AUTONOMOUS MAZE SOLVING ROBOT.

By

MUSFIQUR RAHMAN, ID: 122015001

A project report submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Electronics and Telecommunication Engineering.

Supervised By:

Faysal Hakim,
Junior Lecturer
Department of Electrical and Electronic Engineering (EEE)
University of Liberal Arts Bangladesh (ULAB)

Electronics and Telecommunication Engineering
University of Liberal Arts Bangladesh
House 56, Road 4/A @ Satmosjid Road,
Dhanmondi, Dhaka-1209,
Bangladesh.

Date, 2nd May, 2017

DECLARATION

This project report is submitted to the Electronics and Telecommunication Engineering, University of Liberal Arts Bangladesh in partial fulfillment of the requirements for the degree of Bachelor of Science. So, I hereby, declare that this project report is based on the surveys found by me. Materials of work found by other researchers are mentioned by reference. This project report, neither in whole, nor in part, has been previously submitted for any degree.

Musfiqur Rahman
ID: 122015001
Email Address: musfiqur.rahman.ete@ulab.edu.bd

ACKNOWLEDGEMENT CERTIFICATE

The project report entitled "AUTONOMOUS MAZE SOLVING ROBOT" is submitted to the Electronics and Telecommunication Engineering, University of Liberal Arts Bangladesh, Dhaka in partial fulfillment of the requirements for the degree of Bachelor of Science.

Dated: 25th April, 2017

S. M. Mahbubur Rahman, PhD
Professor and Head
Department of Electrical and Electronic Engineering (EEE)
Department of Electronics and Telecommunication Engineering (ETE)
University of Liberal Arts Bangladesh (ULAB)

Signature & Date

Faysal Hakim,
Junior Lecturer
Department of Electrical and Electronic Engineering (EEE)
University of Liberal Arts Bangladesh (ULAB)

Signature & Date

PREFACE

Firstly, I would like to express my gratitude to the almighty Allah for providing me health, patience and knowledge to successfully complete this project.

I would like to thank to my honorable teacher **Prof. S. M. Mahbubur Rahman**, Head of the Department of Electrical and Electronic Engineering & Department of Electronics and Telecommunication Engineering, University of Liberal Arts Bangladesh, who arranged this project opportunity for me.

I submit my highest appreciation to my supervisor Mr. **Faysal Hakim**, Junior Lecturer, Department of Electrical and Electronic Engineering, University of Liberal Arts Bangladesh for his beneficial advices, support and guidance throughout this project.

A special thank for Lab assistant **Khorshed Alam,** my friend **Imran Hosain** and my beloved junior **Pronab Biswas** to give me time during my lab work. Thanks to all academic staff whose makes me reach this level of education.

Finally, thanks to my parents, family and friends for their help, encouragement and guidance, who without I would not be where I am today.

ABSTRACT

Maze solving problem is a very old problem, but still now it is considered as an important field of robotics. This field is based on decision making algorithms. The main aim of this project is to make an Arduino based efficient autonomous maze solver robot. Two simple mazes solving algorithms “Wall following algorithm” and “Flood fill algorithm” are used to make this robot. In this project Hardware development, software development and maze construction had been done. For performance testing, the robot will implement to solve 4×4 maze. Capability of finding the shortest path is also verified.

TABLE OF CONTENTS

LIST OF FIGURE

Chapter 1

Introduction

This chapter is going to provide an introduction about this project. These following topics have been described in this chapter, General information about Autonomous Maze solving Robot, Problem Statement, objective of the project, Project organization and Thesis outline.

Introduction

A maze is a network of paths, typically from an entrance to exit. The concept of Maze approximately thousand years old [1]. Which was invented in Egypt. From then, many mathematician made various algorithm 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, which is “Decision Making Algorithm”. Cause, this robot will be placed in unknown place, and it requires to have a good decision making capability. There are many types of maze solving robot using various type of algorithms.

In this project, 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. It will also follow the *Flood fill* algorithm for finding the shortest path.

1.1 Autonomous Maze solving Robot

An autonomous robot is a category of robot that can perform tasks intelligently depending on themselves, without any human assistance [2]. Maze Solving Robot is one of the most popular autonomous robots. It is a small self-reliant robot that can solve a maze from a known starting position to the center area of the maze in the shortest possible time. A maze solving robot make multiple runs in a maze, first it create a map of the maze layout and store it in its memory, then run through a shortest path [3].

1.2 Problem Statement

To design a hardware for maze solving robot, construct a software with the combination of wall following and flood fill algorithms then implement the software in the hardware of maze solving robot. At last make a maze 6×6 maze to verify the robot.

1.3 Objectives

- Understand and implement the wall follower and Flood fill algorithm.
- Design and build ultrasonic sensors array.
- Design and build Arduino based hardware.
- Programmed the robot to solve the maze.
- Make a small 4×4 maze.

1.4 Thesis Overview

This thesis is divided into **7** chapters. Each chapter discuss ion the different issues which related to this project. The outline of each chapter is stated in the paragraphs below.

Basic introduction about autonomous robot, maze solving robot have been described in **Chapter 1**. Project statement and objectives of the project have also been described in this chapter. **Chapter 2** covers the important background information and history about the Maze solving robot. Many important theories, method and algorithms on maze solving problem is also given there. This section will start with providing some research about Wall Following Algorithm and Fill Flood Algorithm. **Chapter 3** is going to give some details for each component including features, specifications and how it operates. Important Algorithms which is applied in this project is described in **Chapter 4**. Methodology, Software development, Hardware implementation is described in **Chapter 5.** In **Chapter 6,** simulated result and important discussion is described. Conclusion and Further development is described in chapter 7.

Chapter 2

Literature Review

This chapter is going to provide important background information and history about the Maze solving robot. Many important theories, method and algorithms on maze solving problem is also given there. This section will start with providing some research about Wall Following Algorithm and Fill Flood Algorithm.

2.1 Background History

In the middle of the 20th century, Maze solving problems become an important field of robotics [4]. In the year of 1972, editors of IEEE Spectrum magazine came up with the concept of micro-mouse which is a small microprocessor controlled vehicle with self-intelligence and capability to navigate a critical maze [5][6]. Then in May 1977, the fast US Micro mouse contest, called "Amazing Micromouse Maze Contest" was announced by IEEE Spectrum. From then, this type of contest became more popular, and many type of maze solving robots are developed every year.

Late 1970s the designs of the maze solving robots designs were used to have huge physical shapes that contain many blocks logic gates. Figure 1.1 and 1. 2 show the example of early the maze solving robots (micro mouse). Due to technological development the physical size of the robot becomes smaller and the features of the robot becomes modern.

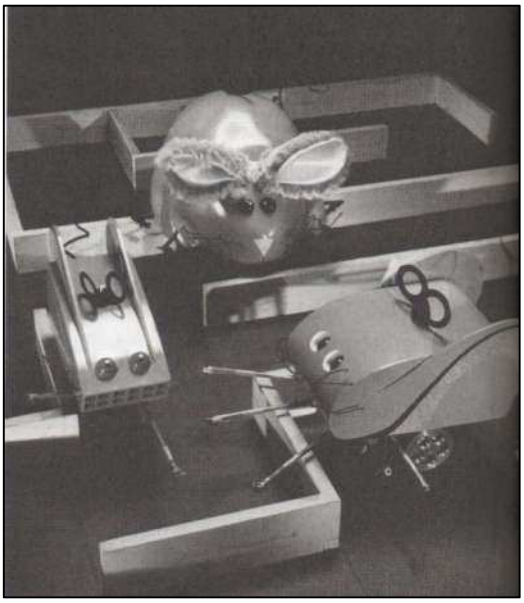


Figure-2.1

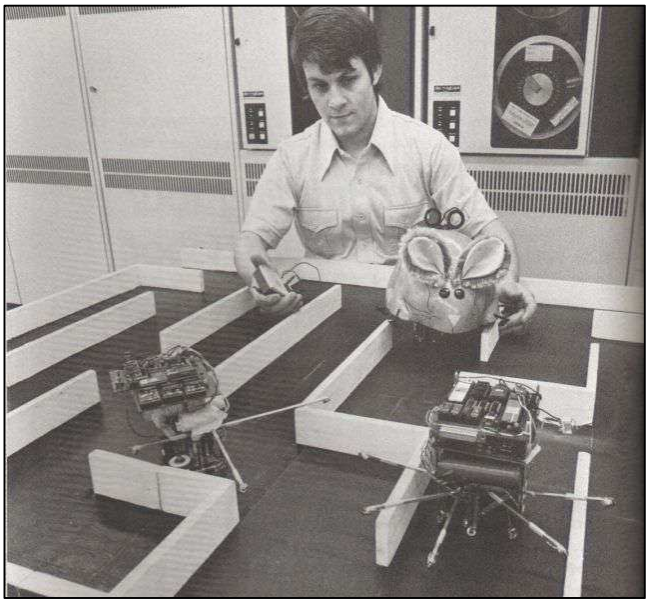


Figure-2.2

Figure-2.1 & 2.2: Old generation of micro-mouse robots [5].

2.2 Micromouse using Wall Following Algorithm by University of East London, 1999 [7].

In the year 1999, Michael Gims, Sonja Lenz and Dirk Becker from University of East London developed a micro mouse. They used a non-graph theory algorithm, **Wall Following Algorithm.** But their robot did not move intelligent in the map and it could not solve maze with loop.

