Mini Project Report on

Autonomous Maze Solving Robot

SUBMITTED BY

JOBANPUTRA DARSH PARAG (24)
MOURYA AARYA ANAND (33)
NANDAWANA VATSAL MANILAL (34)

UNDER GUIDANCE OF

PROF. SHEETAL MAHADIK

DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCEENGINEERING

SHREE L. R. TIWARI COLLEGE OF ENGINEERING,

KANAKIA PARK, MIRA ROAD (E) – 401 107

UNIVERSITY OF MUMBAI

Academic Year 2022–2023

CERTIFICATE

This is to certify that the requirements for the Mini Project entitled "Autonomous Maze Solving Robot" have been successfully completed by following SE ECS students.

JOBANPUTRA DARSH PARAG (24)
MOURYA AARYA ANAND (33)
NANDAWANA VATSAL MANILAL (34)

in fulfillment of Mumbai University in the Department of Electronics and Computer Science Engineering, Shree L. R. Tiwari College of Engineering, Mira Road (E) – 401107 for Academic year 2022 - 2023.

Internal Examiner | **External Examiner**

H.O.D.
(Electronics Engineering)

PRINCIPAL
Shree L. R. Tiwari College of Engg.

ACKNOWLEGDEMENT

Special thanks to our Guide ***Prof. Sheetal Mahadik*** for assisting us to partially complete our Mini Project on ***“Autonomous Maze Solving Robot”.***

We'd like to express our gratitude to Prof. Sheetal Mahadik Madam, our Project Coordinator (Mini Project), for guiding us through the process to fulfil Mumbai University's Mini Project requirements.

Special thanks to Head of Department ***Prof. Manjiri M. Gogate*** madam for her constant motivation, without which this work would not have become successful.

We are also thankful to our Principal ***Dr. Deven Shah*** sir for his continuous encouragement throughout the process.

We would also like to thank our **Electronics and Computer Science Engineering Department** for providing us online/offline facility and labs, which helped us constantly in increasing our technical knowledge, and to write this report.

INDEX

ABSTRACT

The project aims to develop an efficient maze-solving robot based on the Wall following algorithm and collision avoidance concepts using Arduino. The project involves hardware and software development, as well as maze construction for performance testing. The maze-solving algorithm is widely used in real-world scenarios such as mining and military decision-making. To avoid trapping and falling into infinite loops, the proposed method addresses this issue. The maze-solving robot has applications in various fields, including intelligent traffic control, mining, greenhouses, and the military.

INTRODUCTION

Autonomous robots have become increasingly popular and are used in a wide range of applications, including manufacturing, agriculture, and healthcare, transportation, and space exploration, among others. These robots are equipped with advanced sensors, processors, and software that enable them to perform complex tasks and make decisions based on the data they receive from their environment.

In the case of maze-solving robots, the robot's sensors detect the maze walls, and the robot's processor analyses the data to determine the robot's position and orientation within the maze. The robot then uses algorithms to plan a path through the maze and navigate to the end point, avoiding dead-ends.

To build a maze-solving robot, engineers and designers need to consider various factors such as the robot's size, weight, power source, sensors, control systems, and programming. The robot's physical design should allow it to move quickly and smoothly through the maze, while its control systems and algorithms should be optimized for speed, accuracy, and efficiency.

Overall, an autonomous robot is a type of robot that can perform tasks without human intervention, using its own sensors, algorithms, and decision-making capabilities. A maze-solving robot is an example of an autonomous robot that is designed to navigate through a maze and reach the end point using sensors such as proximity sensors, light sensors, or line-following sensors, and algorithms that allow it to make decisions based on the data it receives. Maze-solving robots are often used in competitions, as they require a combination of mechanical engineering, programming, and An autonomous robot is a type of robot that can perform tasks without human intervention, using its own sensors, algorithms, and decision-making capabilities. A maze-solving robot is an example of an autonomous robot that is designed to navigate through a maze and reach the end point using sensors such as proximity sensors, light sensors, or line-following sensors, and algorithms that allow it to make decisions based on the data it receives. Maze-solving robots are often used in competitions, as they require a combination of mechanical engineering, programming, and sensor technology to build and operate successfully. autonomous robots such as maze-solving robots represent a significant advancement in robotics technology, and they have the potential to revolutionize the way we live and work by performing tasks that are dangerous, tedious, or impossible for humans.

PROBLEM STATEMENT

To design hardware for maze solving robot, construct a program with the combination of wall following algorithm and collision avoidance then implement it in the hardware of maze solving robot. At last, make a maze to verify the path planning of the robot.

Designing a maze-solving robot involves creating both hardware and software components. The hardware component consists of the physical robot that will navigate through the maze, while the software component involves developing the algorithm that willguide the robot in finding its way through the maze.

The first step is to design the hardware for the robot. This will involve selecting the appropriate motors, sensors, and other components that will be needed to build the robot. Some of the sensors that will be required include ultrasonic sensors, which will help the robot detect obstacles in its path while moving. The hardware is in place, the next step is to develop the software algorithm that will guide the robot through the maze. The algorithm will be based on a combination of wall-following and collision avoidance techniques. The wall-following algorithm will allow the robot to follow the walls of the maze, while the collision avoidance algorithm will help the robot detect and avoid obstacles in its path.

To implement the algorithm in the hardware, the software will need to be written in a programming language that is compatible with the robot's microcontroller. The algorithm will need to be tested and refined until it is able to successfully navigate the maze.

