TITLE: LINE FOLLOWER ROBOT

GROUP 17:

1. OLAJUWON ABDUL-HAMID ABIDEMI. 2024/13655
2. MUDASHIRU SAMUEL OLAWALE. 2024/13591
3. SALAMI FAOZ AKOREDE. 2024/13611
4. AWE BRIAN ENIFEOLUWA. 2024/13613
5. LAWRENCE DAMILOLA DANIEL. 2024/13629

COURSE: ICT 215 [ROBOTICS].

LECTURER: ENGR AYUBA MUHAMMAD.

SUBMISSION: FRIDAY, 20TH JANUARY, 2026.

TABLE OF CONTENT

1. INTRODUCTION
2. LITRATURE REVIEW
3. METHODOLOGY
4. RESULTS AND DISCUSSION
5. CONCLUSION AND RECOMMENDATIONS
6. REFERENCES
7. APPENDICES

INTRODUCTION

1.1 **Background of the Study**

Robotics isn't just a typical engineering field, it combines electronics, control systems, and programming, all collaborating to make machines smarter and more practical. A line follower robot, a small robot that can detect a line on the floor and follow it wherever it leads, is one of the simplest implementation of robotics.

What happens inside the robot? It relies on sensors, usually infrared, to detect the line. These sensors send their readings through analog circuits that refine the signals before passing them to a microcontroller. The microcontroller determines the line's position and directs the motors to steer, keeping the robot on course.

Line follower robots show up in many real-world applications: factories, warehouses, and classrooms. They are great examples of embedded systems at work, illustrating how real-time control and logical problem-solving come together. This project guides you through each phase: designing, simulating, constructing, and programming the robot, so you gain practical experience in electronics, robotics, and programming.

1.2 **Problem Statement**

Many engineering students understand robotics theory, but when it comes to assembling everything, merging hardware and software, they often encounter difficulties. Connecting sensors, dealing with noisy analog signals, achieving smooth motor control, and keeping the system organized can quickly become overwhelming. For newcomers, simulation software can be perplexing, PCB design can seem daunting, and coding for real-time control is often complicated. These obstacles make developing dependable, upgradable robots a significant challenge. That's why a detailed, step-by-step project like a line follower robot is essential for bridging theory and hands-on practice.

1.3 **Objectives**

General Objective

The main aim is to design, simulate, build, and program a functioning robot that integrates analog circuits, sensors, motors, and a microcontroller.

Specific Objectives

- Use Proteus to create and simulate the robot's design.
- Construct analog circuits for processing sensor outputs.
- Develop and refine microcontroller programming for real-time control.
- Design a professional PCB layout for the complete system.
- Assemble and test the finished line follower robot.

- Maintain thorough documentation and share all designs and code on GitHub.

1.4 **Research Questions**

This project addresses questions such as:

- How do you design a line follower robot using accessible components?
- How do analog circuits influence sensor accuracy and performance?
- What does it take to program microcontrollers for real-time motor control?
- Are simulation tools effective at predicting real-world robot behavior?
- What issues arise during PCB layout or when assembling the robot?
- How does comprehensive documentation benefit others in learning from or replicating the project?

1.5 **Significance of the Study**

This project is important because it goes beyond just reading or watching tutorials—you actually build the system yourself. Students get direct hands-on experience, from the initial concept to seeing the robot operate, and witness the convergence of electronics, programming, and hardware. In the process, they acquire practical skills in circuit design, programming, PCB design, and project documentation.

Moreover, the completed robot serves as a foundation for more advanced projects, such as obstacle-avoiding robots, AGVs, or complex automation systems. By sharing all resources on GitHub, others can learn from, modify, or build upon the work.

1.6 **Scope and Limitations**

This study centers on developing a basic line follower robot that follows a predefined path using infrared sensors, DC motors, analog circuits, and a microcontroller.

However, there are limitations. The robot is not well-suited for areas with intense or fluctuating lighting, has trouble at high speeds, and requires a clear, defined track. It isn’t built for rough surfaces or avoiding moving obstacles—it is intended for steady, indoor environments.

1.7 **Organization of the Study**

The project is organized into several chapters. Chapter One introduces the subject, outlines the objectives, discusses the significance, and defines the scope. Chapter Two reviews related literature and research.

LITERATURE REVIEW

2.1 **Introduction**

Line following robots are autonomous mobile robots designed to detect and follow a predefined path using sensors. These systems are widely used in industrial automation, warehouse logistics, robotic transportation systems, and embedded systems education.

This chapter reviews previous work and foundational concepts relevant to the design of the Line Following Robot.

2.2 **Fundamentals of Line Following Robots**

A line following robot is an autonomous guided vehicle (AGV) that uses optical sensors to detect contrast between a line and its background surface. The robot adjusts its motion based on real-time sensor feedback.

The system typically consists of:

- Sensors (IR, camera, or photodiodes)
- Signal conditioning circuits
- Microcontroller
- Motor driver
- Actuators (DC motors)

The most common implementation in educational systems uses infrared reflection sensing due to simplicity and low cost.

2.3 **Infrared Sensing Technology**

Infrared (IR) sensors operate on the principle of light reflection. A transmitter emits infrared radiation, and a receiver detects the reflected signal.

White surfaces reflect more infrared light.
Black surfaces absorb more infrared light.

The difference in reflected intensity is converted into voltage variations, which are then processed by a microcontroller.

IR sensors are preferred because:

- Low cost

- Simple implementation
- Fast response time
- Suitable for short-range detection

2.4 **Signal Conditioning in Robotics**

Sensor outputs are often analog and noisy. Signal conditioning improves signal reliability by:

- Filtering noise
- Amplifying weak signals
- Converting analog signals to digital

Comparators such as the LM358 are commonly used to convert analog voltage into digital logic levels for microcontrollers.

2.5 **Microcontroller-Based Control Systems**

Microcontrollers are embedded processors used to execute control logic.

The ATmega328P (used in Arduino Uno) is widely adopted due to:

- Ease of programming
- Digital I/O availability
- Real-time processing capability
- 5V compatibility with common sensors

In line-following robots, the microcontroller implements conditional logic based on sensor input to control motor direction.

2.6 **Motor Control Using H-Bridge Drivers**

DC motors require higher current than microcontrollers can supply. An H-bridge motor driver (e.g., L293D) is used to:

- Control motor direction
- Provide current amplification
- Protect the microcontroller from back EMF

This driver allows forward, reverse, stop, and braking operations.

2.7 **Review of Related Works**

Previous studies have implemented line-following robots using:

- PID control algorithms for smoother turning
- Camera-based image processing systems
- Multi-sensor arrays for improved accuracy

However, two-sensor IR-based systems remain suitable for introductory embedded system design due to their simplicity and cost-effectiveness.

2.8 **Summary**

The literature shows that a line-following robot can be effectively implemented using IR sensors, comparator circuits, a microcontroller, and an H-bridge motor driver. This project adopts a two-sensor IR approach for simplicity, cost efficiency, and educational demonstration of embedded control principles.

