INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 1, January 2026 | ISSN: 2456-4184 | IJNRD.ORG

MULTI MODE MAZE SOLVER ROBOT USING ARDUINO UNO

[1]G. L. Saranya, [2]Inumula Hari Charan, [3]Ippili Vedha Sri, [4]Jarajapu Deepak

[1]Assistant Professor, [2]Student, [3]Student, [4]Student
[1]Department of Electronics and Communication Engineering,
[1]Lendi Institute of Engineering and Technology (Autonomous), Vizianagaram, India

Abstract : Autonomous mobile robots are increasingly used in industrial automation and intelligent navigation systems. This project presents the design and implementation of a dual-mode autonomous robot capable of performing line following and maze solving operations using Arduino. The robot utilizes infrared sensors for path detection and ultrasonic sensors for obstacle-based maze navigation. Path optimization techniques are implemented to improve accuracy, reduce traversal time, and enhance reliability. Experimental results demonstrate effective navigation in both structured line paths and unstructured maze environments.

IndexTerms **- Line Following Robot, Maze Solving Robot, Arduino UNO, IR Sensors, Ultrasonic Sensors**

I. INTRODUCTION

Autonomous robots are increasingly adopted in automation, logistics, and intelligent systems due to their ability to operate with minimal human intervention and high operational precision. Line-following robots are widely used in assembly lines and material handling systems, where consistent navigation along predefined paths is required. Maze-solving robots, on the other hand, are essential in autonomous navigation research for studying obstacle avoidance, real-time decision-making, and path optimization in dynamic environments.

Integrating both navigation strategies into a single robotic platform enhances system versatility and practical usability. This work focuses on the development of a dual-mode autonomous robot capable of switching between line-following and maze-solving operations based on sensor feedback and predefined logic. Infrared sensors provide surface detection for path tracking, while ultrasonic sensors measure distances for obstacle detection and wall-following. The Arduino UNO processes sensor data in real time and generates motor control signals to ensure smooth motion, reliable navigation, and low computational overhead suitable for embedded platforms.

II. RELATED WORK.

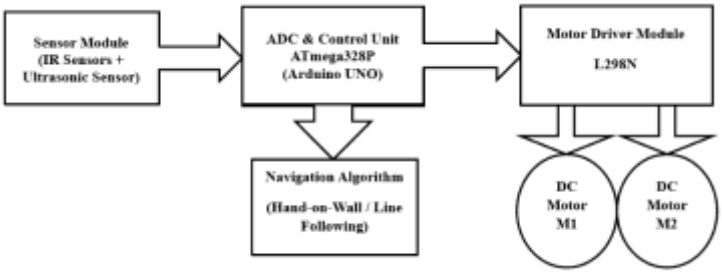
Several studies have explored line following robots using infrared sensors and simple control logic. Ultrasonic sensors are widely used for obstacle detection due to their accuracy and low cost. Maze solving techniques such as hand-on-wall (Wall-Following) algorithms are commonly implemented in embedded robotic systems. However, limited work has focused on combining these functionalities into a single robot.

III. SYSTEM OVERVIEW.

The proposed system is built around an Arduino Uno microcontroller, which serves as the central processing and control unit of the robotic platform. It is interfaced with multiple IR sensors for precise line detection, allowing the robot to distinguish the line path from the surrounding surface. An ultrasonic sensor is also integrated to measure distances and detect obstacles, enabling the robot to sense and respond to its environment effectively.

The system uses an L298N motor driver to interface the low-power signals from the microcontroller with the higher-power requirements of the DC motors, which drive the robot's wheels and enable forward, backward, and turning movements. A regulated power supply ensures stable voltage and current delivery to all components, improving overall reliability and performance.

The Arduino continuously monitors and processes data from the IR and ultrasonic sensors in real time. Based on the sensor inputs and the selected navigation mode, it executes control logic to generate appropriate motor signals. In line-following mode, the robot tracks a predefined path by continuously adjusting its direction using IR sensor feedback. In maze-solving mode, it navigates complex environments by detecting obstacles, estimating distances with the ultrasonic sensor, and making dynamic movement decisions. This dual-mode operation allows the robot to perform flexible and intelligent navigation, making it suitable for a wide range of autonomous applications.



block diagram of the proposed robotic system

The block diagram represents the overall structure and operation of the robotic system, which consists of a sensor module, an Arduino Uno–based microcontroller, and a motor driver connected to DC motors. The sensor module detects line position and obstacles and sends analog signals to the microcontroller through the ADC.

