Faculty of Science + Engineering

Manchester Metropolitan University

Algorithms for Maze Solving Robot

Name	Mohamed Alsubaie
MMU ID	09562211
Supervisor	Dr. Dave Southall
Subject	Project
Unit code	64ET3590
Course	BEng (Hons) Computer and Communication Engineering

Declaration of Originality

I confirm that, unless stated otherwise, this work is the result of my own efforts. These efforts include the originality of written material, diagrams or similar pictorial material, electrical or electronic hardware, computer software and experimental results. Where material is drawn from elsewhere, references to it are included. I am aware that the University penalties for plagiarism can be severe.

Signed: ______________________________________
Date: ______________________________________

Project Title: Algorithms for Maze Solving Robot
Done By: Mohamed Alsubaie
Supervisor: Dr. Dave Southall
Date: March 2013

Abstract

The problem of solving a maze is approximately 30 years old. However, it is still considered as an important field of robotics. It is based on one of the most important areas of robot, which is “Decision Making Algorithm”. This project is going to cover two well known maze solving algorithm which are Wall Following Algorithm and modified Fill Flood Algorithm. It will involves designing a programme capable of showing a simulation result for all the steps required to solve a maze, as well as building a robot and test it on 3x3 maze.

Acknowledgment

Firstly, I would like to express my gratitude to God for providing me health, patience and knowledge to successfully complete this project. I submit my highest appreciation to my supervisor Dr. David Southall for his beneficial advices, support and guidance throughout this project. A special thank for all academic staff whose makes me reach this level of education. Thanks to all technicians staffs for providing me the required equipments and component. I would also like to thanks the Ministry of Education, Kingdom of Bahrain for there sponsorship and for providing me the opportunity to completing my studies in a good university. Finally, thanks to my parents, family and friends for their help, encouragement and guidance, who without I would not be where I am today.

Table of Contents

Chapter *Page Number*

List of Figures

List of Tables

Report Structure

Chapter 1 provides an introduction to the project definition as well as specifies the aim and objectives.

Chapter 2 provides the important background information about the history of micro-mouse robot and micro-mouse competition.

Chapter 3 builds the foundation for this project. It covers important theories, methods and algorithms that will helps in design and building the robot.

Chapter 4 provides some important information about all the hardware and software that will be used to implement this project.

Chapter 5 describes the used algorithms as well se the design and architecture of the robot.

Chapter 6 demonstrates the results of the simulation and experimental result.

Chapter 7 discuss the obtained result and summarised the main findings.

Chapter 1

Introduction

This chapter is going to provide an introduction about this project. It will start with general information about autonomous robots including the Maze Solving Robot, and it will also cover the aim and objectives for the project.

Chapter 1: Introduction

Autonomous robots are robots that can perform tasks intelligently depending on themselves, without any human assistance. Maze Solving Robot, which is also called "Micro-Mouse 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 centre area of the maze in the shortest possible time. This robot attracted people from all phases of life. Micro-Mouse Competitions have been taken a place for about 3 decades. There are many algorithms and techniques have been discovered and used to solve the maze.

1.1 About the Project

This project will cover two popular maze solving algorithms. It will start with implementing the basic Wall Follower Algorithm, showing the advantages and weakness of this technique. Then, it is going to show the ability of Fill Flood Algorithm to tackle these problems. In addition, a robot will be tested on a small maze by applying the fill flood algorithm.

1.2 Aims

- The aim of this project is to deign algorithms capable of leading a robot to solve a maze.

1.3 Objectives

- Understand and implement the wall follower algorithm.
- Understand and implement the modified fill flood algorithm.
- Use multi-dimensional array for storing the maze walls.
- Apply some navigation and mapping strategies to show a simulation results for the full process involves in solving the maze.
- Design and build a sensor board.
- Build a small maze.
- Programme a robot to solve the maze using fill flood algorithm.

Chapter 2

Background

This chapter is going to provide important background information about the history of micro mouse robot. In addition, there will also be some useful information about micro-mouse competition history and rules.

Chapter 2: Background

2.1 Micro Mouse Robot

The problem of solving a maze is approximately 30 years old. However, it is still considered as an important field of robotics. It is based on one of the most important areas of robot, which is “Decision Making Algorithm”. The robot will be placed in unknown environment, and it requires having a good decision making capability. The development of micro-mouse algorithms improved gradually, starting from a basic wall follower up to complicated levels as Flood fill algorithm (Sharma and Robeonics, 2009). Early micro-mouse designs are used to have huge physical shapes that contain many blocks logic gates. Figure-1 shows an example of how micro-mouse robots were look like in the late 1970s. Due to the technology development, the physical shape of the new generation of micro-mouse robot becomes mush smaller than before.

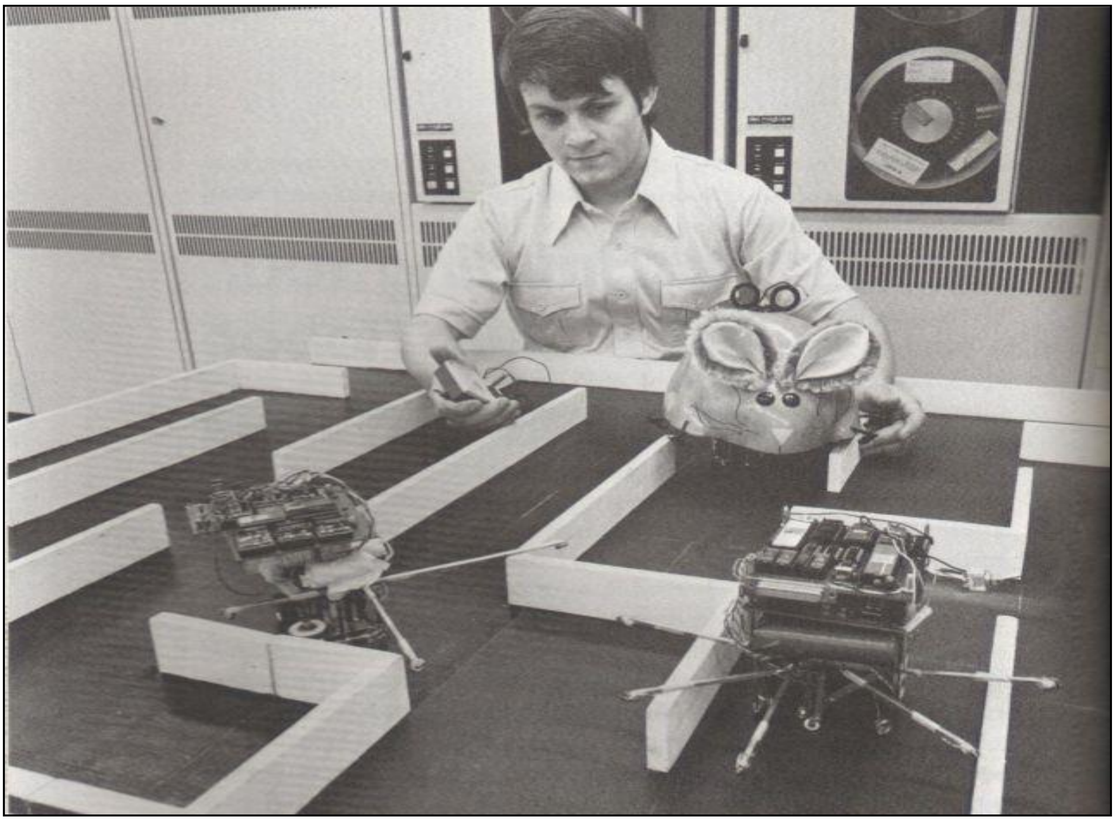


Figure-1: Old generation of micro-mouse robots (Cyberneticzoo, 2010).

2.2 Micro Mouse Competitions

There are many competitions based on autonomous robots that inspired engineers from all around the world. On of the most popular competitions is micro-mouse. This competition has been organised since the 1970s in many different countries. It has some slight changes since it's begin. As it shown in figure-2, there is a wooden maze consist of 16 x 16 grid of cells. The robot must find it path to the middle area of the maze from a given starting position in less than 10 minutes. In addition, the robot is required to meet the competition rules. The robot width and length should be no more than 25cm, while there are no boundaries for the robot height (IEEE UCSD, 2009).

Figure-2: Micro-mouse competition in United State (IEEE UCSD, 2009).

Chapter 3

Literature Review

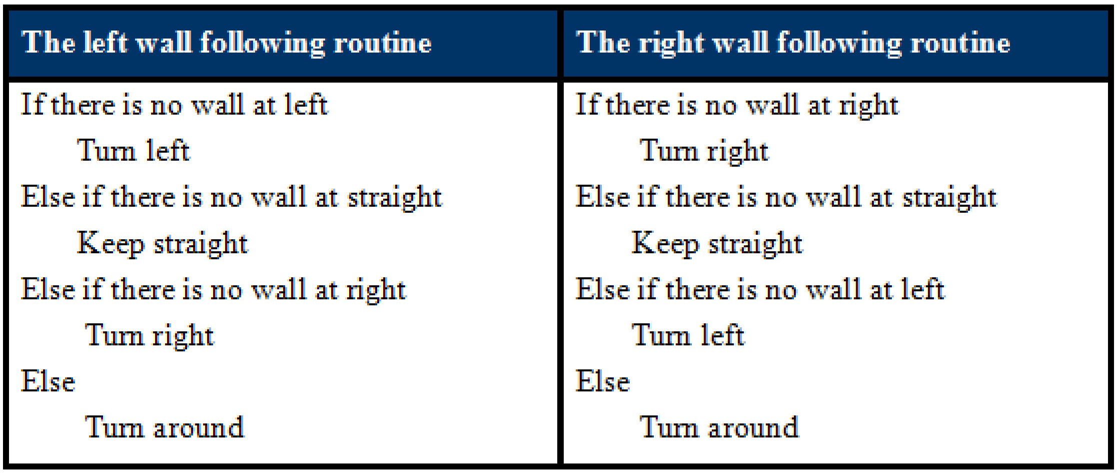
This chapter builds the foundation for this project. It is going to cover many important theories, method and algorithms that will helps to complete design and build the robot. It will start with providing some research about Wall Following Algorithm and Fill Flood Algorithm. Then, it is going to provide essential theories for controlling robot movement, direction and speed.

Chapter 3: Literature Review

3.1 Wall Following Algorithm

The wall following algorithm is one of the simplest techniques that can be used to solve a maze. Essentially, the robot will take its direction by following either left or right wall. 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. The instructions involved in following wall for both left and right is given in table-1 below. However, this algorithm is no longer considered to be an efficient method in solving the maze. This because the wall follower algorithm will fail to solve some maze construction, such as a maze with a closed loop region (Mishraand and Bande, 2008).

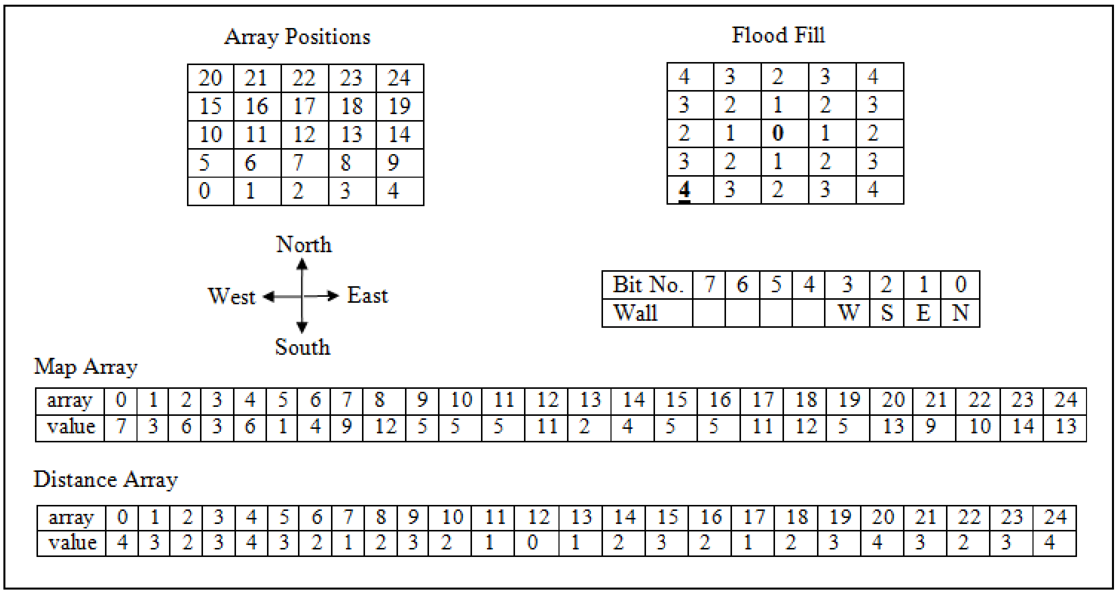
Table-1: Left and right wall following routine.



The left wall following routine	The right wall following routine
If there is no wall at left Turn left Else if there is no wall at straight Keep straight Else if there is no wall at right Turn right Else Turn around	If there is no wall at right Turn right Else if there is no wall at straight Keep straight Else if there is no wall at left Turn left Else Turn around

3.2 Fill Flood Algorithm

This is one of the most efficient maze solving algorithms. The fill flood algorithm is derived from the “Bellman Ford Algorithm”. It paved new methods by which complex and difficult mazes can be solved without having any troubles. 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 (Mishra and Bande, 2008). Implementing this algorithm requires to have two arrays. The first array is holding the walls map values, while the other one is storing the distance values (Southall, 2007).



Array Positions

20	21	22	23	24
15	16	17	18	19
10	11	12	13	14
5	6	7	8	9
0	1	2	3	4

Flood Fill

4	3	2	3	4
3	2	1	2	3
2	1	**0**	1	2
3	2	1	2	3
4	3	2	3	4

Bit No.	7	6	5	4	3	2	1	0
Wall					W	S	E	N

Map Array

array	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
value	7	3	6	3	6	1	4	9	12	5	5	5	11	2	4	5	5	11	12	5	13	9	10	14	13

Distance Array

array	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
value	4	3	2	3	4	3	2	1	2	3	2	1	0	1	2	3	2	1	2	3	4	3	2	3	4

Figure-3: Fill flood algorithm using one-dimensional arrays (Southall, 2007).

Every time the mouse arrives in a cell it will perform the following steps:

(1) Update the wall map.

(2) Flood the maze with the new distance values (only if necessary).

(3) Decide which neighbouring cell has the lowest distance value.

(4) Move to the neighbouring cell with the lowest distance value.

Some recent researches suggested a new method to improve and minimise the number of arrays, using a single multi-dimensional array. This is the method that is going to be used in this project; therefore all the results will be stored in only one multi-dimensional array.

3.3 Pulse Width Modulation

Pulse Width Modulation “PWM” is a simple method of controlling analogue devices via a digital signal through changing or modulating the pulse width. This is a very clever way to use electricity. By powering an analogue device with a pulse wave switches between on “5V” and off “0V” at a fast rate the device will behave as it getting a steady voltage some where between 0V and 5V (McRoberts, 2010). Figure-4 shows an example of a PWM with a duty cycle of 25%, which is gives a voltage value of 1.25V. The term of duty cycle describes the amount of time in the period that the pulse is on or high. It is typically specified as a percentage of the full period. 100% duty cycle means that the PWM wave is fully on.

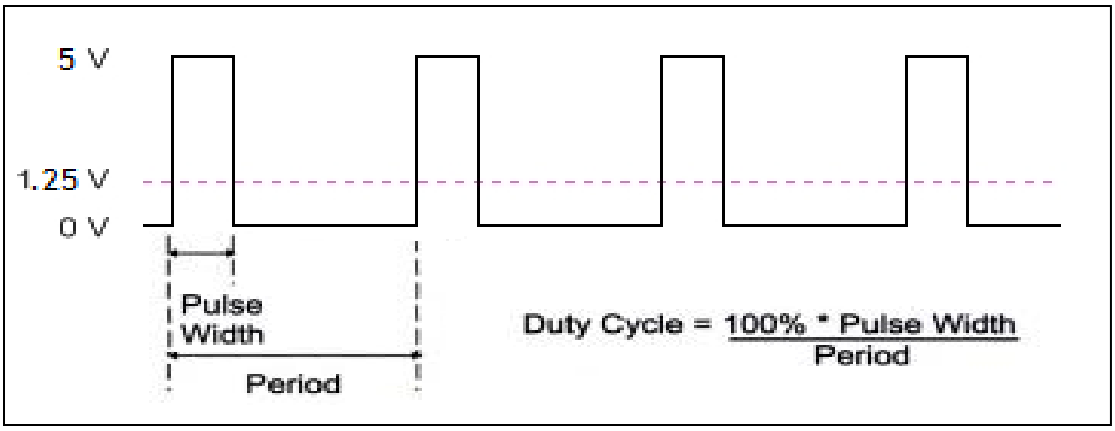


Figure-4: An example of a PWM of 25% duty cycle.

The main advantage of PWM is that power loss in the switching devices is very low. Power loss, is the product of voltage and current, which is in this case close to zero. Where other methods can cause to waste more power, such as using resistor will lead to loss the power in form of heat. Therefore, PWM is an efficient way to driving motors, heaters, or lights in varying intensities or speeds.

3.4 H-Bridge

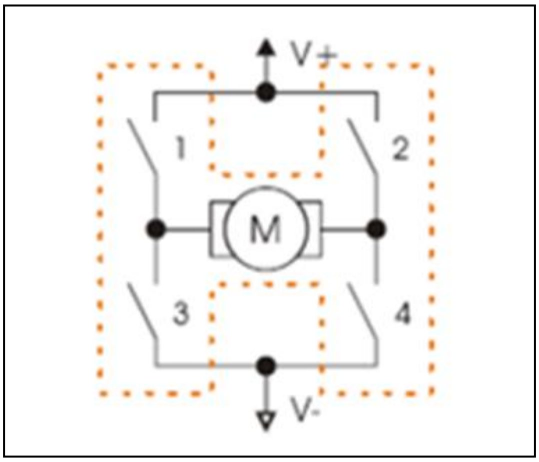


Figure-5: H-bridge with four switches (McRoberts, 2010).

In order to change the direction of a motor, the direction of current flows and the polarity of the voltage of the motor terminal should be reversed. To achieve this, four switch need to be connected in the following arrangement shown in figure-5, and for obvious reasons it called H-bridge (Warren, Adams and Molle, 2011). Two switches controls the path of the current from the positive power supply to both motor terminals, and the other two switches controls the path of the current from the negative power supply to both motor terminals (McRoberts, 2010).

To make the motor turn in forward direction, both S1 and S4 should be closed with opining S2 and S3 as it shown in the first configuration in figure-3. On the other hand, reveres the configuration with closing S2 and S3 and opining S1 and S4 as in the second configuration will make the motor turn in the reverse direction. The third and forth configuration shown in the figure below is about creating an electric brake, which is going to stop the motor from turning in any direction.