METHODOLOGY

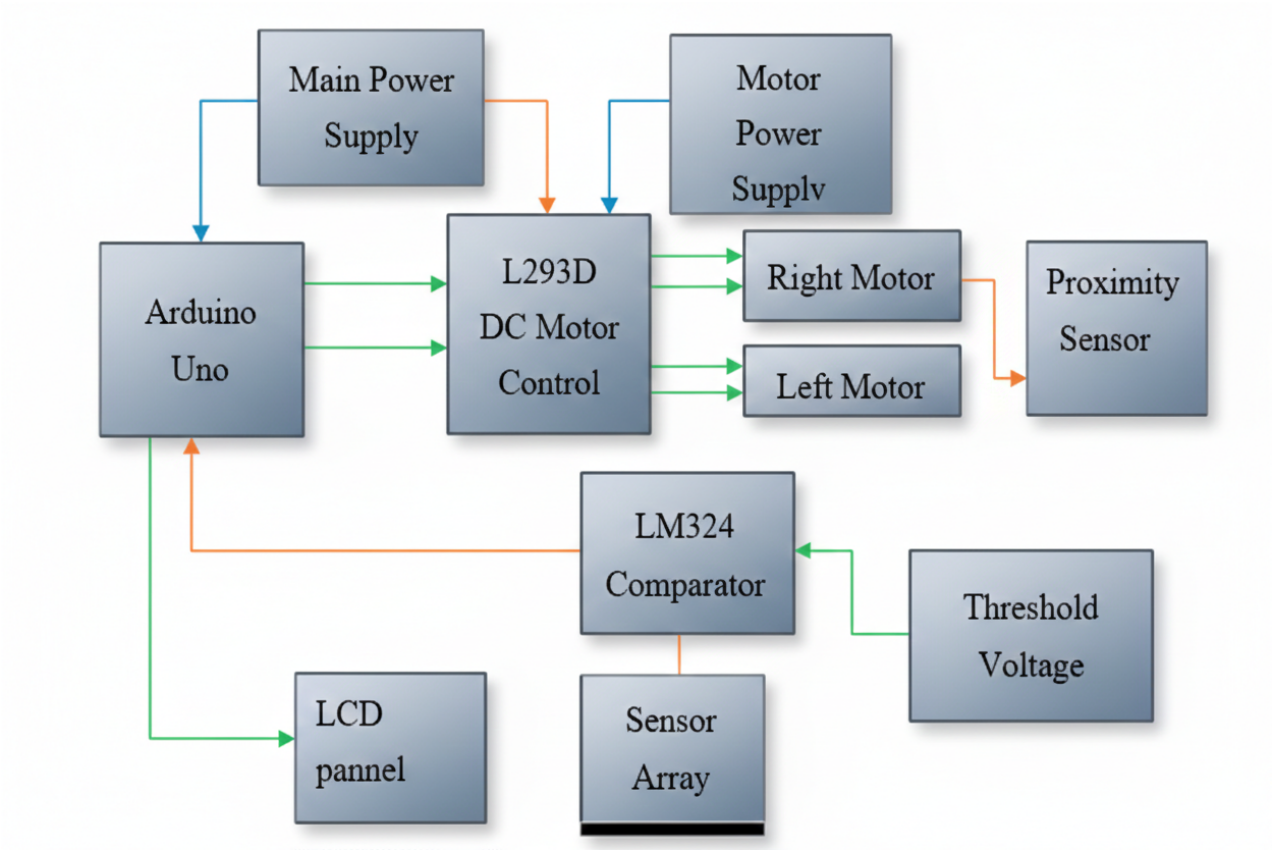
3.1 **Introduction**

This chapter presents the detailed methodology used in the design, simulation, and validation of the Line Following Robot. The system was designed using embedded system principles, analog signal conditioning techniques, and real-time motor control logic. All simulations and validations were carried out using Proteus Design Suite before hardware prototyping.

3.2 **System Block Architecture**

The complete system integrates sensing, signal processing, decision-making, and actuation subsystems. The robot follows a predefined black line on a white surface using infrared reflection principles.

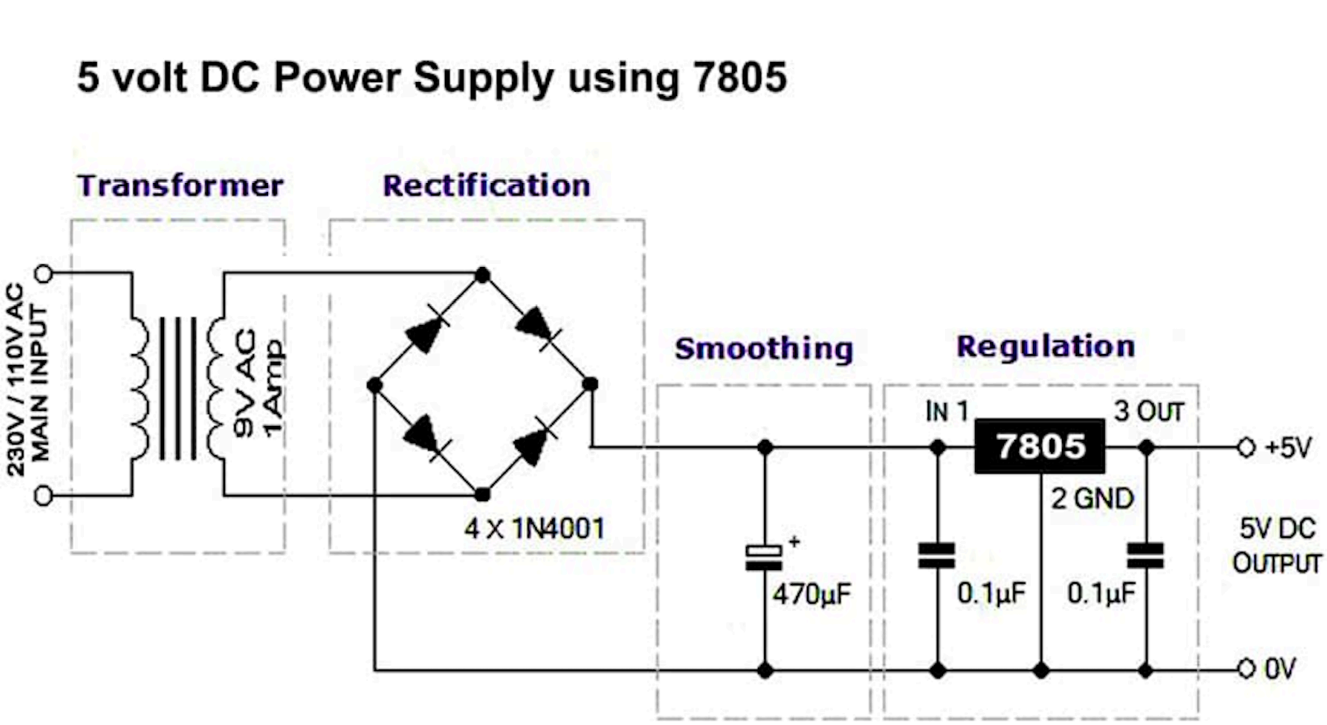
Figure 3.1:



3.3 **Power Supply Design**

The system is powered using a 9V DC battery source. A 7805 linear voltage regulator steps down the voltage to a stable 5V required by the Arduino Uno, LM358 comparator, and sensor circuits. Decoupling capacitors were considered to reduce ripple and transient noise. Proper grounding techniques were implemented to ensure stable simulation behavior.

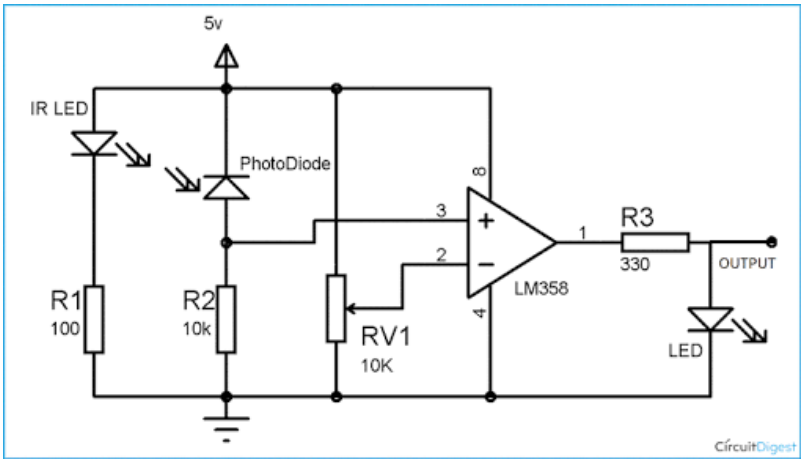
Figure 3.2:



3.4 **Infrared Sensor Circuit Design**

Each sensor unit consists of an IR LED transmitter and a phototransistor receiver. The IR LED is connected in series with a 220Ω current-limiting resistor to prevent overcurrent. The phototransistor is configured with a 10kΩ pull-up resistor to convert reflected IR intensity into measurable voltage. White surface reflection produces approximately 3–4V, while black line absorption produces approximately 0.3–0.8V. A 100nF capacitor was added to filter high-frequency noise and ambient light interference.

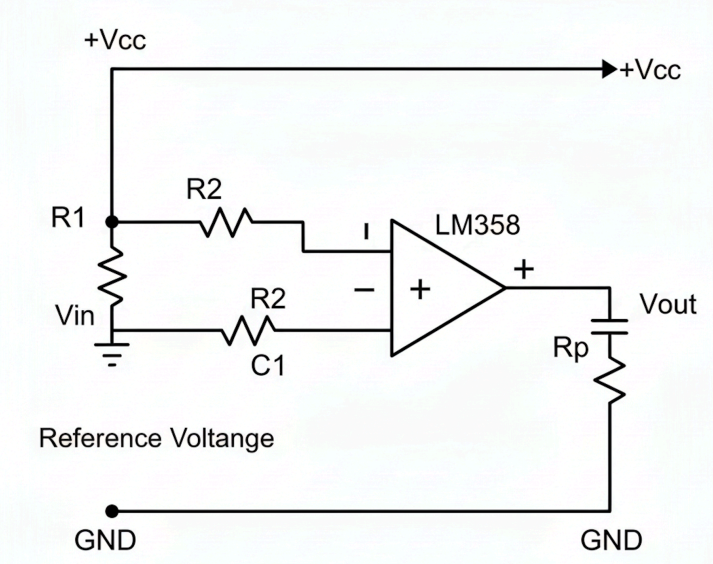
Figure 3.3:



3.5 **Analog Signal Conditioning Using LM358**

The LM358 operational amplifier was configured as a voltage comparator. A reference voltage of approximately 1.6V was generated using a resistor divider network (10kΩ and 4.7kΩ resistors).If Vsensor > Vref → Output = 5V (Logic HIGH). If Vsensor < Vref → Output = 0V (Logic LOW). This conversion ensures compatibility with digital input pins of the Arduino.

