MAZE SOLVING ROBOT USING ARDUINO

End-Term Project Report For

Practical Robotics Projects with Arduino (CSE 4571)

Submitted by

ANSHUMAN PATTANAYAK	2241016334
MUGDHAA PATTANAYAK	2241016337
SANJEEB KUMAR PUSTI	2241016499
SHREETI BISWAL	2241002217
SURYAKANTA NISHANK	2241014157
TANAYA DASH	2241004201

B. Tech. 7 th Semester,CSE

Project Supervisor

Dr. BHABASIS MOHAPATRA (Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Institute of Technical Education and Research

SIKSHA'O'ANUSANDHAN DEEMED TO BE UNIVERSITY

Bhubaneswar, Odisha, India.

(January, 2026)

DECLARATION

We certify that

a. The work contained in this report is original and has been done by us under the guidance of our supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. We have followed the guidelines provided by the Department in preparing the report.

d. Whenever we have used materials (data, theoretical analysis, figures, and text) from other sources, we have given due credit to them by citing them in the text of the report and giving their details in the reference.

Name of the Student	Registration Number	Signature
Anshuman Pattanayak	2241016334	
Mugdhaa Pattanayak	2241016337	
Sanjeeb Kumar Pusti	2241016499	
Shreeti Biswal	2241002217	
Suryakanta Nishank	2241014157	
Tanaya Dash	2241004201	

REPORT APPROVAL

The report entitled

TITLE OF THE PROJECT

by

ANSHUMAN PATTANAYAK	**2241016334**
MUGDHAA PATTANAYAK	**2241016337**
SANJEEB KUMAR PUSTI	**2241016499**
SHREETI BISWAL	**2241002217**
SURYAKANTA NISHANK	**2241014157**
TANAYA DASH	**2241004201**

is approved for

Practical Robotics Projects with Arduino (CSE 4571)

in Computer Science and Engineering

Dr. Bhabasis Mohapatra

Date: _____

Place: _____

CONTENTS

ABSTRACT

The rapid growth of robotics and embedded systems has enabled the development of autonomous machines capable of operating with minimal human intervention. One important application of such systems is a maze solving robot, which is designed to navigate through an unknown maze environment and determine a valid path from the starting point to the destination using logical decision-making techniques. This project, titled "Maze Solving Robot using Arduino," focuses on the design, development, and implementation of an autonomous robotic system that efficiently solves a maze using sensor-based navigation.

The proposed system is built around an Arduino microcontroller, which functions as the central processing unit of the robot. Sensors such as infrared (IR) or ultrasonic sensors are used to detect walls, obstacles, and available paths within the maze. The sensor data is continuously monitored and processed by the microcontroller to make real-time navigation decisions. Based on predefined maze-solving logic, the Arduino controls the robot's movement by regulating the direction and speed of DC motors through a motor driver module, allowing the robot to move forward, turn left or right, or stop when required.

The robot employs a rule-based maze-solving algorithm, such as the left-hand or right-hand rule, which ensures systematic and reliable navigation without requiring prior knowledge of the maze layout. This approach offers simplicity, ease of implementation, and suitability for educational and prototype-level applications. The system operates in a fully autonomous manner without external control.

The hardware design prioritizes cost-effectiveness, simplicity, and modularity, enabling easy modification and future expansion. The software is developed using the Arduino Integrated Development Environment (IDE) and written in embedded C, ensuring efficient interaction between sensors and actuators. Extensive testing was conducted across different maze configurations to evaluate performance. The results confirm that the robot accurately detects obstacles, makes appropriate navigation decisions, and successfully reaches the destination with minimal error.

Overall, this project demonstrates the practical integration of robotics, embedded systems, automation, and control logic. It provides valuable hands-on experience in hardware interfacing, algorithm implementation, and real-time decision-making. The concepts explored have potential applications in autonomous navigation, warehouse automation, rescue missions, and intelligent robotic systems, and the project establishes a strong foundation for future enhancements such as advanced algorithms, path optimization, wireless communication, and vision-based navigation.

Chapter 1: Introduction

In recent years,robotics andembedded systemshave gainedsignificant importance due to their wide-ranging applications in automation, artificial intelligence, and real- world problem solving. One such practical application is the development of a maze- solving robot. A maze-solving robot is an autonomous system designed to navigate through a maze from a starting point to a destination without human intervention. The robot uses sensors, control algorithms, and decision-making logic to detect obstacles, choose appropriate paths, and reach the goal efficiently. This project focuses on the design and development of a maze-solving robot using an Arduino microcontroller. Arduino is a widely used open-source electronics platform known for its simplicity, flexibility, and affordability. By integrating Arduino with sensors and motors, the robot is capable of sensing its surroundings, processing environmental data, and executing movement commands in real time. The maze- solving robot demonstrates the practical implementation of concepts such as embedded programming, sensor interfacing, control systems, and algorithmic thinking. The maze-solving robot has applications in fields such as autonomous navigation, warehouse automation, rescue operations, and robotics research. Through this project, students gain hands-on experience in hardware integration and software development while solving a real- world navigation problem.

1.1. Motivation

- The primary motivation behind developing a maze-solving robot is to understand and implement autonomous decision-making in robotic systems. Navigation through an unknown environment is a fundamental challenge in robotics, and maze solving provides an ideal platform to study this problem in a controlled manner. By designing a robot that can independently analyze paths
- and make movement decisions, students can bridge the gap between theoretical knowledge and practical implementation. Another important motivation is the growing demand for intelligent autonomous systems in modern industries. Robots that can navigate complex environments are widely used in applications such as automated guided vehicles (AGVs), robotic vacuum cleaners, and exploration robots. This project helps students develop skills that are directly relevant to such real-*1* world applications.

1.2. Design Goals

Thedesigngoals ofthe maze-solving robot are established to ensure that the system is efficient, reliable, and easy to implement. The robot is designed with simplicity, accuracy, and adaptability in mind while maintaining low cost and ease of maintenance. The primary design goals include:

- Developing an autonomous robot capable of navigating a maze without human assistance.
- Ensuring accurate obstacle detection using sensors.
- Implementing an effective decision-making algorithm for path selection.
- Achieving smooth and controlled movement of the robot.
- Designing a modular and scalable system that allows future improvements.