LITERATURE SURVEY

"An Autonomous Robot Navigation System for Unknown Maze- Like Environments" by P. M. Khedkar and P. R. Deshmukh. This paper presents an algorithm for autonomous maze-solving using a combination of sensors and path planning techniques.

"An FPGA-Based Autonomous Maze Solving Robot" by K. Wang, K. Zhou, and X. Gao. This paper presents a hardware implementation of a maze-solving robot using a field-programmable gate array (FPGA)and various sensors.

"Real-Time Maze Solving Algorithm for an Autonomous Robot"by A. M. Alazzam, A. Al-Hourani, and M. Al-Qadi. This paper proposes a real-time maze-solving algorithm for an autonomous robot that is capable of finding the shortest path through a maze.

"Design and Implementation of a Low-Cost Autonomous MazeSolving Robot" by M. D. Aron, R. Patel, and D. D. Doye. This paper describes the design and implementation of a low-cost autonomous maze-solving robot using an Arduino microcontroller and various sensors.

"Autonomous Maze Solving Robot Using Raspberry Pi" by R. B. Agarwal, S. A. Rajpoot, and M. S. Rathore. This paper presents the design and implementation of a maze-solving robot using a Raspberry Pi and various sensors.

NAME OF PAPER	NAME OF AUTHORS	PAPER DETAILS	POINTS TAKEN
https://www.researchgate.net/publication/228955195_Simple_real-time_obstacle_avoidance_algorithm_for_mobile_robots	IOAN SUSNEA • UNIVERSITATEA DUNAREA DE JOS GALATI VIOREL MÎNZU • UNIVERSITATEA DUNAREA DE JOS GALATI G. VASILIU • UNIVERSITATEA DUNAREA DE JOS GALATI	January 2009 Conference: The 8th WSEAS International Conference on (CIMMACS'09) At: Puerto de la Cruz, SPAIN Volume: ISBN: 978-960-474-144-1; Proceedings Of The 8th Wseas International Conference On (CIMMACS'09)	Collision avoidance methods referred
https://www.koreascience.or.kr/article/CFKO201114258941250.pdf	• GAN ZHEN YE • DAE-KI KANG	Div. of Computer and Information Engineering, Dongseo University	Maze Design, and path planning
https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-019-1396-2	• Seyyed Mohammad Hosseini Rostami • Arun Kumar Sangaiah • Jin Wang & • Xiaozhu Liu	Published: 18 March 2019 EURASIP Journal on Wireless Communications and Networking	More advances in collision avoidance
https://www.sciencedirect.com	• MUSFIQUR RAHMAN	Department of Electrical and Electronic Engineering (EEE) University of Liberal Arts Bangladesh (ULAB)	Block Diagram referred

GENERAL OVERVIEW OF PROJECT

Background Information, Essential Theories, Methods, and Algorithms: This section should provide a brief overview of the background and history of the Maze Solving Robot, including essential theories, methods, and algorithms used for maze-solving. It should also cover the key features of the robot, such as its sensors, processors, control systems, and programming languages. This section should provide a foundation for the rest of the paper and demonstrate the author's understanding of the relevant concepts.

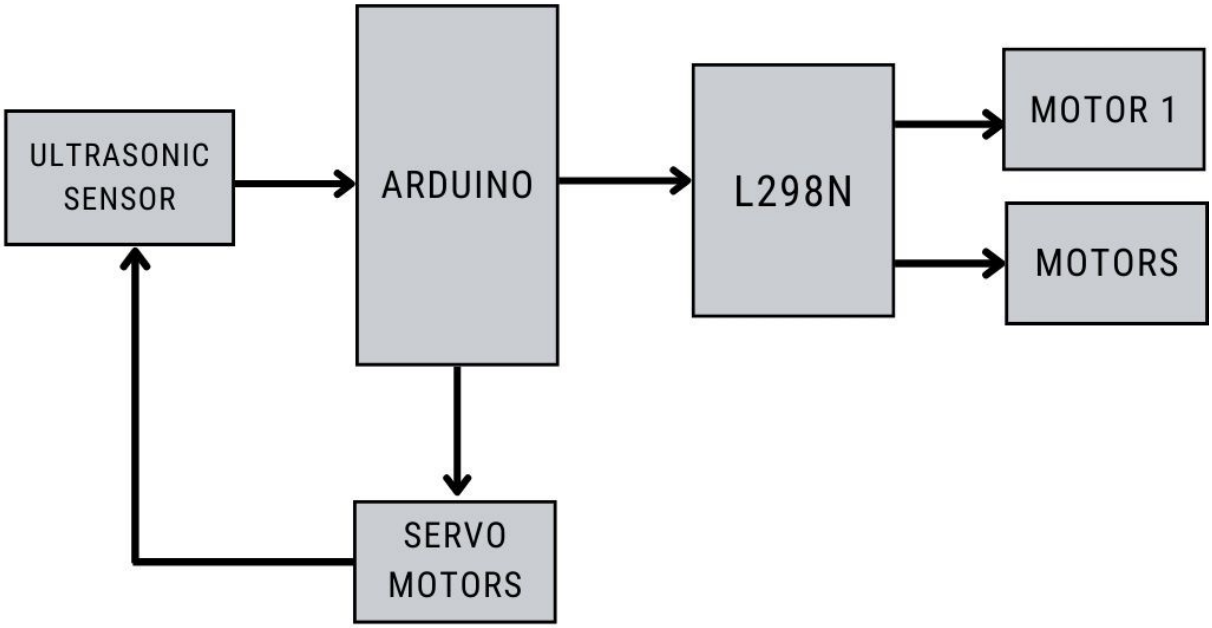
Details for Each Component Including Features, Specifications, and Circuits: This section should provide a detailed description of the different components that make up the Maze Solving Robot, including its features, specifications, and circuits. It should also describe the design and construction process for each component, as well as any challenges encountered during the development process. This section should demonstrate the author's technical skills and understanding of the practical aspects of building a maze-solving robot.