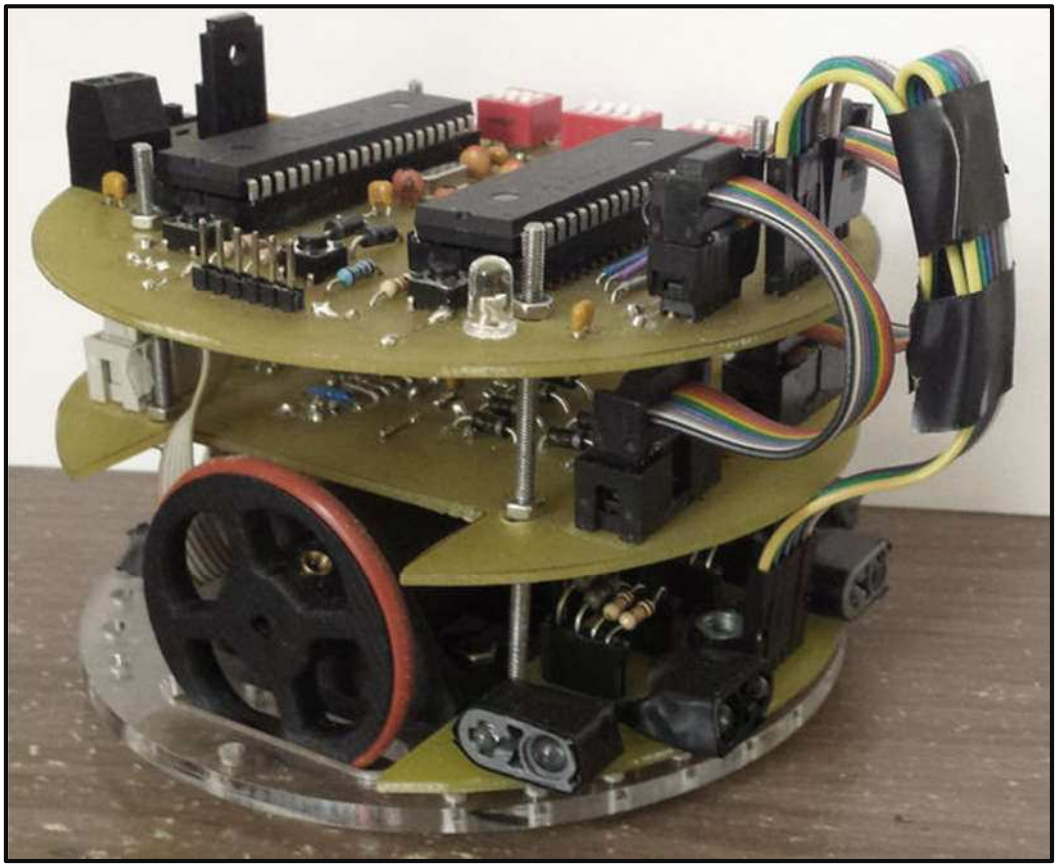


Figure-2.3: Micromouse University of East London [7].

2.3 Micromouse Maze Solving Robot by Chang Yuen Chung, UTM. 2008 [8]

Chang Yuen Chung, a student of Universiti Teknologi Malaysia (UTM) designed a micromouse using **Flood Fill Algorithm**. His robot was designed in three layers so that the robot looks more compact and smaller size. But it was very hard to troubleshoot if there is circuit faulty. Flood Fill Algorithm is one of the graph theory mazes solving algorithm. In paper [8], Chang Yuen Chung claimed that this Flood Fill Algorithm is able to find the shortest path but more memory is required for execution.

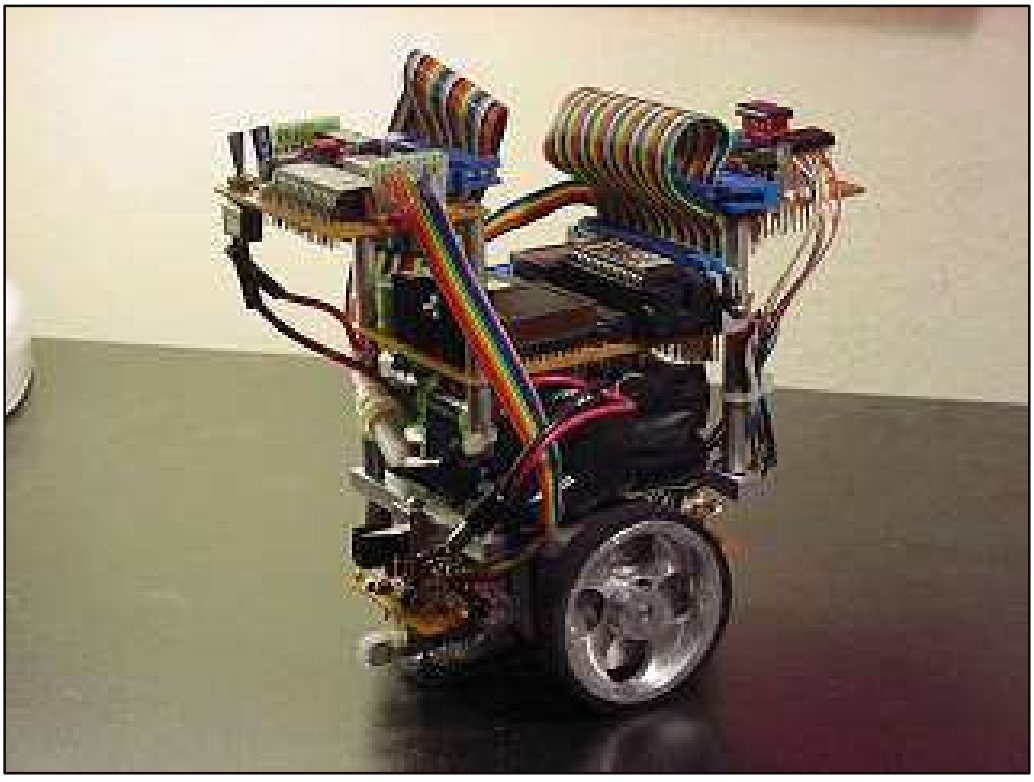


Figure-2.4: Micromouse by Chang Yuen Chung [8]

2.4 Research papers study

For gathering more information about maze solving robot and its algorithm, I read many research papers. Abu Bakor and Issa proposed a Hybrid Wall Follower Algorithm to solve a maze [9]. M.M Thu and N.N. Win designed an Ultrasonic sensor based maze solving robot [10].

2.5 Pulse Width Modulation

For controlling the motors speed, pulse width modulation (PWM) is used. Pulse width modulation is a simple method of controlling analogue devices via a digital signal through changing or modulating the pulse width [11]. An analogue device is become digital by powering it with a pulse signal switches between on and off (5V and 0V). This digital control is used to create a square wave. The duty cycle is defined as the percentage of digital 'High' to digital 'Low' plus digital 'High' pulse-width during a PWM period. The average DC voltage value for 0% duty cycle is zero; with 25% duty cycle the average value is 1.25V (25% of 5V). With 50% duty cycle the average value is 2.5V, and if the duty cycle is 75%, the average voltage is 3.75V and so on. So, by varying the average voltage, the motor speed can be controlled.

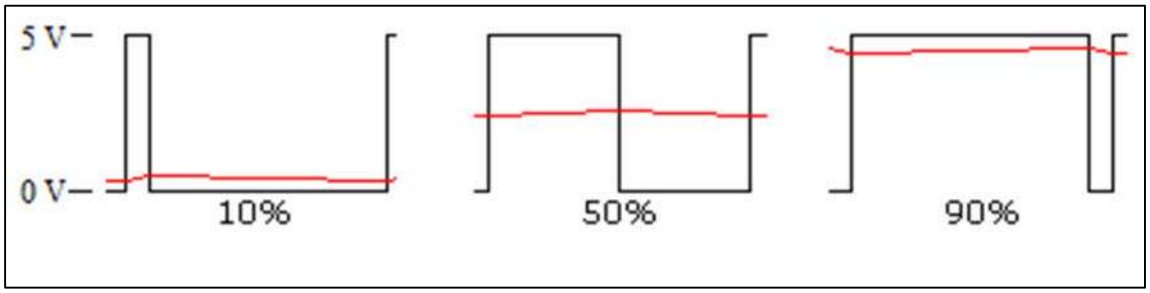


Figure-2.5: Example of PWM wave.

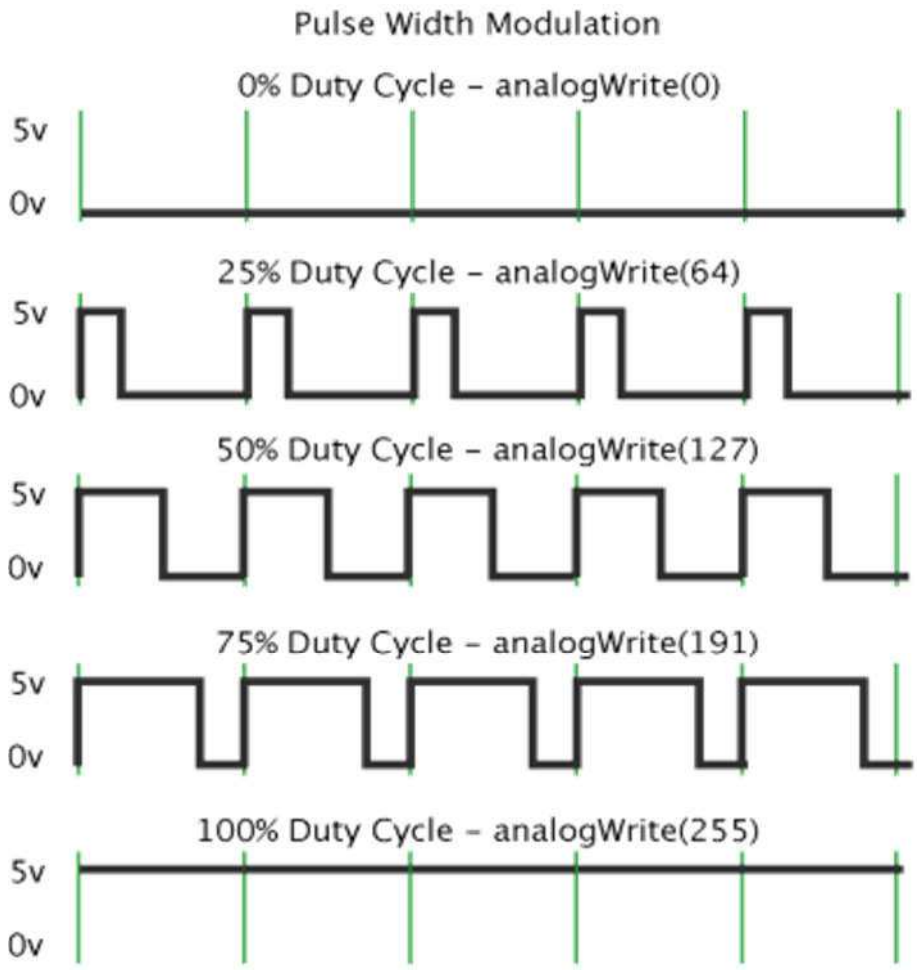


Figure-2.6: Various Duty cycle with Arduino (motor speed) code [12].

The power loss in PWM switching devices is very low. In many cases it is near to 0. So, this is the main advantage of PWM. While being used, resistors will tend to loss more power because of its heat dissipation. So, PWM is efficient in controlling motors.

Chapter 3

Hardware Description

This chapter covers the important information about all hardware which is used in this project. It is going to give some details for each component including features, specifications and how it operates.