The ATmega328P microcontroller processes the converted digital sensor data and generates control signals based on the selected navigation mode. These signals are sent to the L298N motor driver, which supplies the required power to the DC motors (M1 and M2) to control the robot's movement. This coordination between sensors, controller, and actuators enables reliable autonomous navigation.

IV. HARDWARE IMPLEMENTATION.

4.1 Robot Chassis and Components

The robot chassis holds all electronic components, including the microcontroller, sensors, motors, and power supply. It provides a stable and compact structure that ensures proper alignment, reduces vibrations, and protects components from damage, thereby improving system reliability and durability.

front view of the robot hardware

top view of the robot hardware

In addition to structural support, the chassis is designed for easy assembly and maintenance. The organized layout allows quick access to components and ensures proper weight distribution and balance, which contributes to smooth and stable movement. Overall, the chassis functions as both a protective enclosure and a key element in achieving reliable autonomous operation.

4.2 Arduino Uno

The Arduino Uno serves as the central processing unit of the robotic system. It is based on the ATmega328P microcontroller, which handles sensor inputs, data processing, and control of actuators such as motors. The board provides multiple digital and analog input/output pins, offering flexibility for interfacing with various sensors and devices. Its real-time processing capability and ease of programming using the Arduino IDE support rapid prototyping and development. Additionally, strong community support, extensive libraries, and clear documentation make it suitable for both beginner and advanced robotics projects.

4.3 Infrared Sensors

Infrared sensors enable the line-following functionality of the robot. Each sensor uses an IR transmitter and receiver to detect reflected light from the surface, allowing identification of black lines on lighter backgrounds. Multiple sensors are mounted across the front of the robot to improve tracking accuracy, especially on curves and sharp turns. The sensors continuously send feedback to the Arduino, which adjusts motor control in real time to keep the robot on the correct path.

4.4 Ultrasonic Sensors

The HC-SR04 ultrasonic sensor is used for obstacle detection and object tracking in the robot. It operates on the time-of-flight principle by emitting ultrasonic pulses and measuring the time taken for the echoes to return from nearby objects. This time is converted into distance, allowing the microcontroller to accurately determine obstacle proximity. The sensor enables safe navigation, collision avoidance, and intelligent decision-making in tasks such as maze solving and object following. Its reliability and ease of integration make it suitable for autonomous navigation applications.

4.5 Motor Driver and DC Motors

The L298N motor driver module interfaces the low-power control signals from the Arduino with the higherpower DC motors. It enables bidirectional motor control, allowing forward, backward, and turning movements. Using Pulse Width Modulation (PWM), the driver precisely controls motor speed. The DC motors provide mechanical movement, and when combined with differential steering, allow smooth turns, accurate path following, and stable motion on curved or uneven surfaces. Together, the motor driver and motors form the actuator system that converts the robot's decisions into physical action.

V. LINE FOLLOWING ALGORITHM.

5.1 Line Follower Code

```
#define enA 10//Enable1 L298 Pin enA
#define in1 9 //Motor1  L298 Pin in1
#define in2 8 //Motor1  L298 Pin in1
#define in3 7 //Motor2  L298 Pin in1
#define in4 6 //Motor2  L298 Pin in1
#define enB 5 //Enable2 L298 Pin enB
#define R_S A0 //ir sensor Right
#define L_S A1 //ir sensor Left
void setup(){ // put your setup code here, to run once
pinMode(R_S, INPUT); // declare if sensor as input
pinMode(L_S, INPUT); // declare ir sensor as input
pinMode(enA, OUTPUT); // declare as output for L298 Pin enA
pinMode(in1, OUTPUT); // declare as output for L298 Pin in1
pinMode(in2, OUTPUT); // declare as output for L298 Pin in2
pinMode(in3, OUTPUT); // declare as output for L298 Pin in3
pinMode(in4, OUTPUT); // declare as output for L298 Pin in4
pinMode(enB, OUTPUT); // declare as output for L298 Pin enB
analogWrite(enA, 255); // Write The Duty Cycle 0 to 255 Enable Pin A for Motor1 Speed
analogWrite(enB, 255); // Write The Duty Cycle 0 to 255 Enable Pin B for Motor2 Speed
delay(1000);
}
void loop(){
if((digitalRead(R_S) == 0)&&(digitalRead(L_S) == 0)){forward();}   //if Right Sensor and Left Sensor are at White color then it will call forward function
if((digitalRead(R_S) == 1)&&(digitalRead(L_S) == 0)){turnRight();} //if Right Sensor is Black and Left Sensor is White then it will call turn Right function
if((digitalRead(R_S) == 0)&&(digitalRead(L_S) == 1)){turnLeft();}  //if Right Sensor is White and Left Sensor is Black then it will call turn Left function
if((digitalRead(R_S) == 1)&&(digitalRead(L_S) == 1)){Stop();} //if Right Sensor and Left Sensor are at Black color then it will call Stop function
}
void forward(){  //forward
digitalWrite(in1, HIGH); //Right Motor forward Pin
digitalWrite(in2, LOW);  //Right Motor backword Pin
digitalWrite(in3, LOW);  //Left Motor backword Pin
digitalWrite(in4, HIGH); //Left Motor forward Pin
}
void turnRight(){ //turnRight
digitalWrite(in1, LOW);  //Right Motor forward Pin
digitalWrite(in2, HIGH); //Right Motor backword Pin
digitalWrite(in3, LOW);  //Left Motor backword Pin
digitalWrite(in4, HIGH); //Left Motor forward Pin
}
void turnLeft(){ //turnLeft
digitalWrite(in1, HIGH); //Right Motor forward Pin
digitalWrite(in2, LOW);  //Right Motor backword Pin
digitalWrite(in3, HIGH); //Left Motor backword Pin
```

```
digitalWrite(in4, LOW);  //Left Motor forward Pin
}
void Stop(){ //stop
digitalWrite(in1, LOW); //Right Motor forward Pin
digitalWrite(in2, LOW); //Right Motor backword Pin
digitalWrite(in3, LOW); //Left Motor backword Pin
digitalWrite(in4, LOW); //Left Motor forward Pin
}
```