1.2.1. Purpose

The purpose of this project is to design and implement a functional maze-solving robot using an Arduino microcontroller. The robot is intended to autonomously detect obstacles, identify possible paths, and select the optimal route to reach the destination. This project aims to demonstrate the integration of hardware and software components in a real-time embedded system. It also serves as a learning platform for understanding microcontroller programming, sensor interfacing, motor control, and algorithm implementation. The robot acts as a practical example of autonomous navigation and intelligent control systems.

1.2.2. Scope

Thescope ofthisproject is limited to the development of a small-scale maze-solving robot that operates in a predefined maze environment. The robot uses basic sensors such as ultrasonic or infrared sensors to detect walls and obstacles. The decision- making process is based on predefined algorithms rather than complex machine learning techniques. The project includes:

- Design of the robot chassis and mechanical structure.
- Interfacing sensors and motors with the Arduino board.
- Development of control logic for movement and turning.
- Implementation of a maze-solving algorithm. Testing
- and validation of the robot in a physical maze.

The project does not include advanced features such as vision-based navigation, simultaneous localization and mapping (SLAM), or wireless communication.However,the design allows for future expansion to include such features.

1.2.3. Applicability

Themaze-solvingrobot has wide applicability in both academic and practical domains. Academically, it is highly useful for students studying robotics, electronics, and computer science as it provides hands-on experience with embedded systems and autonomous control. Practically, the concepts used in this project can be applied to real-world navigation systems such as:

- Automated guided vehicles in warehouses. Autonomous
- robots used in search and rescue missions. Industrial
- robots navigating factory floors. Smart service robots
- used in homes and offices.

The project also serves as a foundation for more advanced robotic systems that requireintelligentnavigationand obstacle avoidance.

1.3. Problem Statement

Theproblemaddressedinthisproject is to design and develop an autonomous robot that can successfully navigate through a maze from a starting point to an endpoint without human intervention. The robot must be capable of detecting walls and obstacles, making decisions at intersections, and choosing the correct path to reach the goal. The challenge lies in accurately sensing the environment, processing sensor data in real time, and implementing a reliable maze-solving algorithm using limited computational resources of the Arduino microcontroller. The robot must also handle uncertainties such as sensor noise, uneven surfaces, and minor mechanical errors while maintaining stable and efficient movement.

1.4. Organization of the Report

This reportisorganized intoseveralchapters, each describing a specific aspect of the maze-solving robot project:

- **Chapter 1:** Introduction
  Presents the overview of the project, motivation, design goals, problem statement, structure of the report.
- **Chapter 2:** Literature Review
  Discusses existing maze-solving techniques, related robotic systems, and previous work in autonomous navigation.
- **Chapter 3:** System Architecture and Hardware Design
  Describes the overall system architecture, hardware components used, and their interconnections.
- **Chapter 4:** Software Design and Algorithm
  Explains the Arduino programming, maze-solving logic, and control algorithms implemented in the robot.
- **Chapter 5:** Implementation and Testing
  Details the construction of the robot, testing procedures, and experimental results.
- **Chapter 6:** Results and Discussion
  Analyzes the performance of the robot and discusses its efficiency and limitations.
- **Chapter 7:** Conclusion and Future Scope
  Summarizes the project outcomes and suggests possible enhancements for future development.

Chapter 2: Literature Survey

The literature survey provides an overview of existing research, techniques, and autetocnhonmolougsic naal vidgaevtieolno psmysetnemtss . rIet lhaetelpds into u ndmearzseta-snodlivnign gh owro sbiomtsi lar problems have been addressed previously and identifies the methodologies, algorithms, and hardware platforms commonly used. This chapter also highlights the limitations of earlier approaches, which guided the design and implementation of the proposed system. Maze-solving robots have been an area of interest in robotics research for several decades. They serve as a benchmark problem for testing autonomous navigation, decision-making algorithms, and sensor integration. Over time, different maze-solving strategies and hardware configurations have been proposed, ranging from simple rule-based methods to advanced artificial intelligence-based solutions.

2.1. Background work done so far

Early maze-solving robots mainly used simple rule-based techniques such as the left-hand rule and right-hand rule, where the robot follows one wall of the maze until it reaches the exit. These methods are easy to implement and require minimal computational resources, but they do not always guarantee the shortest path and may fail in complex mazes with loops. With improvements in microcontroller technology, more systematic algorithms such as depth-first search (DFS) and breadth-first search (BFS) were introduced. BFS-based approaches are capable of finding the shortest path but require more memory, while DFS requires less memory but may result in longer routes. These algorithms improved navigation accuracy compared to simple wall-following methods. The use of Arduino microcontrollers became popular due to their low cost, ease of programming, and availability of open- source libraries. Many studies implemented Arduino-based maze-solving robots using ultrasonic and infrared sensors for obstacle detection and motor driver modules for movement control.

In recent years, advanced maze-solving approaches have been developed using mapping and path optimization techniques. Algorithms such as Flood Fill and Dijkstra's algorithm have been widely used in competitive robotics events like micromouse competitions. These techniques enable the robot to create an internal map of the maze and compute the shortest path to the destination. Although these methods offer high efficiency and optimal solutions, they require more processing power and memory, making them less suitable for basic Arduino platforms without optimization. Some studies also investigated the use of machine learning and computer vision for maze navigation. Camera-based systems combined with image processing algorithms allow robots to identify paths visually.

However, such systems require powerful processors and are computationally expensive. As a result, they are not commonly used in low-cost educational robots and are mostly limited to research and industrial applications. From the reviewed literature, it is evident that rule- based and sensor-driven approaches remain the most practical solutions for low-cost maze-solving robots. Arduino-based implementations using ultrasonic or IR sensors and simple decision-making algorithms offer a good balance between performance, cost, and complexity. These findings influenced the design of the proposed system, which adopts a priority-based decision-making logic to ensure reliable navigation while maintaining simplicity and affordability.