Methodology, Software Development, and Hardware Implementation: This section should provide a detailed description of the methodology used to develop the Maze Solving Robot, including the software development process and the hardware implementation. It should cover the programming languages used, the algorithms implemented, and the sensors and processors used to build the robot. The section should also describe the testing and validation process used to ensure the robot's functionality and performance. This section should demonstrate theauthor's ability to apply the theoretical concepts and technical skills described in the previous sections.

Result and Critical Discussion: This section should present the results obtained from the testing and validation process, including the robot's success rate in solving mazes and the time taken to complete the task. Itshould also provide a critical discussion of the results, including any limitations or challenges encountered during the development process and suggestions for future improvements. This section should demonstrate the author's ability to analyze and interpret data and draw meaningful conclusions from the research.

Conclusion and Further Development: The conclusion should summarize the main findings of the research and provide a clear answer to the research question or problem statement. It should also highlight the significance of the research and its potential impact on the field of robotics. The section on further development should outline future research directions and improvements that can be made to the Maze Solving Robot, such as enhancing its accuracy and speed, improving its sensors and processors, or expanding its range of capabilities. This section should demonstrate the author's ability to think critically and creatively about the research topic.

BLOCK DIAGRAM



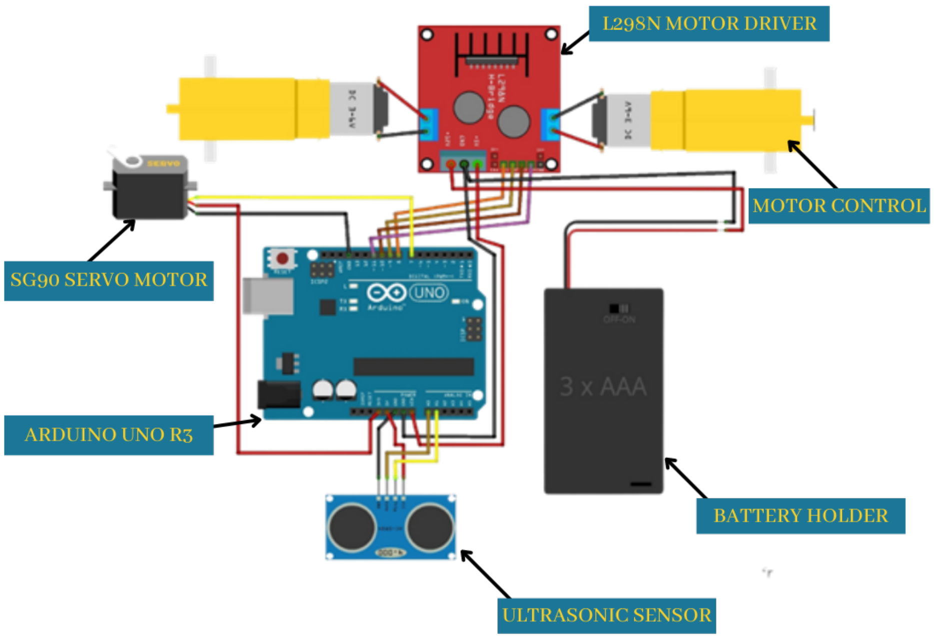
The block diagram for a maze-solving robot using a sensor, Arduino, L298N motor driver, two motors, and a servo motor would include several components and connections. At the heart of the robot is the Arduino board, which serves as the brain of the robot. The sensor, which could be an ultrasonic or infrared sensor, is connected to the Arduino board and detects obstacles and walls within the maze. The Arduino board then processes the data from the sensor and sends commands to the L298N motor driver.

The L298N motor driver controls the two motors that power the robot's movement. The motor driver receives commands from the Arduino board and translates them into signals that drive the motors forward, backward, or turn them in different directions. The motors themselves would be connected to the motor driver, and their movements would determine the direction and speed of the robot's movement through the maze. The block diagram for a maze-solving robot using a sensor, Arduino, L298N motor driver, two motors, and a servo motor would include several components and connections. At the heart of the robot is the Arduino board, which serves as the brain of the robot. The sensor,which could be an ultrasonic or infrared sensor, is connected to the Arduino board and detects obstacles and walls within the maze. The Arduino board then processes the data from the sensor and sends commands to the L298N motor driver.

The L298N motor driver controls the two motors that power the robot's movement. The motor driver receives commands from the Arduino board and translates them into signals that drive the motors forward, backward, or turn them in different directions. The motors themselves would be connected to the motor driver, and their movements would determine the direction and speed of the robot's movement through the maze.

HARDWARE SCHEME

CIRCUIT DIAGRAM :



SCHEMATICS FOR AUTONOMOUS ROBOT

Sensor:

The robot has a sensor that is, Ultrasonic sensor. This sensors help the robot detect the walls and obstacles in the maze.

Arduino board:

The board features 14 digital input/output pins, 6 analog input pins, and a USB connection for programming and communication with a computer. The Arduino Uno R3 can be programmed using the Arduino Integrated Development Environment (IDE) and supports a wide variety of programming languages, including C and C++.

