October 2018

"*Maze Solving Robot using Arduino*"

Submitted for CAL in B.Tech

Robotics and its Applications
(CSE 3011)

By

Group Members:-

Mahima Bhargava (15BCE1075)
Jayen Jhaveri (15BCE1111)
Pushpit Bagga (15BCE1136)
Tushar Pahuja (15BCE1252)
Osho Agyeya (15BCE1326)

Under the Guidance of
Prof. Rekha.D
(School of Computing Science Engineering)

Project Topic

To improvise upon existing algorithm, design hardware for maze solving robot, and then implement the Wall following and Shortest Path algorithms in the hardware of maze solving robot.

Abstract

Maze solving problem is a very old problem, but still now it is considered as an important field of robotics. This project is aimed at making an Arduino based intelligent maze solver robot. The main modules in the project are:

- Hardware development
- Software development
- Maze construction

For performance testing, the robot will implement to solve square maze.
Capability of finding the shortest path is also verified.

INTRODUCTION

A maze is a network of paths, typically from an entrance to exit. The concept of Maze approximately thousand years old and numerous mathematicians have made various algorithms to solve the maze.
Now a days, maze solving problem is an important field of robotics. It is based on one of the most important areas of robot and there are many types of maze solving robot using various type of algorithms.
In this project, we first implement the system design of Maze solving robot consist obstacle avoidance ultrasonic sensors and then sensors will detect the wall. To solve the maze, this robot will apply wall following algorithms such as left or right hand rule. We then implement our 'Shortest path algorithm' which traverses the shortest path in the maze after the maze wall inputs have been entered.

JOINTS AND DOF

Wheels
Type of Joint → Twisting joint (Type - T)
Degree of freedom = 1

ALGORITHMS

Wall Following Algorithm

Wall following algorithm is the most common algorithm for maze solving. The robot will take its direction by following either left or right wall. This algorithm is also called, left hand-right hand rules. Whenever the robot reaches a junction, it will sense for the opening walls and select its 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 a reliable 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.

Pseudocode for Wall following algorithm

Right wall following routine	Left wall following routine
if there is no wall at right,	if there is no wall at left
turn right	turn left
else	else
if there is no wall at straight	if there is no wall at straight
keep straight	keep straight
else	else
if there is no wall at left	if there is no wall at right
turn left	turn right
else	else
turn around	turn around

Shortest Path Algorithm

This algorithm finds the shortest path in the maze after it receives the maze information such as the positioning of the walls and the start and end points. The new algorithm shall solve the maze by considering the maze as a 2D matrix.

Pseudocode for the shortest path algorithm

1. Initially the destination cell position would be pushed into a queue with the *distance* value set as 0. Then the values removed from the front of the queue would be used to update the neighbours of the cell with *distance + 1*.

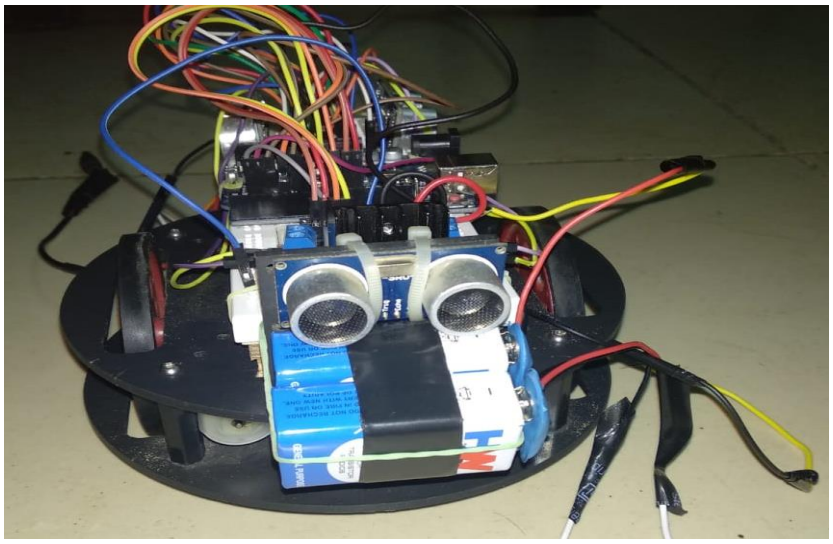
2. After the initialization of the initial matrix, we start from the source position and push the source position in the stack and perform the following steps while the stack is not empty:

- Check the values of the neighboring cells and find the *minimum value* from them.
- Check if the current cell value is 1 greater than the *minimum value*.
- If true then simply move to the cell with the minimum value else update the current cell with *1 + minimum* and push all the neighboring cells in the stack.