3.0 Constructed maze solver

This project consists of several hardware components such as Arduino, Ultrasonic sonar sensors, Motor driver, Voltage Regulator, Batteries, etc. In figure 3.1 show the full project.

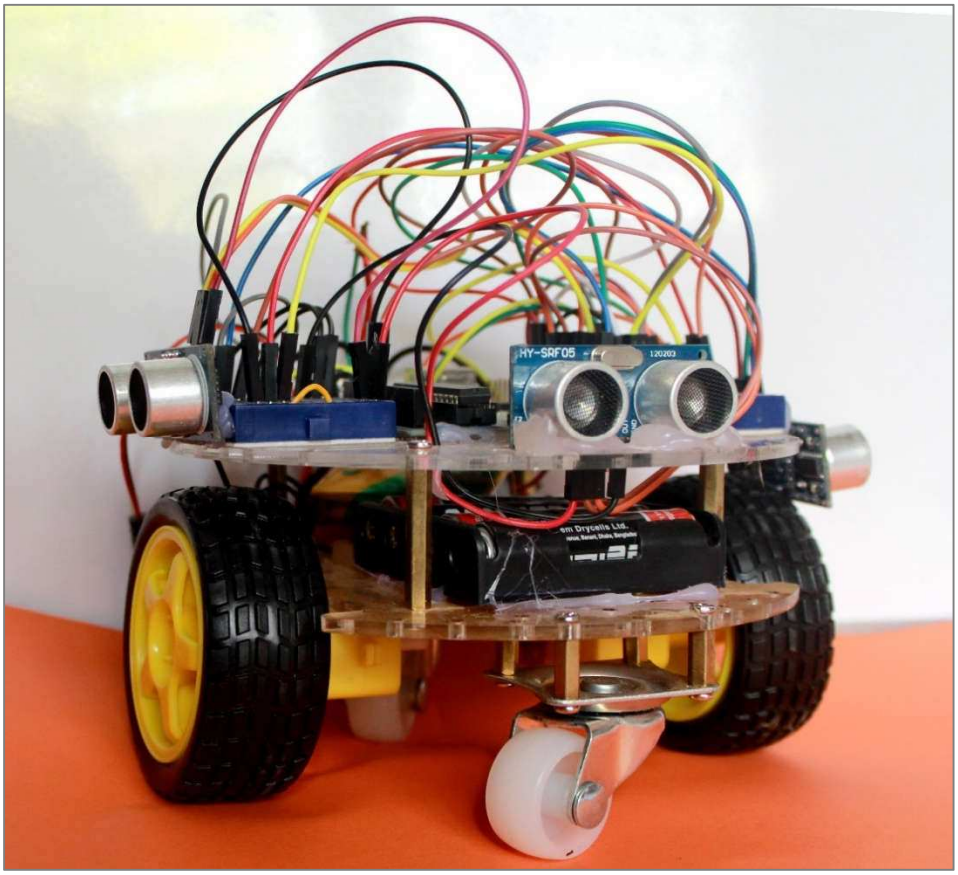


Figure-3.1: Autonomous Maze solving Robot

3.1 Arduino

The heart of this project is Arduino. All program of this project is stored in its microprocessor.

Arduino is an open source hardware development board. Arduino hardware consists of an open hardware design with an Atmel AVR processor [13]. Arduino programming language is used to program the processor.

There are many types of Arduino board available in the market. But, in this project Arduino UNO has been used. Figure 3.2 shows the Arduino UNO board.

Figure-3.2: Arduino UNO [14]**.**

Arduino UNO is based on the ATmega328P microprocessor. It has 14 input/output pins. 6 digital pin can be used as PWM outputs. It has 6 analog input pins. It has a 16 MHz quartz crystal. It contains USB connection port, dc power port. The microcontroller is programmed via Arduino

Software (IDE). Figure 3.3 shows the pin mapping of Arduino UNO and Figure 3.4 shows the IDE of Arduino.

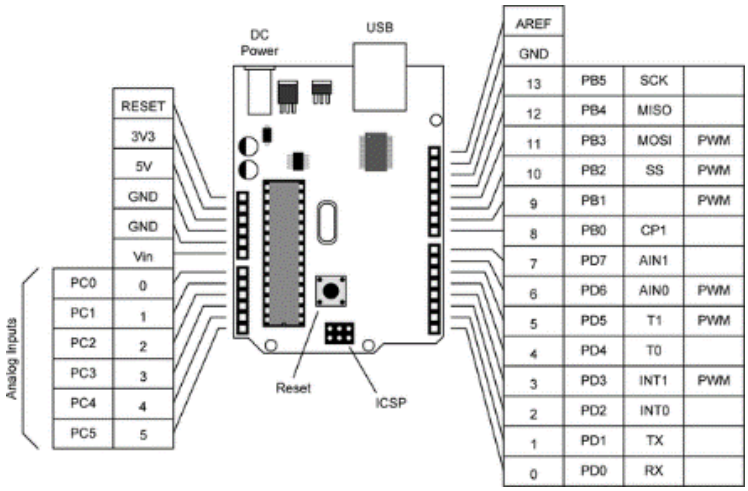


Figure-3.3: Pin Mapping of Arduino UNO [15]**.**

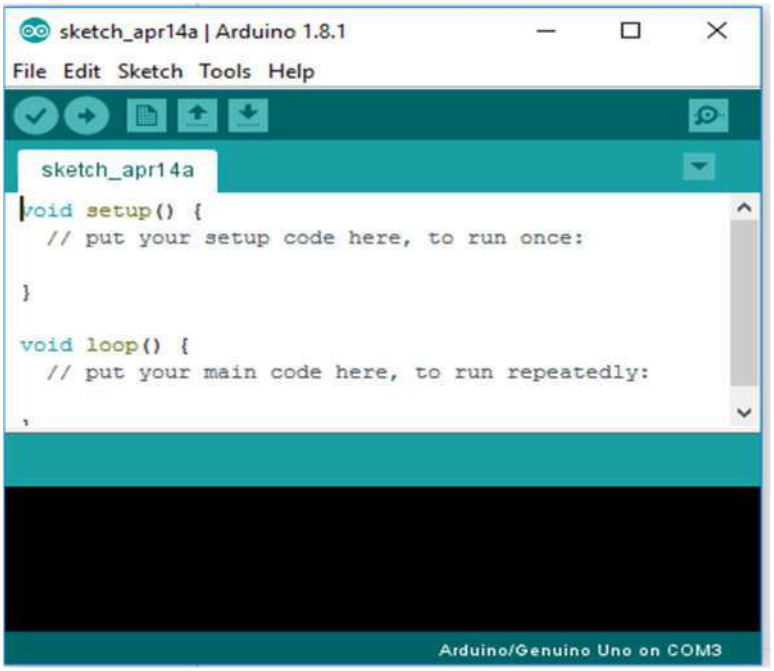


Figure-3.4: Arduino Software (IDE).

3.2 ATmega328P

ATmega328P is an 8-bit AVR RISC-based microcontroller. It has 32KB ISP flash memory, 1024B EEPROM, 2KB SRAM, 23 general purpose I/O lines, 6-channel 10-bit A/D converter, 32 general purpose working registers, 3 flexible timers, serial programmable USART, a byte-oriented 2-wire serial interface and SPI serial port [16].

Figure-3.5: ATmega328P [16].

3.3 Ultrasonic ranging sensor (HC-SR04)

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves [17]. HC-SR04 (figure- 3.6) is one of the common ultrasonic sensor. 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. The received signal go to the circuit for measured the distance. In figure-3.7, a working diagram of sonar sensor is given.

The formula of measuring distance using sonar sensor is given below:

$$distance = \frac{speed\ of\ sound\ \times time\ taken}{2}$$

Figure-3.6: HC-SR04 [17]

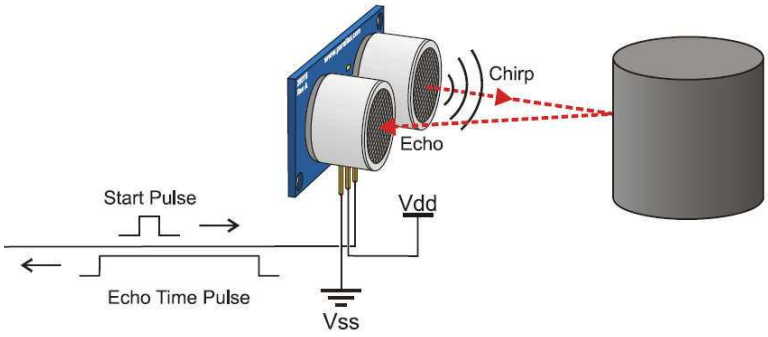


Figure-3.7: Working diagram of Ultrasonic sensor.

Features [18]:

- Operating Voltage: 5V DC
- Operating Current: 15mA
- Measure Angle: 15°
- Ranging Distance: 2cm - 4m

3.4 L298 Motor driver

This dual bidirectional motor driver. Most popular L298 Dual H-Bridge Motor Driver Integrated Circuit is used here [19].

Features [19]**:**

- Logic voltage- 5V
- Drive voltage- 5V to 35V
- Logic Current- 0mA-36mA
- Drive current- 2A (MAX each bridge)
- Maximum Power – 25W

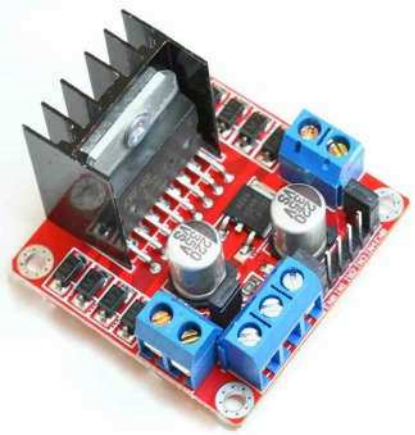


Figure-3.8: L298 Motor driver.