Chapter 3: Design Scheme

This chapter explains the complete design methodology of the maze-solving robot, including the system design, architecture, component selection, implementation process, and design improvements. The aim of this design scheme is to develop a reliable and efficient autonomous robot using an Arduino microcontroller that can successfully navigate through a maze environment.

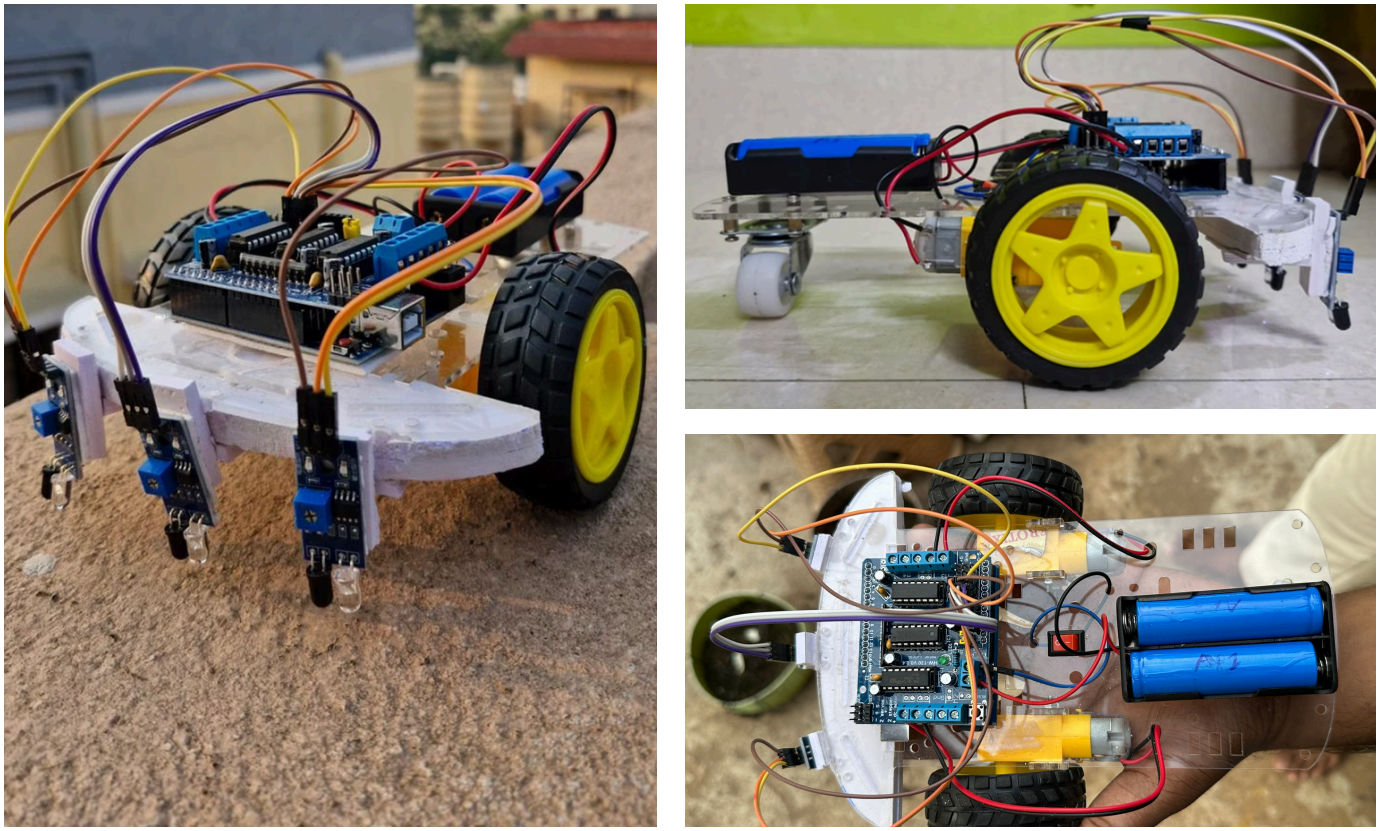


Fig. 3.0 Final Design of The Robot

3.1. SystemDesign

The system design of the maze-solving robot is based on the integration of sensing, processing, and actuation units. The robot continuously senses the surrounding environment using distance sensors to detect walls and obstacles present in the maze. The collected sensor data is processed by the Arduino microcontroller, which executes the maze-solving algorithm and generates appropriate control signals. These control signals are then sent to the motor driver circuit, which controls the movement of the DC motors. The system operates in a closed-loop manner, allowing the robot to make real-time decisions and correct its movement based on sensor feedback.

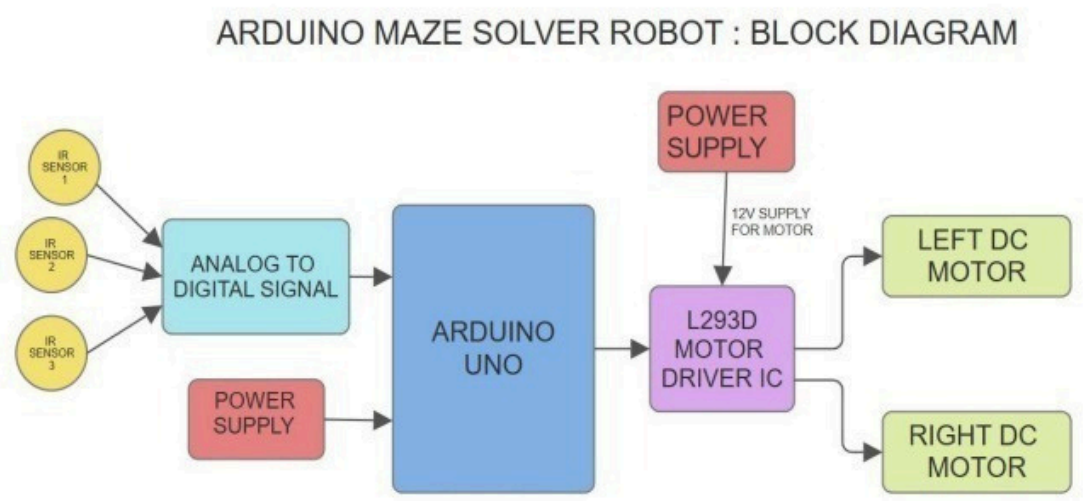


Fig. 3.1 System Block Diagram of Maze-Solving Robot

3.2. Architecture

The architecture of the proposed maze-solving robot follows a layered approach to ensure simplicity and efficiency. The input layer consists of ultrasonic or infrared sensors that detect obstacles and measure the distance from maze walls. The processing layer includes the Arduino microcontroller, which receives sensor data and applies decision-making logic to determine the direction of movement. The output layer comprises the motor driver module and DC motors, which execute the movement commands such as forward motion, left turn, right turn, or stop. This structured architecture improves modularity and makes the system easier to debug and modify.