3. Keep checking the direction of the robot face and the direction to be next followed by the robot and keep updating the route array to get the final path from the source position to the target position.

HARDWARE DESCRIPTION

This project consists of several hardware components such as Arduino, Ultrasonic sonar sensors, Motor driver, Voltage Regulator, Batteries, etc.



1. Arduino UNO

The base of this project is Arduino. Entire code is stored in its microprocessor. It is an open source hardware development board. Arduino hardware consists of an open hardware design with an Atmel AVR processor. Arduino programming language (C) is used to program the processor. It is based on the ATmega328P microprocessor.

2. Ultrasonic ranging sensor (HC-SR04)

It is a device that can measure the distance to an object by using sound waves. It has two part, one is transmitter another is receiver. A pulse is triggered in the trigger pin, then a sound wave send to the object, then the reflected wave from the object is received by the receiver which is used to calculate the distance.

Left and right sensors are used to detect the left and right wall. The front sensor is used to map the front wall.

3. L298 Motor Driver

This dual bidirectional motor driver. It is used as an actuator to control the end effector of the robot i.e. the wheels.

4. Voltage Regulator

The voltage regulator is used to supply constant 5V to the ultrasonic sensors. A very common voltage regulator IC is 7805 which has been used in this project.

5. DC Motor

Two DC gear motors have been used to drive this robot. This motor acts as an actuator for the robot.

6. Wheels

Two wheels would be attached to the robot chassis which acts as the end effectors of the robot.

7. Robot Chassis

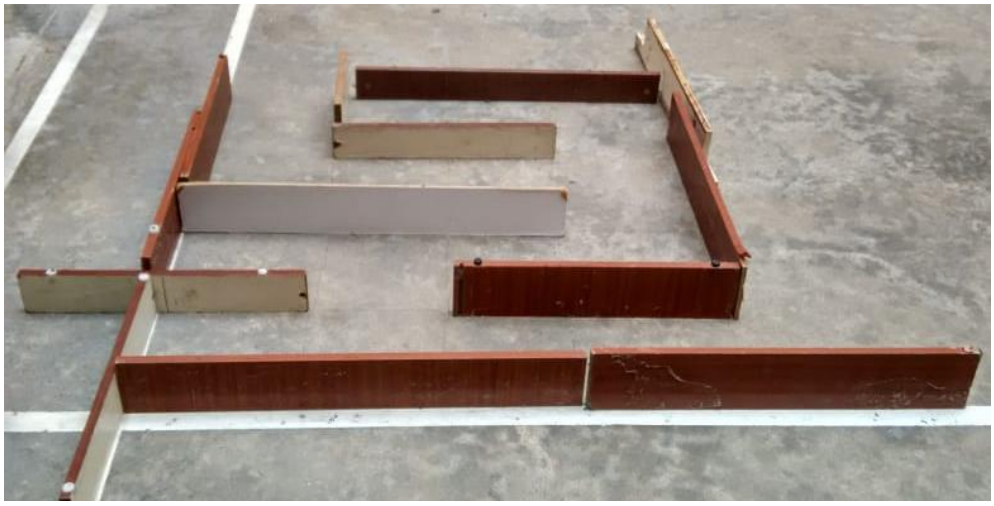
It acts as the main structure of the robot.

LANGUAGE USED

Initially, a C/C++ code had been written to understand and improvise upon the working of the algorithm. Later, this was converted into Arduino code.