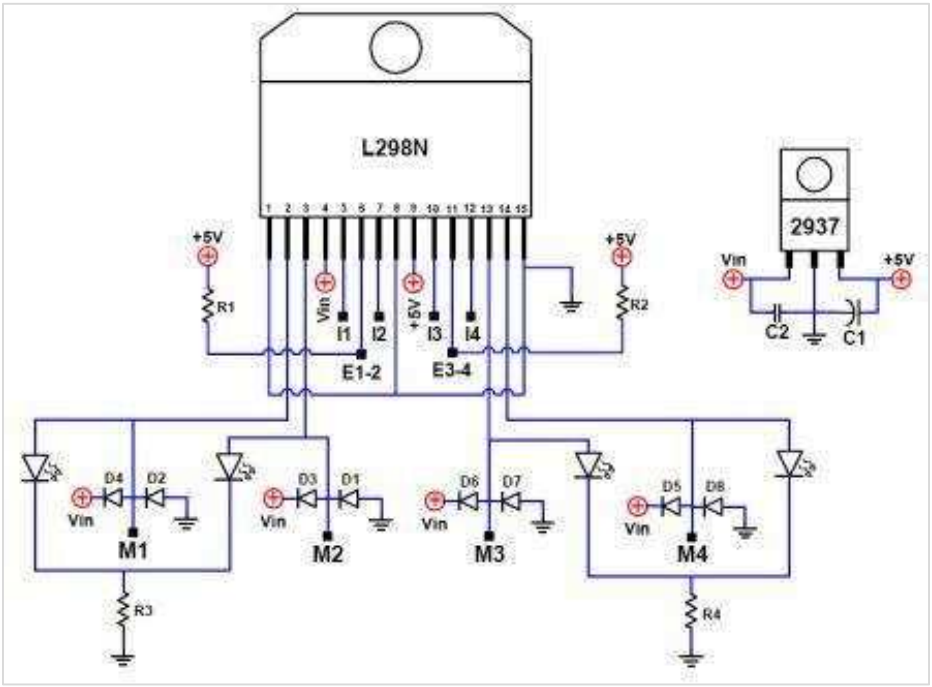


Figure-3.9: Circuit diagram of L298 Motor driver [20]

3.5 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.

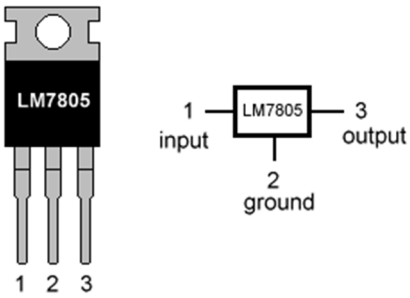


Figure-3.10: Pin out of LM7805 [21].

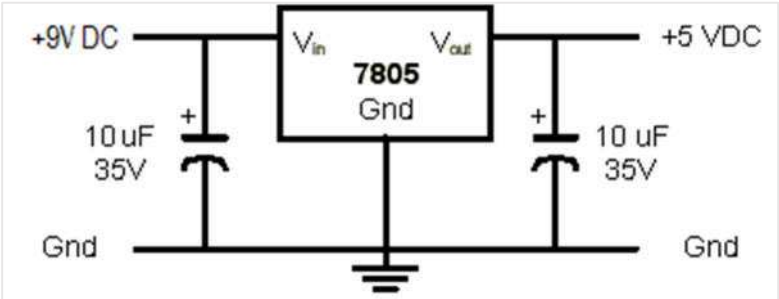


Figure-3.11: Circuit diagram of (5V) Voltage Regulator

3.6 Motors

Two DC gear motors have been used to drive this robot. This gear motor is ideal for robotic car or line-tracing robot.

Features

- Operating voltage: 3V ~ 6V DC
- Speed without load: 800g.cm
- Maximum torque: 90±10rpm
- Reduction ratio: 1:48
- Load current: 190mA (max. 250mA)

Figure-3.12: DC Gear motor with wheel.

3.7 Batteries

4 AA batteries have been used to power the motor driver. Each battery relates to other in series connection. One 9V battery is used to power the Arduino board, another is used to power the voltage regulator.

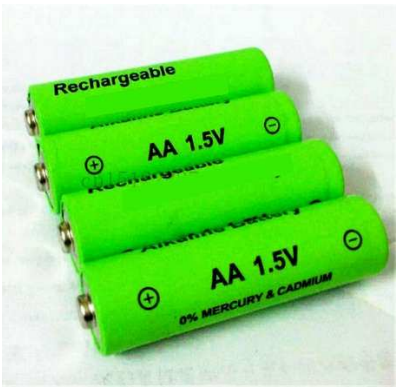


Figure-3.13: AA Battery

Figure-3.14: 9V battery

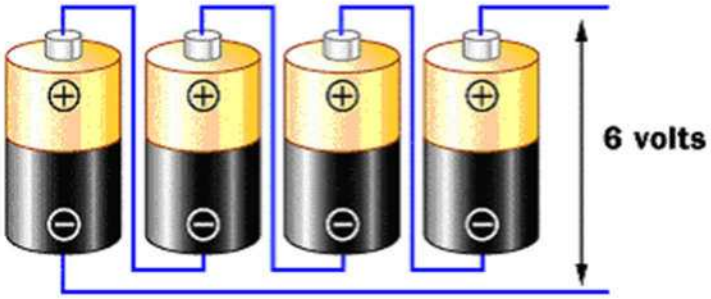


Figure-3.15: Series connection of AA batteries

Chapter 4

Algorithms

This chapter will give the details about the algorithms. Mainly, Wall following algorithm and Flood fill algorithm are in this project.

4.1 Wall following algorithm

The most common algorithm for maze solving robot is Wall following algorithm. 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 [22].

The instructions used in the algorithm for both left and right wall is given in a table below:

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

Table-1: Left-Right wall following routine

4.2 Flood fill algorithm

The most efficient Maze solving algorithm is flood fill algorithm. It is derived from "Bellman Ford Algorithm [23]". The algorithm works by assigning value for all cells in the maze, where these values indicate the steps from any cell to the destination cell [24]. The first array is holding the walls map values, while the other one is storing the distance values [25].

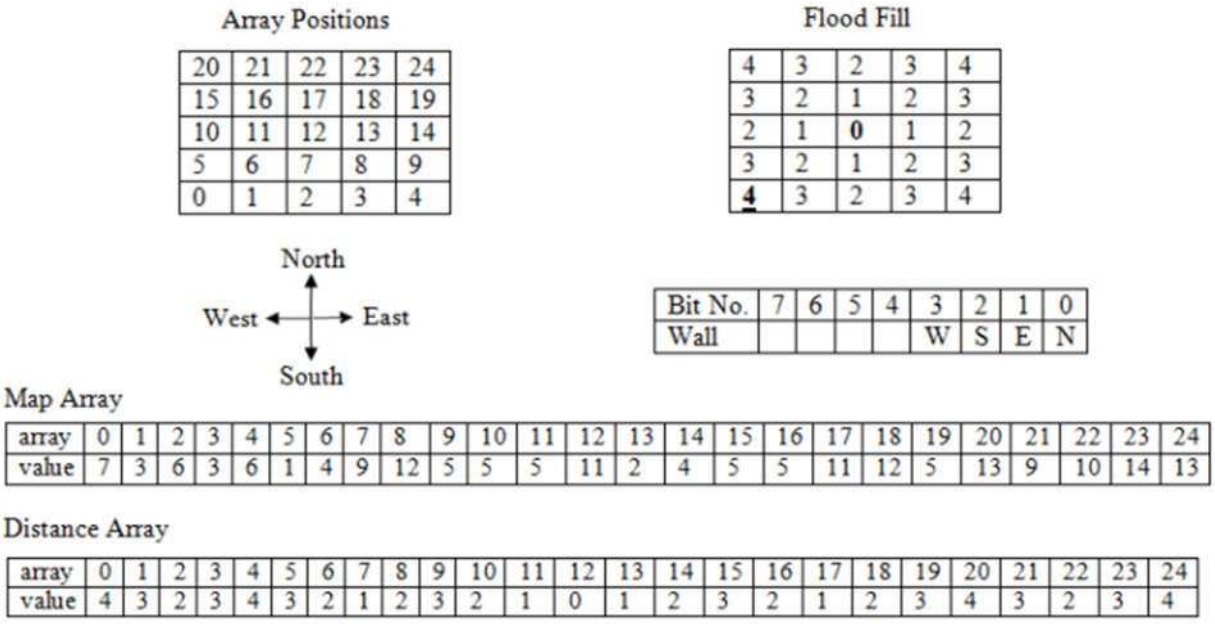


Figure-4.1: Fill flood algorithm using one-dimensional arrays [25]**.**

In every cell, robot will follow the following steps:

1. Update the wall map.
2. Flood the maze with the new distance values.
3. Decide which neighboring cell has the lowest distance value.
4. Move to the neighboring cell with the lowest distance value.

Chapter 5

Methodology

This chapter covers the flow chart of this project, block diagram of this robot, physical implementation, software development, software implementation etc.

5.1 Project Flow

Figure 5.1 below, shows the work flow of this project.

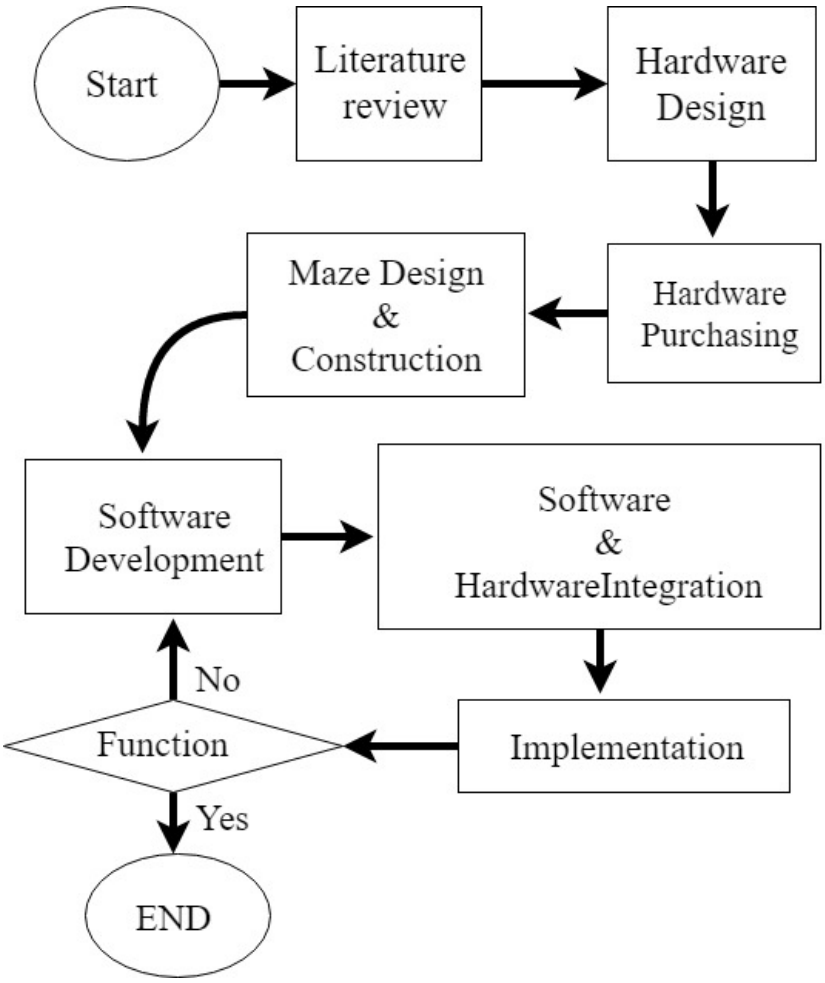


Figure-5.1: Flow chart of this project.