5.2 Algorithm Description

The robot hardware interfaces with the Arduino Uno through digital and analog pins. The L298N motor driver is connected to six digital pins, where enA (pin 10) and enB (pin 5) control motor speed using PWM, and pins in1–in4 control the direction of the right and left DC motors. Infrared sensors connected to analog pins A0 (right) and A1 (left) detect line contrast on the surface. In the setup phase, motor pins are configured as outputs and sensor pins as inputs, with both motors set to maximum speed using PWM. The loop function continuously reads the IR sensor values to control robot movement. When both sensors detect white, the robot moves forward; when one sensor detects black, it turns in the corresponding direction; and when both sensors detect black, the robot stops. Movement is implemented through dedicated functions—forward(), turnRight(), turnLeft(), and Stop()—which control the motor driver pins to achieve smooth and accurate line-following using differential motor control.

VI. MAZE SOLVING ALGORITHM.

6.1 Maze Solver Code

```
const int trigPinFront = A5;
const int echoPinFront = A4;
const int trigPinRight = A3;
const int echoPinRight = A2;
const int trigPinLeft = A1;
const int echoPinLeft = A0;

const int in1 = 9;
const int in2 = 8;
const int in3 = 7;
const int in4 = 6;
const int enA = 10;
const int enB = 5;

#define PWM 225  // Reduced speed for better control
#define DIS 30   // Increased detection distance
#define TURN_DELAY 350  // Standard turn duration

void setup() {
  Serial.begin(9600);

  // Sensor pins
  pinMode(trigPinFront, OUTPUT);
  pinMode(echoPinFront, INPUT);
  pinMode(trigPinRight, OUTPUT);
  pinMode(echoPinRight, INPUT);
  pinMode(trigPinLeft, OUTPUT);
  pinMode(echoPinLeft, INPUT);

  // Motor pins
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
}



void loop() {
  // Get current distances
```

```
  int front = FrontSensor();
  int right = RightSensor();
  int left = LeftSensor();

  // Debugging output
  Serial.print("F:"); Serial.print(front);
  Serial.print("cm R:"); Serial.print(right);
  Serial.print("cm L:"); Serial.print(left);
  Serial.println("cm");

  // Improved decision logic
  if (front < DIS) {
    if (right > DIS && left > DIS) {
      // Only front obstacle - turn right by default
      turn_right();
      delay(TURN_DELAY);
    }
    else if (right < DIS && left > DIS) {
      // Front and right blocked - strong left turn
      turn_left();
      delay(TURN_DELAY * 1.5);
    }
    else if (right > DIS && left < DIS) {
      // Front and left blocked - strong right turn
      turn_right();
      delay(TURN_DELAY * 1.5);
    }
    else {
      // All sides blocked - reverse and turn right
      reverse();
      delay(500);
      turn_right();
      delay(TURN_DELAY);
    }
  }
  else if (left < DIS*0.7) {  // More sensitive left detection
    smooth_left();
  }
  else if (right < DIS*0.7) {  // More sensitive right detection
    smooth_right();
  }
  else {
    forward();
  }
}

//----- Improved Sensor Functions -----//
long FrontSensor() {
  digitalWrite(trigPinFront, LOW);
  delayMicroseconds(4);
  digitalWrite(trigPinFront, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPinFront, LOW);
  long dur = pulseIn(echoPinFront, HIGH);
  return dur / 58;
}

long RightSensor() {
  digitalWrite(trigPinRight, LOW);
  delayMicroseconds(4);
  digitalWrite(trigPinRight, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPinRight, LOW);
  long dur = pulseIn(echoPinRight, HIGH);
  return dur / 58;
}
```

```
long LeftSensor() {
  digitalWrite(trigPinLeft, LOW);
  delayMicroseconds(4);
  digitalWrite(trigPinLeft, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPinLeft, LOW);
  long dur = pulseIn(echoPinLeft, HIGH);
  return dur / 58;
}

//----- Enhanced Movement Functions -----//
void forward() {
  digitalWrite(in1, HIGH); digitalWrite(in2, LOW);
  digitalWrite(in3, LOW); digitalWrite(in4, HIGH);