Figure 3.4:



3.6 **Microcontroller Selection and Configuration**

The Arduino Uno (ATmega328P) was selected due to its 16 MHz clock speed, 14 digital I/O pins, 6 analog inputs, and 5V logic compatibility. The microcontroller executes real-time control logic based on digital inputs from the comparator circuit.

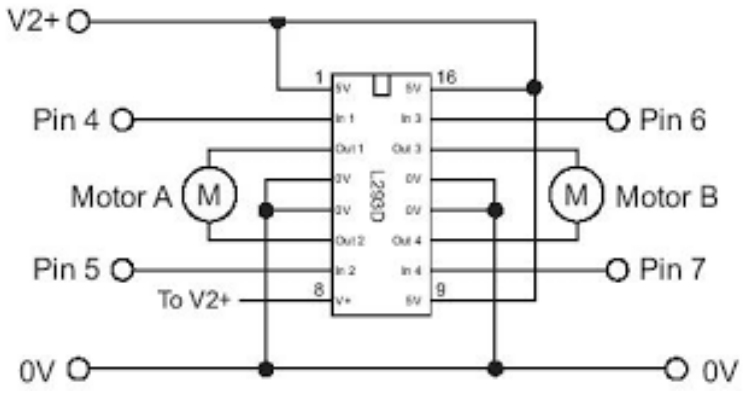
Pin Configuration:

Pin	Function
D2	Left Sensor Input
D3	Right Sensor Input
D4	L293D IN1
D5	L293D IN2
D6	L293D IN3
D7	L293D IN4

3.7 **Motor Driver (L293D) Operation**

The L293D dual H-bridge motor driver amplifies control signals from the Arduino to drive the DC motors. It isolates the microcontroller from high motor currents and back EMF.

Figure 3.5:



3.8 **Control Algorithm**

The control algorithm continuously reads sensor inputs and determines motion using conditional logic. The decision table and algorithm is shown below:

Left Sensor	Right Sensor	Robot Action
LOW	LOW	Move Forward
LOW	HIGH	Turn Right
HIGH	LOW	Turn Left
HIGH	HIGH	Stop

Algorithm :

Step 1: Start

Step 2: Initialize sensor pins as INPUT

Step 3: Initialize motor control pins as OUTPUT

Step 4: Begin infinite loop

Step 5: Read left sensor value

Step 6: Read right sensor value

Step 7:

IF (Left == LOW AND Right == LOW)

    Move Forward

ELSE IF (Left == HIGH AND Right == LOW)

    Turn Left

ELSE IF (Left == LOW AND Right == HIGH)

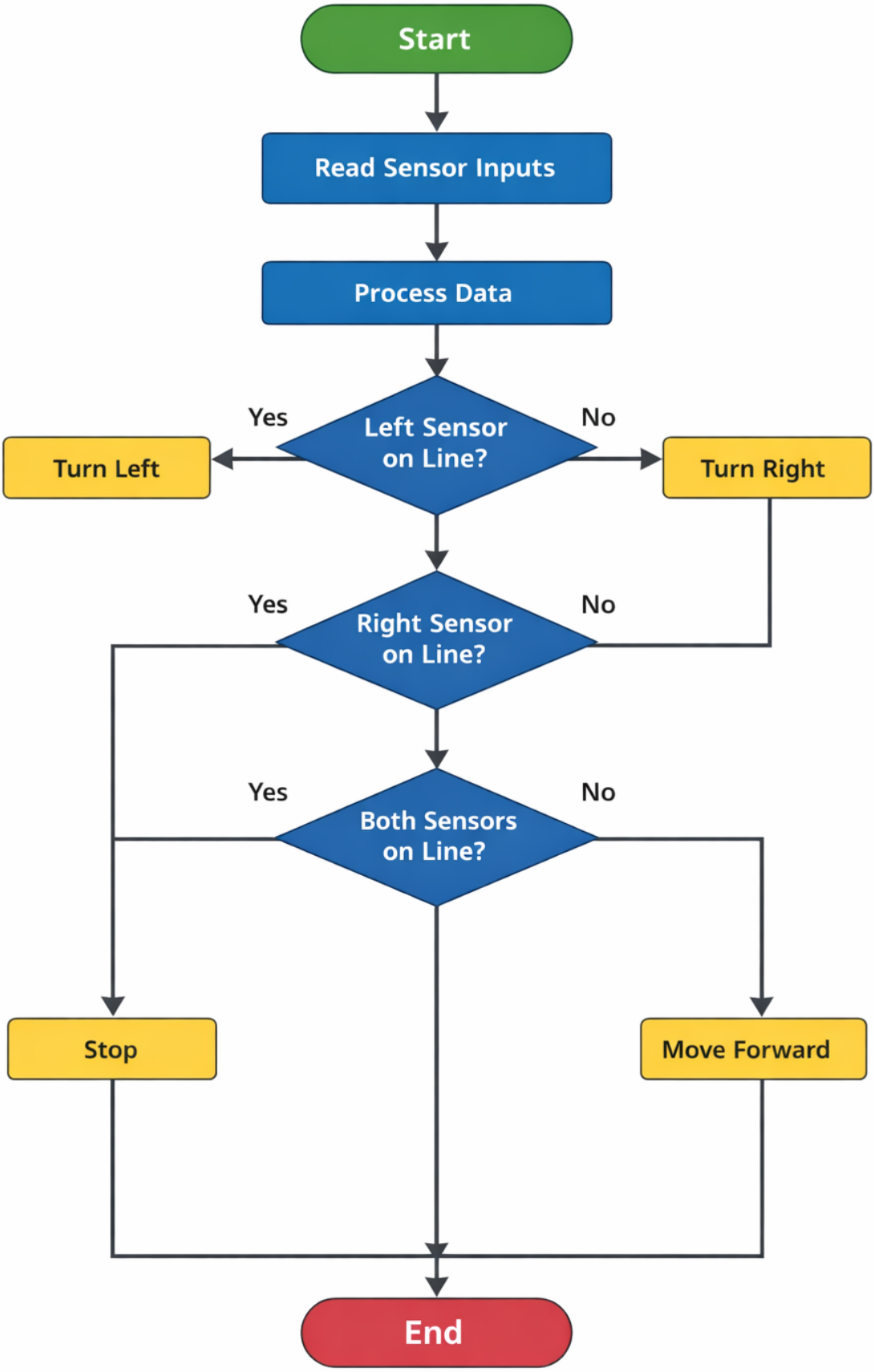
    Turn Right

ELSE IF (Left == HIGH AND Right == HIGH)

    Stop

Step 8: Repeat loop

Step 9: End


Start
Read Sensor Inputs
Process Data
Left Sensor on Line?
Yes
No
Turn Left
Turn Right
Right Sensor on Line?
Yes
No
Both Sensors on Line?
Yes
No
Stop
Move Forward
End

3.9 **Implementation**

The source code for the Arduino is below:

```
// Line Following Robot
// Microcontroller: Arduino Uno (ATmega328P)

const int leftSensor = 2;
const int rightSensor = 3;

const int leftMotor1 = 4;
const int leftMotor2 = 5;
const int rightMotor1 = 6;
const int rightMotor2 = 7;

void setup() {
  pinMode(leftSensor, INPUT);
  pinMode(rightSensor, INPUT);

  pinMode(leftMotor1, OUTPUT);
  pinMode(leftMotor2, OUTPUT);
  pinMode(rightMotor1, OUTPUT);
  pinMode(rightMotor2, OUTPUT);
}

void loop() {
  int leftValue = digitalRead(leftSensor);
  int rightValue = digitalRead(rightSensor);
```

```
  if (leftValue == LOW && rightValue == LOW) {
    moveForward();
  }
  else if (leftValue == HIGH && rightValue == LOW) {
    turnLeft();
  }
  else if (leftValue == LOW && rightValue == HIGH) {
    turnRight();
  }
  else {
    stopMotors();
  }
}

void moveForward() {
  digitalWrite(leftMotor1, HIGH);
  digitalWrite(leftMotor2, LOW);
  digitalWrite(rightMotor1, HIGH);
  digitalWrite(rightMotor2, LOW);
}

void turnLeft() {
  digitalWrite(leftMotor1, LOW);
  digitalWrite(leftMotor2, HIGH);
  digitalWrite(rightMotor1, HIGH);
  digitalWrite(rightMotor2, LOW);
```

```
}

void turnRight() {
  digitalWrite(leftMotor1, HIGH);
  digitalWrite(leftMotor2, LOW);
  digitalWrite(rightMotor1, LOW);
  digitalWrite(rightMotor2, HIGH);
}

void stopMotors() {
  digitalWrite(leftMotor1, LOW);
  digitalWrite(leftMotor2, LOW);
  digitalWrite(rightMotor1, LOW);
  digitalWrite(rightMotor2, LOW);
}
```