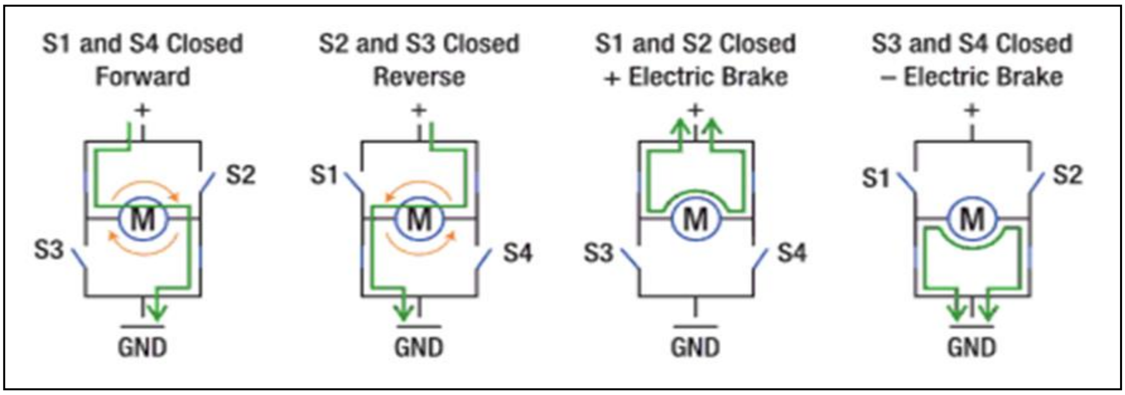


Figure-6: Acceptable H-bridge states (Warren, Adams and Molle, 2011).

However, it can be easily notice that there is a danger in this circuit. As it shown in figure-7, if S1 and S3 have been closed, there will be short-circuit as the positive power supply will connected with the negative power supply, and that causes circuit damage, overheating, fire or explosion. The same thing will apply with closing both S3 and S4 at the same time (Warren, Adams and Molle, 2011).

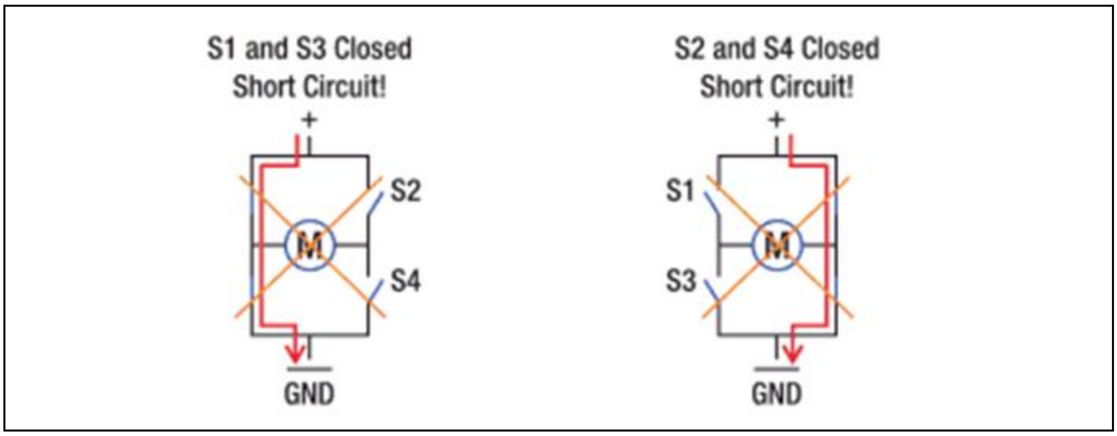


Figure-7: Shoot-through H-bridge states (Warren, Adams and Molle, 2011).

3.5 Back EMF

Motor starts spinning when it powered up. However, when the power removed, the motor will not stop immediately, it will keep spinning under it own inertia until comes to a stop. During that time while the motor is turning without an applied power, an electric current is generating which is known as “Back EMF”, that can makes a series problem and damage the micro-controller, as well as the other component (McRoberts, 2010).

Diodes are used to protect the circuit from the reverse voltage “back EMF”. They work as a valve by allowing a current to flow in one direction but not the other. Therefore, it is necessarily to place them in the circuit around the areas which have a danger of the power being reversed either by user error or via phenomena such as back EMF (McRoberts, 2010).

Chapter 4

Hardware and Software Description

This chapter is going to provide important information about all hardware and software that will be used in this project. It is going to give some details for each component including features, specifications and how it operates.

Chapter 4: Hardware and Software Description

4.1 Arduino Uno

The Arduino Uno is a microcontroller board that contains everything needed to support the microcontroller, see figure-8. It is made up of an Atmel AVR Microprocessor “ATmega328”, a 5V linear regulator, a power jack, a USB connection, an ICSP header, a reset button, a 16 MHz crystal oscillator which is enable the Arduino to operate at the correct speed by sending the right time pulses with specified frequency, and an Atmega8U2 that programmed as a USB-to serial converter (McRoberts, 2010). The important features are summarized in table-2 below.

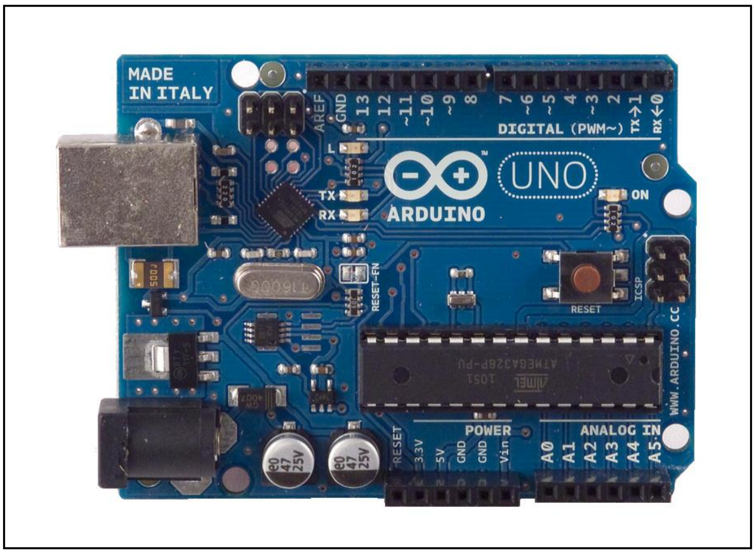
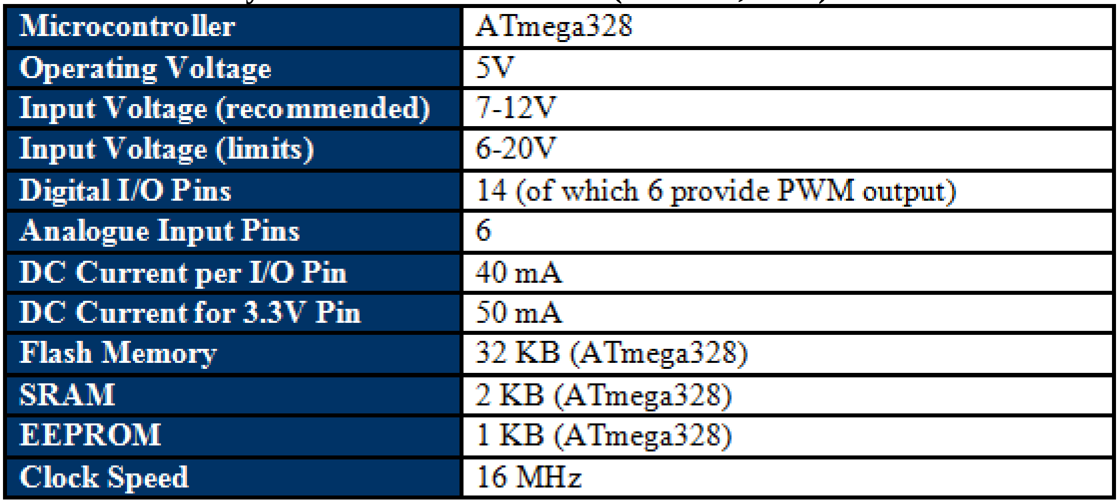


Figure-8: Arduino Uno (Arduino, 2012).

Table-2: Summary of Arduino Uno features (Arduino, 2012).



Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analogue Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

4.2 Motor Driver

There are many different ways that can be used to drive a DC motor. Due to the reason of having many shields that are compatible with Arduino, a motor driver shield which is called “Ardumoto” has been selected for this project. As it shown in figure-9, Ardumoto comes fully assembled, and that makes it very easy to use, simply plug it directly into the top of the Arduino. This shield can control the speed and direction of two motors. It is based on the L298 dual full h-bridge driver, which is able to handle a total DC current up to 4A “2A per each channel”. There are a blue and yellow LEDs connected in both output channels in order to indicate the active direction. All driver lines are diode protected from back EMF (Makezine, 2009).

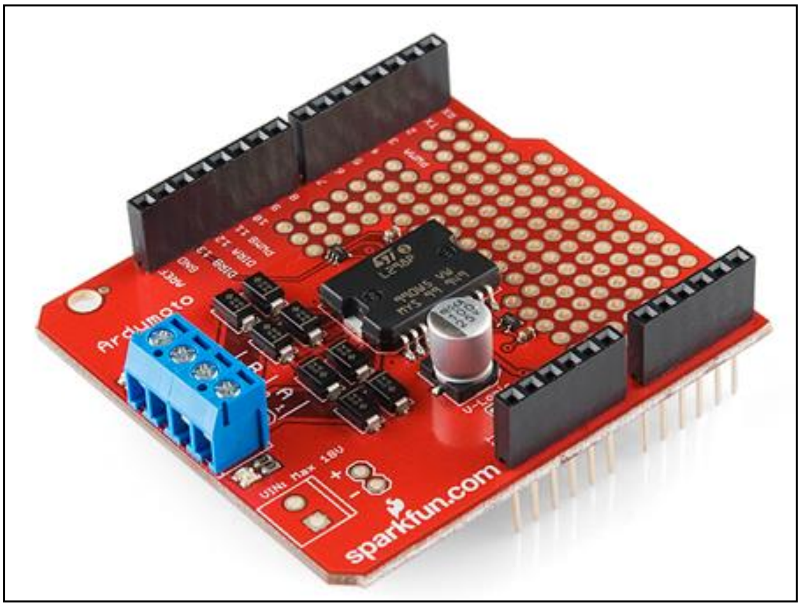


Figure-9: Ardumoto Shield (Makezine, 2009).

4.3 Reflective Sensor

The RPR-220 is a common reflectance sensor often used in sensing objects for short distance between 1cm and 5cm. In this project, this sensor will be used to detect the maze walls. As shown in figure-10, RPR-220 sensor is comprised of two parts, an infrared emitted diode and an infrared phototransistor. The sensor works by shining an IR emitted diode down and measuring how much of that light reflected back to the IR phototransistor, see figure-11.

Figure-10: RPR220 reflective sensor.

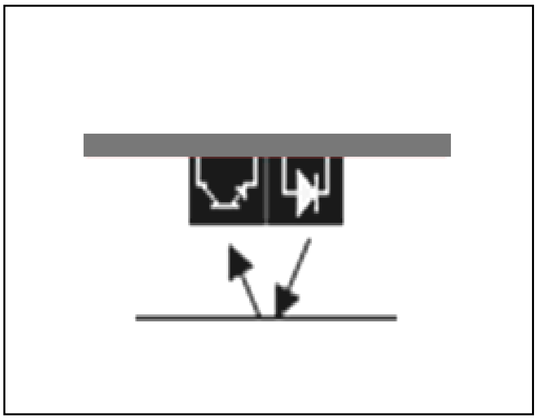


Figure-11: IR reflective sensor construction and operation.

4.4 Remote Control

A remote control is a well known electronics component that can control and operate many modern devices wirelessly from a short line of sight distance, usually a television, DVD players, stereo systems and even laptops. As it shown in figure-12, remote control is a small handheld object with an array of buttons allows a variety of selections. The remote control handset transmit a different pulses of infrared light depending on the pressed button through a light emitting diode “LED”, which is placed on the top of the remote control. This infrared light has a wavelength of 940 nm and it is invisible to the human eye, but it can be visible in a purple light on camera (Monk, 2010).

Figure-12: IR remote control

4.5 IR Receiver Sensors

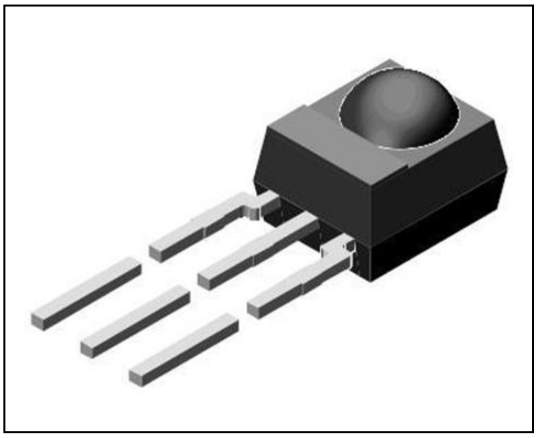


Figure-13: TSOP34840 IR receivers module (Vishay, 2011).

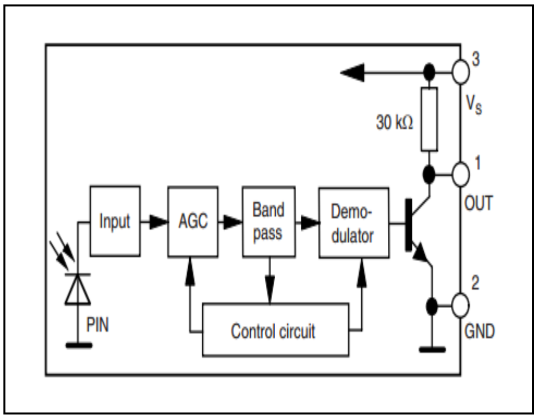


Figure-14: Block diagram of TSOP34840 (Vishay, 2011).

IR receiver sensor is a tiny microchip with a photocell that able to detect infrared light. Every television and DVD player has one these sensors in the front to detect the infrared signal transmitted from the remote control. Figure-14 shows a block diagram of the TSOP34840 IR receiver module that combines an infrared photodiode, with all the amplification, filtering, smoothing, and demodulation to produce a digital output that can be decoded through a microprocessor and carries out the command. As it shown in the datasheet graphs provided in appendix-2, the peak frequency detection is at 38 KHz with a range from about 35 KHz to 41 KHz but the sensitivity will drop off so that it will not detect as well from afar. Moreover, the peak LED wavelength is 940 nm with a sensitivity range from 800nm to 1100nm (Vishay, 2011).

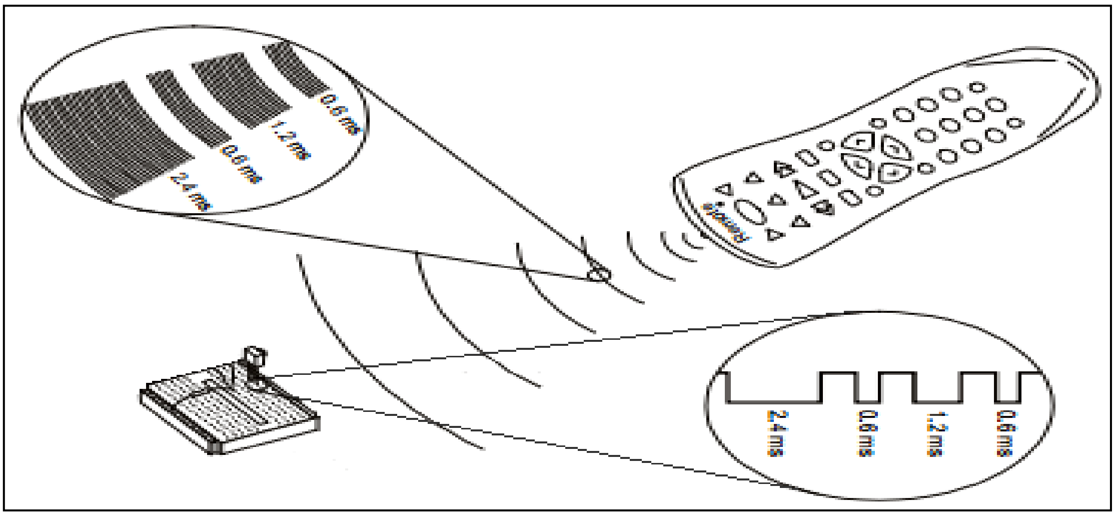


Figure-15: Receiving the transmitted IR Pulses (Monk, 2010).

4.6 Robo Jr Chassis

Robo Jr is a low cost chassis that has all the features required to build a robot. It is provided with two geared dc motor, wheels, battery compartments as well as provision for circuit board mounting. The Robo Jr body is a suitable solution to use for robotics project because it allows testing the controller design and give a fast prototyping for the robot performance (Active-Robot, 2012).

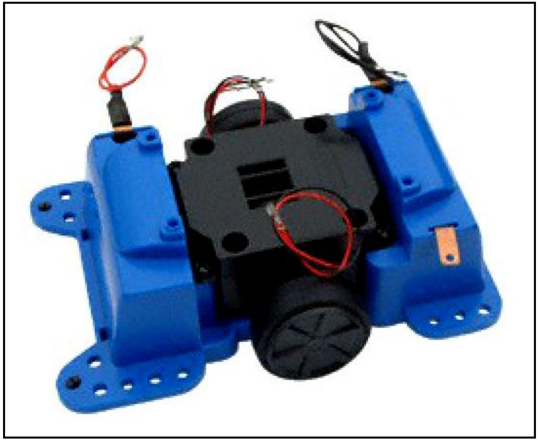
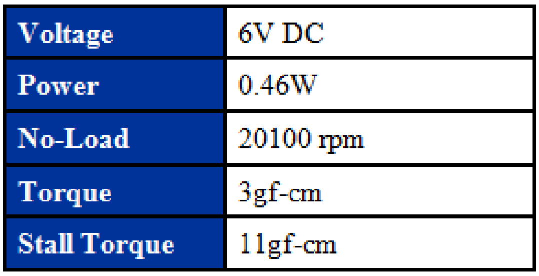


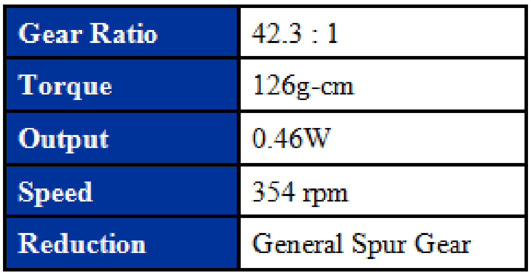
Figure-16: Robo Jr chassis (Active-Robot, 2012).

Table-3: Motor specifications.



Voltage	6V DC
Power	0.46W
No-Load	20100 rpm
Torque	3gf-cm
Stall Torque	11gf-cm

Table-4: Gear box specifications.



Gear Ratio	42.3 : 1
Torque	126g-cm
Output	0.46W
Speed	354 rpm
Reduction	General Spur Gear

4.7 Arduino IDE

Arduino boards are programmed by using an Arduino Integrated Development Environment, which is free software that can be downloaded from the Arduino website. The C programming language is used to write a computer programme, which is a set of step by step instructions that it to convert to a machine code “hex file” and then upload to the Arduino board using a USB cable. It also support a serial communication, which is provides the ability to see the results of the inputs and outputs of the microprocessor, see figure-18. Moreover, Arduino provides a good opportunity to learn and shear ideas because it is an open-source. This is means that examples, codes and libraries can be taken freely by anyone to do what they like with them. As well as allowing people to learn, help and discuss with each others through the Arduino website.