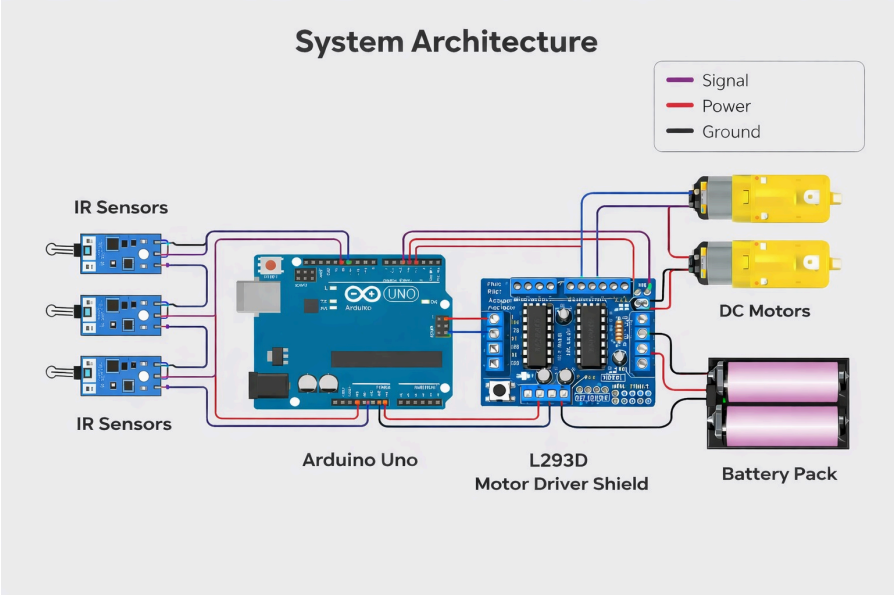


Fig. 3.2 Architecture of the Maze-Solving Robot

3.3. Component Design

The component design focuses on selecting suitable hardware that ensures reliability, low cost, and compatibility with Arduino. The Arduino Uno is used as the main controller due to its ease of programming and sufficient input/output pins. Ultrasonic or infrared sensors are used for obstacle detection because of their accuracy and low power consumption. A motor driver module such as L298N or L293D is employed to control the speed and direction of DC motors, as the Arduino alone cannot drive motors directly. DC motors with wheels provide the necessary movement, while a battery-based power supply ensures portability of the robot.