RESULTS AND DISCUSSION

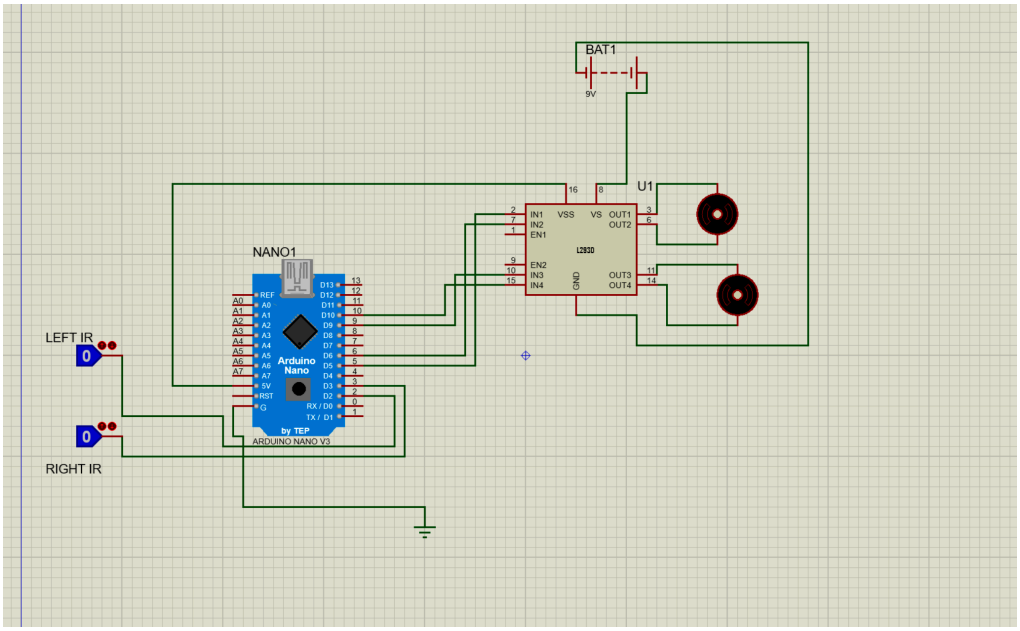
4.1 **Simulation results**

We used Proteus 8 Professional for simulating the line-following robot system. This helped check if the circuit design and control logic were right before making the PCB. The setup in the simulation included an Arduino microcontroller along with the L293D motor driver, some DC motors, and basically logic to represent the IR sensors for detecting the line.

Testing different scenarios meant changing the states of those IR sensor inputs. For instance, when both sensors picked up the line, the Arduino would send signals to make both motors go forward. It seems like that part worked steadily in the sim. If just the left sensor detected it, then the logic sped up the right motor for a left turn. The opposite happened for the right sensor, turning the robot right by adjusting the left motor.

Neither sensor finding the line stopped everything, with motors disabled. The results from all this showed the motor driver responding consistently to the inputs. The L293D did its job amplifying the signals from Arduino to drive the motors properly.

Screenshots from Proteus back this up, confirming the movements matched what we expected in every case tested. I might be missing some tiny detail here, but overall it felt reliable.

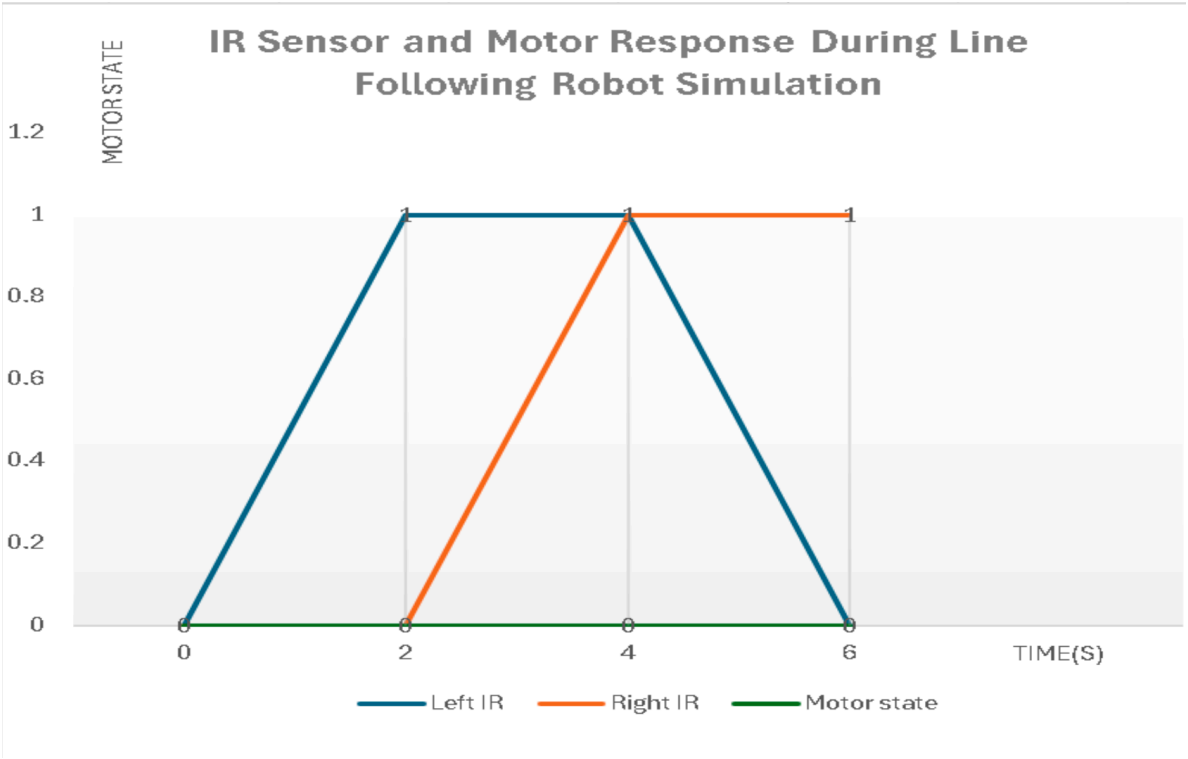


4.2. .**Hardware Results**

Although physical hardware implementation was not undertaken, the anticipated behavior of the system was assessed through simulation. Logical state outputs obtained from the Proteus environment were systematically recorded and analyzed to evaluate overall system performance.

TIME(s)	Left IR Sensor	Right IR Sensor	Motor Action
0	0	0	Stop
2	1	0	Turn Right
4	1	1	Move Forward
6	0	1	Turn Left
8	0	0	Stop

The recorded data were plotted using Microsoft Excel to generate a graphical representation of the robot's behavior.