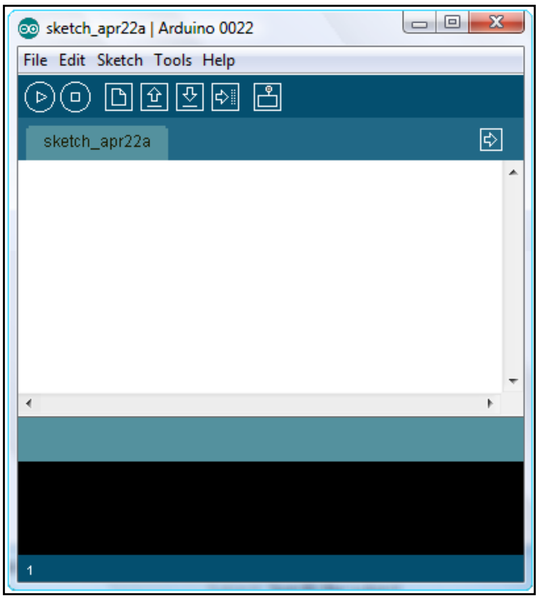
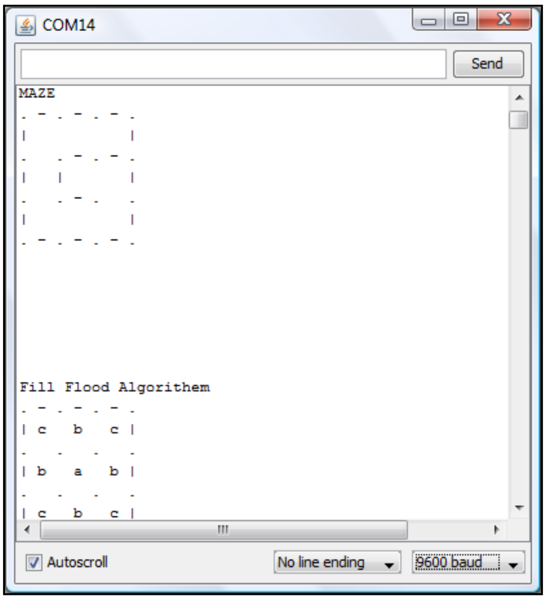


Figure-17: Arduino IDE.



```
MAZE
. - . - . - .
|           |
.     . - . - .
|     |     |
.     . - .     .
|           |
. - . - . - .




Fill Flood Algorithem
. - . - . - .
| c   b   c |
.   .   .   .
| b   a   b |
.   .   .   .
| c   b   c |
```

Figure-18: Serial monitor.

4.8 Fritzing

Fritzing is a good circuit design programme that allows users to document their work in a very nice illustration of who the circuit well look like on a bread board, schematic, as well as PCB (Fritzing, 2011). There are many libraries consist of parts and components including Arduino boards. Fritzing is an open sours programme; therefore people are able to shear there works and designs with others and participate to help each others (Warren, Adams and Molle, 2011).This software used in this project to draw the initial circuit which is provided in appendix-2.

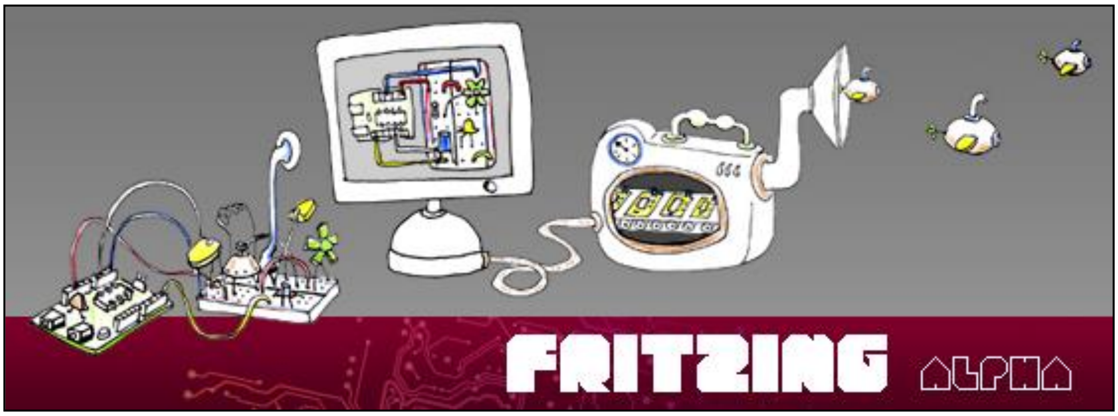


Figure-19: Fritzing software (Fritzing, 2011).

Chapter 5

Method

Before designing and building the robot, it is necessarily to understand and implement a suitable maze solving algorithm. This project will focus on two popular algorithms, which are wall following algorithm and fill flood algorithm. The first section of this project will be about making a programme capable of showing a simulation results for both algorithms, while the other section will be about the implementation of the robot.

Chapter 5: Method

5.1 Simulation Programmes

5.1.1 Simulation Block Diagram

This is a block diagram for the simulation programmes. It only requires an Arduino board and any PC or laptop that has Arduino IDE on it. The Arduino board will generate a simulation results in a multi-dimensional array. Theses result will be sent via USB cable to the computer. Then, the results will appear on the serial monitor, which is provided in the Arduino software.

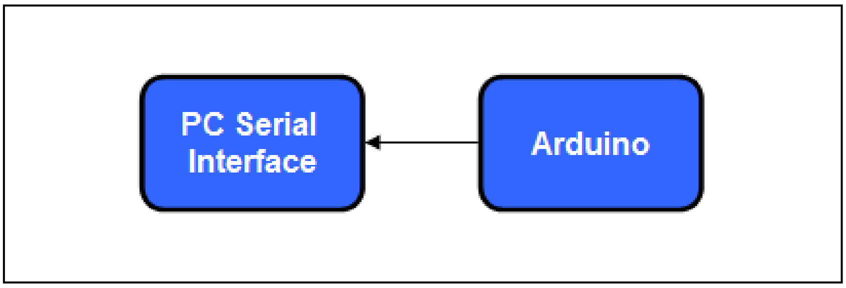


Figure-20: Simulation results block diagram.

5.1.2 Scanning Routine

Each simulation programme will have two multi-dimensional arrays. The first array is holding the values of the test maze, while the second array is representing the maze on the robot brain. As shown in the example below, as long as the mouse "M" is moving it is going to scan the walls of the test maze to its brain. The following code has been written to do the scanning routine. Simply, it copies the test maze.

```
Mouse_Maze[yp-1][xp] = Maze[yp-1][xp];
Mouse_Maze[yp][xp-1] = Maze[yp][xp-1];
Mouse_Maze[yp][xp+1] = Maze[yp][xp+1];
Mouse_Maze[yp+1][xp] = Maze[yp+1][xp];
```

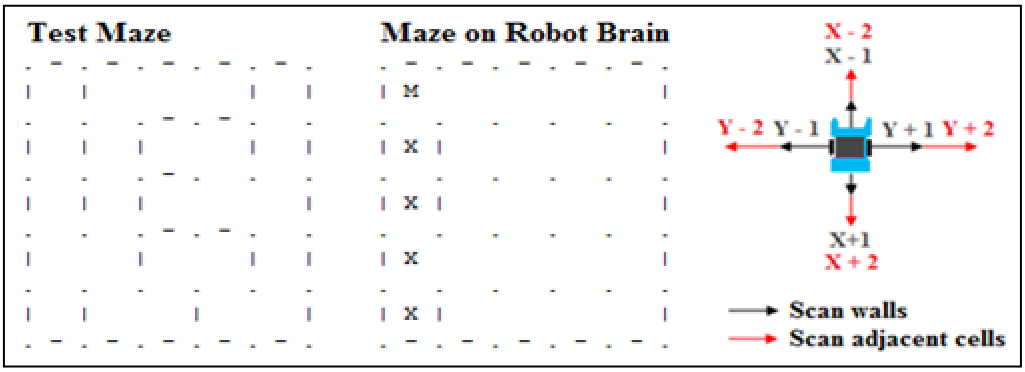


Figure-21: Scanning routine in the multi dimensional array.

In order to get a better looking simulation results, characters values has been used in the multi-dimensional arrays.

5.1.3 Left Hand Wall Algorithm

The left hand wall has been selected, because implementing wall following algorithm required selecting either left or right wall to be followed. The figure below shows the flow chart of the programme. It will start with insert characters '"M", which is represent the current position of the mouse (or robot), and print the first step. Then it will enter the main loop. As long as the robot dose not reaches the goal, it will scan the "test maze" walls and take the suitable direction. The priority in left hand wall is in the following orders: left, straight, right, or u-turn. Before going to the selected direction, the current cell will replaced with "X" and the new cell will replaced with "M". After that, the maze will printed over and over with showing the mouse steps until it reaches to the centre of the maze.

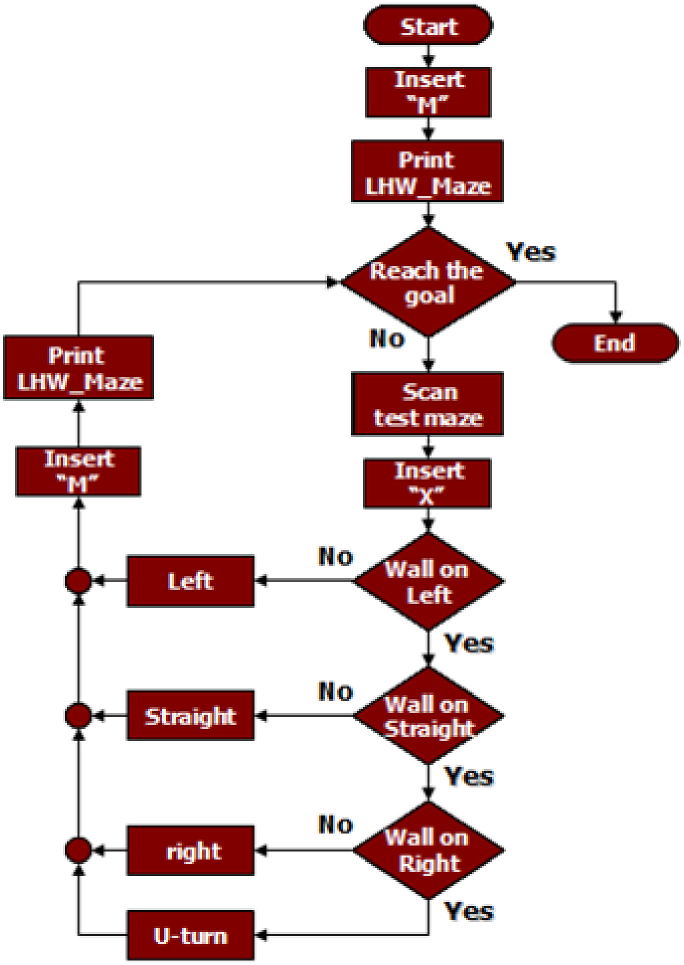
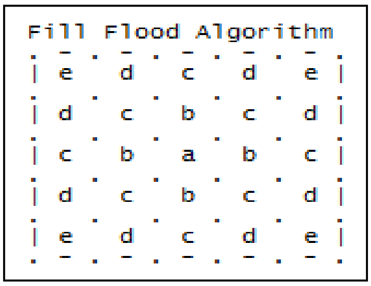


Figure-22: Left hand wall flow chart.

5.1.4 Modified Fill Flood Algorithm

As discussed in chapter 3, the fill flood algorithm works by assigning value for all cells in the maze, where these values indicate the steps from any cell to the destination cell. In figure-3 the array was holding integers values, which will not provide the best look for the simulation results. Therefore, the multi-dimensional array filled with characters informs of aliphatic letters. In the programme character value can be traded as a number using the characters table from Arduino website. For example "a" is equal 97, which should be assumed as zero.



```
Fill Flood Algorithm
. - . - . - . - . - .
| e   d   c   d   e |
. . . . . .
| d   c   b   c   d |
. . . . . .
| c   b   a   b   c |
. . . . . .
| d   c   b   c   d |
. . . . . .
| e   d   c   d   e |
. - . - . - . - . - .
```

Figure-23: Filling the maze with destination values

The flow chart of the algorithm can be seen in figure-24. The programme can be divided into three main parts.

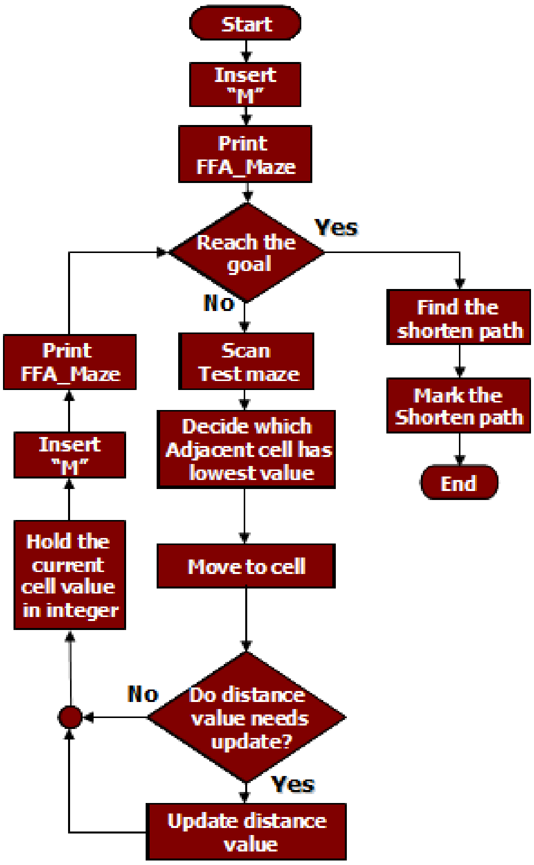


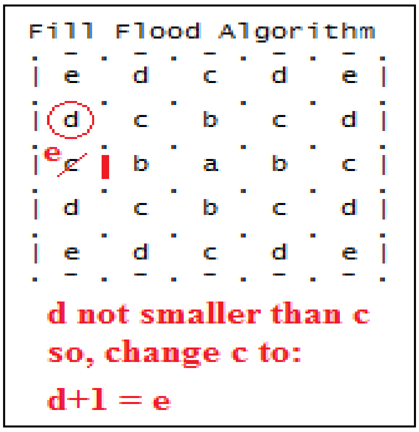
Figure-24: Fill flood flow chart.

- **<u>Part 1: Solve the maze.</u>**

The mouse will start with the first step and then it will enter "solving the maze" loop. This loop will be repeated as long as the mouse not reaches the goal.

In this loop, the mouse will scan the walls of the test maze and decide which adjacent cell has the lowest value.

Then, the mouse will move to the cell and check if it necessarily to update the distance value. Figure-25 is given an example about the updating the cell value. If the adjacent cell value is **More** than the current cell value, [current value = adjacent cell value +1].



```
Fill Flood Algorithm
. - . - . - . - . - .
| e   d   c   d   e |
. . . . . .
| d   c   b   c   d |
. . . . . .
| e c | b   a   b   c |
. . . . . .
| d   c   b   c   d |
. . . . . .
| e   d   c   d   e |
. - . - . - . - . - .
```

d not smaller than c
so, change c to:
d+1 = e

Figure-25: updating the cell value.

After that, the cell value will store in integer “current value” and it will replaced in the maze with character “M” to show the mouse movements. The FFA Maze will printed over and over with showing the mouse steps until it reaches to the centre of the maze. The following code explains how it written on the programme:

```
//set all direction values to high number, 125
//scan test maze walls
//scan adjacent cells
if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}

//select the lowest adjunct cell
if(S_value <= L_value && S_value <= R_value && S_value != 125){direc = 'S';}
else if(L_value <= S_value && L_value <= R_value && L_value != 125){direc = 'L';}
else if(R_value <= S_value && R_value <= L_value && R_value != 125){direc = 'R';}
else{direc = 'D';}

//update value if needed
if(S_value > current_value)
{
    current_value = S_value + 1;  //current value = adjacent cell value +1
}
FFA[yp][xp] = current_value;  //return the actual cell value, the one replaced with “M”
yp = yp - 2;                  //move to cell

current_value = FFA[yp][xp];  //store the new cell value
FFA[yp][xp] = 'M';            //replace it with “M”
PrintFFA();                   //Print the maze
```

➢ **Part 2: Find the shorten path.**

In this process the mouse position will be on the centre of the maze and it should return back to starting position. This process is almost the same as the process of solving the maze process. The only different is, if the adjacent cell value is **Less** than the current cell value, [adjacent cell value = current value +1]. The maze will printed until the mouse reaches the starting position. After completing this process, the FFA Maze will have new values representing the right orders of steps required to reach the centre of the maze.

➢ **Part 3: Mark the shorten path.**

This process is going to mark the shorten path. Since the FFA Maze has the right steps values, marking the shorten path will be straight forward through repeating solve the maze process. Each step the mouse takes will be marked with character “X”.

5.2 Robot Implementation

5.2.1 Robot Block Diagram

The following block diagram gives a quick description of robot design. A 6 volt (4xAA batteries from the chassis) will be provided directly the motors and the Arduino board. Then, the voltage regulator on the Arduino board will regulate the voltage to 5 volts and feed the IR receiver, reflective sensors LEDs, and the motor controller chip L298. There are three reflective sensors used for detecting the maze walls and provide a feed back to the Arduino microcontroller. Motor controller shield will control the speed and direction of the left and the right motor taking it orders from the Arduino. Moreover, an IR remote control is used to make the robot proceed to the next stage through sending an infrared light with different pulses depending on the selected button. TSOP34840 IR receiver will receive these pulses and will produce a digital output that can be decoded by the Arduino. In addition, different colours of LEDs are used to gives more user interface.