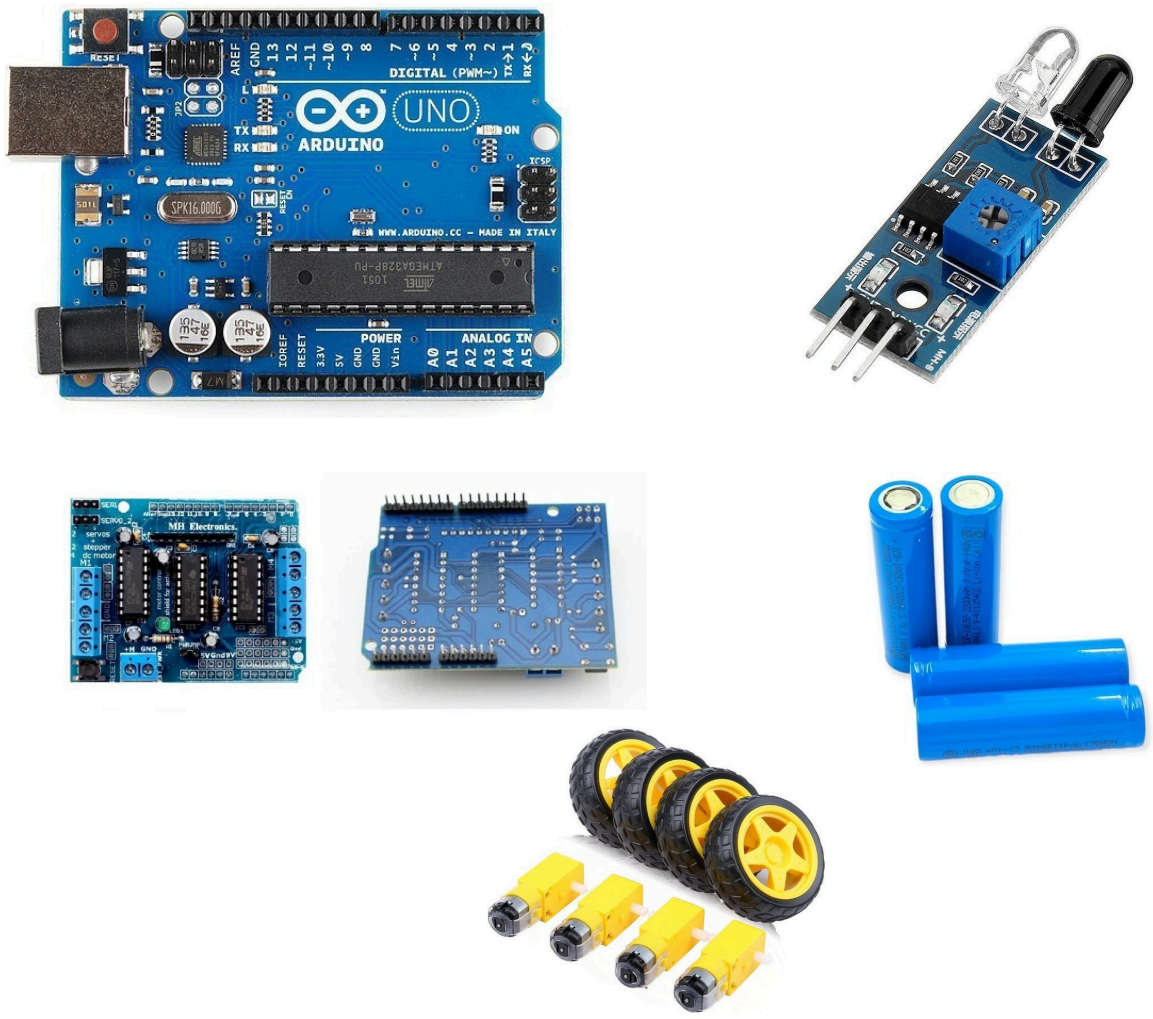


Fig. 3.3 Hardware Components Used

3.4. Implementation

The implementation phase involves assembling the hardware components and developing the control software. Sensors are interfaced with the Arduino to continuously read distance values, while the motor driver is connected to control motor movement. The Arduino program is written to implement a maze-solving logic that allows the robot to choose the appropriate path at intersections. The robot is tested in a predefined maze to verify obstacle detection, turning accuracy, and overall navigation performance. Necessary adjustments are made during testing to improve stability and responsiveness.

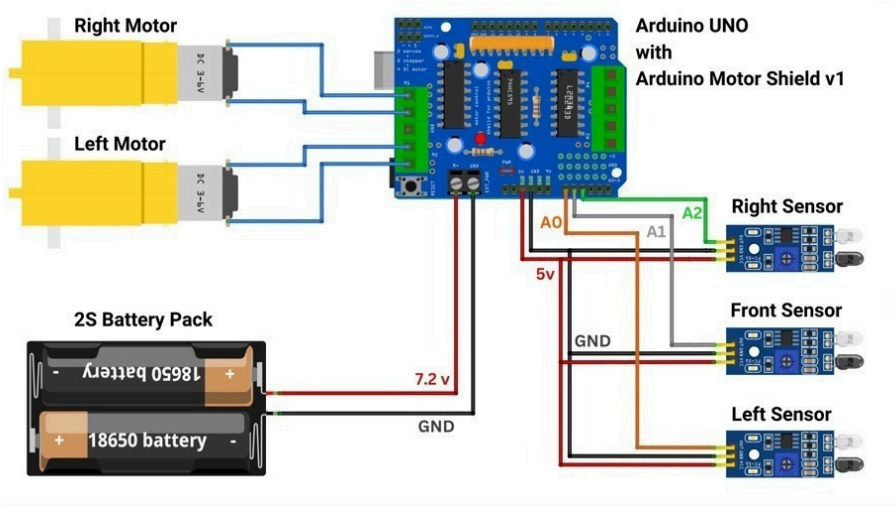


Fig. 3.4 Implementation and Testing Setup

3.5. Design Evolution

During the initial stages, the robot was designed for basic obstacle avoidance. However, through repeated testing and observation, several improvements were made to enhance performance. Sensor positioning was adjusted to improve wall detection accuracy, and motor speed control was refined for smoother turns. The maze-solving logic was also optimized to reduce unnecessary movements. These design modifications resulted in better navigation efficiency and improved reliability of the robot in complex maze paths.