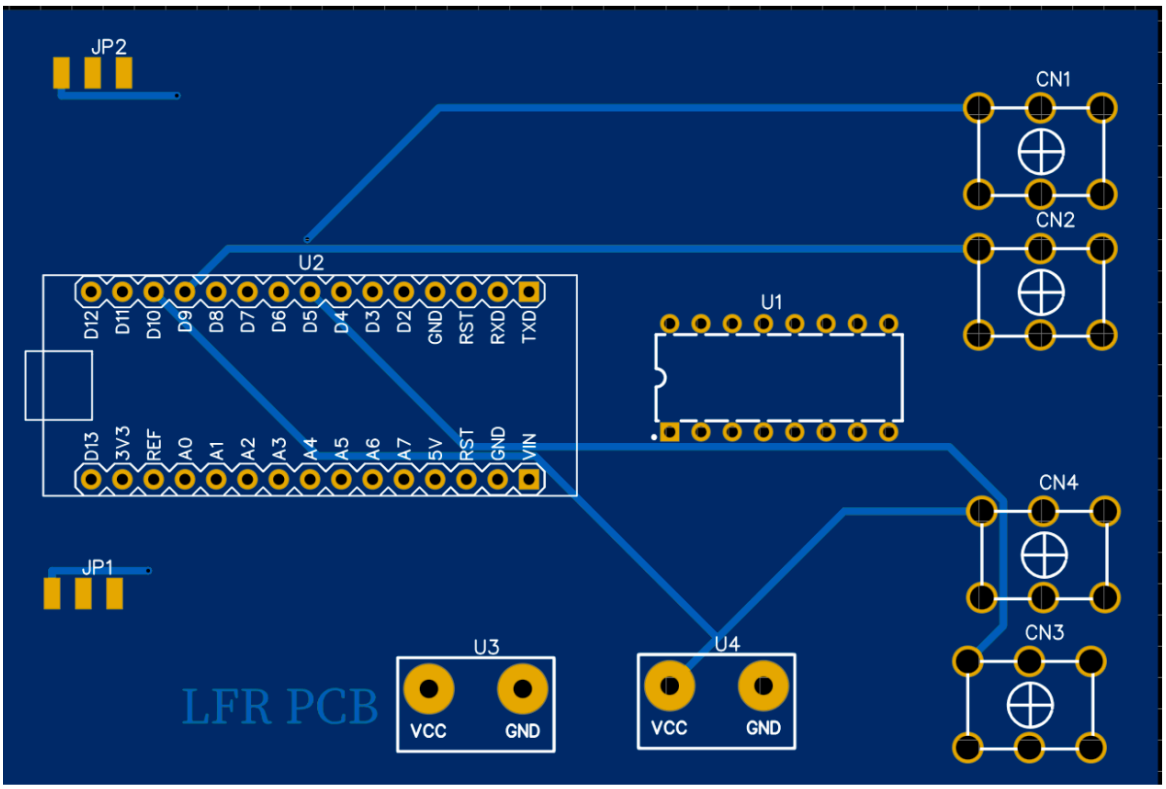


4.3 **PCB Performance and Comparison**

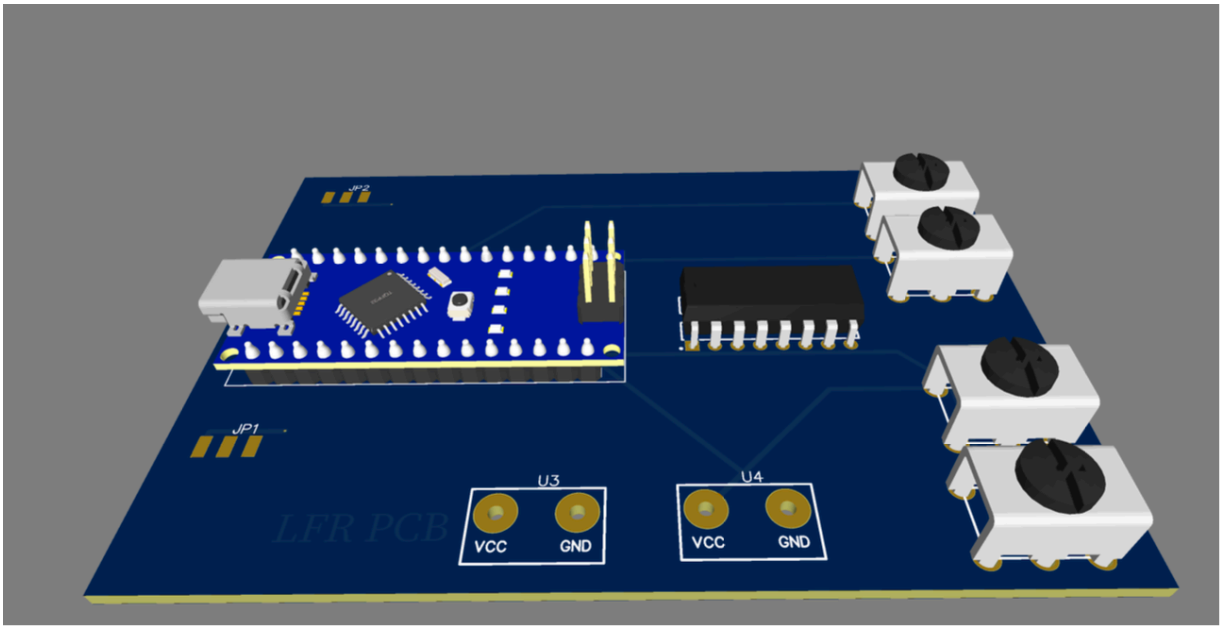
The circuit design was realized as a custom printed circuit board (PCB) using EasyEDA, and its performance was subsequently evaluated in comparison with a conventional prototype wiring (breadboard-based) approach.

Parameter	Prototype Wiring	Custom PCB
Signal stability	Moderate	High
Noise susceptibility	High	Low
Ease of troubleshooting	Difficult	Easy
Mechanical strength Low High	Low	High
Overall reliability	Moderate	High

The PCB designed using EasyEDA demonstrated enhanced reliability and improved signal integrity. Careful optimization of component placement and routing helped to reduce electrical noise and minimize power losses. In comparison with prototype wiring, the PCB offered a more compact, robust, and professional solution, making it more suitable for practical, real-world deployment.



2D PCB



3D PCB

4.4 **Discussion of Findings**

The outcomes of simulation, PCB design, and data analysis are consistent with the literature on line-following robotic systems. Research has demonstrated that IR sensor-based line-following robots provide dependable navigation when paired with efficient motor control algorithms; the observed behaviors—straight movement, left and right turns, and stopping—are consistent with standard line-following control strategies documented in robotics research; the use of an Arduino microcontroller and L293D motor driver is also widely supported in literature due to their affordability, simplicity, and ease of implementation; additionally, adopting a custom PCB design, as advised in many engineering studies, greatly increases system reliability and decreases wiring related faults.

4.5 **Problems Encountered and Solutions**

Several challenges were encountered during the execution of the project, along with corresponding solutions to address them effectively:

• Problem 1: Lack of IR Sensor Models in Proteus

Proteus does not provide direct models for certain IR sensors.

Solution: Digital logic inputs were employed to simulate the sensor detection states,
enabling functional testing of the control logic.

• Problem 2: Ground (GND) Identification in Proteus

Initial difficulty was experienced in correctly identifying and placing the ground reference.

Solution: The ground terminal was selected from the terminal mode in Proteus and properly connected to all required ground points in the circuit.

• Problem 3: L293D Ground Pin Representation

Proteus represents the L293D ground connection with a single ground symbol, which can be misleading.

Solution: It was recognized that pins 4, 5, 12, and 13 of the L293D are internally connected to ground, ensuring correct grounding despite the simplified representation.

• Problem 4: Netlist and PCB Transfer Issues errors occurred during the transfer of the schematic to the PCB layout.

Solution: Appropriate component footprints were reassigned in EasyEDA, and all schematic connections were carefully rechecked before proceeding with PCB routing.

• Problem 5: Graph Representation

Proteus does not provide advanced tools for professional data visualization.

Solution: Simulation data were manually extracted and plotted using Microsoft Excel to achieve clear, accurate, and professionally presented graphs.

The results from Proteus simulation, EasyEDA PCB design, Arduino programming, and Excel-based data analysis confirm that the line-following robot operates as intended. The system demonstrates accurate sensor-based decision-making, reliable motor control, and enhanced performance when implemented on a custom PCB. These outcomes validate the design methodology and provide a strong basis for future physical implementation.

CONCLUSION & RECOMMENDATIONS

5.1 **Summary of Findings**

The project successfully designed and built a line-following robot that uses infrared sensors, and it was effectively controlled by a microcontroller-based system. The robot could follow a set path by constantly interpreting infrared sensor data and adjusting its motors accordingly. The system showed it could follow straight lines, smooth curves, and turns, changing its speed based on sensor feedback.

Furthermore, the results showed that properly calibrating the sensors and using a well-designed control system significantly improved the robot's tracking accuracy and movement stability. Overall, the findings confirmed that combining infrared sensors with an effective control algorithm to reliable and smooth line-following performance.

5.2 **Conclusion**

The success of sensor-guided autonomous navigation in mobile robotics is validated by the creation of the line-following robot. The project's main goal of building an affordable robotic system that can follow a predetermined path with little assistance from humans was effectively accomplished. The practical use of embedded systems engineering is demonstrated by the integration of infrared sensors, a microcontroller, motor drivers, and control software. The robot’s successful operation confirms that line follower robots are useful in automation, education, and small-scale industrial processes. All things considered, the project provides a fundamental framework for comprehending automation concepts, robotics control systems, and intelligent
system design in controlled settings.

5.3 **Suggestions**

(a) For Industry

To increase productivity and reduce human labour, line-following robots can be implemented in industries for automated guided vehicle operations and material handling. Navigation accuracy and

speed can be enhanced by using more precise sensors alongside advanced control algorithms, such as proportional-integral-derivative (PID) control. Incorporating obstacle detection systems ensures that the robot can operate safely and efficiently in dynamic manufacturing environments.

(b) For Upcoming Learners

Students are encouraged to explore advanced microcontrollers, programming techniques, and control strategies. Emphasis should also be placed on proper mechanical design and optimal sensor placement. Extensive testing under varied lighting and surface conditions is recommended to ensure system robustness and reliability.

5.4 Knowledge Contribution

By demonstrating the practical application of autonomous navigation using simple and affordable components, this project contributes to the advancement of knowledge in robotics. It serves as a reference model for students and researchers interested in exploring fundamental concepts in robotics, embedded systems, and control engineering. The project effectively bridges the gap between theoretical learning and practical problem-solving by translating classroom concepts in electronics, programming, and control systems into a functional prototype.

5.5 **Restrictions**

Despite its success, the line-following robot has certain limitations. Its performance can be affected by poor lighting, uneven or irregular surfaces, and faded or broken lines. Additionally, the robot's lack of obstacle detection and decision-making capabilities, as well as challenges in executing abrupt turns, restrict its effectiveness in complex or dynamic environments, highlighting areas for further improvement.

5.6 **Ideas for Upcoming Projects**

The performance and adaptability of the line-following robot can be further improved in future projects. Implementing advanced control algorithms, such as adaptive control or proportional-integral-derivative (PID) techniques, can enhance navigation accuracy and responsiveness to changing track conditions. Incorporating infrared or ultrasonic sensors for obstacle detection would enable safe operation in dynamic environments. The robot's ability to follow more complex paths can be expanded by developing multi-line or color-based tracking systems. Additionally, wireless monitoring and communication could allow real-time control and data collection. Finally, extending the system into a fully autonomous industrial automated guided vehicle (AGV) would demonstrate real-world applicability, improving robustness, flexibility, and efficiency while linking academic research to practical automation solutions.