5.2 Project Overview

The hardware chosen to complete this project is Arduino based two-wheel mobile robot. Three ultrasonic sonar sensors have been used to map the maze and solve the wall maze.

5.2.1 Block Diagram

Figure-5.2: Block diagram for the maze solving robot.

5.3 Robot Implementation

5.3.1 Physical configuration

Figure-5.3, 5.4, 5.5 shows the physical configuration of the robot. The robot body (chassis) is three layered. And it is made of laser cut fiberglass.

Arduino and motor driver circuit has been placed on the top robot chassis. Three ultrasonic sonar sensors have also been placed on the top chassis. The power section has been placed at the middle part of the chassis. Two 9V batteries, 4 AA battery holders and 5V voltage regulator circuit has

been placed on it. At bottom part, two gear motors with wheel and two caster wheels have been attached.

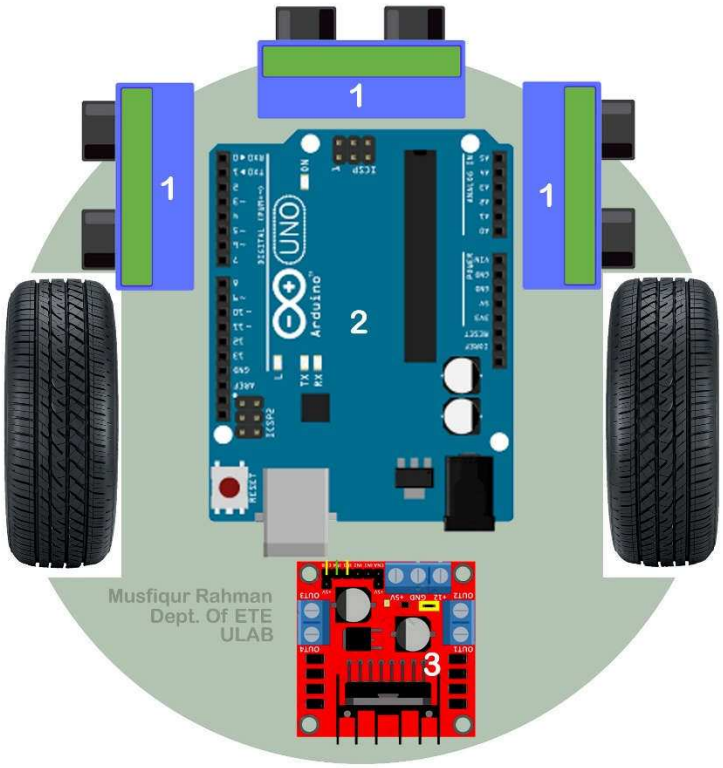


Figure-5.3: Physical Configuration (TOP)

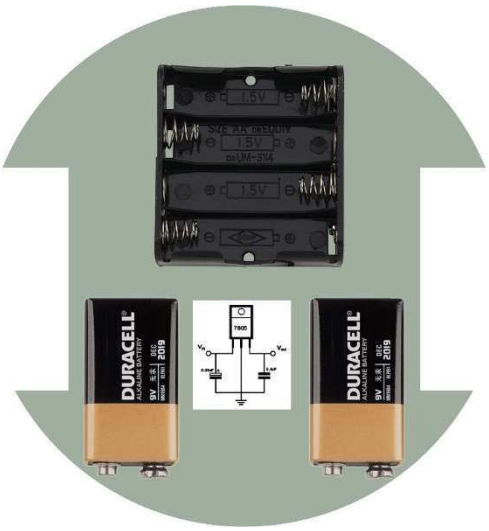


Figure-5.4: Physical Configuration (Middle)

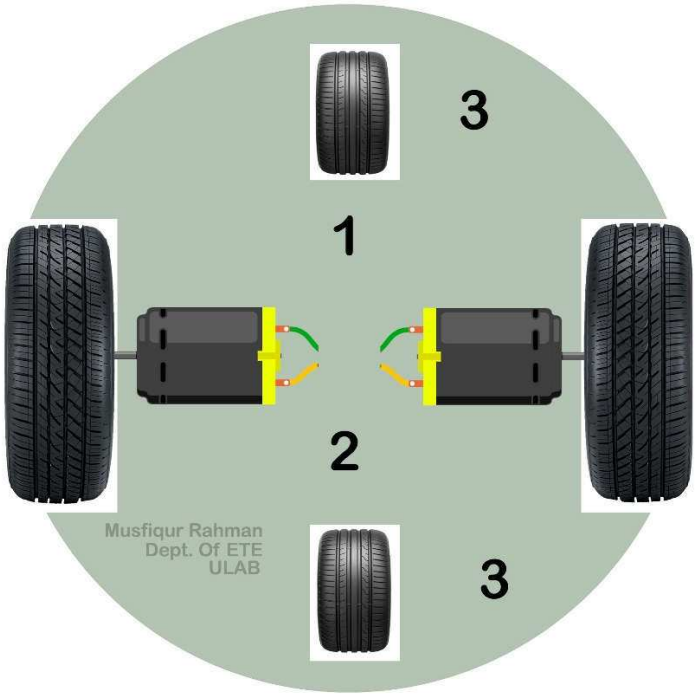


Figure-5.5: Physical Configuration (bottom)

Equipment list Table:

TOP PART	MIDDLE PART	BOTTOM PART
1. Ultrasonic Sonar sensors. 2. Arduino. 3. 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,

5.4 Robot Program

This robot is designed as a wall maze solving robot. So, the hardware and software part is designed to solve this type of maze. Left and right sensors flow the left and right wall. The front sensor is used to map the front wall. At first, it solves the maze using wall following algorithm. At the same time, it makes a map of the maze using all sensors. It stores the map value cell by cell in its memory. After the first run it use the flood fill algorithm to find the shortest path. It collects the

map value from Arduino memory to make a one-dimensional array and apply the flood fill algorithm. This is the full program of the robot. Figure- 5.6 bellow, shows the Working function of this robot.

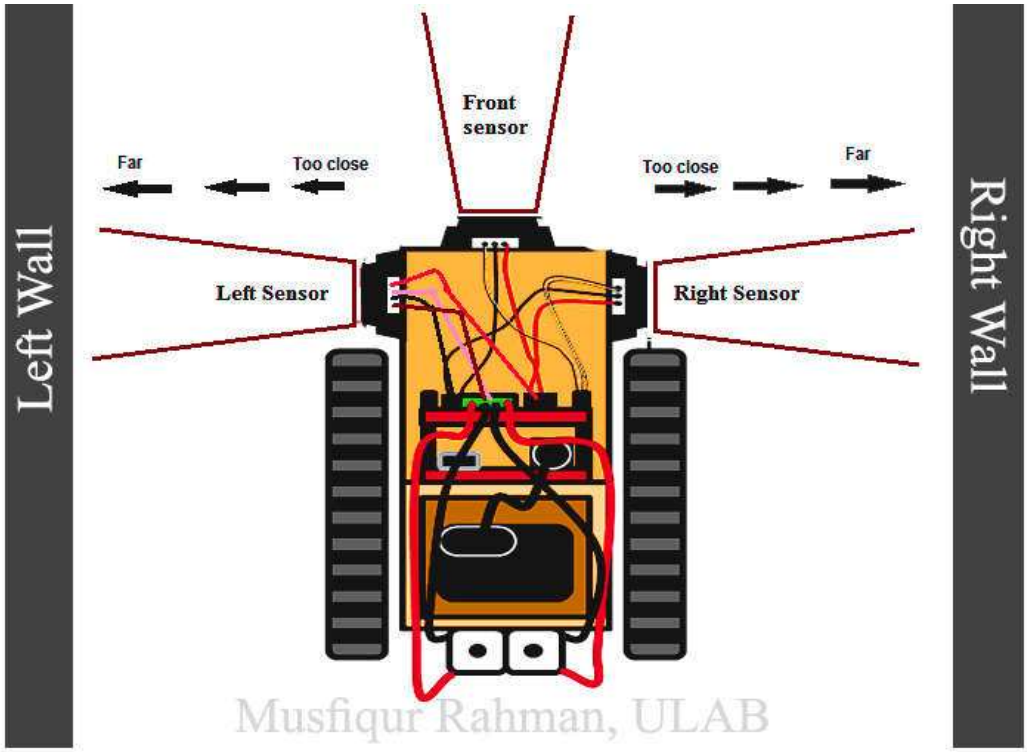


Figure-5.6: Working function of this maze solving robot.

5.5 Software Development

The maze running software have been developed using two known maze solving algorithms.

5.5.1 Wall following code

The first section of the code, if the right wall and left wall distance is equal or less than 5cm than both motors are in forward mode. So, the robot move straight. If, the right wall is more than 5cm, it is considered as open wall, than the robot turn into right. Same as, if the left wall is more than 5cm, it is considered as open wall, than the robot turn into left. And this function is continued in a loop till the maze is solved.

5.5.2 Flood fill code

The flood fill algorithm based Arduino code is used to find the shortest path of the maze. When the robot pass a wall, it set a value for every cell. It calculates the values by itself and drive the robot in shortest path.

5.6 Maze construction

A 4×4 physical maze is constructed to verify the robot performance. And simulate the flood fill algorithm.

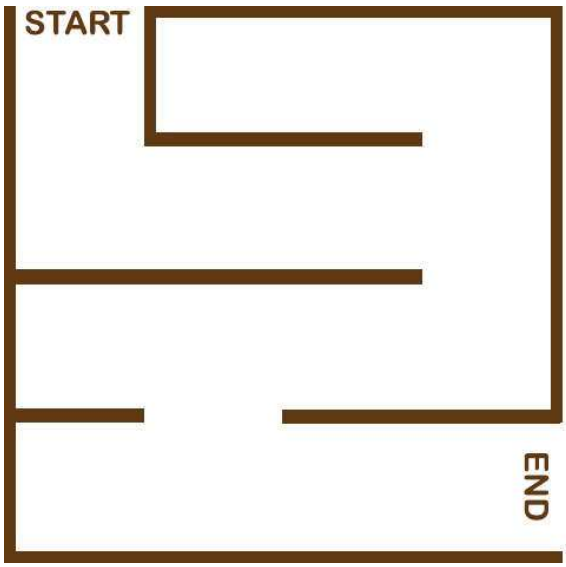


Figure-5.7: 4×4 Maze design.