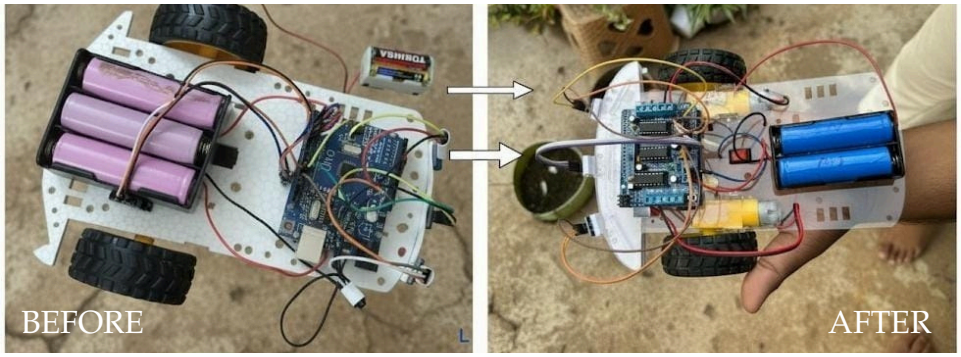


Fig. 3.5 Design Evolution of the Robot

CODE:

```
#include <AFMotor.h> // Include the Adafruit Motor Shield library for motor control
// Motor Definitions
AF_DCMotor motorA(1); // Motor A connected to terminal M1 on the motor shield
AF_DCMotor motorB(3); // Motor B connected to terminal M3 on the motor shield
// Sensor Pin Definitions
const int leftSensor = A0; // Left IR sensor pin
const int frontSensor = A1; // Front IR sensor pin
const int rightSensor = A2; // Right IR sensor pin
// Movement Parameters
const int forwardSpeed = 100; // Speed for forward movement
const int TurningSpeed = 100; // Speed for turning movements
const int turnDelay = 25;      // Delay for completing a turn
const int uTurnDelay = 30; // Delay for completing a U-turn
void setup() {
// Configure sensor pins as input
pinMode(leftSensor, INPUT);
pinMode(frontSensor, INPUT);
pinMode(rightSensor, INPUT);
// Initialize serial communication for debugging
Serial.begin(9600);
}
void loop() {
// Read sensor values (0 or 1)
int leftValue = digitalRead(leftSensor);
int frontValue = digitalRead(frontSensor);
int rightValue = digitalRead(rightSensor);
// Combine sensor states into a single value for switch-case logic
int sensorState = (leftValue << 2) | (frontValue << 1) | rightValue;
// Decision-making based on sensor states
switch (sensorState) {
   case 0b000:
     uTurn(); // Perform a U-turn
     Serial.println("Stop");
     break;
        case 0b010:
         moveForward(); // Move forward
         Serial.println("Move Forward");
         break;
         case 0b111:
         turnLeft(); // Turn left
         Serial.println("Turn Left");
        break;
         case 0b100:
         turnLeft(); // Turn left
         Serial.println("Turn Left");
         break;
         case 0b110:
         turnLeft(); // Turn left
         Serial.println("Turn Left");
         break;
         case 0b001:
         turnRight(); // Turn right
         Serial.println("Turn Right");
         break;
```

```
 case 0b011:
 turnRight(); // Turn right
 Serial.println("Turn Right");
 break;
 case 0b101:
 stopMotors(); // Stop the motors
 Serial.println("Turn Left");
 break;
 default: // Unknown sensor state
 stopMotors(); // Stop the motors as a safety measure
 Serial.println("Unknown State");
 break;
 }


}

// Function to move forward
void moveForward() {
motorA.setSpeed(forwardSpeed); // Set speed for motor A
motorB.setSpeed(forwardSpeed); // Set speed for motor B
motorA.run(FORWARD); // Move motor A forward
motorB.run(FORWARD); // Move motor B forward
}

// Function to turn left
void turnLeft() {
motorA.setSpeed(TurningSpeed - 20); // Reduce speed of motor A for smoother turn
motorB.setSpeed(TurningSpeed); // Set speed for motor B
motorA.run(BACKWARD); // Move motor A backward
motorB.run(FORWARD); // Move motor B forward
delay(turnDelay); // Delay to complete the turn
}


// Function to turn right
void turnRight() {
motorA.setSpeed(TurningSpeed); // Set speed for motor A
motorB.setSpeed(TurningSpeed - 20); // Reduce speed of motor B for smoother turn
motorA.run(FORWARD); // Move motor A forward
motorB.run(BACKWARD); // Move motor B backward
delay(turnDelay); // Delay to complete the turn
}


// Function to stop the motors
void stopMotors() {
motorA.run(RELEASE); // Release motor A
motorB.run(RELEASE); // Release motor B
}


// Function to perform a U-turn
void uTurn() {
motorA.setSpeed(TurningSpeed); // Set speed for motor A
motorB.setSpeed(TurningSpeed); // Set speed for motor B
motorA.run(FORWARD); // Move motor A forward
motorB.run(BACKWARD); // Move motor B backward
delay(uTurnDelay); // Delay to complete the U-turn
}
```

Chapter 4: Testing , Analysis, and Evaluation

This chapter discusses the testing procedures adopted to evaluate the performance of the maze-solving robot. Various testing criteria were defined to ensure that the robot functions correctly, meets the intended project objectives, and performs efficiently within a maze environment. The evaluation of the system is carried out based on its relevance to the problem statement, the effectiveness of its navigation and decision-making capabilities, and the overall efficiency of its operation under different test conditions.

4.1.Testing Criterions

The testing criteria were designed to assess the overall functionality and performance of the maze-solving robot. Testing was carried out in a controlled maze environment with different paths, turns, and dead ends. The robot was observed for its ability to detect obstacles, make correct navigation decisions, and reach the destination without human intervention. Multiple test runs were conducted to ensure consistency and reliability of results. The graph illustrates the performance of the maze-solving robot across multiple test runs. It is observed that the time taken to solve the maze decreases gradually with each test run. This improvement is due to better sensor response, optimized motor control, and refined decision-making logic. The results indicate that the robot performs efficiently and consistently after initial testing, validating the effectiveness of the implemented design and algorithm.

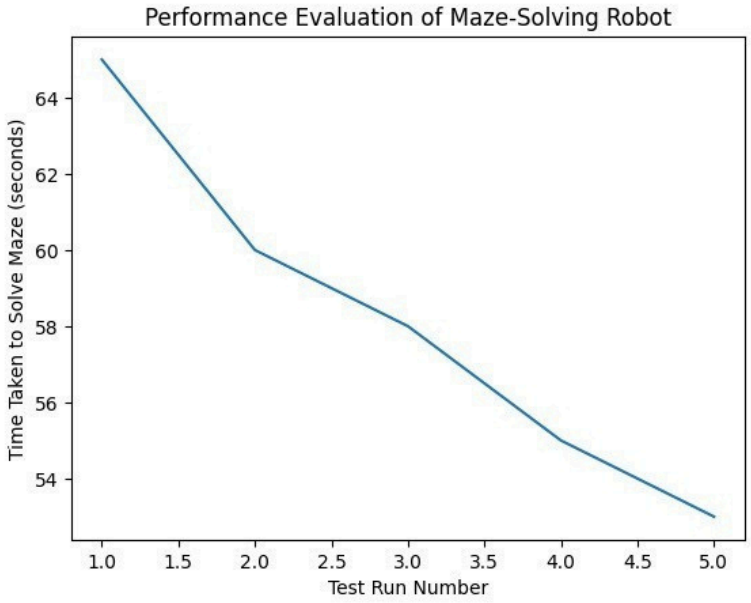


Fig 4.1 Performance Evaluation Graph

4.1.1. Relevance

Relevance refers to how well the testing conditions match the actual objectives of the project. The test cases were designed to closely resemble real maze scenarios, including straight paths, junctions, and blocked routes. This ensured that the robot's behavior during testing accurately reflected its intended use. By testing the robot in environments similar to real-world maze structures, the evaluation remained relevant to the project goals of autonomous navigation and obstacle avoidance.

4.1.2 Effectiveness

Effectiveness measures the ability of the robot to successfully solve the maze and reach the target location. During testing, the robot's decision-making accuracy, turning precision, and obstacle detection capability were evaluated. The robot was considered effective if it consistently chose valid paths, avoided collisions, and completed the maze without manual assistance. The results demonstrated that the implemented algorithm enabled reliable navigation and successful maze completion under normal operating conditions.