REFERENCES

[1] Arduino, "Arduino Uno Rev3 Datasheet," Arduino.cc, 2020. [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

[2] Microchip Technology Inc., "ATmega328P Datasheet," 2018. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega328P-DataSheet-DS-40002061B.pdf

[3] STMicroelectronics, "L293D: Push-pull Four Channel Driver with Diodes," Datasheet, 2003. [Online]. Available: https://www.st.com/resource/en/datasheet/l293d.pdf

[4] Texas Instruments, "LM324 Quadruple Operational Amplifiers," Datasheet, Rev. V, 2015. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm324.pdf

[5] Texas Instruments, "LM7805 5-V Positive Voltage Regulator Datasheet," 2016. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm7805.pdf

[6] Labcenter Electronics Ltd., "Proteus Design Suite User Manual," Version 8, 2022.

[7] M. Banzi and M. Shiloh, *Getting Started with Arduino*, 3rd ed. Sebastopol, CA, USA: Maker Media, 2014.

[8] S. Monk, *Programming Arduino: Getting Started with Sketches*, 2nd ed. New York, NY, USA: McGraw-Hill Education, 2016.

[9] J. Iqbal, "Design and Implementation of a Line Following Robot," *International Journal of Computer Applications*, vol. 100, no. 7, pp. 15–20, 2014.

[10] A. A. Bature, M. A. Sadiq, and U. A. Kolo, "Design and Construction of an Autonomous Line Following Robot," *International Journal of Engineering Research & Technology (IJERT)*, vol. 7, no. 4, pp. 234–239, 2018.

[11] S. M. Kim, Y. Choi, and J. Suh, "Applications of the Open-Source Hardware Arduino Platform in the Mining Industry: A Review," *Applied Sciences*, vol. 10, no. 14, p. 5018, 2020. doi: 10.3390/app10145018

[12] C. H. Choi, "Reference designs for embedded controls," in *2011 Frontiers in Education Conference (FIE)*, 2011, pp. S2F-1–S2F-6. doi: 10.1109/FIE.2011.6142744

[13] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Upper Saddle River, NJ, USA: Pearson, 2005.

[14] N. S. Nise, *Control Systems Engineering*, 7th ed. Hoboken, NJ, USA: Wiley, 2015.

[15] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 2nd ed. Cham, Switzerland: Springer, 2016.

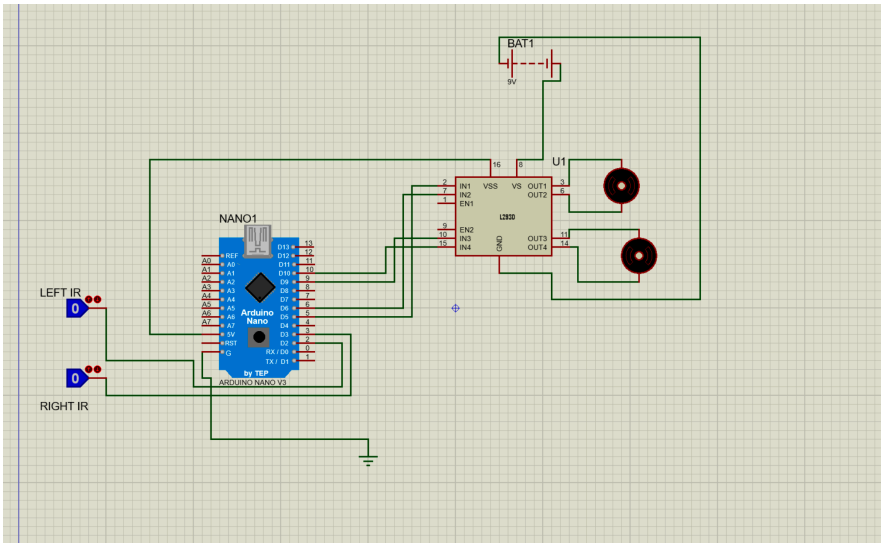
APPENDICES

- Sources Code:

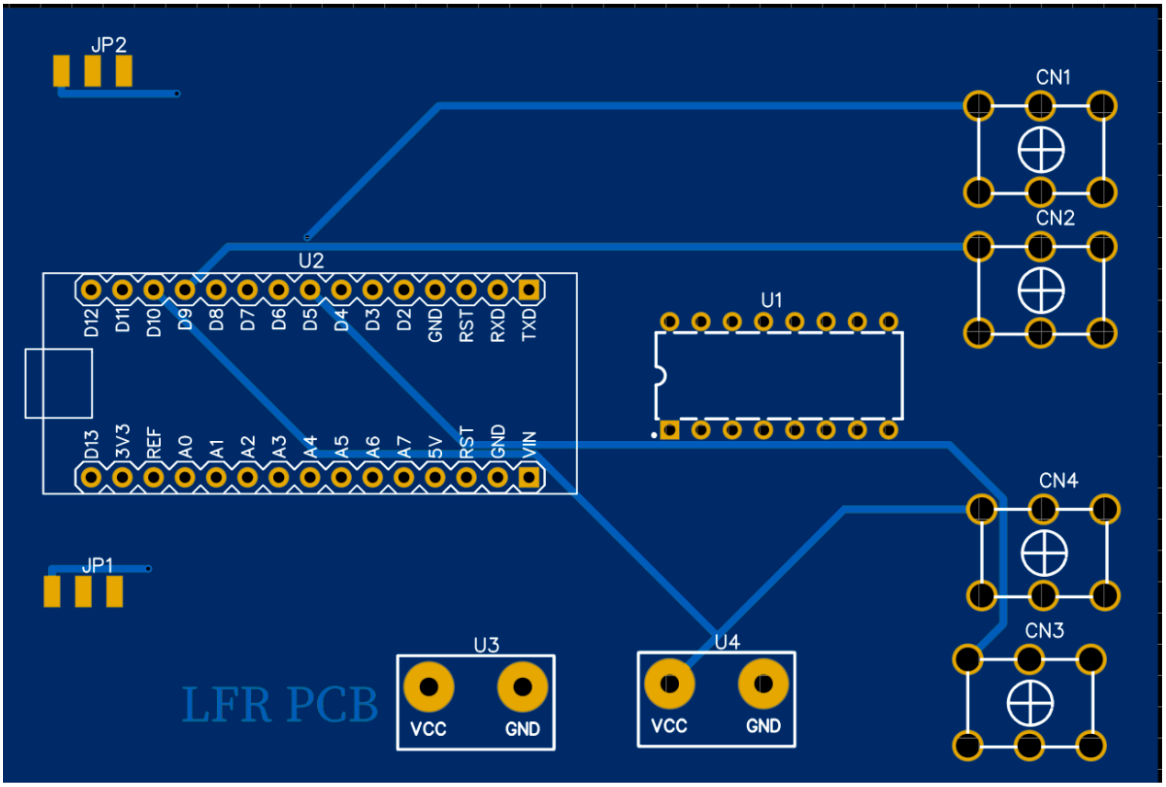
```
// ========================================================
// Line Following Robot
// Microcontroller: Arduino Uno (ATmega328P)
// ========================================================

// Sensor Pins
const int leftSensor = 2;
const int rightSensor = 3;

// Motor Driver Pins (L293D)
const int leftMotor1  = 4;
const int leftMotor2  = 5;
const int rightMotor1 = 6;
const int rightMotor2 = 7;

void setup() {
  // Configure sensor pins
  pinMode(leftSensor, INPUT);
  pinMode(rightSensor, INPUT);

  // Configure motor control pins
  pinMode(leftMotor1, OUTPUT);
  pinMode(leftMotor2, OUTPUT);
  pinMode(rightMotor1, OUTPUT);
  pinMode(rightMotor2, OUTPUT);
}

void loop() {

  int leftValue  = digitalRead(leftSensor);
  int rightValue = digitalRead(rightSensor);

  // Both sensors detect line
  if (leftValue == LOW && rightValue == LOW) {
    moveForward();
  }

  // Left sensor off line
  else if (leftValue == HIGH && rightValue == LOW) {
    turnLeft();
  }

  // Right sensor off line
  else if (leftValue == LOW && rightValue == HIGH) {
    turnRight();
```

```
  }

  // Both sensors off line
  else {
    stopMotors();
  }
}

// ================== Motor Functions ====================

void moveForward() {
  digitalWrite(leftMotor1, HIGH);
  digitalWrite(leftMotor2, LOW);
  digitalWrite(rightMotor1, HIGH);
  digitalWrite(rightMotor2, LOW);
}

void turnLeft() {
  digitalWrite(leftMotor1, LOW);
  digitalWrite(leftMotor2, HIGH);
  digitalWrite(rightMotor1, HIGH);
  digitalWrite(rightMotor2, LOW);
}

void turnRight() {
  digitalWrite(leftMotor1, HIGH);
  digitalWrite(leftMotor2, LOW);
  digitalWrite(rightMotor1, LOW);
  digitalWrite(rightMotor2, HIGH);
}

void stopMotors() {
  digitalWrite(leftMotor1, LOW);
  digitalWrite(leftMotor2, LOW);
  digitalWrite(rightMotor1, LOW);
  digitalWrite(rightMotor2, LOW);
}
```