Figure-5.8: Maze Construction.

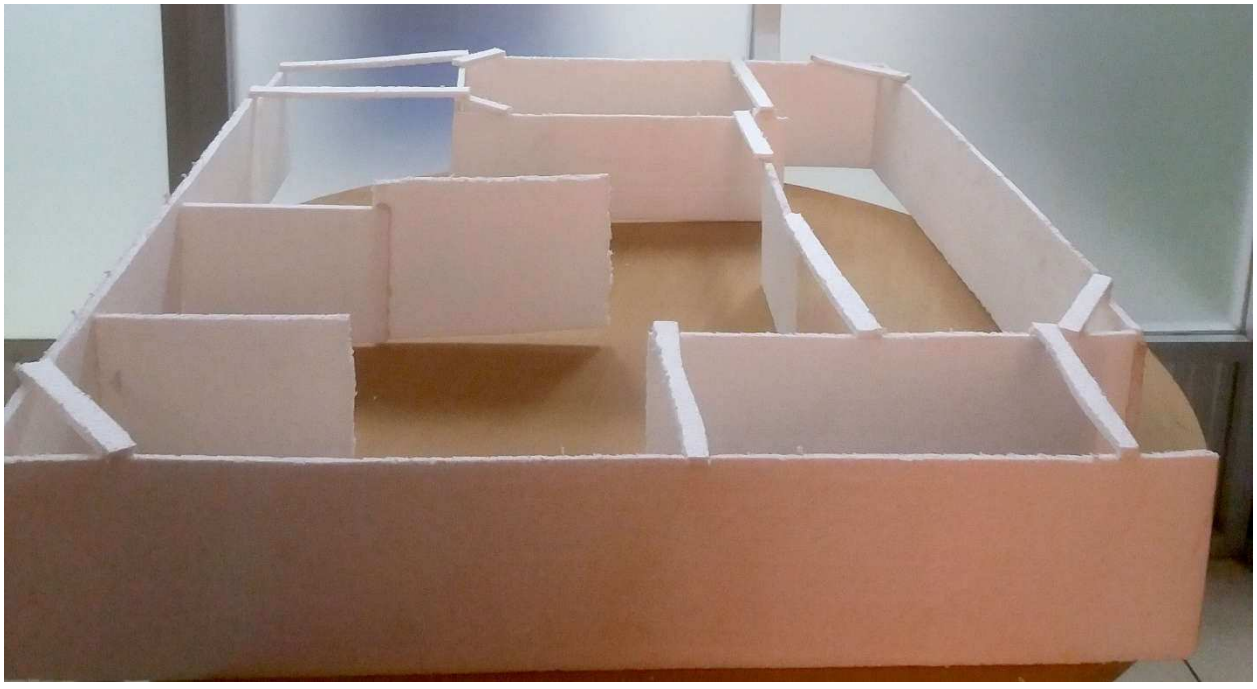


Figure-5.9: The constructed Maze (4×4=16 cells)

Chapter 6

Result & Discussion

This chapter covers the simulated results, compare results with others system, and discussion about this project.

6.1 Ultrasonic sonar response

Figure 6.1 shows the Time vs. Distance Graph of Left, right and Front ultrasonic sonar sensor. Ultrasonic sonar is used to measure the distance of wall. The values are measured by using serial monitor of Arduino IDE.

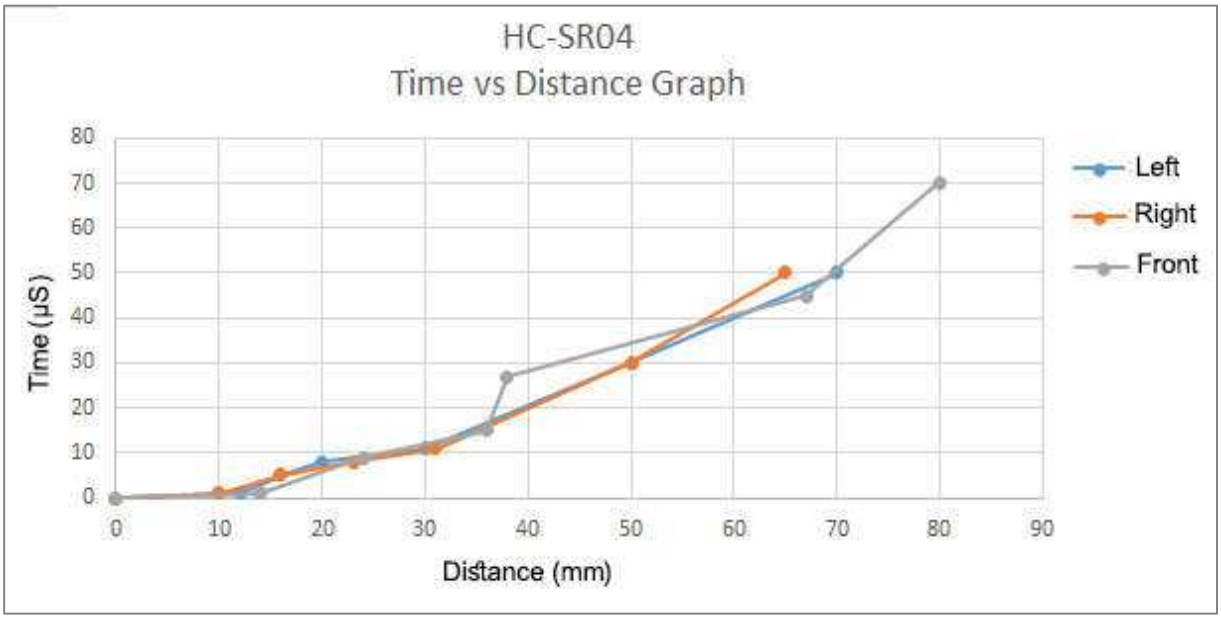


Figure-6.1: Time vs. Distance Graph of HC-SRO4 (Ultrasonic sonar sensors)

6.2 Run on Maze

6.2.1 Wall follower

The robot solves the maze by following the wall. In real maze, this robot was tested. In figure 6.2, left and right wall is closed and front wall was opened so, it moves forward. In figure 6.3, both left and right wall is open and front wall is closed. So, as defined as code, it turns right first and faced front wall was closed, left and right walls was also closed. Then, it makes 180° turn. After that, both left and right wall are closed but front wall is open. So, it moves forward. In the maze, it works in this manner and solves the maze.

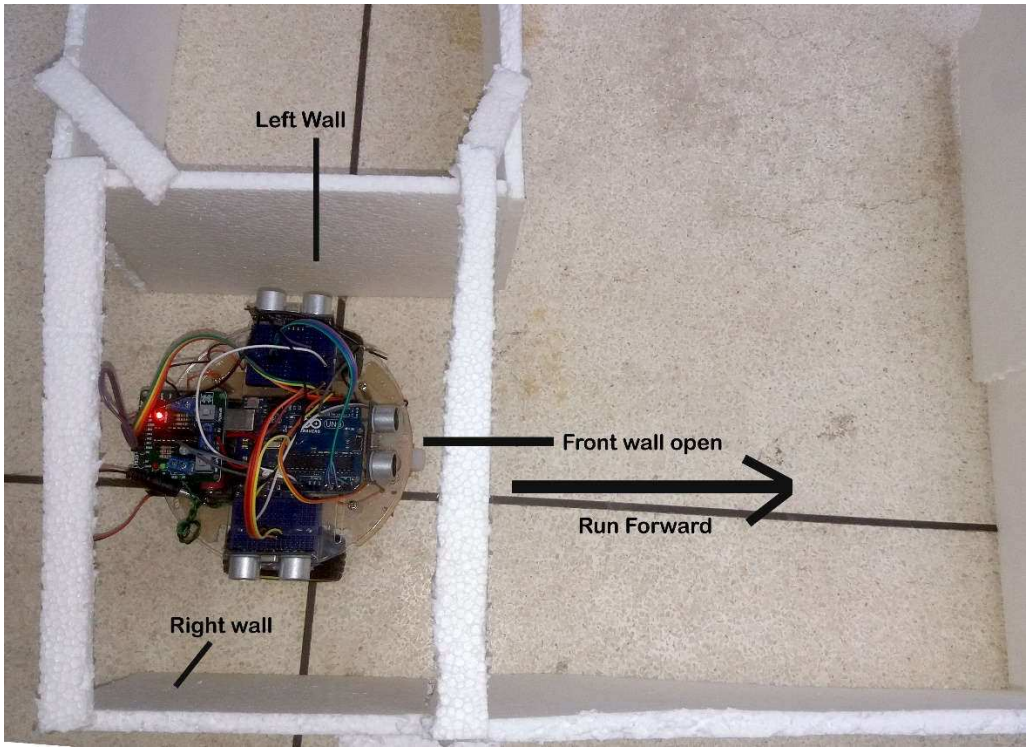


Figure-6.2: Run in maze. (R&L wall is closed, F wall is opened)