Motor Control:

The robot has motors that are controlled by the microcontroller. The motor control allows the robot to move forward, backward, and turnleft or right.

Power Supply:

The robot needs a power supply such as batteries or a power adapter to operate.

SG90 servo motor:

The SG90 servo motor can be used to rotate the sensors to cover a wider area or change their direction of view.

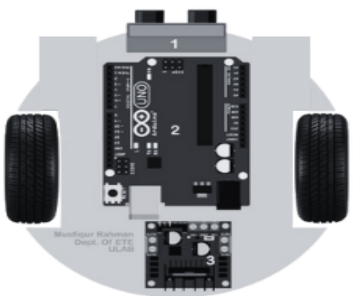
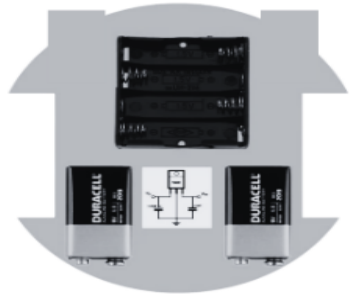
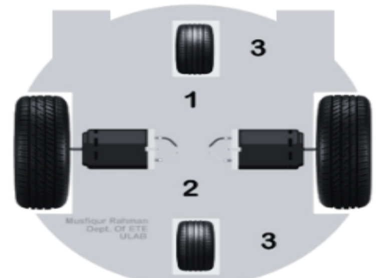
L298N Motor Driver:

The motor driver receives commands from the Arduino board and translates them into signals that drive the motors forward, backward, or turn them in different directions.

Chassis:

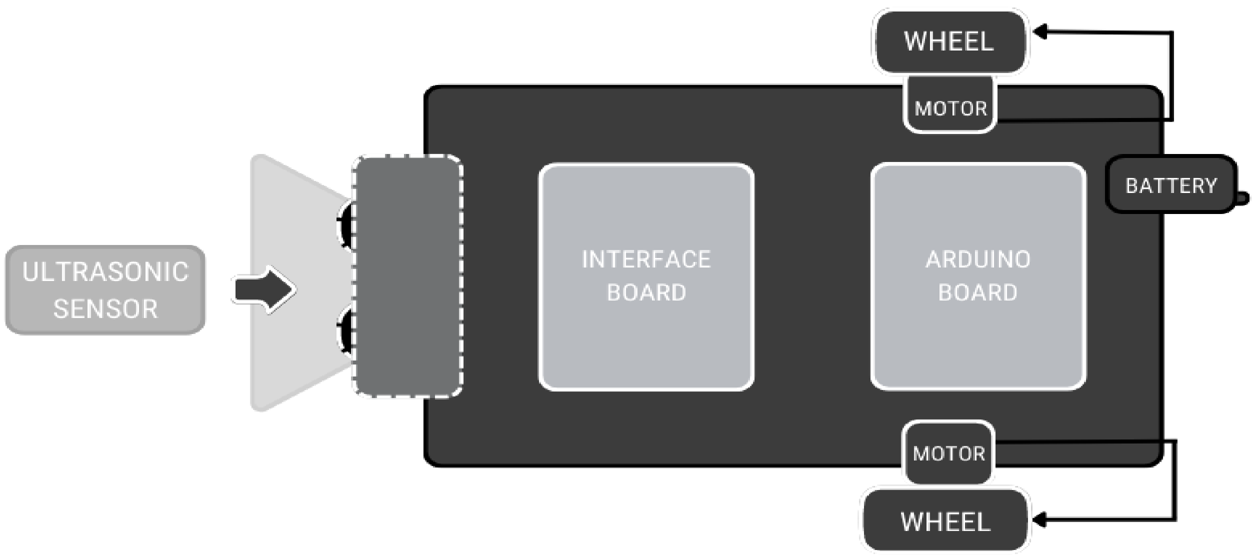
The robot has a chassis that houses all the components. The chassis is designed to be small and able to move so that it can navigate through the maze.

DESIGN &COMPONENTS

TOP PART	MIDDLE PART	BOTTOM PART
1. Ultrasonic Sonar sensors. 2. SG90 servo motor: 3.Arduino. 4. L298 Motor Controller	1. AA battery holder. 2. 2×9V batteries. 3. Voltage Regulator.	1. DC motor with Wheel. 2. DC motor with Wheel. 3. Caster wheels,
		

The top part of the robot contains the Ultrasonic Sonar sensors, SG90 servo motor, Arduino, and L298 Motor Controller. The Ultrasonic Sonar sensors are used to detect obstacles and measure distances, while the SG90 servo motor is used for controlling movement in robotic arms and other components. The Arduino is the microcontroller board that serves as the brain of the robot, while the L298 Motor Controller is used for precise motor control independently. The middle part of the robot consists of the power supply components, including an AA battery holder, 2x9V batteries, and a voltage regulator. These components are essential for providing power to the various electronic components in the robot.

