sensor
#define ir2 12     // Left Sensor
#define ir3 11    // Middle Sensor
#define ir4 10    // Right Sensor
#define ir5 8    // Rightmost Sensor

int baseSpeed = 140;
int maxSpeed = 200;

float Kp = 10.0;   // Proportional constant
float Ki = 0.9;   // Integral constant
float Kd = 5.0;   // Derivative constant

int previousError = 0;
float integral = 0;
int lastKnownError = 0;

void setup() {
  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  pinMode(ir1, INPUT);
  pinMode(ir2, INPUT);
  pinMode(ir3, INPUT);
  pinMode(ir4, INPUT);
  pinMode(ir5, INPUT);

  Serial.begin(9600);
}

void loop() {
  int s1 = digitalRead(ir1) == LOW ?  1 : 0;
  int s2 = digitalRead(ir2) == LOW ?  1 : 0;
  int s3 = digitalRead(ir3) == LOW ?  1 : 0;
  int s4 = digitalRead(ir4) == LOW ?  1 : 0;
  int s5 = digitalRead(ir5) == LOW ?  1 : 0;
```

```
  // Calculate error
  int weights[5] = { -2, -1, 0, 1, 2 };
  int error = 0, totalActiveSensors = 0;
  int sensors[5] = { s1, s2, s3, s4, s5 };

  for (int i = 0; i < 5; i++) {
    error += sensors[i] * weights[i];
    totalActiveSensors += sensors[i];
  }

  if (totalActiveSensors == 0) {
    if (lastKnownError < 0) setMotors(-baseSpeed / 2, baseSpeed / 2);
    else if (lastKnownError > 0) setMotors(baseSpeed / 2, -baseSpeed / 2);
    else setMotors(-baseSpeed / 2, baseSpeed / 2);
    return;
  }

  error /= totalActiveSensors;
  lastKnownError = error;

  float proportional = error * Kp;
  integral = constrain(integral + error, -50, 50);
  float integralTerm = integral * Ki;
  float derivative = (error - previousError) * Kd;
  int pid = proportional + integralTerm + derivative;

  previousError = error;

  int dynamicBaseSpeed = map(abs(error), 0, 2, baseSpeed, baseSpeed / 2);
  int leftSpeed = constrain(dynamicBaseSpeed + pid, 0, maxSpeed);
  int rightSpeed = constrain(dynamicBaseSpeed - pid, 0, maxSpeed);

  setMotors(leftSpeed, rightSpeed);

  Serial.print("Error: ");
  Serial.print(error);
  Serial.print(" | PID: ");
  Serial.print(pid);
  Serial.print(" | Left: ");
  Serial.print(leftSpeed);
  Serial.print(" | Right: ");
  Serial.println(rightSpeed);
}

void setMotors(int leftSpeed, int rightSpeed) {
  analogWrite(enA, abs(rightSpeed));
  digitalWrite(in1, rightSpeed > 0 ? HIGH : LOW);
  digitalWrite(in2, rightSpeed > 0 ? LOW : HIGH);

  analogWrite(enB, abs(leftSpeed));
  digitalWrite(in3, leftSpeed > 0 ? HIGH : LOW);
  digitalWrite(in4, leftSpeed > 0 ? LOW : HIGH);
}
```