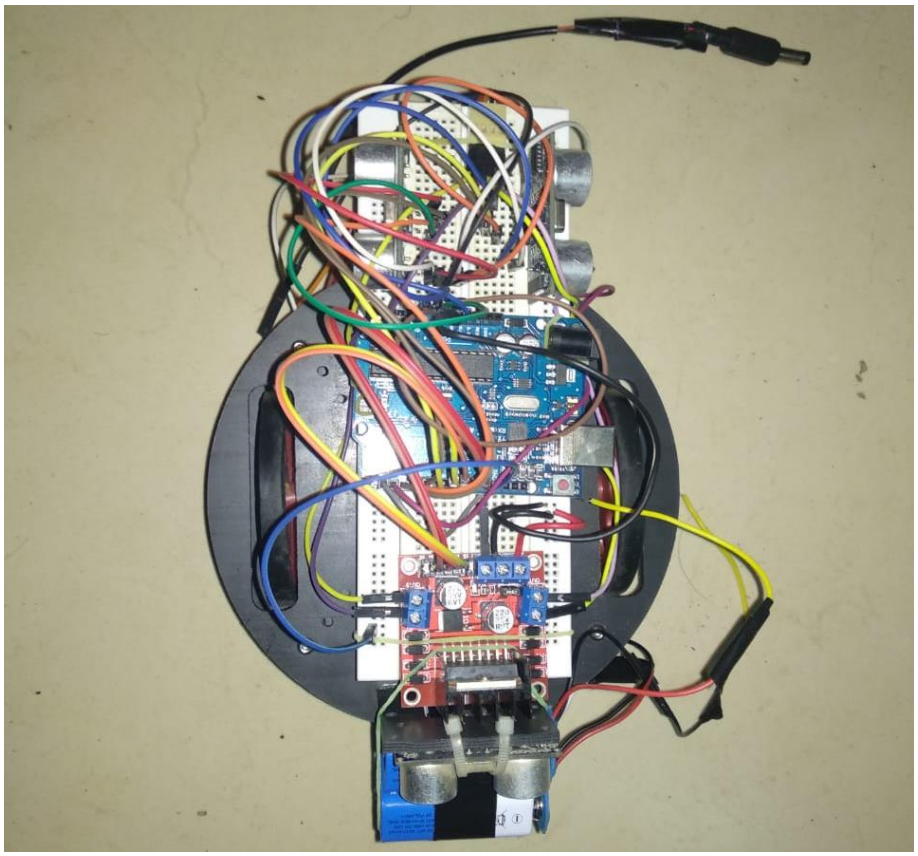
WORK SNIPPETS

The Maze constructed with wooden slabs



Robot in motion

The connections of the robot



CONTRIBUTIONS

1. Hardware

Pushpit, Jayen and Mahima were responsible for the hardware implementation of the project.
Starting from the gathering of the hardware components to understanding and testing each component, to building the complete robot for testing the software was done by them.
Additionally the maze construction was done too for testing the robot in the maze environment. Two mazes were built:

- With thermacol
- With wood slabs

Individual Contributions

Pushpit
While implementation of the hardware and maze was done as a team effort as mentioned above, he:

- Implemented the testing of components which are ultrasonic sensors and Arduino
- Checked the ultrasonic sensors with an Arduino Code and an LED, where LED will glow if sensor distance is less than 20 cm. This experiment was performed with all 3 sensors connected to Arduino to test the sensor component with Arduino.
- Developed the software code to test the turning of the wheels in a particular direction with a specified rotational speed for each wheel rotation that is left forward/backward and right forward/backward.
- Assembled the robot chassis with screws with fixing the components on the body of the robot.
- Implemented connections of Arduino and ultrasonic sensors.

Constraints/Problems Faced:

- Constant Battery Drainage due the high power consumption of the robot drained the battery very quickly. For which 2 3.7V batteries had to be connected in parallel, used to supply power to Arduino
- Due to battery drainage, the wheels rotated in different speeds.
- The sensors required 5V of power supply each, but here the ultrasonic sensors received 3V of power supply each.
- 2 batteries caused the robot to be heavy and replacing with lower weight batteries misbalanced the robot, thereby allowing only 1 wheel touch the surface at a particular time. For which the heavier battery had to be used and weights were added to the robot for the wheels to touch the surface together.
- Due to large number of wires, the signal bursting of the ultrasonic sensors were interfered, therefore wires had to be circled around the sensors.