Finally, the bottom part of the robot contains two DC motors with wheels and caster wheels. These motors are used for driving the robot forward and turning, while the caster wheels help to provide stability and balance



Based on the diagram provided. The Ultrasonic Sonar sensor is used for detecting obstacles and measuring distances, while the interface board serves as a connection point between the sensor and the Arduino board. The Arduino board is the brain of the robot and can be programmed to control the motors that drive the wheels. The wheels are connected to DC motors that provide the movement necessary for the robot to navigate. Finally, the battery serves as the power source for the robot, allowing it to function

HARDWARE DETAILS

Consists of several hardware components:

- Arduino
- Ultrasonic sonar sensors
- Motor driver
- DC gear motors X2
- Servo motor
- Batteries 6 volts
- SG90 Servo Motor
- caster wheel

Arduino: The Arduino board is a microcontroller-based development board that is used in many robotics projects. It consists of an ATMega microcontroller, digital and analog input/output pins, and various other components. The board can be programmed using the Arduino program

based on C/C++. The Arduino board can communicate with other components such as sensors and motor drivers through its input/output pins.

Ultrasonic sonar sensors: Ultrasonic sensors are commonly used in robotics projects to detect obstacles in the path of the robot. They emit high-frequency sound waves and then measure the time it takes for the sound waves

to bounce back off of an object. This allows the sensor to calculate the distance to the object. Ultrasonic sensors can be connected to the Arduino board through digital or analog input pins.

Motor driver: The motor driver is a circuit that is used to control the speed and direction of the robot's motors. In the maze-solving robot project, a motor driver such as the L298N can be used to control two DC motors independently. The motor driver receives commands from the Arduino board and then sends signals to

the motors that determine their speed ar

DC gear motors: The DC gear motors are used to power the robot's movement through the maze. They consist of a DC motor and a gearbox that is designed to reduce the speed of the motor and increase

its torque. The motors can be connected to the motor driver through screw terminals or other connectors.

SG90 Servo Motor (180° Rotation) The SG90 servo motor is a small, lightweight, and inexpensive servo motor that is widely used in hobbyist projects, robotics, and other applications that require precise

control of movement. It is a 18 meaningit can rotate at most 180 degrees in either direction. The SG90 servo motor is controlled using a pulse width modulation (PWM) signal

Batteries: Batteries are used to power the maze-solving robot. In this project, a 6-volt battery can be used to power the motors and the Arduino board. The battery can be connected to the Arduino board through a

voltage regulator to ensure a stable supply o

Caster: A caster wheel is a type of wheel commonly used in many different applications, including robotics, furniture, and material handling equipment. It is designed to rotate around a vertical axis

WIRING CONNECTION

ARDUINO TO ULTRASONIC SENSOR	
ARDUINO	**ULTRASONIC SENSOR**
5v	Vcc
GND	GND
Trig	A1
Echo	A0

ARDUINO TO MOTOR DRIVER	
ARDUINO	**MOTOR DRIVER**
pin 8	**In1**
pin 9	In2
pin 10	In3
pin 11	In1

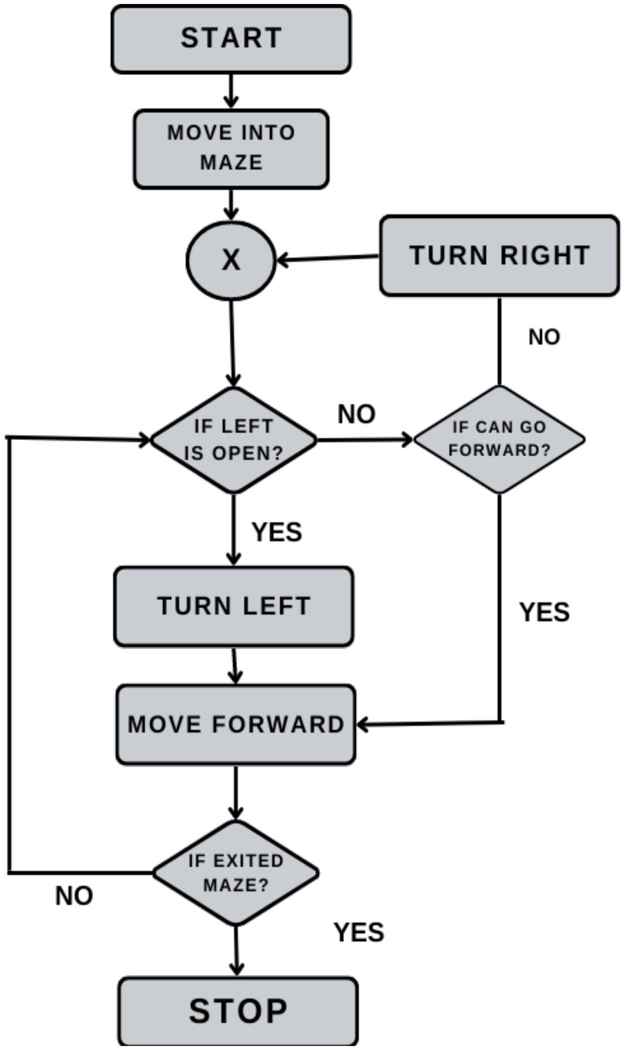
ARDUINO TO SERVO MOTOR	
ARDUINO	**SERVO MOTOR**
5v	**Vcc**
GND	GND
Pin 7	Data

SOFTWARE SCHEME

The Arduino integrated development environment (IDE) is a cross- platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version

2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program argued to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

FLOW CHART



1. Begin at the starting point of the maze.
2. Move forward until an obstacle is detected by the Ultrasonic Sonar sensor.
3. Use the sensor to detect the distance and location of the obstacle.
4. Determine the best course of action based on the obstacle's location and distance. For example, turn left or right to avoid the obstacle.
5. Move in the chosen direction.
6. Repeat steps 2-5 until the robot reaches the end of the maze.
7. Stop and indicate that the maze has been solved.