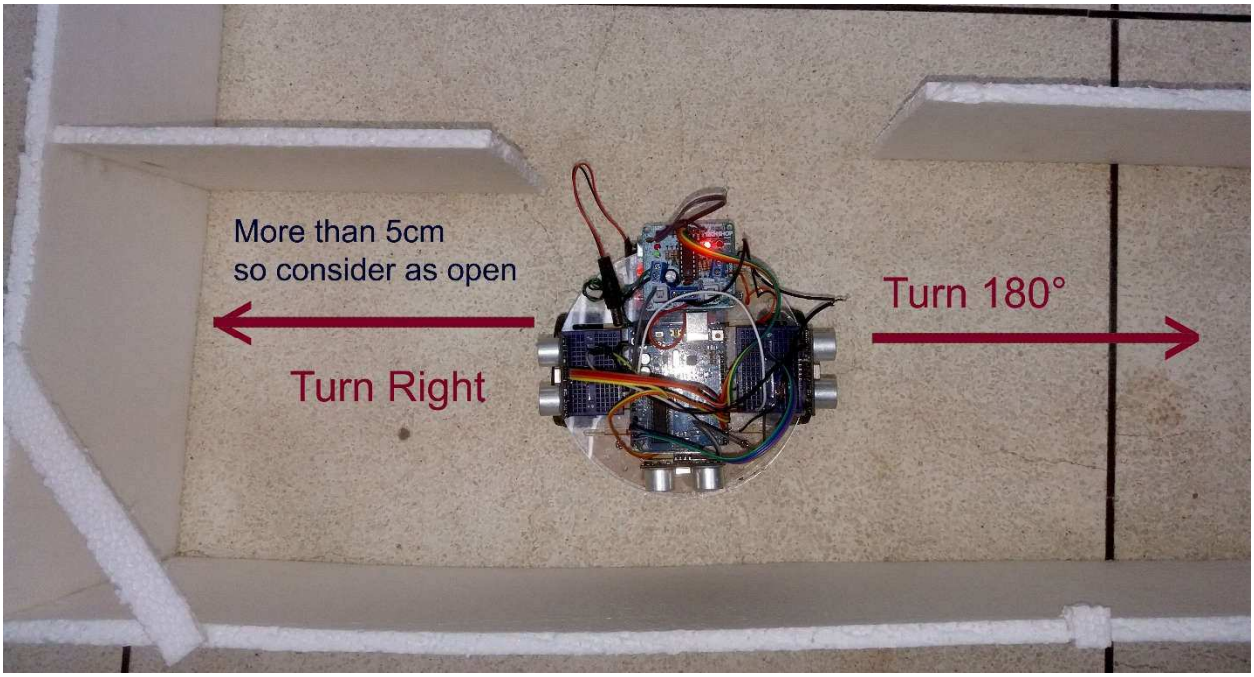


Figure-6.3: Run in maze. (180° Turn)

6.2.2 Flood fill

When the robot solves the maze, it maps the maze and stores all the values of the maze. During second run time, it designs the maze as a flood fill algorithm. It sets the end cell at zero (0) and set all higher values one after another.

Figure-6.4, 6.5, 6.6 show the result of Flood fill algorithm as a shortest pathfinder.

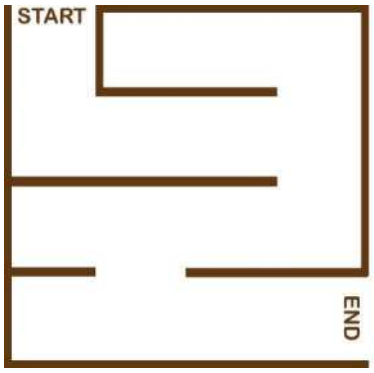


Figure-6.4 : The Maze

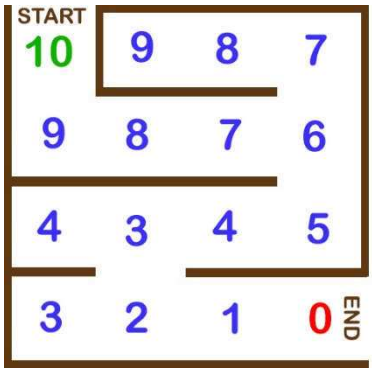


Figure-6.5: Apply Flood Fill

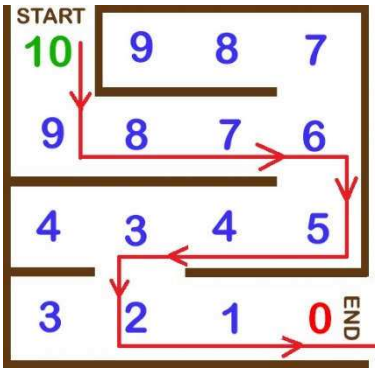


Figure-6.6: Shortest pathfinder

6.3 Application of this project

Nowadays robots are widely used in various critical and dangerous find. This project is based on decision making algorithms. So, it can be used in various intelligent fields. It can be used as a rescue operation, navigation problems, search operation, medical attention, military search and rescue, etc.

There are many caves that are like mazes where humans could get lost in. This robot can find its way out again. It can also be used in too small or dangerous cave where human can't enter.

Not only the caves, but also in a big shopping mall, it's possible that one can lose track of the paths. This robot can be used as a guide in that case too

6.4 Cost Analysis

The total cost of this project is 2090 taka. It is very cheap price for a robot. Mobile robot requires a chassis for assembling all equipment. The cost of the chassis which is used in this project is only 400 taka. But, building a chassis might cost more than 600 to 1000 taka.

The price of Arduino Uno is 550 taka. However, the cost of Arduino can be reduced by using PIC microcontroller. But, the circuit might be more complex and need external cost for run that microprocessor.

Ultrasonic sonar (HR-SR04) has been used in this project. Per sensor cost is only 120taka. But other sensors are more expensive.

Two DC motors with wheels cost only 360 taka. Price of motor controller is 300 taka. AA batteries are used to minimize the cost. It takes only 120 taka.

Table-2: The used parts list and the total cost of the project.

No.	Parts required	Price	Quantity	Total
01	Robot Chassis	400	×1	400
02	DC motor with Wheel	180	×2	360
03	Ultra sonic sonar	120	×3	360
04	Arduino UNO	550	×1	550
05	Motor Controller	300	×1	300
06	Batteries	120	--	120
			Sub Total :	**2090**

Chapter 7

Conclusion And Recommendations

This chapter covers the conclusion and further development of the project.

7.1 Conclusion

As a conclusion, the two mazes solving algorithm have successfully been implemented in the robot and the objectives of the project have been achieved. The first algorithm was wall following algorithm. The basic method shows a good result for solving the maze. But, due to lack of self-intelligence, it failed to solve the maze in the shortest way. And it could not solve to close loop maze. So, an efficient method has been used to find the shortest path that is flood fill algorithm method. After applying all methods, the robot was trained in a real maze. Several tests has been run to ensure the best performance of the robot.

This project helps to improve various important information about robotics, knowledge about many decision making algorithms. It's also helped to learn about many electronics components such as motor driver, sensors, etc. This gained knowledge will have a significant impact on future work.

7.2 Further Development

This robot is little bit slow, so new method should be developed. The size of the robot can be made smaller. In this project, the ultrasonic sensor is used for mapping the maze. Other efficient navigating sensors can be used. At last, it needs development in its wheels and body to make it comfortable in rough surface

Bibliography/ References

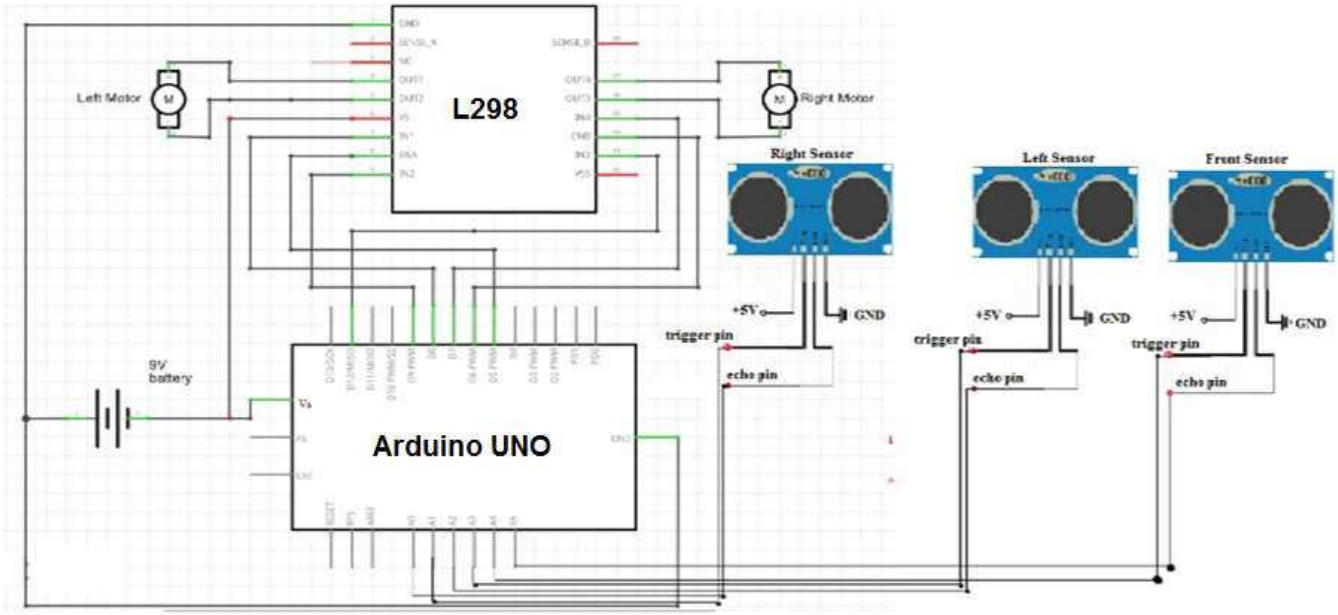
[1] W. Matthews, *Mazes and labyrinths*, 1st ed. New York: Dover Publications, 2013.
[2] Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory, Sadik, A.M.J. *International Conference on Artificial Intelligence and Computational Intelligence*, 10.1109/AICI.2010.18, pp.52-56
[3] B. Clough, "Rooter - A MicroMouse Maze Solving Robot", 2007.
[4] R. Zheng, R. Hill and M. Gardener, *Engaging older adults with modern technology*, 1st ed. Hershey, PA: Information Science Reference, 2013.
[5] "Maze Solvers Archives - cyberneticzoo.com", *cyberneticzoo.com*, 2017. [Online]. Available: http://cyberneticzoo.com/category/mazesolvers/. [Accessed: 12- Apr- 2017].
[6] "Spectral lines: Announcing the Amazing Micro-Mouse Maze Contest - IEEE Xplore Document", *Ieeexplore.ieee.org*, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/6367601/. [Accessed: 12- Apr- 2017].
[7] MICHAEL GIMS, S. L. D. B. 1999. *Micromouse-Microprocessor Controlled Vehicle.* Bachelor of Engineering, University of East London.
[8] CHANG, Y. C. 2009. *Micromouse Maze Solving Robot.* Bachelor of Engineering, Universiti Teknologi Malaysia.
[9] A. Bakarsayutisaman and I. Abdramane, "Solving a Reconfigurable Maze using Hybrid Wall Follower Algorithm," *International Journal of Computer Applications*, vol. 82, no. 3, pp. 22–26, 2013.
[10] M.M Thu, N.N. Win. "Micromouse Maze Solving", *International Journal of Science, Engineering and Technology Research (IJSETR)* Volume 5, Issue 9, September 2016.
[11] M. Kazimierczuk, *Pulse-width modulated DC-DC power converters*, 1st ed. Chichester, West Sussex: Wiley, 2016.
[12] Arduino - PWM", *Arduino.cc*, 2017. [Online]. Available: https://www.arduino.cc/en/Tutorial/PWM. [Accessed: 13- Apr- 2017].
[13] M. McRoberts, *Beginning Arduino*, 1st ed. [Berkeley, Calif.]: Apress, 2013.
[14] R3, S. Edition, S. Edition, A. 5V/16MHz, A. 3.3V/8MHz, S. Arduino, S. (White), A. R3, S. V3.2, S. 101, S. OpenLog and S. Module, "Arduino Uno - R3 - DEV-11021 - SparkFun Electronics", *Sparkfun.com*, 2017. [Online]. Available: https://www.sparkfun.com/products/11021. [Accessed: 13- Apr- 2017].
[15] "4. Arduino Technical Details [Book]", *Safari*, 2017. [Online]. Available: https://www.safaribooksonline.com/library/view/arduino-a-technical/9781491934319/ch04.html. [Accessed: 13- Apr- 2017].
[16] "ATmega328P - Microcontrollers and Processors", *Microchip.com*, 2017. [Online]. Available: http://www.microchip.com/wwwproducts/en/ATmega328P. [Accessed: 13- Apr- 2017].
[17] "Ultrasonic Sensor | What is an Ultrasonic Sensor?", *Education.rec.ri.cmu.edu*, 2017. [Online]. Available: http://education.rec.ri.cmu.edu/content/electronics/boe/ultrasonic_sensor/1.html. [Accessed: 13- Apr- 2017].