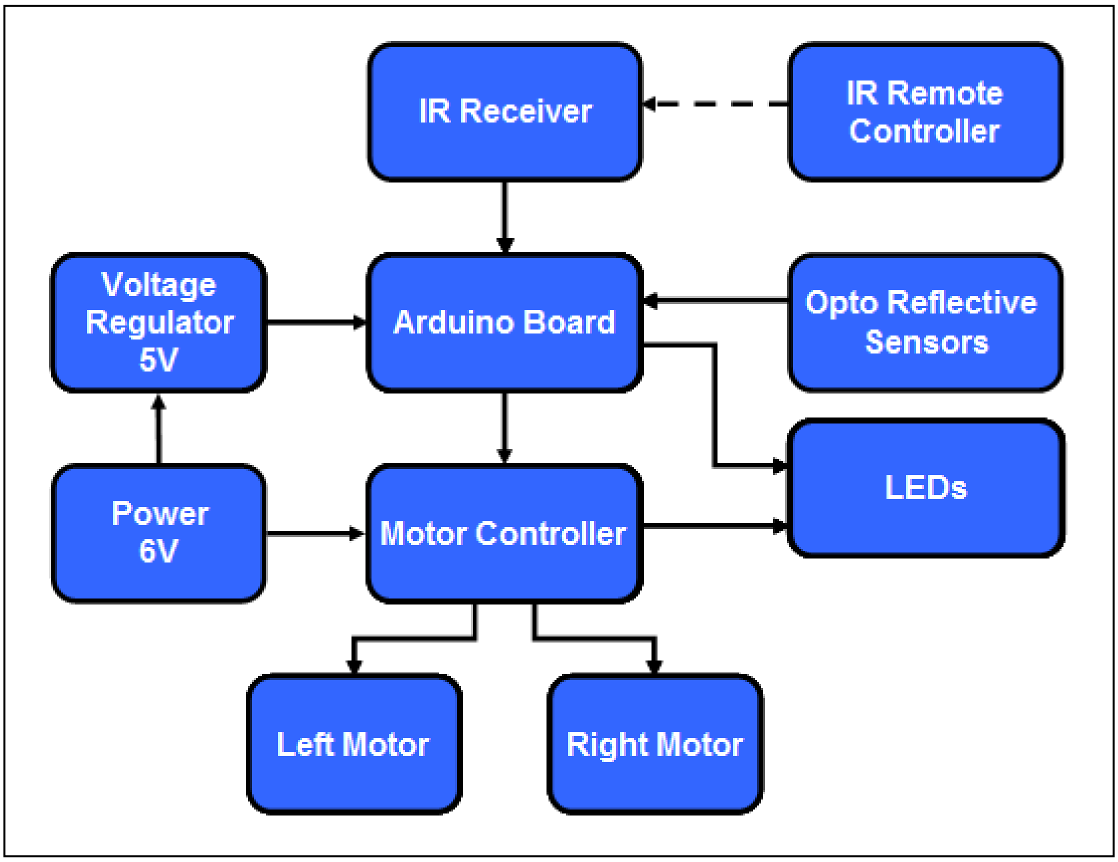


Figure-26: Block diagram for the maze solving robot.

5.2.2 Circuit Diagram

This is the circuit diagram for the maze solving robot. It has a very simple connection. The motor shield has been plugged on the top of the Arduino board and its outputs have been connected to the DC motors. Then, IR receiver has been connected to digital pin number 3. Finally the sensor board attached to the Arduino board. The sensor board contains three RPR220 infrared reflective sensors each with a current limiting resistor for infrared diode as well as pull up resistor for the phototransistor. The full schematic diagram of this project is included in appendix-1.

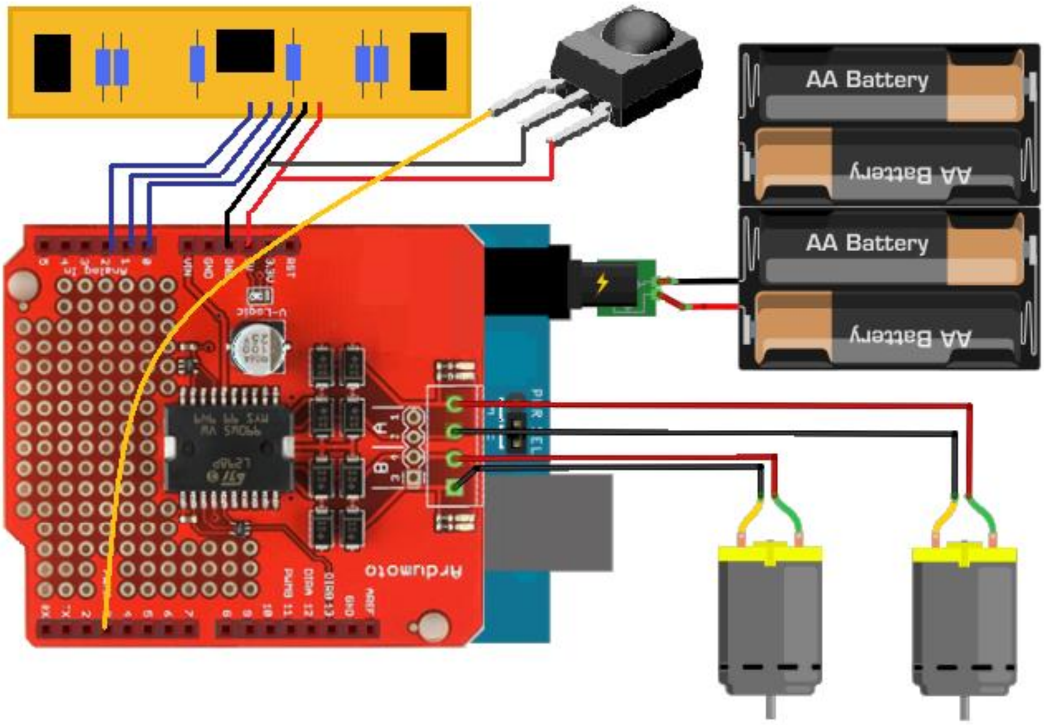


Figure-27: Circuit diagram for the maze solving robot.

5.2.3 Robot Physical Configuration

Figure-28 shows the physical configuration of the robot. Arduino and motor controller shield has been placed at the centre of the robot chassis, in order to have a good balance. Moreover, the sensor board, which is containing of reflective sensors located on the right, straight and left has been attached at the front of the robot to sensing the maze walls. In addition the infrared receiver has been placed toward the top to increase the detection area.

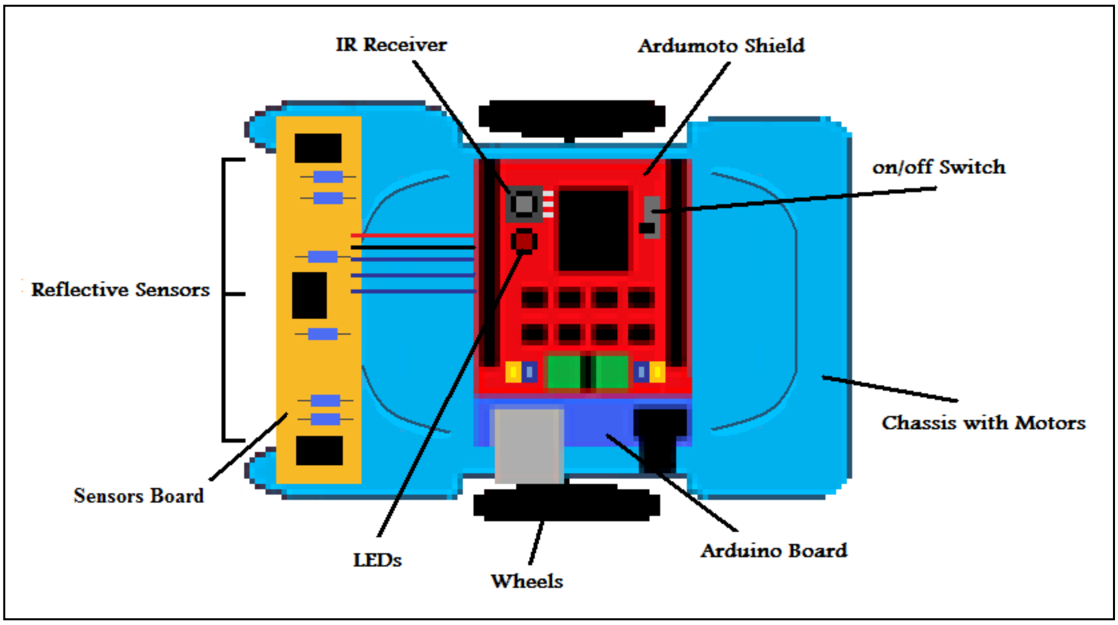


Figure-28: Maze solving robot physical configuration

5.2.4 Robot Programme

The same programme for fill flood algorithm has been used with some additional instructions to makes the robot work.

1. The programme required to read the walls using the sensors. This could be done using the analogue to digital converter on the Arduino.

```
//Read rensors
left_sensor = analogRead(0);
front_sensor = analogRead(1);
right_sensor = analogRead(2);
```

2. The programme needs to have instructions that will drive the motors. The following code is an example for moving forward:

```
//set both motor high
digitalWrite(dir_L, HIGH);
digitalWrite(dir_R, HIGH);
speed_L = 60;
speed_R = 60;
```

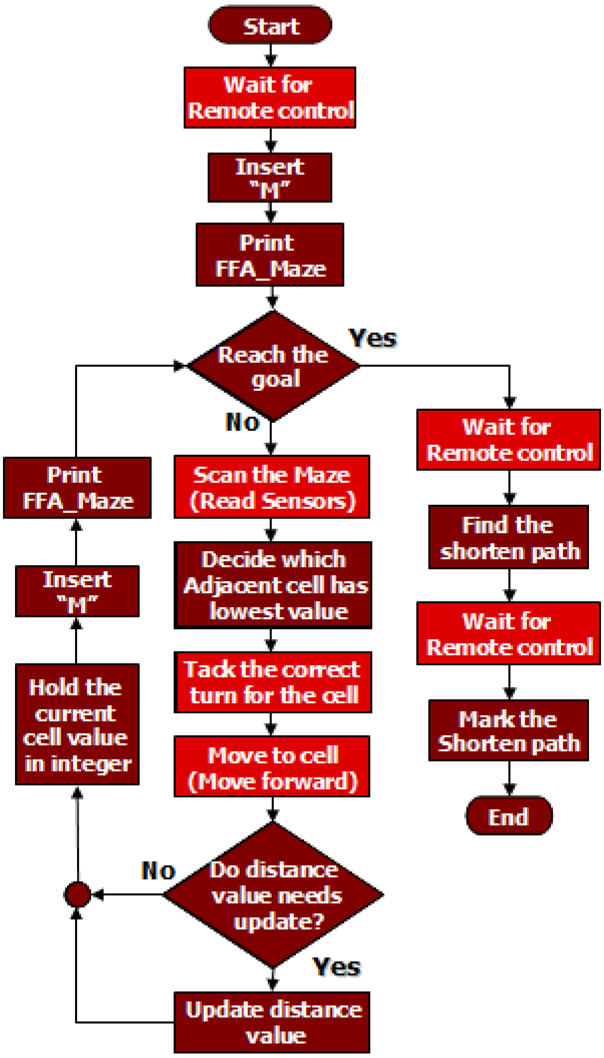


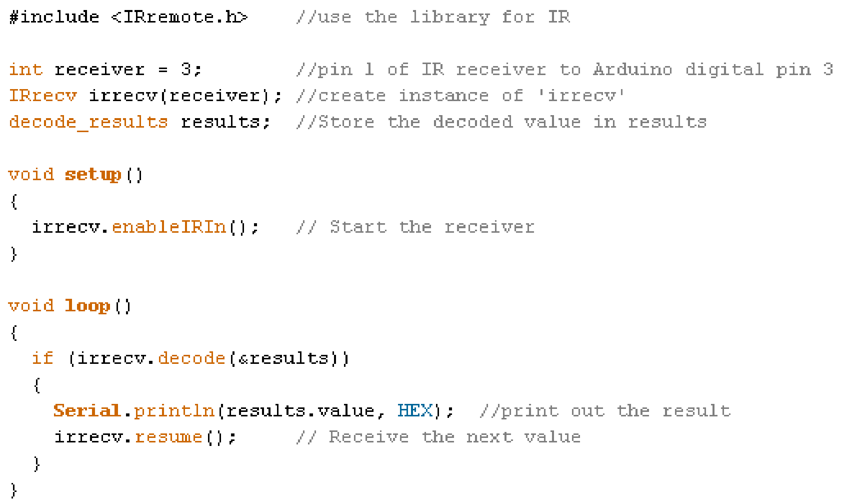
Figure-29: Robot programme flow chart.

When the robot moves to the cell it enter the straight function, which will involves reading the sensors and prevent the robot from hitting the maze walls.

```
move = 0;
while(move < 9)
left_sensor = analogRead(0);    //Read sensors
front_sensor = analogRead(1);
right_sensor = analogRead(2);

digitalWrite(dir_L, HIGH);      //set both motor high
digitalWrite(dir_R, HIGH);
speed_L = 60;
speed_R = 60;

//move slightly to the left if the right sensor is near to the wall
if(right_sensor > active)
{    speed_L = 0;
     speed_R = 60;
}

 //move slightly to the right if the left sensor is near to the wall
 if(left_sensor > active)
{    speed_L = 60;
     speed_R = 0;
}

analogWrite(pwm_L, speed_L);    //upload the speed to the motors
analogWrite(pwm_R, speed_R);
move++;
delay(100);
```

Moreover, the infrared receiver has been added as an extra feature for the robot. It will prevent the robot from starting the next stage except with the user permission. Therefore, after each stage the robot will wait for the user to press (CH-) button.

The infrared receiver will generate a different digital output depending on the received pulses of the infrared light coming from the remote control. This digital output will be decoded by programming the Arduino board with a suitable programme that obtains the result in a form of numbers. The decoding process can be easily achieved through using “IRremote.h” library, which is made for Arduino. The following code shows how the programme obtains the result from the IR receiver.

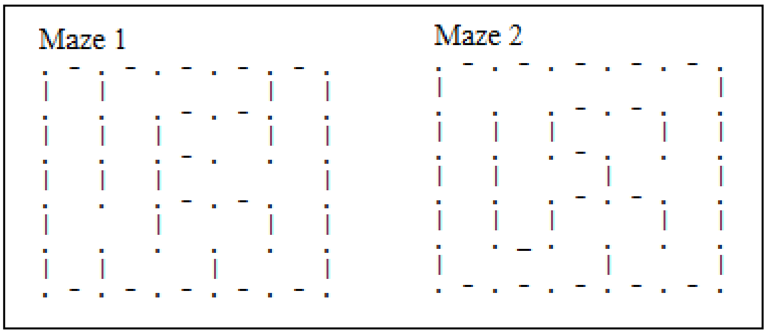


```
#include <IRremote.h>     //use the library for IR

int receiver = 3;         //pin 1 of IR receiver to Arduino digital pin 3
IRrecv irrecv(receiver); //create instance of 'irrecv'
decode_results results;  //Store the decoded value in results

void setup()
{
  irrecv.enableIRIn();   // Start the receiver
}

void loop()
{
  if (irrecv.decode(&results))
  {
    Serial.println(results.value, HEX);  //print out the result
    irrecv.resume();     // Receive the next value
  }
}
```

5.3 Testing Procedure

5.3.1 Simulation Test

Both left hand wall and fill flood algorithms programmes have been loaded with two different mazes. After running the programmes the simulation result has been appear on the serial monition, the results are included on the next chapter.



```
Maze 1                  Maze 2
.-.-.-.-.-.             .-.-.-.-.-.
| |     | |             |         |
. . .-.-. .             . . .-.-. .
| | |   | |             | | |   | |
. . .-. . .             . . .-. . .
| | |     |             | |   |   |
. . .-.-. .             . . .-.-. .
|   |   | |             | | |   | |
. . . . . .             . .-. . . .
| |   |   |             |     |   |
.-.-.-.-.-.             .-.-.-.-.-.
```

Figure-30: Mazes for testing simulations programme.

5.3.2 Robot Test

A 3 x 3 maze, which is shown in figure-31, has been built in order to test the robot. At the beginning a simulation test has been run to make sure that the algorithm work well. Then, robot has been placed on the starting position and tested on the real maze. Both simulation and experimental result are provided on the next chapter.

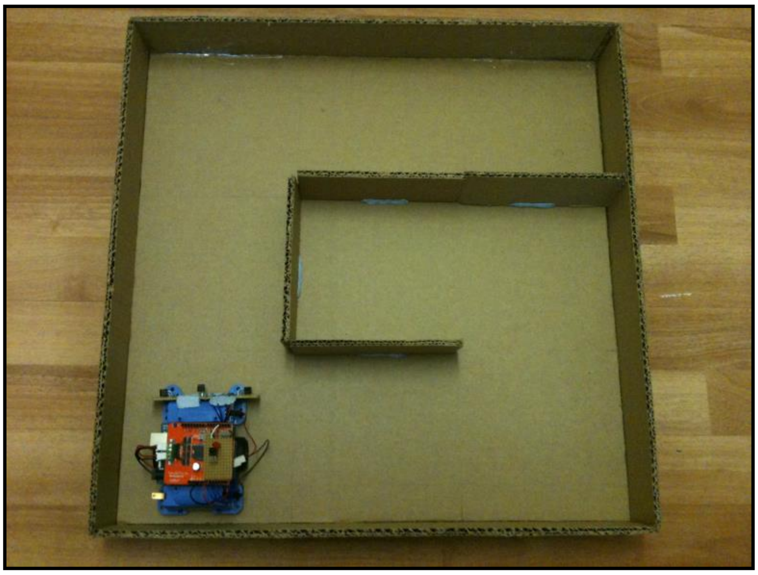


Figure-31: The robot on a 3 x 3 maze.

Chapter 6

Results

This chapter is going to show the results obtained during this project. It will start with showing the simulation result for both left hand wall and fill flood algorithms. Then it will show the experimental result for the robot on a 3x3 maze.

Chapter 6: Results

6.1 Simulation Results

After running the simulation programmes for both left hand wall and fill flood algorithms the following result has been optioned. The results for Maze-1 are shown in figure-32, while Maze-2 results are shown in figure-33. The full simulation results for Maze-1 are included in appendix-4.

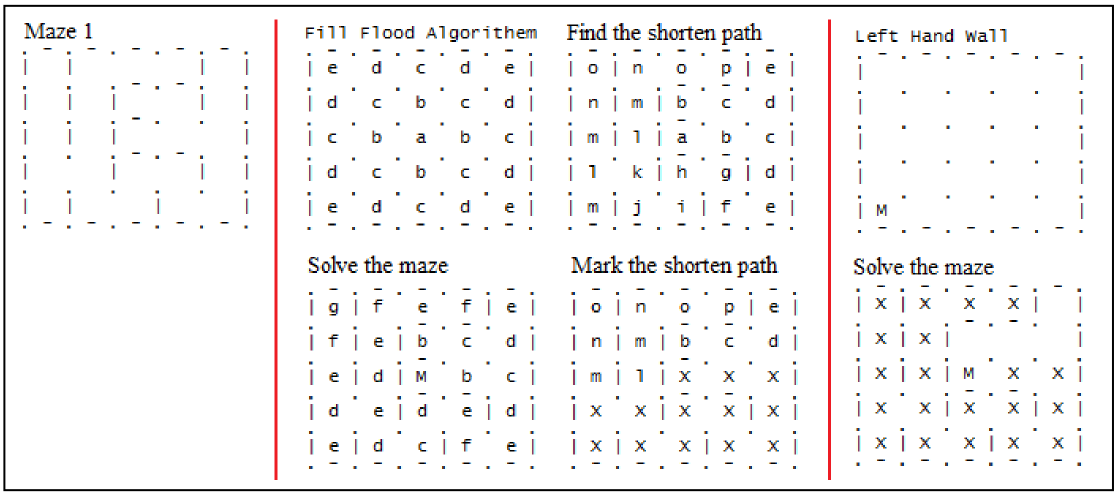


Figure-32: Simulation results for maze 1.