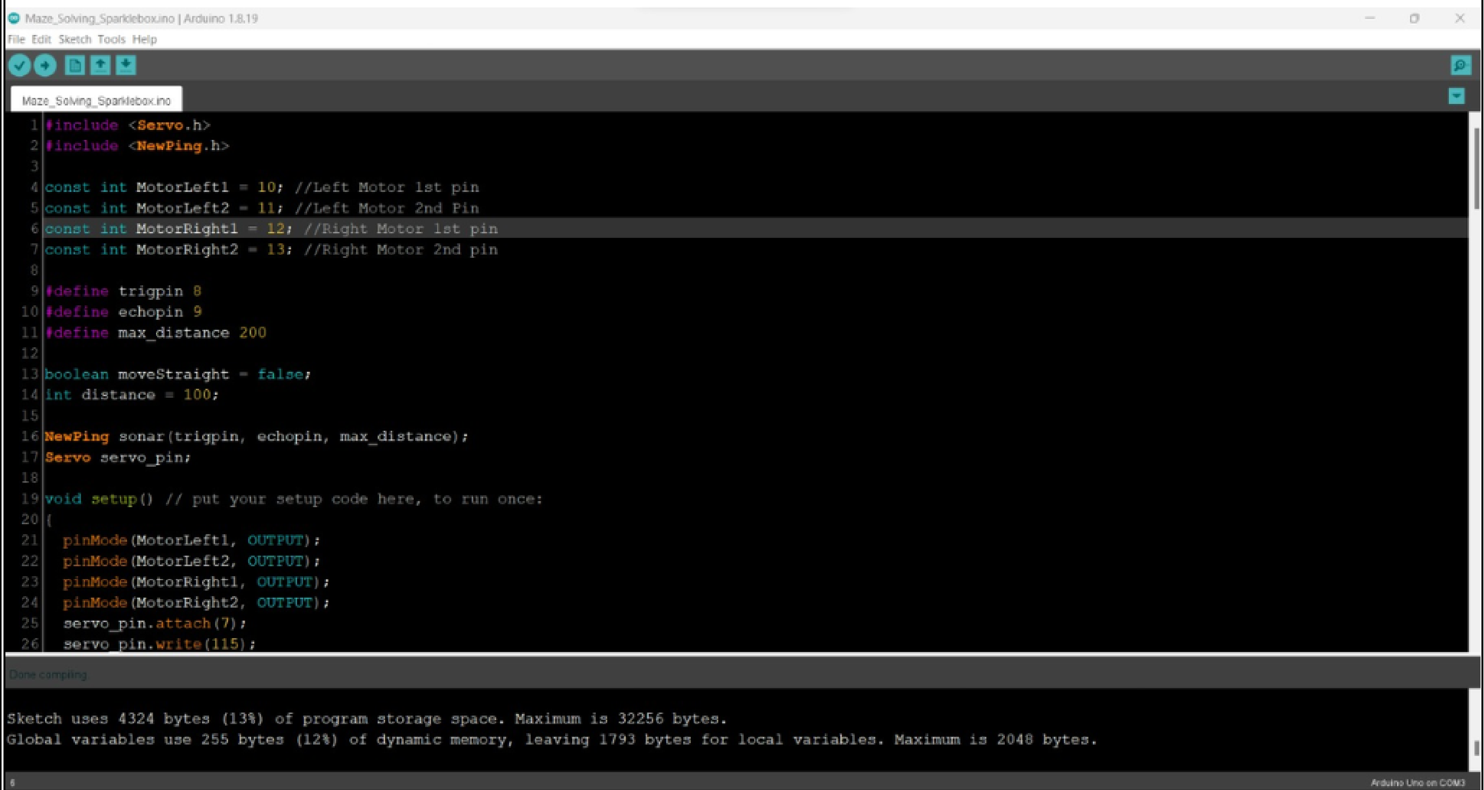
ALGORITHM

1. At START robot moves forward.
2. If robot detects obstacle in range x m in front by ultrasonic sensor.
3. Sensor turns right.
4. Sensor turns left.
5. .If wall detected on right, robot turns left by x deg, then movesforward.
6. If wall detected on left, robot turns right by x deg, then movesforward
7. If wall detected on front, right, left then ROBOT STOPS.

PROGRAMMING THE ROBOT: ARDUINO CODE

```
#include <Servo.h>        //Servo motor library. This is standard library
#include <NewPing.h>            //Ultrasonic sensor function library. You must install this library
//our L298N control pins
const int LeftMotorForward = 8; const int LeftMotorBackward = 9; const int
RightMotorForward = 10; const int RightMotorBackward = 11;
//sensor pins
#define trig_pin A1 //analog input 1 #define echo_pin A0 //analog input 2
#define maximum_distance 200 boolean goesForward = false; int distance = 100;
NewPing sonar(trig_pin, echo_pin, maximum_distance); //sensor function Servo servo_motor;
//our servo name
void setup(){
pinMode(RightMotorForward, OUTPUT); pinMode(LeftMotorForward, OUTPUT);
pinMode(LeftMotorBackward, OUTPUT); pinMode(RightMotorBackward, OUTPUT);
servo_motor.attach(3); //our servo pin
servo_motor.write(115);

delay(2000);
distance = readPing(); delay(100);
distance = readPing(); delay(100);
distance = readPing(); delay(100);
distance = readPing(); delay(100);
}
void loop(){
int distanceRight = 0; int distanceLeft = 0; delay(50);
if (distance <= 20){ moveStop(); delay(300); moveBackward(); delay(400); moveStop();
delay(300);
distanceRight = lookRight(); delay(300);
distanceLeft = lookLeft(); delay(300);
if (distance >= distanceLeft){ turnRight();
moveStop();
}
else{

turnLet(); moveStop();
}
}
else{ moveForward();
}
distance = readPing();
}
int lookRight(){ servo_motor.write(50); delay(500);
int distance = readPing(); delay(100); servo_motor.write(115); return distance;
}
```