Jayen

While implementation and hardware was done as a team effort as mentioned above, his role involved more towards the hardware component testing and assembling. He:

- Implemented the connections of the motor driver L298n module with the Arduino
- Implemented the software code to check if the motor driver performs the desired commands from the Arduino and that if the robot's wheel is rotating or not.
- Wired and re-wired the connections of the robot whenever needed according to testing.
- Implemented the power distribution and connected 2 batteries:
  (1) One for the motor driver
  (2) One for the Arduino

Constraints/Problems Faced:

- The motor driver did not receive enough power through the 9V battery supply. It wasn't enough to drive the wheels of the robot. For which 2 9V batteries had to be connected in parallel to provide enough power supply to the motor driver.
- The rubber tires of the robot was a huge disadvantage for the robot to run on multiple surfaces.
- The robot could only run properly on medium rough surfaces. Like concrete
- Initially the sensors and motor driver weren't working together, for which the code had to be re-framed.

Mahima

While implementation and hardware was done as a team effort as mentioned above, her role involved:

- More towards the hardware component gathering and maze construction.
- In collecting every component required according to cost analysis made earlier, required by the project and robot's hardware construction.
- In testing the wires before connection made to test for fused or broken wires
- In checking scanning connections made to the Arduino and every component to check for ground and positive wires connected correctly with the batteries.
- In building the maze's base as cardboard and building the walls with thermacol
- In cutting and acquiring the wooden slabs to be cut of particular dimension to be used as walls in final testing.
- In calculations of the matrix developed for the robot, to implement the maze construction according to each cell dimension of the matrix which was 30cm x 30cm.

Constraints/Problems Faced:

- The robot wasn't able to run on the cardboard surface when testing the robot, hence it had to be removed.
- The thermacol walls had a lower strength tolerance, hence while testing the robot's performance, walls used to break. For which wooden slabs were given as the solution with strength tolerance much higher than thermacol.
- Due to battery drainage and robot's movement and turning issue according to delay, the cell's dimension had to be recalculated and maze had to be altered.

2. Software

The software team was responsible for the creation of the two Arduino C files, one for the implementation of Wall's algorithm and the other for shortest path algorithm.
The team was involved in writing pseudocode, actual code, and documentation, remarking, and testing whether the robot parts are working in synchronization with the code and updating the code as and when required.
Both the algorithms are a result of collaboration from both the members along with individual roles.

Individual Contributions

Osho Agyeya

- Determining the correct echo pins and trigger pins for integration of software into hardware.
- Determining the distance values obtained from three Ultrasonic sensors for detecting walls in left, right and front using speed and time values.
- Implementing the code for left turn, forward movement, right turn, stopping and 180 degree rotation of the robot along with determining the correct delay values.
- Transferring the above stated movement code and delay values into the shortest path algorithm and changing values as required.
- Bug fixing and testing of both code files for errors.

Constraints/Problems Faced:

- Constant battery drainage caused high fluctuation of delay values during testing resulting in undesirable turning angles.
- Datatype of distance values calculated for sensors had to be adjusted to make sure that there is no overflow of values which could have led to negative distance values.
- Delay values had to be adjusted according to surfaces (hostel floor, badminton ground) due to variable friction.
- Bug fixing and testing of both code files for errors.

Tushar Pahuja

- Conversion of the input maze into a 2D matrix along with wall information.
- Initialization of matrix values as stated in the algorithm using queue.
- Coding function for obtaining next step depending upon current orientation of the robot face.
- Changing initial matrix values according to the proposed algorithm using stack.
- Calculating final path from source to destination.

Constraints/Problems Faced:

- Due to the absence of standard data structures like stacks, queues, pairs in the Arduino C, the data structures and their functions had to be implemented fully from scratch which took a lot of time and effort.
- Due to the small amount of memory being supported by the Arduino, the data types of stacks and queues and other values had to be adjusted to fit the memory requirements of the board.
- The proposed algorithm was tested for various maze and wall configurations and the output path from the source to the destination was cross checked to verify the correctness of the algorithm which introduced a lot of bugs and hence corrections in the algorithm.

CONCLUSION

In our project, the shortest path algorithm gave accurate results of simulations and gave good results depending on the surface and batteries. After the starting computational delay, there was no decision making delay while traversing the shortest path.