4.1.3 Efficiency

Efficiency refers to how optimally the robot completes the maze in terms of time, movement, and resource usage. The robot's speed, number of turns, and unnecessary movements were analyzed during multiple test runs. Efficient performance was indicated by smooth movement, minimal delays at junctions, and reduced power consumption. The Arduino-based control system ensured quick processing of sensor data, resulting in timely decisions and efficient maze traversal.

Chapter 5: Socio-Economic Issues associated with the Project

This chapter discussesthesocio-economicaspects relatedtothemaze-solving robot pro ject. It includes cost considerations, safety aspects, global impact, and the importance of lifelong learning. The chapter highlights how such a project contributes not only to technical development but also to social awareness and economic feasibility.

5.1. Detailed Cost Analysis

Cost analysis plays a crucialrole in determining the feasibility of any engineering project. The maze-solving robot is designed to be low-cost and affordable, making it suitable for educational and small-scale applications. By using commonly available components such as Arduino boards, sensors, and DC motors, the overall project cost is minimized without compromising performance. This cost-effective approach ensures that the project can be easily replicated by students and institutions with limited resources.

5.1.1. Cost Analysis

The total cost of the maze-solving robot depends on the selection of components and their market availability. The Arduino microcontroller forms the core of the system and is relatively inexpensive. Sensors such as ultrasonic or infrared modules add moderate cost but provide essential functionality. Additional expenses include motor drivers, DC motors, wheels, chassis, and power supply. Overall, the project is economically viable and demonstrates how functional robotic systems can be developed within a limited budget.

5.1.2. Bill of Materials

The bill of materials for the maze-solving robot includes all the essential hardware components required for the successful implementation of the project. It covers the Arduino microcontroller, sensors, motor driver module, DC motors, wheels, chassis, power supply, and connecting wires. These components were selected based on availability, cost-effectiveness, and compatibility with the system. The complete list of components along with their quantities and costs is shown in the bill of materials image provided below for reference.*5*

Cost Estimation Table

Cost Head	Description	Approx. Cost (INR)
Control Unit	Arduino Uno + WiFi R3 (ATmega328p + ESP8266)	550 – 650
Sensing Unit	Digital IR Sensor Module (LM393) × 4	150 – 200
Motor Driver	L293D 3-in-1 Motor Driver Shield for Arduino	200 – 250
Actuators	Two DC Geared Motors with 2WD Chassis Kit	350 – 450
Power Supply	18650 Li-ion Batteries (3.7V, 2200mAh) × 3	250 – 300
Battery Holders	Single & Triple 18650 Battery Holders	70 – 100
Mechanical Structure	Robot chassis, wheels, motor mounts	150 – 200
Wiring / PCB / Connectors	Jumper wires, connectors, small PCB	150 – 200
Others	Nuts, bolts, spacers, casing	100 – 150
Total estimated cost: ₹1,600 – ₹1,800		

(The cost may vary depending on component brand, quality, and local market availability).

Fig. 5.1 Bill of Materials of the Maze-Solving Robot

5.2. Safety issues

Safety isanimportant consideration in the design and operation of the maze-solving robot. Since the robot operates at low voltage levels, the risk of electrical hazards is minimal. Proper insulation of wires and secure mounting of components reduce the chances of short circuits and mechanical failure. During testing, care must be taken to avoid collisions that could damage the robot or surrounding objects. Overall, the project is safe for educational usewhen basic precautions are followed.

5.3. Global Impact

The maze-solvingrobot project has a positive global impact, particularly in the fields of education and automation. It promotes hands-on learning and encourages students worldwide to explore robotics and embedded systems. The concepts used in this project are applicable to real-world problems such as automated transportation,warehouse robotics, and search-and-rescue operations. By fostering innovation and technical skills, such projects contribute to the global advancement of automation and intelligent systems.

5.4. Lifelong Learning

This project encourages lifelong learning by motivating students to continuously upgrade their technical knowledge and skills. Working on a maze-solving robot exposes learners to practical challenges in programming, electronics, and system integration. The experience gained through this project builds a strong foundation for future learning in advanced robotics, artificial intelligence, and embedded system design. It also instills problem-solving abilities and adaptability, which are essential skills in a rapidly evolving technological world.

Chapter 6: Engineering Tools and Standards used in the Project

ENGINEERING TOOLS USED:

- **ArduinoIDE:**Used for writing,compiling, and uploading the control program to the Arduino microcontroller.
- **Arduino Uno Microcontroller:** Acts as the main processing unit of the maze-solving robot.
- **IR Sensors:** path detection in the maze.
- **Motor Driver Module (L293D):** Used to control the speed and direction of DC motors.
- **DC Motors and Wheels:** Used for movement and navigation of the robot.
- **Jumper Wires:** Used for circuit connections and prototyping.
- **Power Supply / Battery Pack:** Provides required electrical power to the robot.
- **Computer / Laptop:** Used for programming, simulation, and debugging.

SOFTWARE TOOLS USED:

- **EmbeddedC++/ArduinoProgramming Language:** Used to implement the maze-solving logic.
- **Serial Monitor:** Used for debugging sensor values and decision-making outputs.
- **Flowcharts:** Used to design and visualize the maze-solving algorithm.

Chapter 7: Problems, faults, bugs, challenges

7.1. Problems

- Inaccurate path detection due to improper sensor alignment.
- Irregular movement caused by unequal motor speeds.
- Difficulty in detecting sharp turns and dead ends in the maze.
- Slippage of wheels on smooth surfaces affecting navigation.
- Delay in response when multiple paths are detected simultaneously.

7.2. Faults

- Loose wiring connections leading to intermittent operation.
- Power supply fluctuations affecting sensor performance.
- Motor driver overheating during prolonged usage.
- Battery voltage drop reducing motor efficiency.
- Sensor malfunction due to dust or external interference.

7.3. Bugs

- Logical errors in the maze-solving algorithm at junctions.
- Delay in sensor data processing causing late decisions.
- Incorrect condition handling in the Arduino program.
- Infinite loop conditions during wrong path detection.
- Improper threshold values causing false obstacle detection.