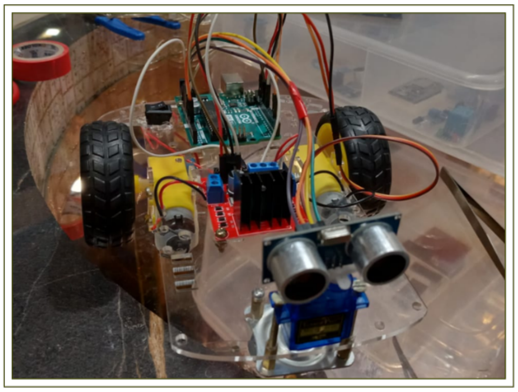
```
int lookLeft(){ servo_motor.write(170); delay(500);
int distance = readPing(); delay(100); servo_motor.write(115); return distance; delay(100);
}
int readPing(){ delay(70);
int cm = sonar.ping_cm(); if (cm==0){
cm=250;
}
return cm;
}
void moveStop(){
digitalWrite(RightMotorForward, LOW); digitalWrite(LeftMotorForward, LOW);
digitalWrite(RightMotorBackward, LOW); digitalWrite(LeftMotorBackward, LOW);
}
void moveForward(){ if(!goesForward){ goesForward=true;
digitalWrite(LeftMotorForward, HIGH); digitalWrite(RightMotorForward, HIGH);
digitalWrite(LeftMotorBackward, LOW); digitalWrite(RightMotorBackward, LOW);
}
}
void
moveBackward(){ goesForward=false;

digitalWrite(LeftMotorBackward, HIGH); digitalWrite(RightMotorBackward, HIGH);
digitalWrite(LeftMotorForward, LOW);
digitalWrite(RightMotorForward, LOW);
}
void turnRight(){
digitalWrite(LeftMotorForward, HIGH); digitalWrite(RightMotorBackward, HIGH);
digitalWrite(LeftMotorBackward, LOW); digitalWrite(RightMotorForward, LOW);
delay(500);
digitalWrite(LeftMotorForward, HIGH); digitalWrite(RightMotorForward, HIGH);
digitalWrite(LeftMotorBackward, LOW); digitalWrite(RightMotorBackward, LOW);
}
void turnLeft(){
digitalWrite(LeftMotorBackward, HIGH); digitalWrite(RightMotorForward, HIGH);
digitalWrite(LeftMotorForward, LOW); digitalWrite(RightMotorBackward, LOW); delay(500);
digitalWrite(LeftMotorForward, HIGH); digitalWrite(RightMotorForward, HIGH);
digitalWrite(LeftMotorBackward, LOW); digitalWrite(RightMotorBackward, LOW);
```



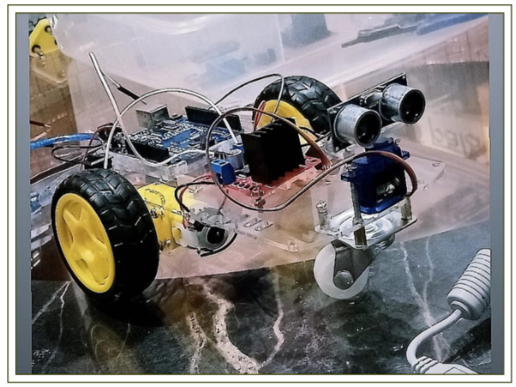

```
#include <Servo.h>
#include <NewPing.h>

const int MotorLeft1 = 10; //Left Motor 1st pin
const int MotorLeft2 = 11; //Left Motor 2nd Pin
const int MotorRight1 = 12; //Right Motor 1st pin
const int MotorRight2 = 13; //Right Motor 2nd pin

#define trigpin 8
#define echopin 9
#define max_distance 200

boolean moveStraight = false;
int distance = 100;

NewPing sonar(trigpin, echopin, max_distance);
Servo servo_pin;

void setup() // put your setup code here, to run once:
{
  pinMode(MotorLeft1, OUTPUT);
  pinMode(MotorLeft2, OUTPUT);
  pinMode(MotorRight1, OUTPUT);
  pinMode(MotorRight2, OUTPUT);
  servo_pin.attach(7);
  servo_pin.write(115);
```

```
Sketch uses 4324 bytes (13%) of program storage space. Maximum is 32256 bytes.
Global variables use 255 bytes (12%) of dynamic memory, leaving 1793 bytes for local variables. Maximum is 2048 bytes.
```

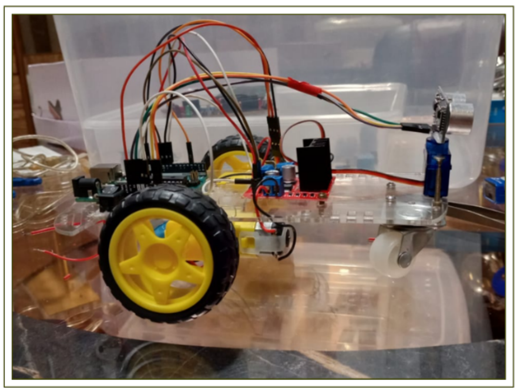
OUTPUT



front view



top view



side view

maze

The output of the project was an Arduino-based autonomous maze-solving robot that successfully solved the maze problem using the wall-following algorithm. The robot's hardware and software development were completed, and the maze construction and performance testing confirmed the robot's efficiency in maze-solving. The project provided valuable insights into the field of robotics and presented a challenging opportunity to apply programming and electronics skills. The outcome of the project demonstrated the importance of decision-making algorithms in robotics and showed the potential for creating more advanced and sophisticated autonomous robots capable of solving complex mazes and exploring unknown environments.