[18] "Ultrasonic Sensor - HC-SR04 - SEN-13959 - SparkFun Electronics", *Sparkfun.com*, 2017. [Online]. Available: https://www.sparkfun.com/products/13959. [Accessed: 13- Apr- 2017].
[19] Z. China, "Motor Driver 2A Dual L298 H-Bridge [L298N] - US $4.00 : HAOYU Electronics : Make Engineers Job Easier", *Hotmcu.com*, 2017. [Online]. Available: http://www.hotmcu.com/motor-driver-2a-dual-l298-hbridge-p-45.html. [Accessed: 14- Apr- 2017].
[20] "Bipolar Stepper Motor Driver Using L298", *Archiveclear.jimdo.com*, 2017. [Online]. Available: https://archiveclear.jimdo.com/2016/01/24/bipolar-stepper-motor-driver-using-l298/. [Accessed: 14- Apr- 2017].
[21] "LM7805 Datasheet, PDF - Alldatasheet", *Alldatasheet.com*, 2017. [Online]. Available: http://www.alldatasheet.com/view.jsp?Searchword=Lm7805. [Accessed: 14- Apr- 2017].
[22] Min Raj Nepali, "A Novel Wall Following Algorithm For Mobile Robots", *Cscjournals.org*, 2017. [Online]. Available: http://www.cscjournals.org/library/manuscriptinfo.php?mc=IJRA-126. [Accessed: 19- Apr- 2017].
[23] "Lecture 21: Single Source Shortest Paths - Bellman-Ford Algorithm", *Faculty.ycp.edu*, 2017. [Online]. Available: http://faculty.ycp.edu/~dbabcock/PastCourses/cs360/lectures/lecture21.html. [Accessed: 20- Apr- 2017].
[24] S. Mishra and P. Bande, "Maze Solving Algorithms for Micro Mouse", *2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, 2008.
[25] D.Southall and D. M, "Modified flood algorithm for a maze solving robot ‘mouse’", 2007.

Appendices

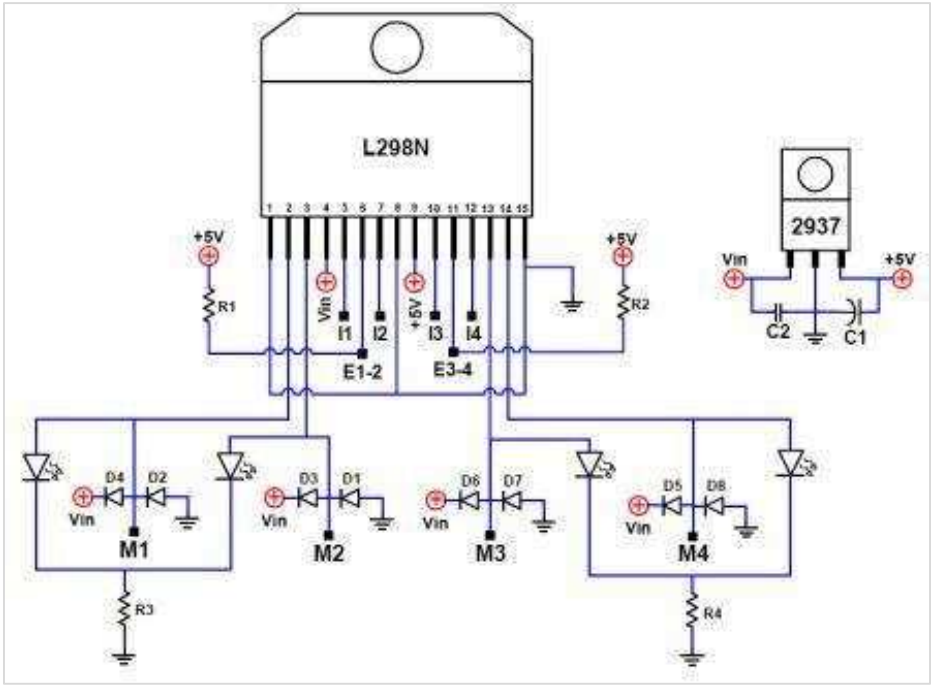
Appendices-1:

Circuit Diagram of full project:



Appendices-2:

Circuit diagram of L298 Motor driver:



Appendices-3:

Arduino Code:

```
//Maze solving Robot;
// Musfiqur Rahman,ULAB;
//email:- musfiqur.rahman.ete@ulab.edu.bd;
//Phone: 01911985323/01611985323;
#define vel_motor_esq 10
#define vel_motor_dir 11
#define e1 8
#define e2 9
#define d1 12
#define d2 7
    int trigger_frente = A4; // front
    int echo_frente = A5; // front
        int trigger_esq = A2;// right
    int echo_esq = A3;// rt
   int trigger_dir = A0;
    int echo_dir = A1
    void setup()
    {
        pinMode(trigger_frente, OUTPUT);
        pinMode(echo_frente, INPUT);
   pinMode(trigger_esq, OUTPUT);
        pinMode(echo_esq, INPUT);
pinMode(trigger_dir, OUTPUT);
        pinMode(echo_dir, INPUT);
               pinMode(vel_motor_esq, OUTPUT);
        pinMode(vel_motor_dir, OUTPUT);
         pinMode(e1, OUTPUT);
        pinMode(e2, OUTPUT);
        pinMode(d1, OUTPUT);
        pinMode(d2, OUTPUT);
        delay(5000);
    }
    void loop()
    {
     long      duracao_frente,      duracao_esq,
duracao_dir, direita, esquerda, frente;
     digitalWrite(trigger_frente, LOW);
     delayMicroseconds(2);
     digitalWrite(trigger_frente, HIGH);
     delayMicroseconds(5);
     digitalWrite(trigger_frente, LOW);
     duracao_frente = pulseIn(echo_frente, HIGH);
     frente = duracao_frente/29/2
     digitalWrite(trigger_esq, LOW);
     delayMicroseconds(2);
     digitalWrite(trigger_esq, HIGH);
     delayMicroseconds(5);
     digitalWrite(trigger_esq, LOW);
     duracao_esq = pulseIn(echo_esq, HIGH);
     esquerda = duracao_esq/29/2;
     digitalWrite(trigger_dir, LOW);
     delayMicroseconds(2);
     digitalWrite(trigger_dir, HIGH);
     delayMicroseconds(5);
     digitalWrite(trigger_dir, LOW);
     duracao_dir = pulseIn(echo_dir, HIGH);
     direita = duracao_dir/29/2;
analogWrite(vel_motor_esq, 0);
     analogWrite(vel_motor_dir, 0); //
```

```
    analogWrite(e1, 0);            //
    analogWrite(e2, 0);            //
    analogWrite(d1, 0);            //
    analogWrite(d2, 0);
      if(frente >8)
        { if(direita >7 && direita< 13)
            {
              analogWrite(vel_motor_esq, 120);
              analogWrite(vel_motor_dir, 150);
          analogWrite(e1, 255);
              analogWrite(e2, 0);
              analogWrite(d1, 0);
              analogWrite(d2, 255);
            }
          if(direita >=13)
            {
              analogWrite(vel_motor_esq, 255);
              analogWrite(vel_motor_dir, 60);
analogWrite(e1, 255);
              analogWrite(e2, 0);
              analogWrite(d1, 0);
              analogWrite(d2, 255);
            }
          if(direita <=7)
            {
              analogWrite(vel_motor_esq, 60);
              analogWrite(vel_motor_dir, 255);
              analogWrite(e1, 255);
              analogWrite(e2, 0);
              analogWrite(d1, 0);
              analogWrite(d2, 255);
           }  }
    if(esquerda <=20 && direita>20 && frente <=8)
dir();
if(esquerda  >20  &&  direita>20  &&  frente  <=8)
dir();
if(direita  <=20  &&  esquerda>20  &&  frente  <=8)
esq();
   if(direita<=20 && esquerda<=20 && frente<=8)
voltar();
    }
    void esq()
      {
        analogWrite(vel_motor_esq, 120);
        analogWrite(vel_motor_dir, 120);
        analogWrite(e1, 0);
        analogWrite(e2, 255);
        analogWrite(d1, 0);
        analogWrite(d2, 255);
        delay(700);
       }
    void dir()
      {analogWrite(vel_motor_esq, 120);
        analogWrite(vel_motor_dir, 120);
        analogWrite(e1, 255);
        analogWrite(e2, 0);
        analogWrite(d1, 255);
        analogWrite(d2, 0);
        delay(800);
      }
    void voltar()
      {
analogWrite(vel_motor_esq, 120);
        analogWrite(vel_motor_dir, 120);
        analogWrite(e1, 255);
        analogWrite(e2, 0);
        analogWrite(d1, 255);
        analogWrite(d2, 0);
        delay(1200);
         }}
```