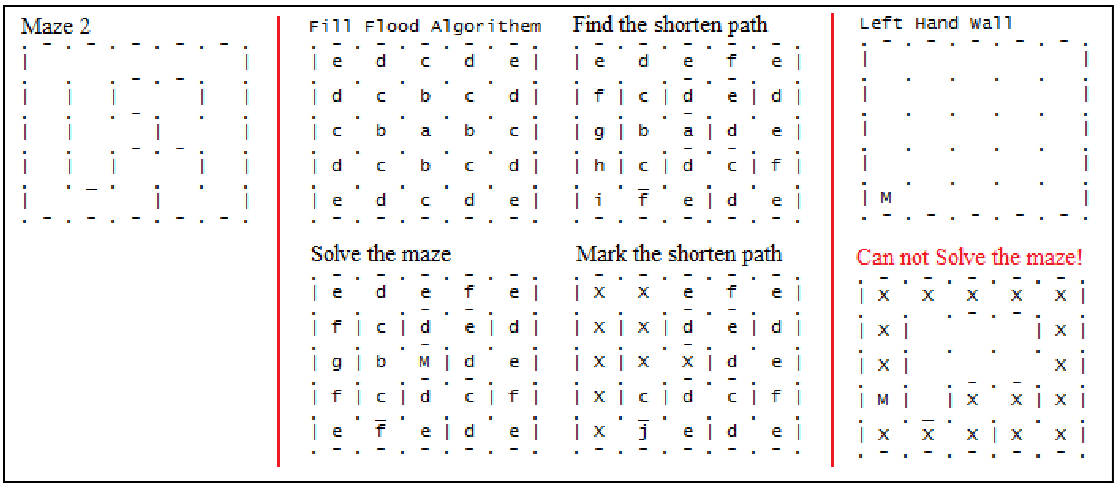
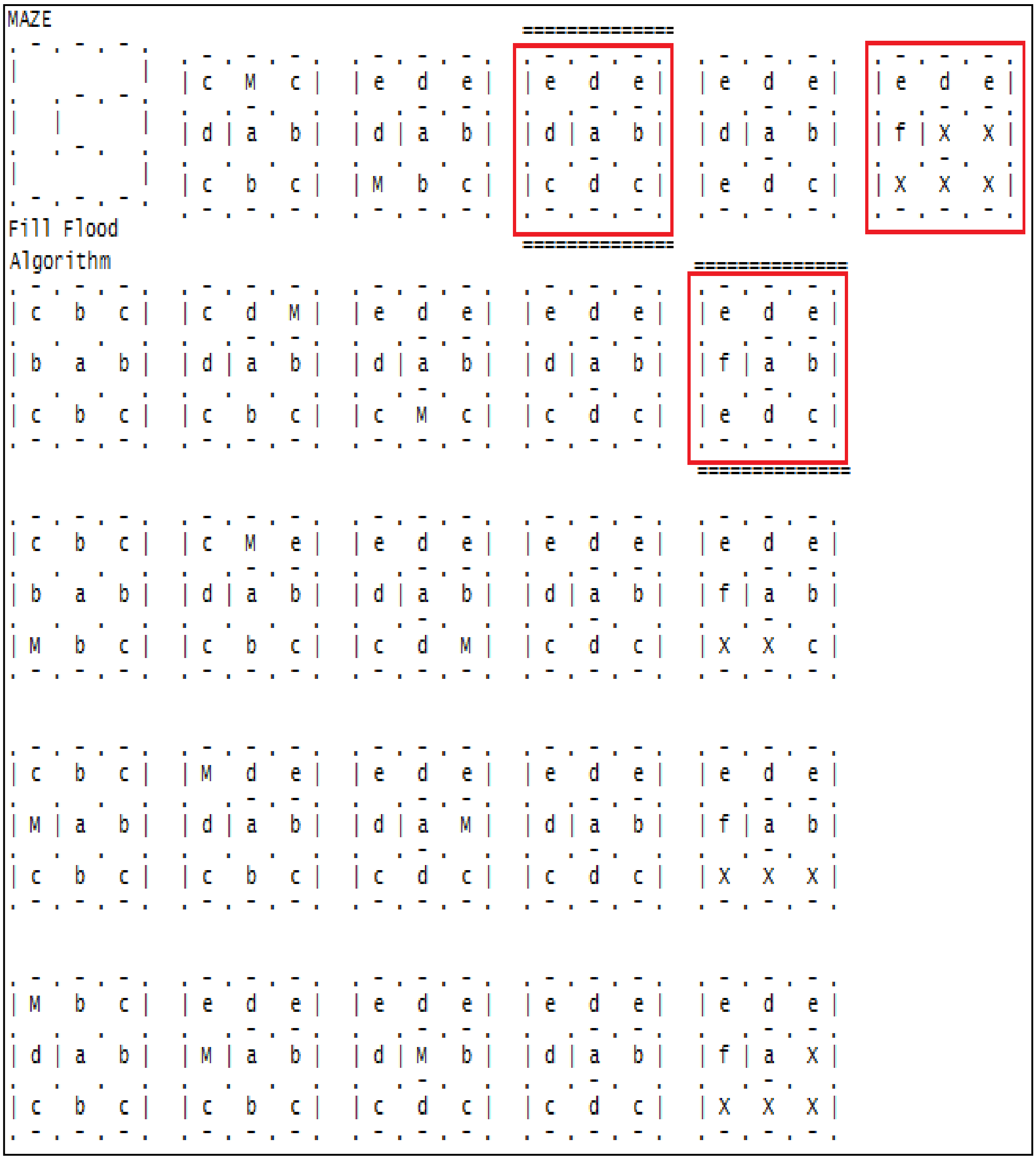


Figure-33: Simulation results for maze 2.

6.2 Experimental Results

The simulation test has been applied for the 3 x 3 maze and the result has been recorded in the figure below. After that, the robot has been tested on the maze. The robot was able to read the remote control signal and it stats the first stage. It was hard to achieve this stage “solving the maze”, but when the robot reaches the shorten path stage it has better response. Figure-35 and figure-36 are giving an illustration about the robot performance.



```
MAZE
                                                  ===============
. - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .
|           |    | c   M   c |    | e   d   e |    | e   d   e |    | e   d   e |    | e   d   e |
.   . - . - .    .   . - .   .    .   . - . - .    .   . - . - .    .   . - . - .    .   . - . - .
|   |       |    | d | a   b |    | d | a   b |    | d | a   b |    | d | a   b |    | f | X   X |
.   . - .   .    .   .   .   .    .   .   .   .    .   . - .   .    .   . - .   .    .   . - .   .
|           |    | c   b   c |    | M   b   c |    | c   d   c |    | e   d   c |    | X   X   X |
. - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .
                                                  ===============
Fill Flood
Algorithm                                                          ===============
. - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .
| c   b   c |    | c   d   M |    | e   d   e |    | e   d   e |    | e   d   e |
.   .   .   .    .   . - . - .    .   . - . - .    .   . - . - .    .   . - . - .
| b   a   b |    | d | a   b |    | d | a   b |    | d | a   b |    | f | a   b |
.   .   .   .    .   .   .   .    .   . - .   .    .   . - .   .    .   . - .   .
| c   b   c |    | c   b   c |    | c   M   c |    | c   d   c |    | e   d   c |
. - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .
                                                                   ===============

. - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .
| c   b   c |    | c   M   e |    | e   d   e |    | e   d   e |    | e   d   e |
.   .   .   .    .   . - . - .    .   . - . - .    .   . - . - .    .   . - . - .
| b   a   b |    | d | a   b |    | d | a   b |    | d | a   b |    | f | a   b |
.   .   .   .    .   .   .   .    .   . - .   .    .   . - .   .    .   . - .   .
| M   b   c |    | c   b   c |    | c   d   M |    | c   d   c |    | X   X   c |
. - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .


. - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .
| c   b   c |    | M   d   e |    | e   d   e |    | e   d   e |    | e   d   e |
.   .   .   .    .   . - . - .    .   . - . - .    .   . - . - .    .   . - . - .
| M | a   b |    | d | a   b |    | d | a   M |    | d | a   b |    | f | a   b |
.   .   .   .    .   .   .   .    .   . - .   .    .   . - .   .    .   . - .   .
| c   b   c |    | c   b   c |    | c   d   c |    | c   d   c |    | X   X   X |
. - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .


. - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .
| M   b   c |    | e   d   e |    | e   d   e |    | e   d   e |    | e   d   e |
.   .   .   .    .   . - . - .    .   . - . - .    .   . - . - .    .   . - . - .
| d | a   b |    | M | a   b |    | d | M   b |    | d | a   b |    | f | a   X |
.   .   .   .    .   .   .   .    .   . - .   .    .   . - .   .    .   . - .   .
| c   b   c |    | c   b   c |    | c   d   c |    | c   d   c |    | X   X   X |
. - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .    . - . - . - .
```

Figure-34: Simulation results for maze the real maze.

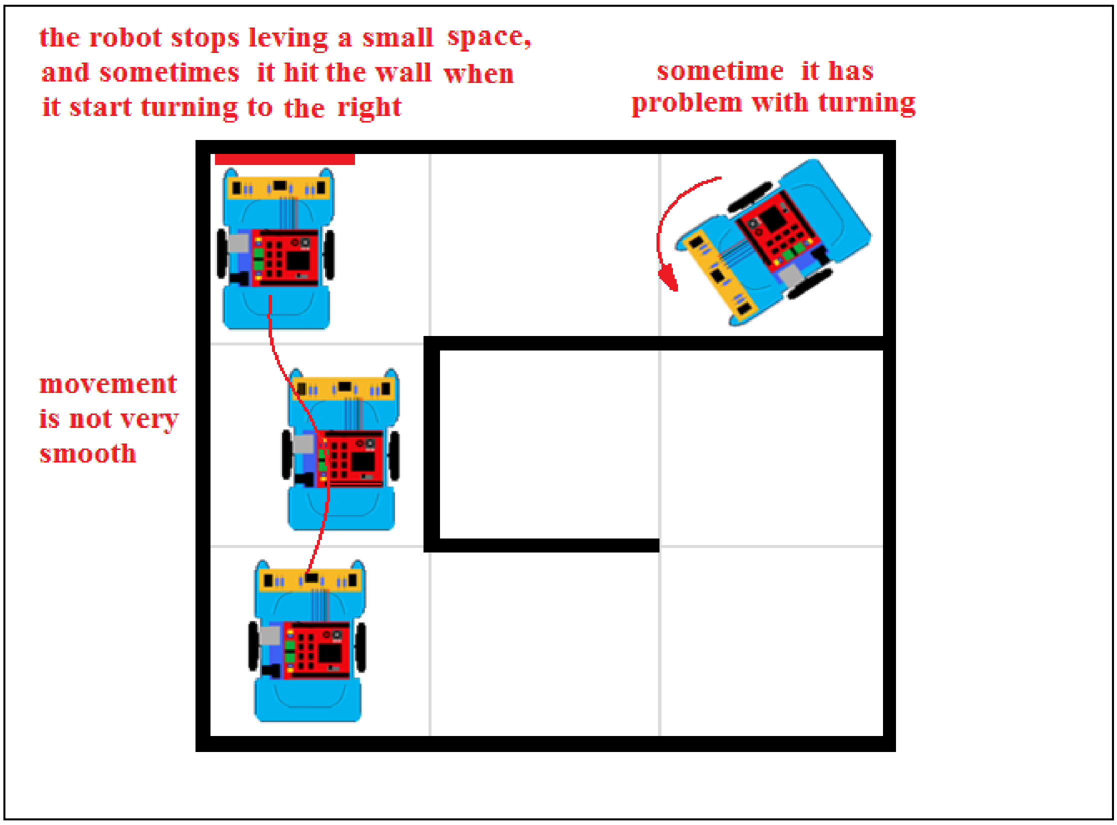

the robot stops leving a small space,
and sometimes it hit the wall when
it start turning to the right
sometime it has
problem with turning
movement
is not very
smooth

Figure-35: Difficulties on solving the maze.

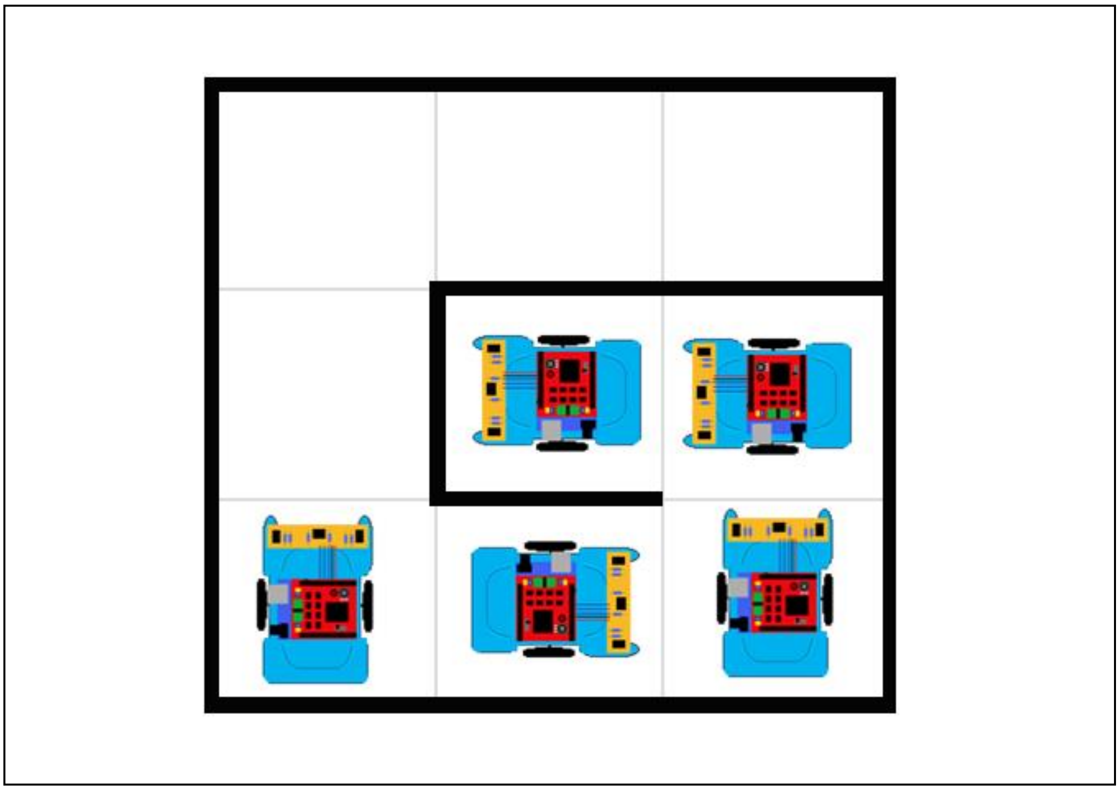


Figure-36: Finding the shorten path.

Chapter 7

Discussion and Conclusion

This chapter is divided into four sections. The first section is going to discuss the obtained results with describing errors and inaccuracy occurred during testing the robot as well as explaining the reasons behind these errors.

The second section will talk about the project management, which will gives an explanation about time planning for the project and providing some details about the cost analysis.

The third and fourth sections are going to summaries the main findings and will end up with some suggestions could be done in order to improve this project in the future.

7.1 Discussion

Left Hand Wall

The left hand wall algorithm was an easy algorithm to implement. The simulation test on for Maze-1 was impressive. By taking the left walls as guide, the mouse was able to reach the goal with the same number of steps as using the modified fill flood algorithm. However, because the absent of intelligence this algorithm was not able to solving Maze-2. The reason behind that is the closed loop path. The mouse has three options arranged in the following order: left, straight or right. In a close loop path the mouse will not be able to execute the suitable option "right" when it should be. That because it will always have to execute the higher priority options which are "left" or "straight". However, this project offer a solution for solving this problem by changing the polarity of the following wall, see further work section for more details.

Modify Fill Flood Algorithm

The modified fill flood algorithm is an efficient method to solve even complicated maze. By depending on the assigning value for all cells in the maze, the algorithm was capable of solving both mazes without having any problems. In this project this algorithm has been extended to find out the shorten path. It can be clearly seen that the shorten path has been found and it marks as it wonted with character "X".

Maze Solving Robot

Since the modified fill flood algorithm proves it efficient, it has been used to lead the robot to solve the maze. A 3x3 maze has been built and simulated. The simulation result, which is shown in fgure-34, describes the way of how the robot will solve the maze. During testing the robot on the real maze, there were some problems in the movement of the robot. This is owing to the time delay on the programme and to the sensors. The threshold value of the sensors on the code can be adjusted, in order to achieve a smoother movement for the robot. Moreover, in order to prevent the robot from hitting the walls or having a long turn, the delay time for movement and turning can be reduced. However, the algorithm works fine on the robot and it will have slight adjustment before the demonstration as it planed on the Gantt chart "further work".

7.2 Project Management

Back to second year, I was enthusiastic about robot, programming and algorithms. I spent most of my summer holiday reading journals and searching about maze solving robot.

During the first week on term 1, a full plan has been set to complete this project. The Gantt chart for the plan is attached on appendix-5. The work has been stick with the plan and all deadlines have been submitted on time.

The supervisor meetings have significant advantages to successfully complete this project. My supervisor Dr. David Southall has a very good background on this kind of projects. He suggested, to focus on the software part and to use a suitable algorithm for this project, which is the Modified Fill Flood Algorithm. He also provided me with a lot of document that helps me to understanding the algorithm.

Applications for this project

This project is based on one of the most important areas of robot, which is "Decision Making Algorithm". Nowadays robots are used widely use especially in factories. The following figure shows a robot used in Porsche Factory for carrying and delivering engines and transmissions in the exact time and sequence to be install into the car. It has a tracking control and safety system which will stop immediately if there is something on the fronot of it (National Geographic, 2012).

Figure-37: application of robots on cars factories (National Geographic, 2012).

Cost Analysis

The total cost of this project is £57.40. In fact, this is a very good price for a robot. Making a mobile robot requires a chassis that it has the ability to contain and provide all the robot needs. The chassis which is used in this project dose not only provide the body for the robot, it also has two DC motors, wheels and a battery holder. Therefore, building a chassis might cost even more than £20.

Arduino Uno board is quite expensive and it cost £20. Even though, the prise can be reduced be making a simple Arduino board, the Arduino Uno has been selected because this prise includes the ATmega328 microcontroller, voltage regulators, and many other futures such as supporting serial communication. Moreover, Arduomoto shield, which is cost only £8, have been used since it compatible with most of Arduino boards.

The robot depends on the IR Reflectance Sensors, which is cost £3, to sense the maze walls and provide the feed back of the robot position.

TSOP34840 IR Receivers and Remote Control have been used to control the robot wirelessly. The cost of both components together is less than £5, which is one of the cheapest solutions for wireless controlling. Other components such as LED, resistor and switch have been added to the robot and they are very cheap.

Table-5: The used parts and the total cost of the project.

No.	Parts Required	Quantity	Prise
1	Robot Chassis (Including DC Motors)	x1	£20
2	Arduino Uno Board	x1	£20
3	Ardumoto Shield	x1	£8
4	RPR IR Reflectance Sensors	X3	£3 (£1 each)
5	TSOP34840 IR Receivers	x1	£0.90
6	Remote Control Handset	x1	£3.50
7	Red LED	x1	£0.50
8	On/Off Switch	x1	£1
9	Resistor	x1	£0.50
Total Cost		**£57.40**	

7.3 Conclusion

In conclusion, the project covers two well known maze solving algorithms. It involves designing programmes that gives a simulation results for all the steps required to solve the maze. The first algorithm was left hand wall. This basic method shows an impressive results in solving the maze, but due to the absent of intelligence, this method fail in solving even is a simple maze with closed loop. There for another algorithm, which is the modified fill flood algorithm, has been used to tackle the problem. It is a very efficient method that will even solve a complicated maze. Moreover, the modified fill flood algorithm has been used to drive robot to solve a real maze. The robot has been able to solve the maze. It was not the best performance, but with few adjustments the robot can works better.