  analogWrite(enA, PWM); analogWrite(enB, PWM);
}

void smooth_left() {
  // Gradual left turn while moving forward
  digitalWrite(in1, HIGH); digitalWrite(in2, LOW);
  digitalWrite(in3, LOW); digitalWrite(in4, HIGH);
  analogWrite(enA, PWM*0.7);  // Left wheel slower
  analogWrite(enB, PWM);
}

void smooth_right() {
  // Gradual right turn while moving forward
  digitalWrite(in1, HIGH); digitalWrite(in2, LOW);
  digitalWrite(in3, LOW); digitalWrite(in4, HIGH);
  analogWrite(enA, PWM);
  analogWrite(enB, PWM*0.7);  // Right wheel slower
}

void turn_left() {
  // Pivot left turn
  digitalWrite(in1, HIGH); digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH); digitalWrite(in4, LOW);
  analogWrite(enA, PWM); analogWrite(enB, PWM);
}

void turn_right() {
  // Pivot right turn
  digitalWrite(in1, LOW); digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW); digitalWrite(in4, HIGH);
  analogWrite(enA, PWM); analogWrite(enB, PWM);
}

void reverse() {
  digitalWrite(in1, LOW); digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH); digitalWrite(in4, LOW);
  analogWrite(enA, PWM); analogWrite(enB, PWM);
}
```

6.2 Navigation Algorithm Description

The Arduino code implements an autonomous obstacle-avoiding robot using three ultrasonic sensors to detect obstacles in the front, left, and right directions. Distance measurements are obtained in real time using the time-of-flight principle and processed by the Arduino Uno to make navigation decisions. The L298N motor driver controls two DC motors with PWM-based speed regulation for smooth movement. When a front obstacle is detected, the robot selects the safest turning direction based on side sensor readings, reverses if all paths are blocked, and then pivots to avoid collisions. If side obstacles are detected while the front is clear, smooth directional adjustments are applied. In the absence of obstacles, the robot moves forward continuously. Dedicated motion functions such as forward, smooth turns, pivot turns, and reverse ensure modular control and stable navigation in complex environments.

VII. RESULTS AND DISCUSSION

The multi-mode maze-solving robot was tested on a structured indoor track consisting of a black line on a white surface with curves, T-junctions, and enclosed maze walls. The setup evaluated line-following accuracy, junction decision-making, and real-time obstacle avoidance. Using IR sensors and PWM-based motor control, the robot maintained accurate line alignment and executed smooth turns through differential speed adjustment.

At junctions, the Hand-on-Wall navigation algorithm enabled autonomous and reliable path selection, while ultrasonic sensors ensured continuous obstacle detection and safe wall clearance. The integration of line-following and obstacle-avoidance algorithms allowed stable navigation even in narrow corridors.

Experimental results demonstrated consistent line tracking and successful junction navigation. Obstacles were reliably detected within approximately 30 cm, allowing timely avoidance maneuvers. Reduced speed in curves and confined spaces improved stability and control. Overall, the results confirm that the proposed robot achieves reliable autonomous navigation with effective obstacle avoidance, showing strong potential for applications in warehouse automation, autonomous delivery, and robotics education.

REFERENCES

[1] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ, USA: Princeton University Press, 2008.
[2] M. Borenstein, H. R. Everett, and L. Feng, "Where am I? Sensors and methods for mobile robot positioning," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 16–35, Mar. 1997.
[3] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. Cambridge, MA, USA: MIT Press, 2011.
[4] N. M. Z. Hashim et al., "Design and development of line following robot using microcontroller," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 10, Oct. 2013.
[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.