7.4. Challenges

- Integrating hardware components with software logic.
- Balancing speed, accuracy, and stability of the robot.
- Ensuring consistent performance in different maze layouts.
- Minimizing noise and error in sensor readings.
- Optimizing code within limited Arduino memory and processing power.

Chapter 8: Teamwork

8.1. Summary of teamwork

8.1.1 Attributes

1	Attends group meetings regularly and arrives on time.
2	Contributes meaningfully to group discussions.
3	Completes group assignments on time.
4	Prepares work in a quality manner.
5	Demonstrates a cooperative and supportive attitude.
6	Contributes significantly to the success of the project.

8.1.2 Score

1=strongly disagree; 2=disagree; 3=agree; 4=strongly agree

Student 1: Suryakanta Nishank **Student 2:** Mugdhaa Pattanayak

Student 4: Sanjeeb Kumar Pusti **Student 3:** Tanaya Dash

Student 5: Anshuman Pattanayak **Student 6:** Shreeti Biswal

Mugdhaa Pattanayak	Suryakanta Nishank	
	Attributes	Student 1
	1	4
	2	4
	3	4
	4	3
	5	4
	6	3
	Grand Total	22

Suryakanta Nishank	Mugdhaa Pattanayak	
	Attributes	Student 1
	1	3
	2	4
	3	4
	4	4
	5	4
	6	4
	Grand Total	23

Signature of Student 1

Signature of Student 2

Sanjeeb Kumar Pusti	Tanaya Dash	
	Attributes	Student 1
	1	3
	2	4
	3	4
	4	4
	5	4
	6	4
	Grand Total	23

Tanaya Dash	Sanjeeb Kumar Pusti	
	Attributes	Student 1
	1	3
	2	4
	3	4
	4	3
	5	4
	6	4
	Grand Total	22

Signature of Student 1

Signature of Student 2

Shreeti Biswal	Anshuman Pattanayak	
	Attributes	Student 1
	1	3
	2	4
	3	4
	4	3
	5	4
	6	4
	Grand Total	22

Anshuman Pattanayak	Shreeti Biswal	
	Attributes	Student 1
	1	3
	2	4
	3	4
	4	4
	5	4
	6	4
	Grand Total	23

Signature of Student 1

Signature of Student 2

Chapter 9: Conclusion

The line following maze solving robot successfully demonstrates the concept of autonomous navigation by combining sensor-based perception with intelligent decision-making. Using infrared sensors to detect the path and a microcontroller to process sensor data, the robot is able to accurately follow lines, identify junctions, and choose the correct direction within a maze. The system operates without any external control, highlighting the effectiveness of real-time feedback and control algorithms in robotic movement. This project not only validates the working of line-following mechanisms but also provides practical exposure to embedded programming, motor control, and sensor calibration, which are essential components of modern robotics.

In addition, the line following maze solving robot serves as a scalable platform for future research and development. By incorporating advanced features such as shortest-path algorithms, memory-based optimization, obstacle detection, or IoT connectivity, the robot can be enhanced to handle more complex environments. The knowledge gained from this project has significant relevance to real-world applications including automated guided vehicles, warehouse automation, and intelligent transportation systems. Overall, this project strengthens problem-solving skills, encourages innovation, and bridges the gap between theoretical concepts and practical implementation in the field of robotics and automation.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 2672{2680. Curran Associates, Inc., 2022. URL http://papers.nips.cc/paper/5423- generative-adversarial-nets.pdf

[2] Circuit Digest, "Human Following Robot Using Arduino and Ultrasonic Sensor," Microcontroller Projects Tutorial. Available at: circuitdigest.com (Original project reference)

[3] Arduino Official Documentation, "Arduino UNO Board Overview and Technical Specifications." Available at: arduino.cc

[4] HC-SR04 Ultrasonic Sensor Datasheet, "Ultrasonic Ranging Module Specifications and Operation."

[5] L298N Motor Driver Datasheet, "Dual Full-Bridge Driver for DC and Stepper Motors."

[6] T. R. Kurfess, "Robotics and Automation Handbook," CRC Press.

[7] S. Niku, "Introduction to Robotics: Analysis, Control, Applications," Wiley Publications.

[8] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza, "Introduction to Autonomous Mobile Robots," MIT Press.

[9] J. F. Engelberger, "Robotics in Practice: Management and Applications," Springer.

[10] G. McComb and M. Predko, "Robot Builder's Bonanza," McGraw-Hill.

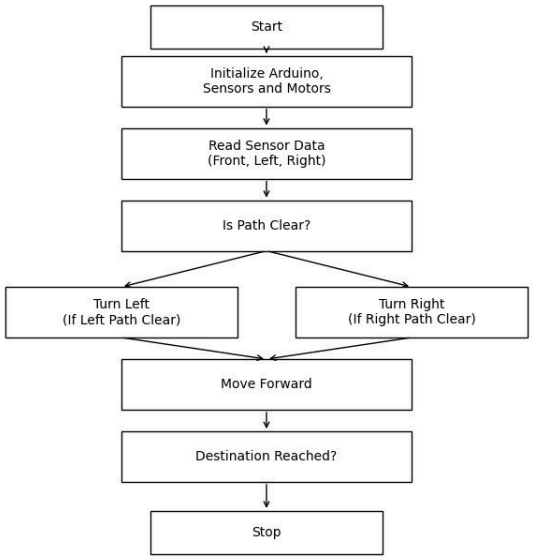
[11] Online Arduino Community Forum, discussion threads on ultrasonic sensor interfacing and motor control.

[12] https://circuitdigest.com/microcontroller-projects/human-following-robot-using-arduino-and-ultrasonicsensor

Appendix

APPENDIX A :

The flowchart represents the working algorithm of the maze-solving robot. The robot starts by initializing the Arduino, sensors, and motors. It continuously reads sensor data to detect obstacles and checks for a clear path. Based on sensor input, the robot decides whether to move forward, turn left, or turn right. This process repeats until the destination is reached, after which the robot stops.



APPENDIX B:

The diagram or photograph illustrates the path followed by the maze-solving robot during testing, showing how it navigates straight paths, junctions, and turns by making decisions based on sensor data while choosing path.