This project provides lots of benefits. Working in this kind of project improves important techniques such as time management, project development and cost analysis. It also covers an extensive information about robotics field which helps to improve programming skills as well as learning about other electronic components such as motor driver, IR reflective sensors and the operating of remote control. This gained knowledge will have a significant impact in my future during working on another project or during working in a factory.

7.4 Further Work

One of the best methods that can be used to improve the robot movement is wheel encoding technique. As it shown in the figure-38, by adding the counting number to the circle circumference formula it possible to find out the distance travelled. In this method an infrared reflective sensor is used to encode the wheel. The wheel should have two different colures or holes, so if the wheel rotates the output of the IR sensor will change from high to low. Then, a microcontroller can be used to apply the equation depending on the changing IR output.

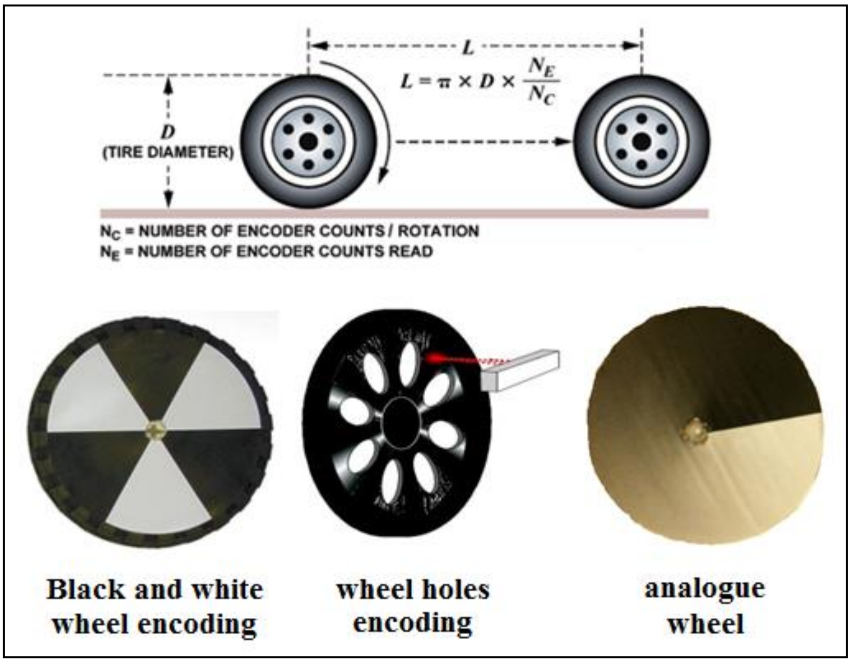
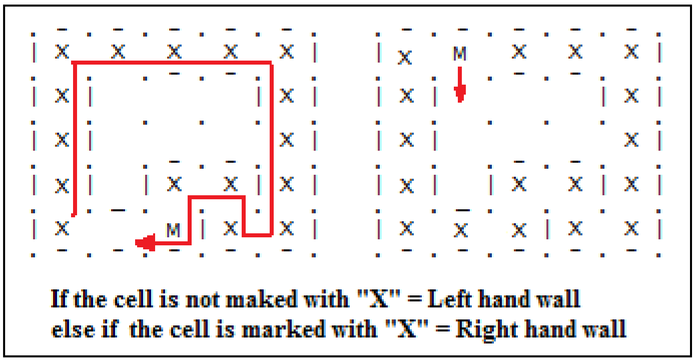

L
$L = \pi \times D \times \frac{N_E}{N_C}$
D
(TIRE DIAMETER)
N_C = NUMBER OF ENCODER COUNTS / ROTATION
N_E = NUMBER OF ENCODER COUNTS READ
Black and white wheel encoding
wheel holes encoding
analogue wheel

Figure-38: Wheel encoding technique.

Moreover, a suggestion has been made to improve the wall following algorithm. Both left hand wall and right-hand wall are going to fail in solving this maze, because it has a closed loop path. However, a solution can be made to solve this problem which is by changing the polarity of the following wall when the robot starts looping, see the figure below.



If the cell is not maked with "X" = Left hand wall
else if the cell is marked with "X" = Right hand wall

Figure-39: solution for wall following algorithm.

References

Active-Robot, 2012. *Robo jr body set.* [online] Available at: <http://www.active-robots.com/robot-parts/robot-chassis/robo-jr-body-set.html> [Accessed 20 March 2013].

Arduino, 2012. Arduino Uno. [online] Available at: <http://arduino.cc/en/Main/ArduinoBoardUno> [Accessed 20 March 2013].

Bellman, R., 1958. On a Routing Problem. *Quarterly of Applied Mathematics*, 16(1), pp.87-90

Chen, Y., Zhao, H., Wan, W. and Yu, X., 2008. A High Efficiency Center-First-Routing-Theorem Algorithm of Micro-Mouse. *Audio, Language and Image Processing.* 10.1109/ICALIP.2008.4590229, pp.788-793.

Cyberneticzoo, 2010. *Amazing Micromouse Maze Contest.* [online] Available at: <http://cyberneticzoo.com/?tag=amazing-micromouse-maze-contest> [Accessed 20 March 2013].

Fritzing, 2011. *About fritzing*. [online] Available at: <http://fritzing.org/> [Accessed 20 March 2013].

IEEE UCSD, 2009. *IEEE Micromouse Rules Region 6 Southwest Area.* [pdf] Available at: <http://ieee.ucsd.edu/files/micromouse-rules.pdf> [Accessed 20 March 2013].

Li, Z., Duan, H., Gao, Y. and Dong, Z., 2011. An extention of height value algorithm for micro-mouse robot applying for exploring unknown region. *Industrial Electronics and Applications,* 10.1109/ICIEA.2011.5975611, pp.370–375.

Makezine, 2009. *Make: holiday gift guide 2009: all-arduino.* [online] Available at: <http://blog.makezine.com/2009/12/08/make-holiday-gift-guide-2009-all-ar/> [Accessed 20 March 2013].

McRoberts, M., 2010. *Beginning arduino*. New York : Apress.

Mishra, S. and Bande, P., 2008. Maze Solving Algorithms for mouse. *Signal Image Technology and Internet Based Systems,* 10.1109/SITIS.2008.104, pp.86-93.

Monk, S., 2010. *30 arduino projects for the evil genius*. New York: McGraw-Hill.

National Geographic, 2012. *Mega Factories- Porsche Panamera*.[video online]. Available at: < http://www.youtube.com/watch?v=74HouUyZITw> [Accessd 20 March 2012].

Sadik, A. M. J., 2010. Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory. *International Conference on Artificial Intelligence and Computational Intelligence*, 10.1109/AICI.2010.18, pp.52-56.

Satkum, 2010. Maze solving robot using Arduino. *WordPress.com* Web blog. [blog] 10 october. Avilable at: <http://satkum.wordpress.com/2010/10/10/maze-solving-robot/> [Accessed 20 March 2013].
Sharma, M., Robeonics, K., 2009. Algorithms for Micro-mouse. *International Conference on Future Computer and Communication,* DOI 10.1109/ICFCC.2009.38, pp.581-585.

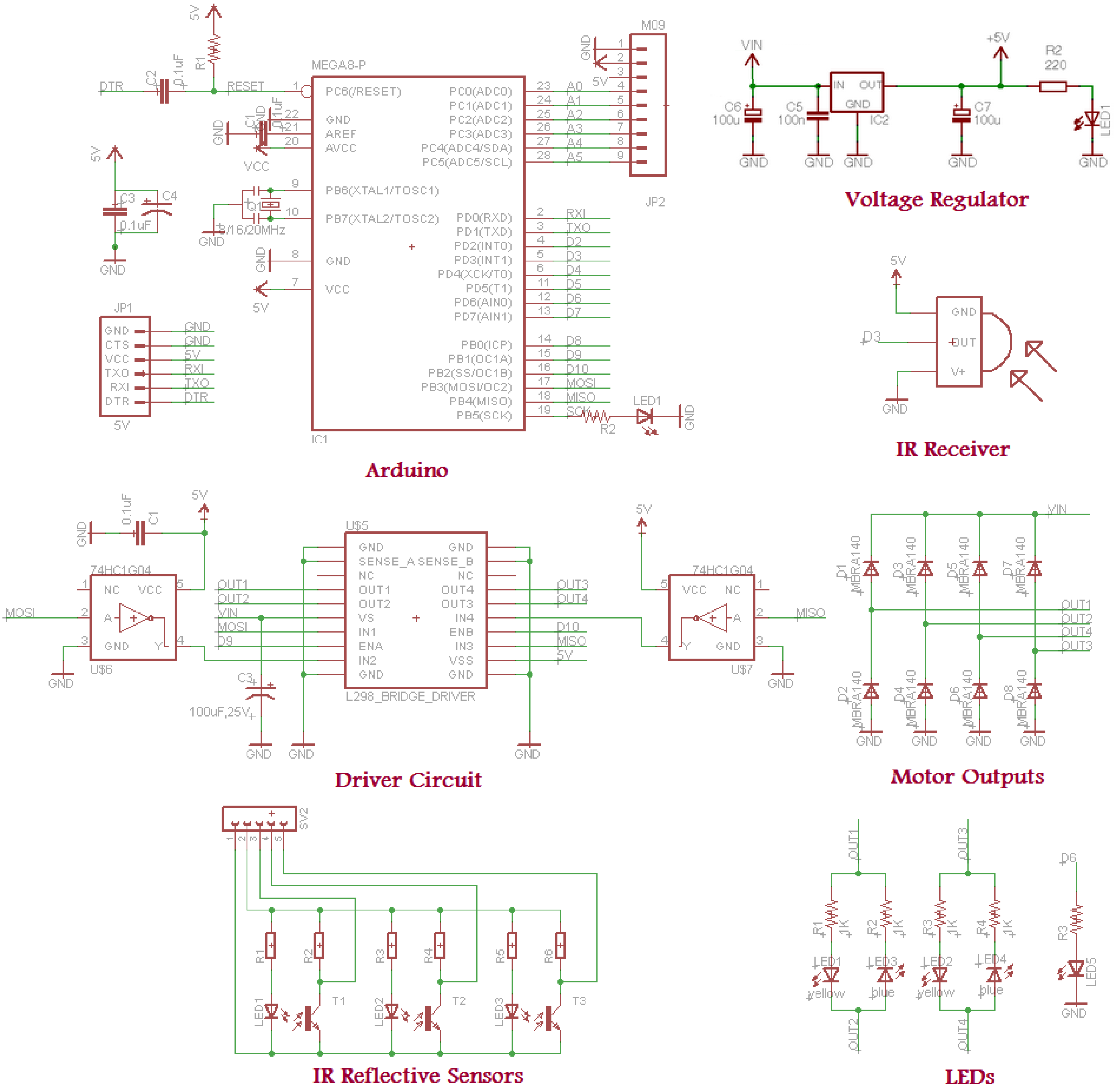
Southall, D. M., 2007. *Modified flood algorithm for a maze solving robot 'mouse'*. [doc]

Vishay, 2011. *TSOP34840 - IR receiver modules for remote control systems*. [online] Available at: < http://www.vishay.com/docs/81732/tsop348.pdf > [Accessed 20 March 2013].

Warren, J. D., Adams, J. and Molle, H., 2011. *Arduino robotics*. New York : Apress.

Appendixes

Appendix-1: Maze solving robot schematic diagram.



Arduino

Voltage Regulator

IR Receiver

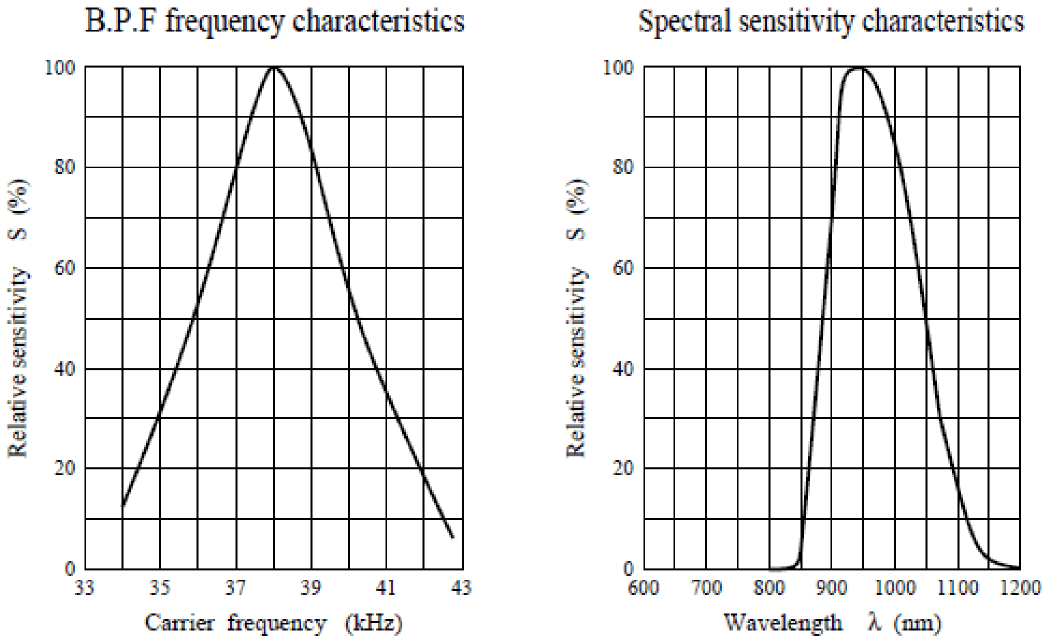
Driver Circuit

Motor Outputs

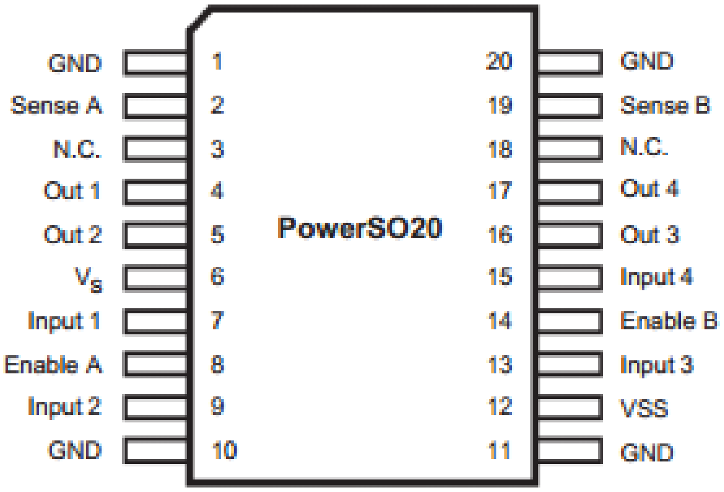
IR Reflective Sensors

LEDs

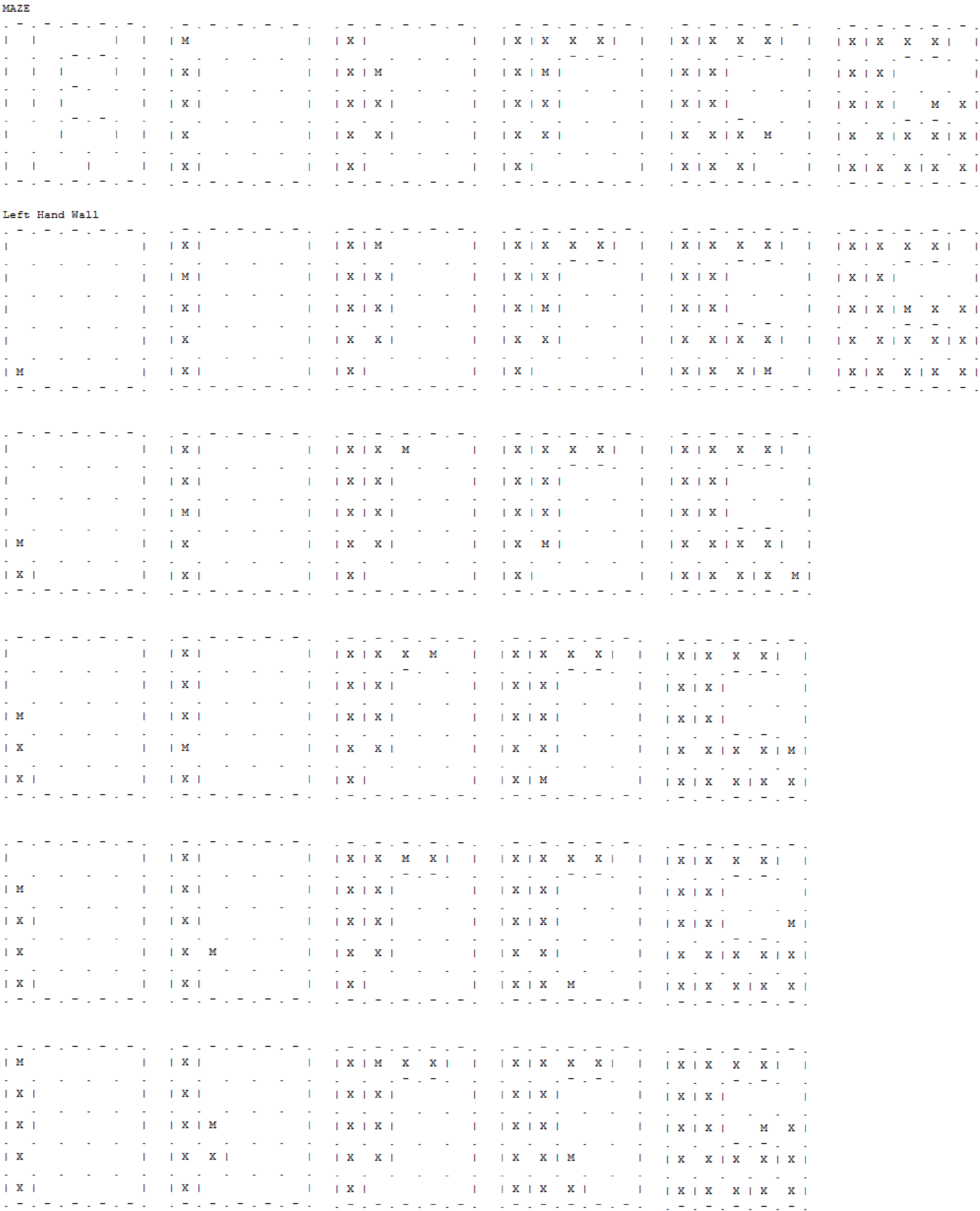
Appendix-2: Frequency and sensitivity of TSOP34840 IR receiver.



Appendix-3: L298 dual h-bridge motor driver.



Appendix-4: Full solution for Maze 1.