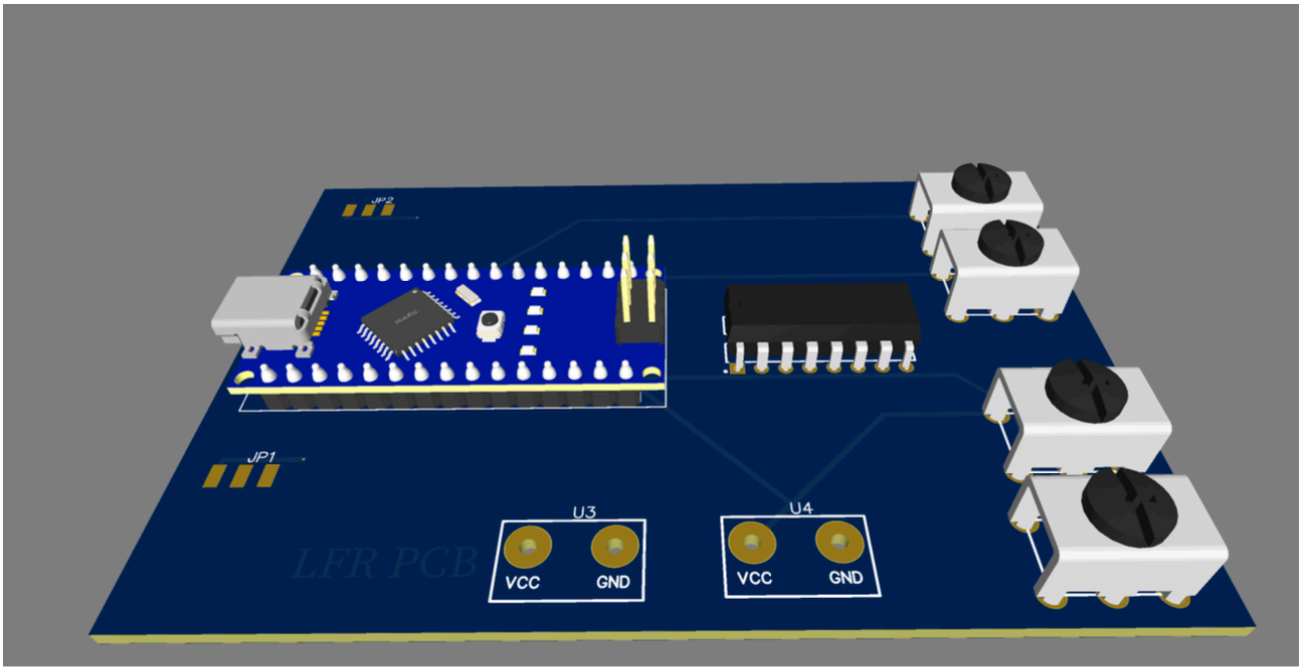
- **Proteus Schematic:**



- PCB Layout:



2D PCB



3D PCB

- **Algorithm:**

Step 1: Start

Step 2: Initialize sensor pins as INPUT

Step 3: Initialize motor control pins as OUTPUT

Step 4: Begin infinite loop

Step 5: Read left sensor value

Step 6: Read right sensor value

Step 7:

IF (Left == LOW AND Right == LOW)

Move Forward

ELSE IF (Left == HIGH AND Right == LOW)

Turn Left

ELSE IF (Left == LOW AND Right == HIGH)

Turn Right

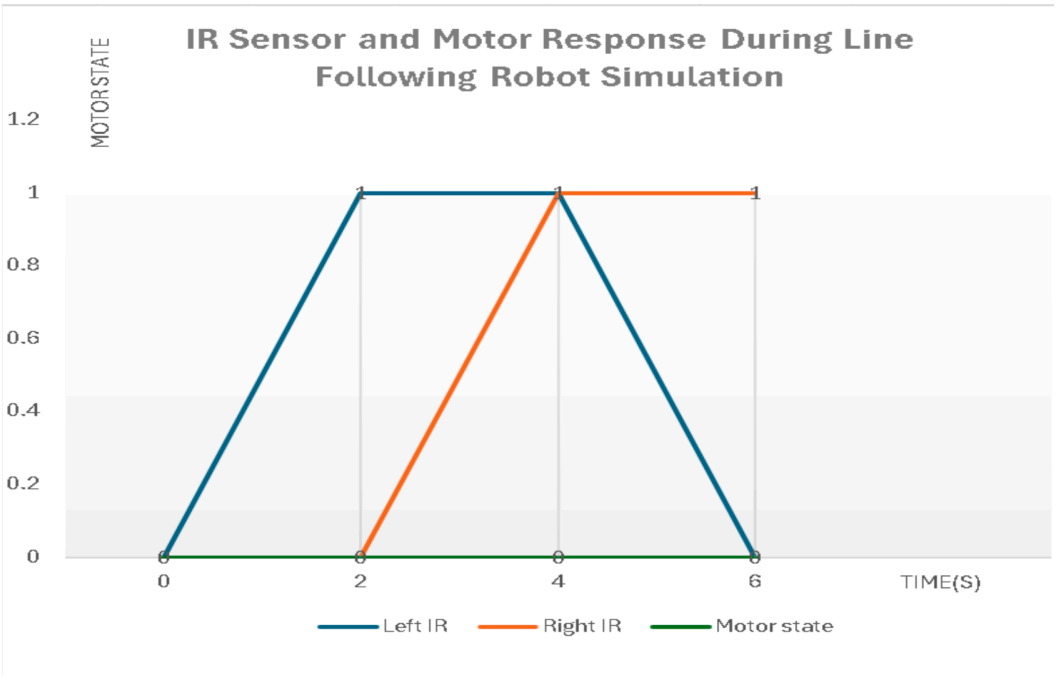
ELSE IF (Left == HIGH AND Right == HIGH)