APPLICATIONS

- Applications of maze solving systems include intelligent traffic control that helps ambulances, fire fighters, or rescuing robots to find their path to their destination.
- **Education:** It can be used as a hands-on educational tool to teach programming, electronics, and robotics to students.
- **Exploration**: It can be used to explore and map unknown or dangerous environments, such as mines or disaster zones, where humans cannot safely access.
- **Industrial automation**: It can be used in industrial settings to transport materials or products through a maze-like environment.
- **Research**: It can be used to conduct research on robotics, artificial intelligence, and machine learning to develop more advanced and sophisticated autonomous robots.

CONCLUSION

- The Arduino-based autonomous maze-solving robot project successfully addressed the maze-solving problem using the wall following algorithm concepts.
- The project demonstrated the importance of decision-making algorithms in robotics, and the robot's hardware and software development were successfully accomplished.
- The maze construction was an essential part of the project, and the performance testing verified that the robot was efficient in solving the maze.
- Overall, this project provided valuable insights into the field of robotics and presented a fun and challenging opportunity to apply programming and electronics skills.
- Further development of this project could lead to the creation of more advanced and sophisticated autonomous robots capable of solving complex mazes and exploring unknown environments.

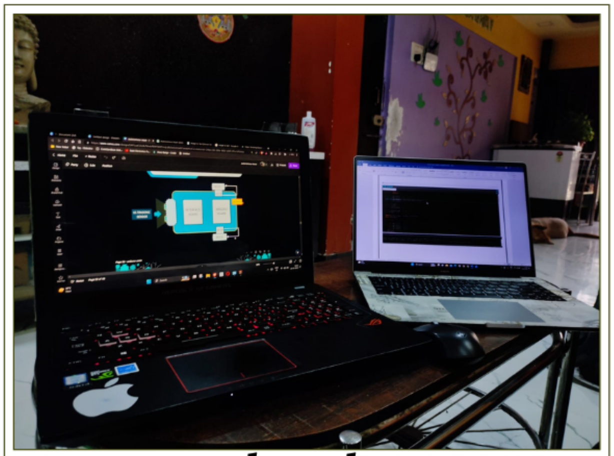
FUTURE SCOPE

- The future of maze-solving robots looks promising with several potential scopes, including:
- Improved Navigation: As technology continues to evolve, maze- solving robots will become even more efficient in navigating through complex environments. Future robots may have better sensors and algorithms that allow them to quickly and accurately navigate through mazes and other complex spaces.
- Multi-Robot Collaboration: In the future, multiple maze-solving robots may work together to solve complex tasks. This could involve collaborative exploration of unknown environments, coordinated search and rescue missions, or even joint construction or assembly tasks.
- Artificial Intelligence: The incorporation of artificial intelligence into maze-solving robots will make them more intelligent and capable. Future robots may be able to learn from their environment and make autonomous decisions based on their observations and experiences.
- Personal Assistance: Maze-solving robots could be used as personal assistants in the future. They could help people navigate through complex environments such as airports or shopping malls, or provide assistance to people with disabilities.
- Healthcare: In the healthcare industry, maze-solving robots could be used to transport medical equipment, medication, or even patients through complex hospital environments.

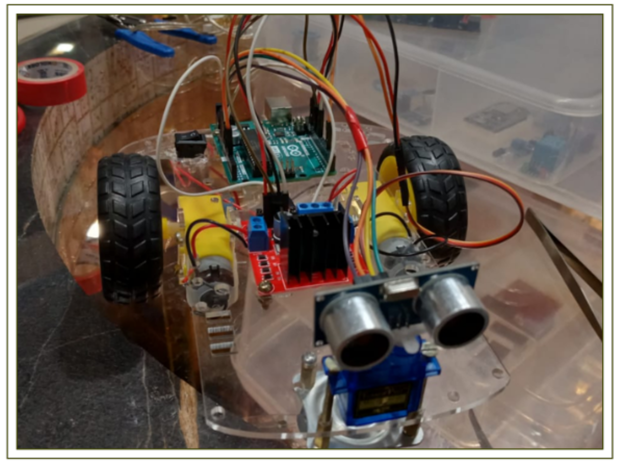
REFERENCES

- https://ieeexplore.ieee.org/document/8109037
- https://www.researchgate.net/publication/321121617_ Intelligent_Maze_Solving_Robot_Based_on_Image_P rocessing_and_Graph_Theory_Algorithms
- https://www.instructables.com/Maze-Solver-Robot-Using-Artificial-Intelligence-Wi/
- https://maker.pro/arduino/projects/how-to-build-an-arduino-based-maze-solving-robot
- https://www.engineersgarage.com/arduino-based-maze-solving-robot/

TEAM COLLABORATION



code and ppt



robot

TEAM