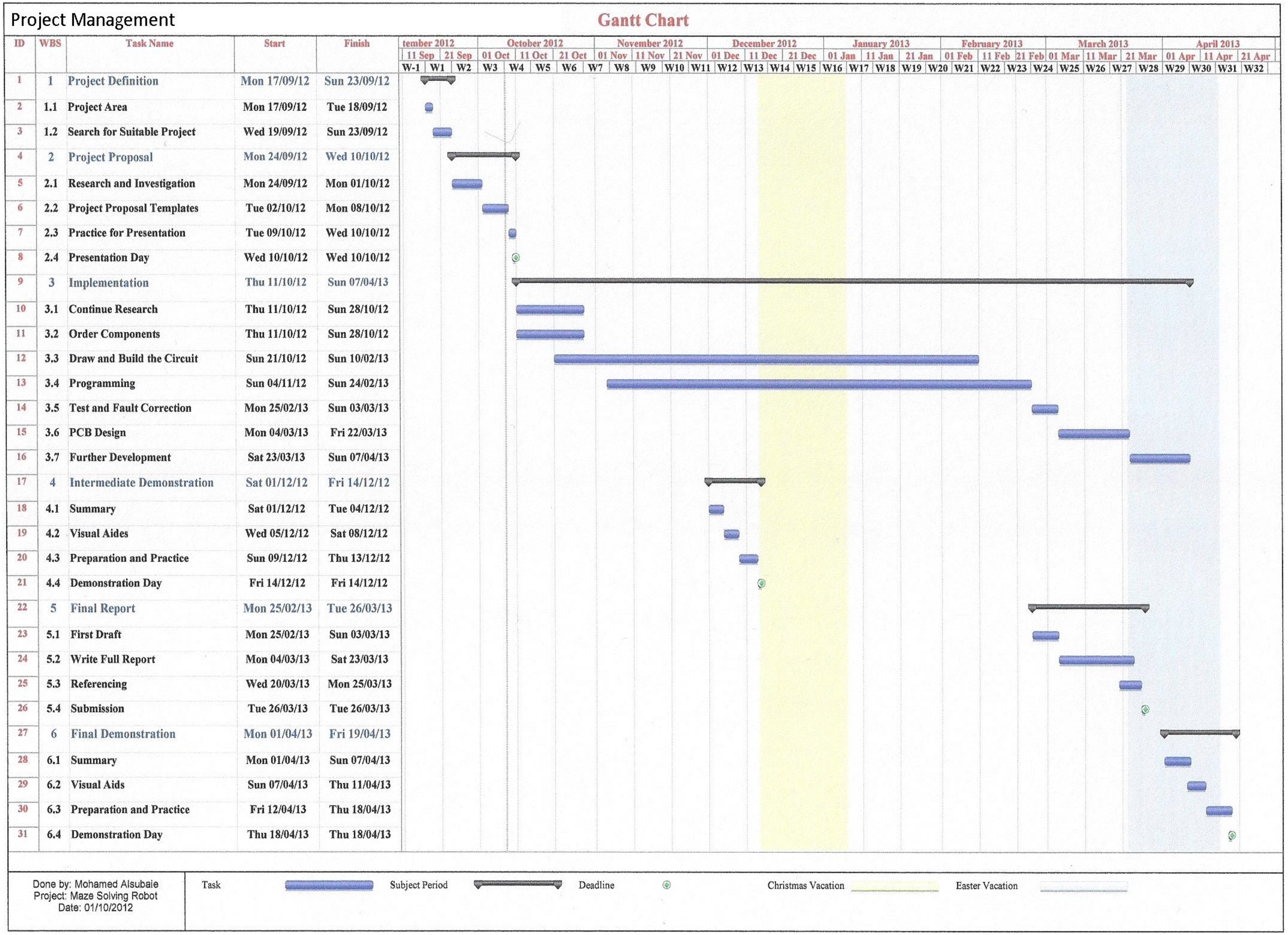
```
MAZE
. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .
|   |           |   |    | M                 |    | X |               |    | X | X   X   X |   |    | X | X   X   X |   |    | X | X   X   X |   |
.   .   . - . - .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   . - . - .   .    .   .   . - . - .   .    .   .   . - . - .   .
|   |   |       |   |    | X |               |    | X | M             |    | X | M |           |    | X | X |           |    | X | X |           |
.   .   . - .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
|   |   |           |    | X |               |    | X | X |           |    | X | X |           |    | X | X |           |    | X | X |     M   X |
.   .   . - . - .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   . - .   .   .    .   .   . - . - .   .
|       |       |   |    | X                 |    | X   X |           |    | X   X |           |    | X   X | X   M     |    | X   X | X   X | X |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
|   |       |       |    | X |               |    | X |               |    | X |               |    | X | X   X |       |    | X | X   X | X   X |
. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .

Left Hand Wall
. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .
|                   |    | X |               |    | X | M             |    | X | X   X   X |   |    | X | X   X   X |   |    | X | X   X   X |   |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   . - . - .   .    .   .   . - . - .   .    .   .   . - . - .   .
|                   |    | M |               |    | X | X |           |    | X | X |           |    | X | X |           |    | X | X |           |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
|                   |    | X |               |    | X | X |           |    | X | M |           |    | X | X |           |    | X | X | M   X   X |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   . - . - .   .    .   .   . - . - .   .
|                   |    | X                 |    | X   X |           |    | X   X |           |    | X   X | X   X |   |    | X   X | X   X | X |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
| M                 |    | X |               |    | X |               |    | X |               |    | X | X   X | M     |    | X | X   X | X   X |
. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .

. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .
|                   |    | X |               |    | X | X   M         |    | X | X   X   X |   |    | X | X   X   X |   |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   . - . - .   .    .   .   . - . - .   .
|                   |    | X |               |    | X | X |           |    | X | X |           |    | X | X |           |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
|                   |    | M |               |    | X | X |           |    | X | X |           |    | X | X |           |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   . - . - .   .
| M                 |    | X                 |    | X   X |           |    | X   M |           |    | X   X | X   X |   |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
| X |               |    | X |               |    | X |               |    | X |               |    | X | X   X | X   M |
. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .

. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .
|                   |    | X |               |    | X | X   X   M     |    | X | X   X   X |   |    | X | X   X   X |   |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   . - .   .   .    .   .   . - . - .   .    .   .   . - . - .   .
|                   |    | X |               |    | X | X |           |    | X | X |           |    | X | X |           |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
| M                 |    | X |               |    | X | X |           |    | X | X |           |    | X | X |           |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   . - . - .   .
| X                 |    | M                 |    | X   X |           |    | X   X |           |    | X   X | X   X | M |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
| X |               |    | X |               |    | X |               |    | X | M             |    | X | X   X | X   X |
. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .

. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .
|                   |    | X |               |    | X | X   M   X |   |    | X | X   X   X |   |    | X | X   X   X |   |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   . - . - .   .    .   .   . - . - .   .    .   .   . - . - .   .
| M                 |    | X |               |    | X | X |           |    | X | X |           |    | X | X |           |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
| X |               |    | X |               |    | X | X |           |    | X | X |           |    | X | X |         M |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   . - . - .   .
| X                 |    | X   M             |    | X   X |           |    | X   X |           |    | X   X | X   X | X |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
| X |               |    | X |               |    | X |               |    | X | X   M         |    | X | X   X | X   X |
. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .

. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .
| M                 |    | X |               |    | X | M   X   X |   |    | X | X   X   X |   |    | X | X   X   X |   |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   . - . - .   .    .   .   . - . - .   .    .   .   . - . - .   .
| X |               |    | X |               |    | X | X |           |    | X | X |           |    | X | X |           |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
| X |               |    | X | M             |    | X | X |           |    | X | X |           |    | X | X |     M   X |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   . - . - .   .
| X                 |    | X   X |           |    | X   X |           |    | X   X | M         |    | X   X | X   X | X |
.   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .    .   .   .   .   .   .
| X |               |    | X |               |    | X |               |    | X | X   X |       |    | X | X   X | X   X |
. - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .    . - . - . - . - . - .
```

```
MAZE
.-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.
| |     | |  |e d c d e|  |g|d c d e|  |g|d c d e|  |g|d M f|e|  |g|f e f|e|  |g|f e f|e|  |g|f e f|e|
. . .-.-. .  . . . . . .  . . . . . .  . . . . . .  . . .-.-. .  . . .-.-. .  . . .-.-. .  . . .-.-. .
| | |   | |  |d c b c d|  |f|c b c d|  |f|M|b c d|  |f|e|b c d|  |f|e|b c d|  |f|e|b c d|  |f|e|b c d|
. . .-. . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .
| | |     |  |M|b a b c|  |M|b a b c|  |e|d|a b c|  |e|d|a b c|  |e|d|a b c|  |e|d|a b c|  |e|d|a b M|
. . .-.-. .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . .-.-. .  . . .-.-. .
|   |   | |  |d c b c d|  |d c b c d|  |d c|b c d|  |d c|b c d|  |d M|b c d|  |d e|d M|d|  |d e|d e|d|
. . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .
| |   |   |  |e|d c d e|  |e|d c d e|  |e|d c d e|  |e|d c d e|  |e|d c d e|  |e|d c|d e|  |e|d c|f e|
.-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.

Fill Flood Algorithem
.-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.
|e d c d e|  |e d c d e|  |g|d c d e|  |g|M c d e|  |g|M e f|e|  |g|f e f|e|  |g|f e f|e|  |g|f e f|e|
. . . . . .  . . . . . .  . . . . . .  . . . . . .  . . .-.-. .  . . .-.-. .  . . .-.-. .  . . .-.-. .
|d c b c d|  |M|c b c d|  |f|c b c d|  |f|e|b c d|  |f|e|b c d|  |f|e|b c d|  |f|e|b c d|  |f|e|b c d|
. . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .
|c b a b c|  |e|b a b c|  |e|b a b c|  |e|d|a b c|  |e|d|a b c|  |e|d|a b c|  |e|d|a b c|  |e|d|a M c|
. . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . .-.-. .  . . .-.-. .
|d c b c d|  |d c b c d|  |M c b c d|  |d c|b c d|  |d c|b c d|  |d e|b c d|  |d e|d e|d|  |d e|d e|d|
. . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .
|e d c d e|  |e|d c d e|  |e|d c d e|  |e|d c d e|  |e|d c d e|  |e|M c d e|  |e|d c|M e|  |e|d c|f e|
.-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.

.-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.
|e d c d e|  |M|d c d e|  |g|d c d e|  |g|d M d e|  |g|f e f|e|  |g|f e f|e|  |g|f e f|e|  |g|f e f|e|
. . . . . .  . . . . . .  . . . . . .  . . .-. . .  . . .-.-. .  . . .-.-. .  . . .-.-. .  . . .-.-. .
|d c b c d|  |f|c b c d|  |f|c b c d|  |f|e|b c d|  |f|M|b c d|  |f|e|b c d|  |f|e|b c d|  |f|e|b c d|
. . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . .-. . .
|c b a b c|  |e|b a b c|  |e|b a b c|  |e|d|a b c|  |e|d|a b c|  |e|d|a b c|  |e|d|a b c|  |e|d|M b c|
. . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . .-.-. .  . . .-.-. .
|d c b c d|  |d c b c d|  |d M|b c d|  |d c|b c d|  |d c|b c d|  |d e|b c d|  |d e|d e|d|  |d e|d e|d|
. . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .
|M|d c d e|  |e|d c d e|  |e|d c d e|  |e|d c d e|  |e|d c d e|  |e|d M|d e|  |e|d c|f M|  |e|d c|f e|
.-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.

.-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.
|e d c d e|  |g|d c d e|  |g|d c d e|  |g|d e M|e|  |g|f e f|e|  |g|f e f|e|  |g|f e f|e|
. . . . . .  . . . . . .  . . . . . .  . . .-.-. .  . . .-.-. .  . . .-.-. .  . . .-.-. .
|d c b c d|  |M|c b c d|  |f|c b c d|  |f|e|b c d|  |f|e|b c d|  |f|e|b c d|  |f|e|b c d|
. . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .
|c b a b c|  |e|b a b c|  |e|M|a b c|  |e|d|a b c|  |e|M|a b c|  |e|d|a b c|  |e|d|a b c|
. . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . .-. . .  . . .-.-. .
|M c b c d|  |d c b c d|  |d c|b c d|  |d c|b c d|  |d c|b c d|  |d e|M c d|  |d e|d e|M|
. . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .  . . . . . .
|e|d c d e|  |e|d c d e|  |e|d c d e|  |e|d c d e|  |e|d c d e|  |e|d c|d e|  |e|d c|f e|
.-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.  .-.-.-.-.-.
```

Shorten Path

```
. - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .
| g | f   e   f | e |  | g | f   e   f | e |  | g | f   e   f | e |  | g | f   e   f | e |  | g | n   o   p | e |  | g | n   o   p | e |  | o | n   o   p | e |  | o | n   o   p | e |
.   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .
| f | e | b   c   d |  | f | e | b   c   d |  | f | e | b   c   d |  | f | m | b   c   d |  | f | m | b   c   d |  | f | m | b   c   d |  | n | m | b   c   d |  | n | m | b   c   d |
.   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .
| e | d | a   b   c |  | e | d | a   b   c |  | e | d | a   b   c |  | e | l | a   b   c |  | e | l | a   b   c |  | e | l | a   b   c |  | m | l | a   b   c |  | m | l | a   b   c |
.   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .
| d   e | d   e | d |  | d   e | d   e | d |  | d   e | h   g | d |  | d   k | h   g | d |  | d   k | h   g | d |  | d   k | h   g | d |  | l   k | h   g | d |  | l   k | h   g | d |
.   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .
| e | d   c | f   e |  | e | d   c | f   e |  | e | d   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | m | j   i | f   e |
. - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .

. - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .
| g | f   e   f | e |  | g | f   e   f | e |  | g | f   e   f | e |  | g | n   e   f | e |  | g | n   o   p | e |  | g | n   o   p | e |  | o | n   o   p | e |
.   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .
| f | e | b   c   d |  | f | e | b   c   d |  | f | e | b   c   d |  | f | m | b   c   d |  | f | m | b   c   d |  | f | m | b   c   d |  | n | m | b   c   d |
.   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .
| e | d | a   b   c |  | e | d | a   b   c |  | e | d | a   b   c |  | e | l | a   b   c |  | e | l | a   b   c |  | e | l | a   b   c |  | m | l | a   b   c |
.   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .
| d   e | d   e | d |  | d   e | d   e | d |  | d   e | h   g | d |  | d   k | h   g | d |  | d   k | h   g | d |  | l   k | h   g | d |  | l   k | h   g | d |
.   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .
| e | d   c | f   e |  | e | d   c | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |
. - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .

. - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .
| g | f   e   f | e |  | g | f   e   f | e |  | g | f   e   f | e |  | g | n   o   f | e |  | g | n   o   p | e |  | g | n   o   p | e |  | o | n   o   p | e |
.   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .
| f | e | b   c   d |  | f | e | b   c   d |  | f | e | b   c   d |  | f | m | b   c   d |  | f | m | b   c   d |  | f | m | b   c   d |  | n | m | b   c   d |
.   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .
| e | d | a   b   c |  | e | d | a   b   c |  | e | d | a   b   c |  | e | l | a   b   c |  | e | l | a   b   c |  | m | l | a   b   c |  | m | l | a   b   c |
.   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .
| d   e | d   e | d |  | d   e | d   g | d |  | d   k | h   g | d |  | d   k | h   g | d |  | d   k | h   g | d |  | l   k | h   g | d |  | l   k | h   g | d |
.   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .
| e | d   c | f   e |  | e | d   c | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |
. - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .

. - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .
| g | f   e   f | e |  | g | f   e   f | e |  | g | f   e   f | e |  | g | n   o   p | e |  | g | n   o   p | e |  | g | n   o   p | e |  | o | n   o   p | e |
.   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .
| f | e | b   c   d |  | f | e | b   c   d |  | f | e | b   c   d |  | f | m | b   c   d |  | f | m | b   c   d |  | n | m | b   c   d |  | n | m | b   c   d |
.   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .  .   .   . - .   .   .
| e | d | a   b   c |  | e | d | a   b   c |  | e | l | a   b   c |  | e | l | a   b   c |  | e | l | a   b   c |  | m | l | a   b   c |  | m | l | a   b   c |
.   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .  .   .   . - . - .   .
| d   e | d   e | d |  | d   e | h   g | d |  | d   k | h   g | d |  | d   k | h   g | d |  | d   k | h   g | d |  | l   k | h   g | d |  | l   k | h   g | d |
.   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .  .   .   .   .   .   .
| e | d   c | f   e |  | e | d   c | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |  | e | j   i | f   e |
. - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .  . - . - . - . - . - .
```

Appendix-5: Gantt chart.