Stop

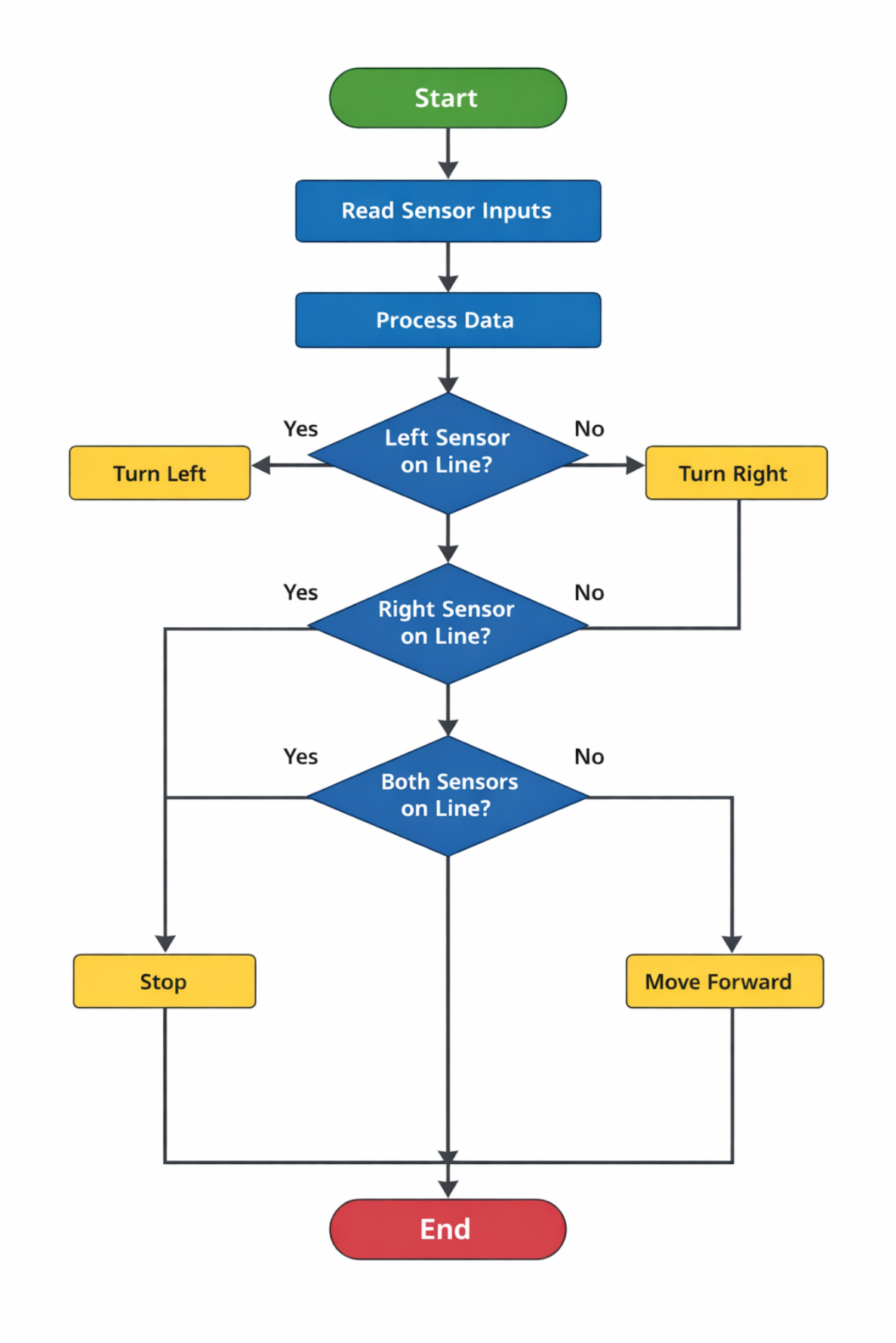
Step 8: Repeat loop

Step 9: End

- **Result Graph:**

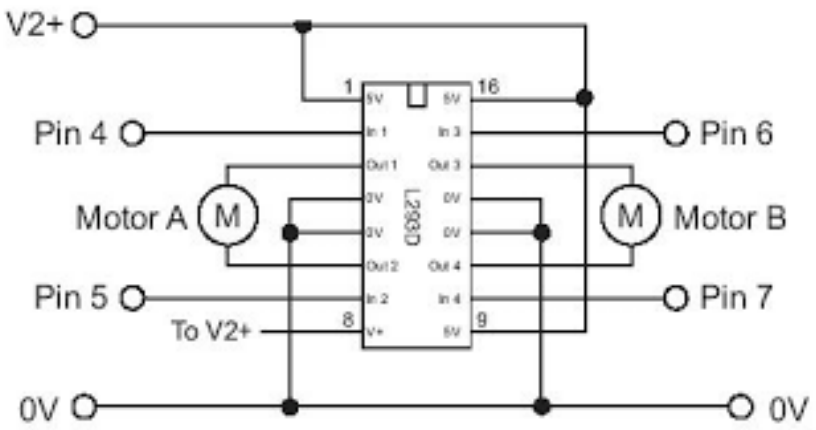


- Flowchart:

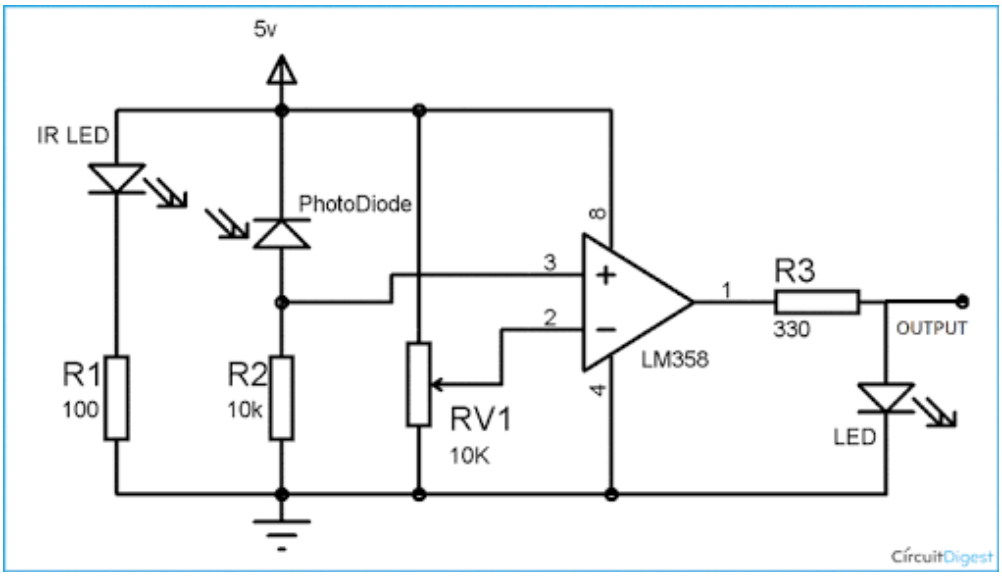

Start
Read Sensor Inputs
Process Data
Left Sensor on Line?
Yes
No
Turn Left
Turn Right
Right Sensor on Line?
Yes
No
Both Sensors on Line?
Yes
No
Stop
Move Forward
End

- **Circuits Diagrams:**

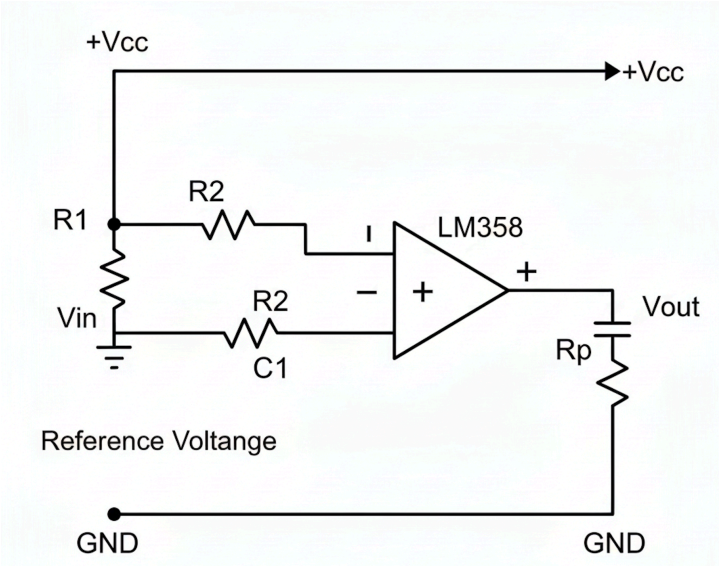
- Motor driver (L293D) Circuit Diagram:



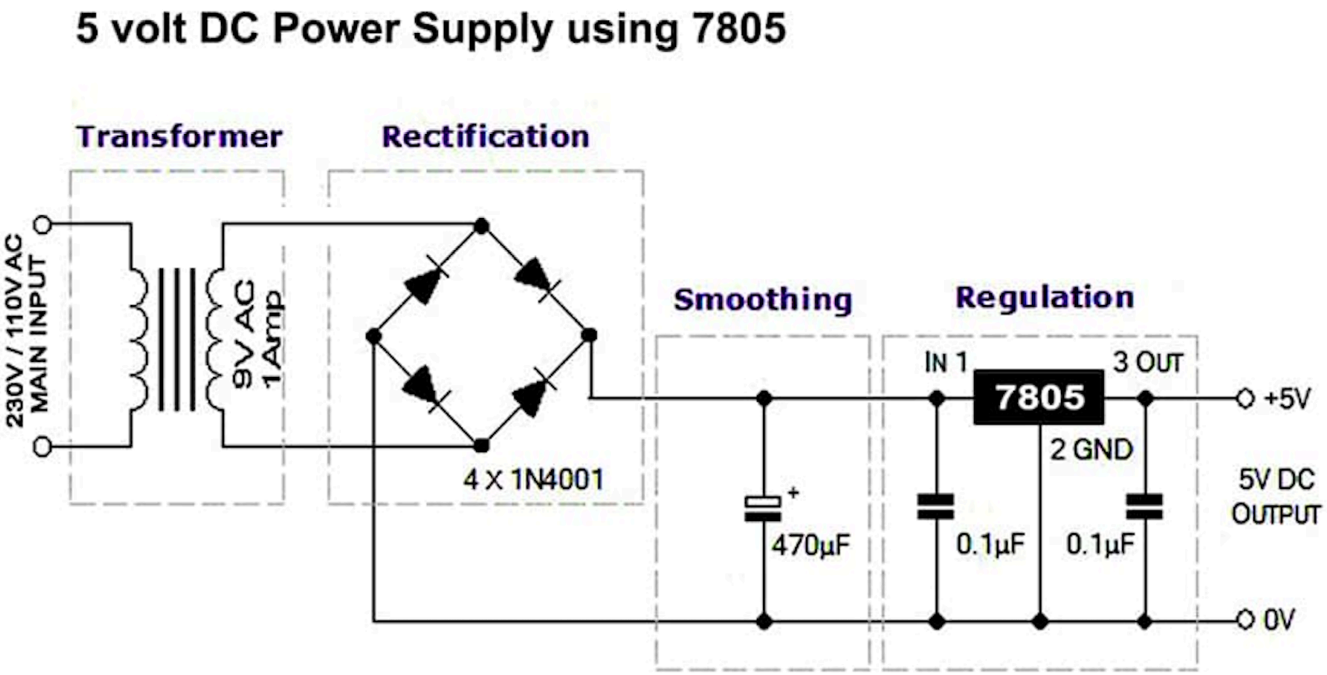
- infrared Circuit Diagram:



- Analog Signal Conditioning Circuit:



-Power Supply :



- **System Block Architecture:**