Maze solver robot

Overview:

The project aims to design and develop an autonomous maze solver robot that navigates through a maze using the left-hand rule wall-following algorithm. The robot will use sensors to detect walls and consciously follow the left boundary, allowing it to explore and eventually find the exit.

Th purpose of this project is to demonstrate the effectiveness of a simple, real-time decision making algorithm in autonomous navigation.

Objective:

The primary objective and goal of this project is to design and build a functional maze solving robot that can navigates through a maze using the left-hand rule wall-following algorithm. The robot should detect the walls, follow the left boundary consistently and find the exit.

Expected outcomes:

- A fully operational autonomous robot capable of solving a maze (using the left hand rule)
- Successful navigation through various maze configuration, demonstrating adaptability to different layout.
- Efficient use of sensors (ultrasonic) for wall detection and navigation
- Improved understanding of robotics, sensor integration, and algorithm problem solving in real world applications.
- A foundation for future enhancement, such as optimizing the model and algorithm, and incorporating other maze solving algorithm/techniques.

Electronics component requirements:

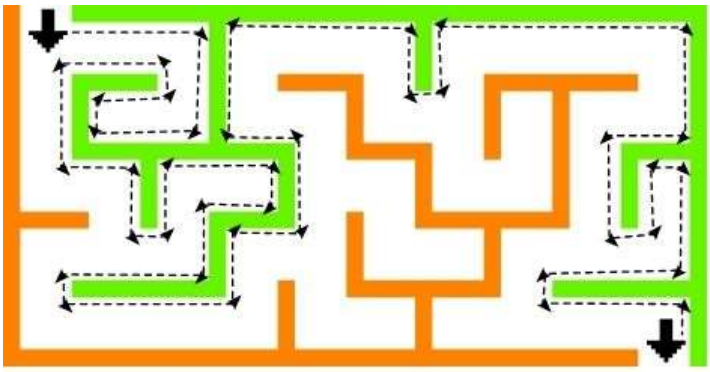
- Arduino Uno
- Ultrasonic ranging sensor (HC-SR04)
- Motor driver (L298)
- DC gear motors
- Voltage Regulator
- Connecting wires
- Batteries

Wall following algorithm

The wall follower algorithm contains two rules which are the right-hand rule and the left-hand rule. Regarding the rules, both rules can be applied to solve the maze in a fast and without any extra memory to solve maze. The rules are based on the simple logic of movements whereby either following along the left side or the right side of the enclosure wall. Following one side of the enclosure wall will form a pathway that leads to the exit.

The robot will take its direction by following either left or right wall. This algorithm also called, left hand-right hand rules. Whenever the robot reaches a junction, it will sense for the opening walls and select it direction giving the priority to the selected wall. By taking the walls as guide, this strategy is capable to make the robot reaches the finish point of the maze without actually solving it. But, this algorithm is not efficient method to solve a maze. Cause, the wall follower algorithm will fail to solve some maze construction, such as a maze with a closed loop region.

This algorithm works well in mazes with connected wall-mazes where a solution path exists as a continuous boundary. Below is an example of such a maze:

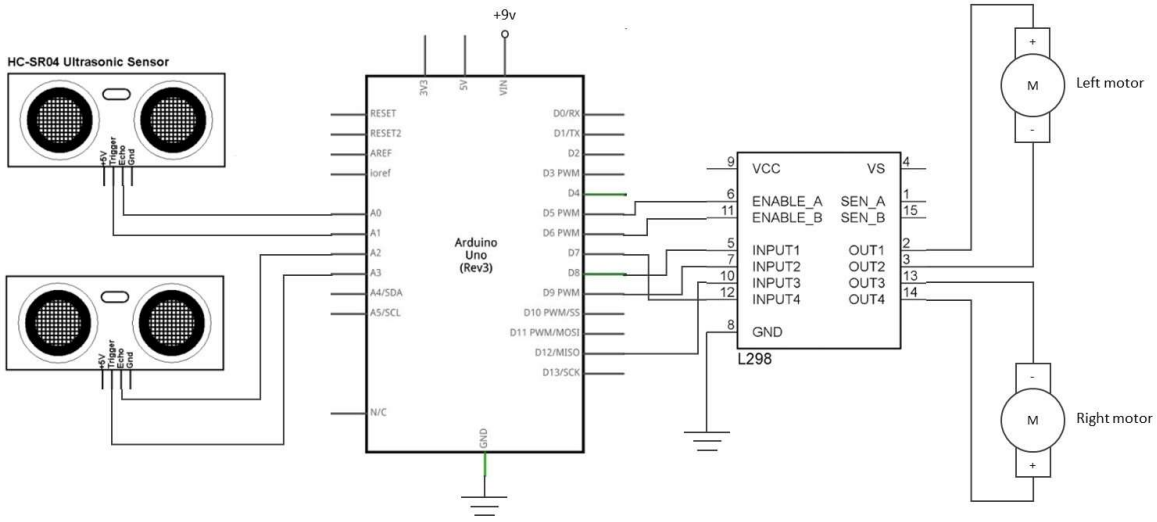


Path traced by left-hand rule

Truth Table for Left and Front Sensors:

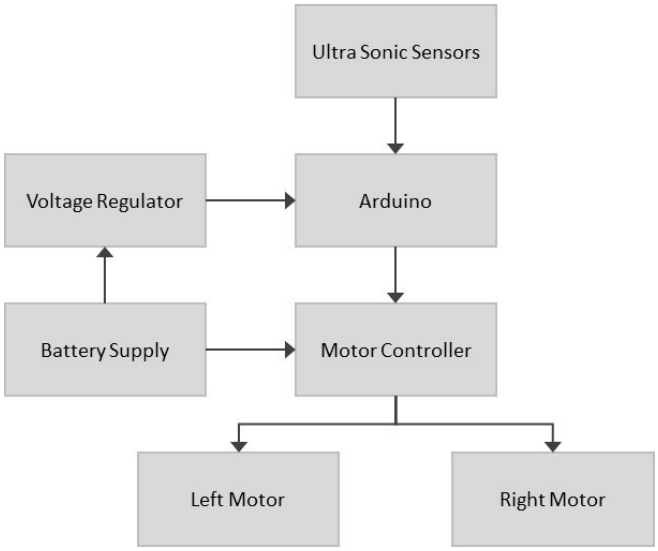
Left	Right	Action	
0	0	Turn Left	(No wall on the left or front, prioritize turning left)
0	1	Turn Left	(Wall in front, no wall on the left)
1	0	Move Forward	(Wall on the left, no wall in front)
1	1	Turn Right	(Wall on the left and front, turn right or U-turn)

Circuit Diagram



Circuit Diagram for Maze Solver Robot

Block Diagram

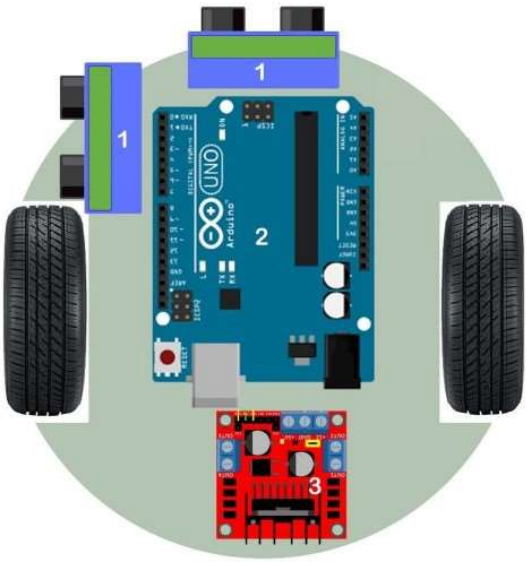


Block Diagram for Maze Solver Robot

Product model Design:

The conceptual design of the main body frame of the maze solving robot.

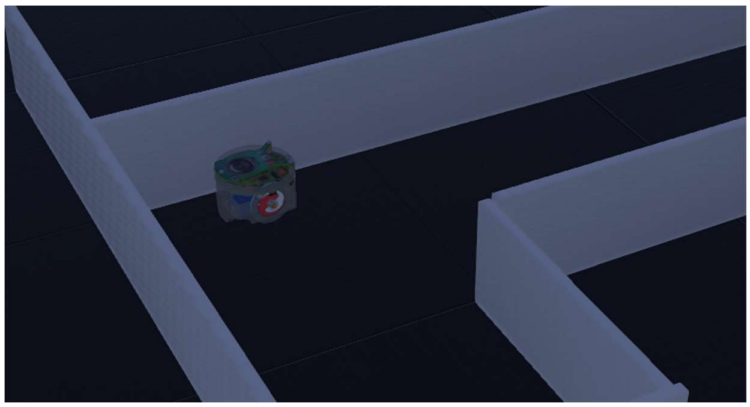
Components	
1	**Left and front Ultrasonic sensors**
2	**Arduino Uno**
3	**L298 Motor Controller**



Simulation tools:

In this project, Webots is used as a simulation tool for testing the design.

Webots is an open source and multi-platform desktop application used to simulate robots. It provides a complete development environment to model, program and simulate robots.



Robot in the simulation tool (Webots)

Controller code for robot:

The controller code is written in C++, using the logic described in Left-hand rule and the truth table of left and front sensor. The main loop for the logic is as shown:

```
// (Left-Hand Rule)

        if (frontWall) {
            // Front is blocked, turn right
            std::cout << "Front blocked: Turn right" << std::endl;
            leftSpeed = 6.28;
            rightSpeed = -6.28;

        } else if (!leftWall) {
            // Left is clear, turn left
            std::cout << "Left clear: Turn left" << std::endl;
            leftSpeed = -6.28;
            rightSpeed = 6.28;

        } else {
            // Move forward (follow the wall)
            std::cout << "Move forward" << std::endl;
            leftSpeed = 6.28;
            rightSpeed = 6.28;
        }
```

Additional Ideas / Future scope:

The left-hand rule maze-solving robot is a simple yet effective approach to navigating through mazes. However, there is significant potential for enhancing its capabilities and exploring new applications. Below are some additional ideas and future scope for improving and expanding the functionality of the robot:

- Enhanced Wall Detection:
  Adding more sensors, can improve the robot's ability to detect walls and make better navigation decisions in complex mazes.
- Maze Mapping:
  Equipping the robot with memory to store and update a map of the maze as it explores can help optimize its path and avoid revisiting the same areas.
- Using encoders or odometry to track the robot's position within the maze can improve navigation accuracy and efficiency.
- Incorporating advanced maze-solving algorithms, for better performance, efficiency and speed.
- Uses advanced technologies to make its own decisions, a smarter robot that doesn’t just follow simple rules.

Applications Beyond Mazes

- Search and Rescue: Adapt the robot for search and rescue operations in disaster-stricken areas where navigation through debris is required.
- Agriculture: Implement the robot for tasks like crop monitoring or weed detection in fields.