Project Management

Gantt Chart

ID	WBS	Task Name	Start	Finish
1	1	Project Definition	Mon 17/09/12	Sun 23/09/12
2	1.1	Project Area	Mon 17/09/12	Tue 18/09/12
3	1.2	Search for Suitable Project	Wed 19/09/12	Sun 23/09/12
4	2	Project Proposal	Mon 24/09/12	Wed 10/10/12
5	2.1	Research and Investigation	Mon 24/09/12	Mon 01/10/12
6	2.2	Project Proposal Templates	Tue 02/10/12	Mon 08/10/12
7	2.3	Practice for Presentation	Tue 09/10/12	Wed 10/10/12
8	2.4	Presentation Day	Wed 10/10/12	Wed 10/10/12
9	3	Implementation	Thu 11/10/12	Sun 07/04/13
10	3.1	Continue Research	Thu 11/10/12	Sun 28/10/12
11	3.2	Order Components	Thu 11/10/12	Sun 28/10/12
12	3.3	Draw and Build the Circuit	Sun 21/10/12	Sun 10/02/13
13	3.4	Programming	Sun 04/11/12	Sun 24/02/13
14	3.5	Test and Fault Correction	Mon 25/02/13	Sun 03/03/13
15	3.6	PCB Design	Mon 04/03/13	Fri 22/03/13
16	3.7	Further Development	Sat 23/03/13	Sun 07/04/13
17	4	Intermediate Demonstration	Sat 01/12/12	Fri 14/12/12
18	4.1	Summary	Sat 01/12/12	Tue 04/12/12
19	4.2	Visual Aides	Wed 05/12/12	Sat 08/12/12
20	4.3	Preparation and Practice	Sun 09/12/12	Thu 13/12/12
21	4.4	Demonstration Day	Fri 14/12/12	Fri 14/12/12
22	5	Final Report	Mon 25/02/13	Tue 26/03/13
23	5.1	First Draft	Mon 25/02/13	Sun 03/03/13
24	5.2	Write Full Report	Mon 04/03/13	Sat 23/03/13
25	5.3	Referencing	Wed 20/03/13	Mon 25/03/13
26	5.4	Submission	Tue 26/03/13	Tue 26/03/13
27	6	Final Demonstration	Mon 01/04/13	Fri 19/04/13
28	6.1	Summary	Mon 01/04/13	Sun 07/04/13
29	6.2	Visual Aids	Sun 07/04/13	Thu 11/04/13
30	6.3	Preparation and Practice	Fri 12/04/13	Thu 18/04/13
31	6.4	Demonstration Day	Thu 18/04/13	Thu 18/04/13

Done by: Mohamed Alsubaie
Project: Maze Solving Robot
Date: 01/10/2012

Appendix-6: Left hand wall algorithm code.

```
//=====Arrays Declarations=====
char Maze[11][11] = { {'.', '-', '.', '-', '.', '-', '.', '-', '.', '-', '.'},
                      {'|', ' ', '|', ' ', ' ', ' ', ' ', ' ', '|', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', '-', '.', '-', '.', ' ', '.'},
                      {'|', ' ', '|', ' ', '|', ' ', ' ', ' ', '|', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', '-', '.', ' ', '.', ' ', '.'},
                      {'|', ' ', '|', ' ', '|', ' ', ' ', ' ', ' ', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', '-', '.', '-', '.', ' ', '.'},
                      {'|', ' ', ' ', ' ', '|', ' ', ' ', ' ', '|', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', ' ', '|', ' ', ' ', ' ', '|', ' ', ' ', ' ', '|'},
                      {'.', '-', '.', '-', '.', '-', '.', '-', '.', '-', '.'} };

char LHW[11][11] = {  {'.', '-', '.', '-', '.', '-', '.', '-', '.', '-', '.'},
                      {'|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|'},
                      {'.', '-', '.', '-', '.', '-', '.', '-', '.', '-', '.'} };

//=====Declarations & Inital values=====
int x = 0;
int y = 0;
int xp = 1;
int yp = 9;
int uTurn = 0;
int i = 0;
char direc = 'S';

//=====Print Maze Function=====
void PrintMaze()
{
   //=====Print-out the maze=====
   x = 0;
   y = 0;
   while(x < 11)
   {
      Serial.print(Maze[y][x]);
      Serial.print(' ');
      x++;

      if(x > 10)
      {
         Serial.print("\n");
         y++;
         x = 0;
         if(y > 10)
         {
           x = 11;
         }
      }
   }
   Serial.print("\n");
   Serial.print("\n");
}

//=====Print LHW Function=====
void PrintLHW()
{
   //=====Print-out the LHW=====
   x = 0;
   y = 0;
   while((x < 11) && (i < 33))
   {
     Serial.print(LHW[y][x]);
     Serial.print(' ');
     x++;

     if(x > 10)
     {
        Serial.print("\n");
        y++;
        x = 0;
        if(y > 10)
        {
          x = 11;
        }
     }
   }
   Serial.print("\n");
   Serial.print("\n");
}

void setup()
{
   Serial.begin(9600); // for serial monitor output

   delay(500);

   //=====MAZE Titel=====
   Serial.print("MAZE");
   Serial.print("\n");
   PrintMaze();     //Call Print Maze Function

   //=====LHW Titel=====
   Serial.print("Left Hand Wall");
   Serial.print("\n");
   LHW[9][1] = 'M';
   PrintLHW();      //Call Print LHW Function
}



void loop()
{
   while(i < 33)
   {
     if (xp == 5 && yp == 5){i = 40;}

     i++;
     LHW[yp][xp] = 'X';

     if(direc == 'S')
     {
       LHW[yp][xp-1] = Maze[yp][xp-1];
       LHW[yp-1][xp] = Maze[yp-1][xp];
       LHW[yp][xp+1] = Maze[yp][xp+1];

       if(LHW[yp][xp-1] == ' '){direc = 'L';}
       else if(LHW[yp-1][xp] == ' '){direc = 'S';}
       else if(LHW[yp][xp+1] == ' '){direc = 'R';}
       else{direc = 'D'; uTurn = 1;}
     }

     else if(direc == 'D')
     {
       LHW[yp][xp+1] = Maze[yp][xp+1];
       LHW[yp+1][xp] = Maze[yp+1][xp];
       LHW[yp][xp-1] = Maze[yp][xp-1];

       if(LHW[yp][xp+1] == ' '){direc = 'R';}
       else if(LHW[yp+1][xp] == ' '){direc = 'D';}
       else if(LHW[yp][xp-1] == ' '){direc = 'L';}
       else{direc = 'S'; uTurn = 1;}
     }

     else if(direc == 'L')
     {
       LHW[yp+1][xp] = Maze[yp+1][xp];
       LHW[yp][xp-1] = Maze[yp][xp-1];
       LHW[yp-1][xp] = Maze[yp-1][xp];

       if(LHW[yp+1][xp] == ' '){direc = 'D';}
       else if(LHW[yp][xp-1] == ' '){direc = 'L';}
       else if(LHW[yp-1][xp] == ' '){direc = 'S';}
       else{direc = 'R'; uTurn = 1;}
     }

     else if(direc == 'R')
     {
       LHW[yp-1][xp] = Maze[yp-1][xp];
       LHW[yp][xp+1] = Maze[yp][xp+1];
       LHW[yp+1][xp] = Maze[yp+1][xp];

       if(LHW[yp-1][xp] == ' '){direc = 'S';}
       else if(LHW[yp][xp+1] == ' '){direc = 'R';}
       else if(LHW[yp+1][xp] == ' '){direc = 'D';}
       else{direc = 'L'; uTurn = 1;}
     }

     LHW[yp][xp] = 'X';
     if(direc == 'S'){yp = yp-2;}
     else if(direc == 'D'){yp = yp+2;}
     else if(direc == 'L'){xp = xp-2;}
     else if(direc == 'R'){xp = xp+2;}
     LHW[yp][xp] = 'M';

     PrintLHW(); //Call Print LHW Function
   }
   while(1);
}
```

Appendix-7: Modified fill flood algorithm code.

```
//=====Arrays Declarations=====
/*
char Maze[11][11] = { {'.', '-', '.', '-', '.', '-', '.', '-', '.', '-', '.'},
                      {'|', ' ', '|', ' ', ' ', ' ', ' ', ' ', '|', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', '-', '.', '-', '.', ' ', '.'},
                      {'|', ' ', '|', ' ', '|', ' ', ' ', ' ', '|', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', '-', '.', ' ', '.', ' ', '.'},
                      {'|', ' ', '|', ' ', '|', ' ', ' ', ' ', ' ', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', '-', '.', '-', '.', ' ', '.'},
                      {'|', ' ', ' ', ' ', '|', ' ', ' ', ' ', '|', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', ' ', '|', ' ', ' ', ' ', '|', ' ', ' ', ' ', '|'},
                      {'.', '-', '.', '-', '.', '-', '.', '-', '.', '-', '.'} };
*/

char Maze[11][11] = { {'.', '-', '.', '-', '.', '-', '.', '-', '.', '-', '.'},
                      {'|', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', '-', '.', '-', '.', ' ', '.'},
                      {'|', ' ', '|', ' ', '|', ' ', ' ', ' ', '|', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', '-', '.', ' ', '.', ' ', '.'},
                      {'|', ' ', '|', ' ', ' ', ' ', '|', ' ', ' ', ' ', '|'},
                      {'.', ' ', '.', ' ', '.', '-', '.', '-', '.', ' ', '.'},
                      {'|', ' ', '|', ' ', '|', ' ', ' ', ' ', '|', ' ', '|'},
                      {'.', ' ', '.', '_', '.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', ' ', ' ', ' ', ' ', ' ', '|', ' ', ' ', ' ', '|'},
                      {'.', '-', '.', '-', '.', '-', '.', '-', '.', '-', '.'} };

char FFA[11][11] = {  {'.', '-', '.', '-', '.', '-', '.', '-', '.', '-', '.'},
                      {'|', 'e', ' ', 'd', ' ', 'c', ' ', 'd', ' ', 'e', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', 'd', ' ', 'c', ' ', 'b', ' ', 'c', ' ', 'd', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', 'c', ' ', 'b', ' ', 'a', ' ', 'b', ' ', 'c', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', 'd', ' ', 'c', ' ', 'b', ' ', 'c', ' ', 'd', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', 'e', ' ', 'd', ' ', 'c', ' ', 'd', ' ', 'e', '|'},
                      {'.', '-', '.', '-', '.', '-', '.', '-', '.', '-', '.'} };


//=====Declarations & Inital values=====
int x = 0;
int y = 0;
int xp = 1;
int yp = 9;
int i = 0;
char direc = 'S';
char S_value;
char D_value;
char L_value;
char R_value;
char current_value;
int U_turn;
int total;

//=====Print Maze Function=====
void PrintMaze()
{
   //=====Print-out the maze=====
   x = 0;
   y = 0;
   while(x < 11)
   {
      Serial.print(Maze[y][x]);
      Serial.print(' ');
      x++;

      if(x > 10)
      {
         Serial.print("\n");
         y++;
         x = 0;
         if(y > 10)
         {
           x = 11;
         }
      }
   }
   Serial.print("\n");
   Serial.print("\n");
}

//=====Print LHW Function=====
void PrintFFA()
{
   //=====Print-out the FFA=====
   x = 0;
   y = 0;
   while((x < 11) && (i < 33))
   {
     Serial.print(FFA[y][x]);
     Serial.print(' ');
     x++;

     if(x > 10)
     {
        Serial.print("\n");
        y++;
        x = 0;
        if(y > 10)
        {
          x = 11;
        }
     }
   }
   Serial.print("\n");
   Serial.print("\n");
}

void setup()
{
   Serial.begin(9600); // for serial monitor output

   delay(500);
   Serial.print("\n");

   //=====MAZE Titel=====
   Serial.print("MAZE");
   Serial.print("\n");
   PrintMaze();     //Call Print Maze Function

   //=====LHW Titel=====
   Serial.print("Fill Flood Algorithem");
   Serial.print("\n");
   current_value = FFA[yp][xp];
   //FFA[yp][xp] = 'M';
   PrintFFA();      //Call Print FFA Function
            FFA[yp][xp] = 'M';
}

void loop()
{
   while(i < 40)
   {
      S_value = 125;
      L_value = 125;
      R_value = 125;
      D_value = 125;

      if(direc == 'S')
      {
        FFA[yp-1][xp] = Maze[yp-1][xp];
        FFA[yp][xp-1] = Maze[yp][xp-1];
        FFA[yp][xp+1] = Maze[yp][xp+1];
        FFA[yp+1][xp] = Maze[yp+1][xp];
```

```
  if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
  if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
  if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
  if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}


  if(S_value <= L_value && S_value <= R_value && S_value != 125){direc = 'S';}
  else if(L_value <= S_value && L_value <= R_value && L_value != 125){direc = 'L';}
  else if(R_value <= S_value && R_value <= L_value && R_value != 125){direc = 'R';}
  else{direc = 'D';}
}

else if(direc == 'D')
{
  FFA[yp+1][xp] = Maze[yp+1][xp];
  FFA[yp][xp+1] = Maze[yp][xp+1];
  FFA[yp][xp-1] = Maze[yp][xp-1];
  FFA[yp-1][xp] = Maze[yp-1][xp];

  if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
  if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
  if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
  if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}

  if(D_value <= L_value && D_value <= R_value && D_value != 125){direc = 'D';}
  else if(R_value <= D_value && R_value <= L_value && R_value != 125){direc = 'R';}
  else if(L_value <= D_value && L_value <= R_value && L_value != 125){direc = 'L';}
  else{direc = 'S';}
}

else if(direc == 'L')
{
  FFA[yp+1][xp] = Maze[yp+1][xp];
  FFA[yp][xp+1] = Maze[yp][xp+1];
  FFA[yp][xp-1] = Maze[yp][xp-1];
  FFA[yp-1][xp] = Maze[yp-1][xp];

  if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
  if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
  if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
  if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}

  if(L_value <= S_value && L_value <= D_value && L_value != 125){direc = 'L';}
  else if(S_value <= L_value && S_value <= D_value && S_value != 125){direc = 'S';}
  else if(D_value <= L_value && D_value <= S_value && D_value != 125){direc = 'D';}
  else{direc = 'R';}
}

else if(direc == 'R')
{
  FFA[yp+1][xp] = Maze[yp+1][xp];
  FFA[yp][xp+1] = Maze[yp][xp+1];
  FFA[yp][xp-1] = Maze[yp][xp-1];
  FFA[yp-1][xp] = Maze[yp-1][xp];

  if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
  if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
  if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
  if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}

  if(R_value <= S_value && R_value <= D_value && R_value != 125){direc = 'R';}
  else if(D_value <= R_value && D_value <= S_value && D_value != 125){direc = 'D';}
  else if(S_value <= R_value && S_value <= D_value && S_value != 125){direc = 'S';}
  else{direc = 'L';}
}

            PrintFFA();      //Call Print FFA Function
            if (xp == 5 && yp == 5){i = 50;}
if(i != 50)
{
  if(direc == 'S')
  {
    if(S_value > current_value)
    {
            current_value = S_value + 1;
          }

          FFA[yp][xp] = current_value;
          yp = yp - 2;
        }

        else if(direc == 'D')
        {
          if(D_value > current_value)
          {
            current_value = D_value + 1;
          }

          FFA[yp][xp] = current_value;
          yp = yp + 2;
        }

        else if(direc == 'L')
        {
          if(L_value > current_value)
          {
            current_value = L_value + 1;
          }

          FFA[yp][xp] = current_value;
          xp = xp - 2;
        }

        else if(direc == 'R')
        {
          if(R_value > current_value)
          {
            current_value = R_value + 1;
          }

          FFA[yp][xp] = current_value;
          xp = xp + 2;
        }

        current_value = FFA[yp][xp];
        FFA[yp][xp] = 'M';
        //PrintFFA();      //Call Print FFA Function
        i++;
      }
  }

  i = 0;
  Serial.println("==========================");

  FFA[5][5] = 'a';
  delay(200);
  PrintFFA();      //Call Print FFA Function

  //if(direc=='S'){direc = 'D'}  //might needed (change back the direction becose it inverted
in the last proccess)
  Serial.println("==========================");

  i = 0;
  xp = 5;
  yp = 5;

  while(i < 40)
   {
      S_value = 125;
      L_value = 125;
      R_value = 125;
      D_value = 125;

      if(direc == 'S')
      {
        FFA[yp-1][xp] = Maze[yp-1][xp];
        FFA[yp][xp-1] = Maze[yp][xp-1];
        FFA[yp][xp+1] = Maze[yp][xp+1];
        FFA[yp+1][xp] = Maze[yp+1][xp];

        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}

        if(S_value <= L_value && S_value <= R_value && S_value != 125){direc = 'S';}
        else if(L_value <= S_value && L_value <= R_value && L_value != 125){direc = 'L';}
        else if(R_value <= S_value && R_value <= L_value && R_value != 125){direc = 'R';}
        else{direc = 'D'; U_turn = 1;}
      }

      else if(direc == 'D')
      {
        FFA[yp+1][xp] = Maze[yp+1][xp];
        FFA[yp][xp+1] = Maze[yp][xp+1];
        FFA[yp][xp-1] = Maze[yp][xp-1];
        FFA[yp-1][xp] = Maze[yp-1][xp];

        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}

        if(D_value <= L_value && D_value <= R_value && D_value != 125){direc = 'D';}
        else if(R_value <= D_value && R_value <= L_value && R_value != 125){direc = 'R';}
        else if(L_value <= D_value && L_value <= R_value && L_value != 125){direc = 'L';}
        else{direc = 'S'; U_turn = 1;}
      }

      else if(direc == 'L')
      {
        FFA[yp+1][xp] = Maze[yp+1][xp];
        FFA[yp][xp+1] = Maze[yp][xp+1];
        FFA[yp][xp-1] = Maze[yp][xp-1];
        FFA[yp-1][xp] = Maze[yp-1][xp];

        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}

        if(L_value <= S_value && L_value <= D_value && L_value != 125){direc = 'L';}
        else if(S_value <= L_value && S_value <= D_value && S_value != 125){direc = 'S';}
        else if(D_value <= L_value && D_value <= S_value && D_value != 125){direc = 'D';}
        else{direc = 'R'; U_turn = 1;}
      }

      else if(direc == 'R')
      {
        FFA[yp+1][xp] = Maze[yp+1][xp];
        FFA[yp][xp+1] = Maze[yp][xp+1];
        FFA[yp][xp-1] = Maze[yp][xp-1];
        FFA[yp-1][xp] = Maze[yp-1][xp];

        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}

        if(R_value <= S_value && R_value <= D_value && R_value != 125){direc = 'R';}
        else if(D_value <= R_value && D_value <= S_value && D_value != 125){direc = 'D';}
        else if(S_value <= R_value && S_value <= D_value && S_value != 125){direc = 'S';}
        else{direc = 'L'; U_turn = 1;}
      }

      PrintFFA();      //Call Print FFA Function
      if (xp == 1 && yp == 9){i = 50;}

      total = 0;
      if(S_value==125){total = total + 1;}
      if(D_value==125){total = total + 1;}
      if(L_value==125){total = total + 1;}
```

```
    if(R_value==125){total = total + 1;}

    if(total < 2)
    {
      U_turn = 0;
    }

    if(direc == 'S')
    {
      if(S_value < FFA[yp][xp] && U_turn==0)
      {
        FFA[yp-2][xp] = FFA[yp][xp] + 1;
      }

      yp = yp - 2;
    }

    else if(direc == 'D')
    {
      if(D_value < FFA[yp][xp] && U_turn==0)
      {
        FFA[yp+2][xp] = FFA[yp][xp] + 1;
      }

      yp = yp + 2;
    }

    else if(direc == 'L')
    {
      if(L_value < FFA[yp][xp] && U_turn==0)
      {
        FFA[yp][xp-2] = FFA[yp][xp] + 1;
      }

      xp = xp - 2;
    }

    else if(direc == 'R')
    {
      if(R_value < FFA[yp][xp] && U_turn==0)
      {
        FFA[yp][xp+2] = FFA[yp][xp] + 1;
      }

      xp = xp + 2;
    }

    //current_value = FFA[yp][xp];
    //FFA[yp][xp] = 'M';
    //PrintFFA();      //Call Print FFA Function
    i++;

                      //PrintFFA();      //Call Print FFA Function
}

    Serial.println("==========================");
i = 0;
xp = 1;
yp = 9;
direc = 'S';
FFA[yp][xp] = 'X';

while(i < 40)
 {
    S_value = 125;
    L_value = 125;
    R_value = 125;
    D_value = 125;

    if(direc == 'S')
    {
      FFA[yp-1][xp] = Maze[yp-1][xp];
      FFA[yp][xp-1] = Maze[yp][xp-1];
      FFA[yp][xp+1] = Maze[yp][xp+1];
```

```
  FFA[yp+1][xp] = Maze[yp+1][xp];

  if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
  if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
  if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
  if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}


  if(S_value <= L_value && S_value <= R_value && S_value != 125){direc = 'S';}
  else if(L_value <= S_value && L_value <= R_value && L_value != 125){direc = 'L';}
  else if(R_value <= S_value && R_value <= L_value && R_value != 125){direc = 'R';}
  else{direc = 'D';}
}

else if(direc == 'D')
{
  FFA[yp+1][xp] = Maze[yp+1][xp];
  FFA[yp][xp+1] = Maze[yp][xp+1];
  FFA[yp][xp-1] = Maze[yp][xp-1];
  FFA[yp-1][xp] = Maze[yp-1][xp];

  if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
  if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
  if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
  if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}

  if(D_value <= L_value && D_value <= R_value && D_value != 125){direc = 'D';}
  else if(R_value <= D_value && R_value <= L_value && R_value != 125){direc = 'R';}
  else if(L_value <= D_value && L_value <= R_value && L_value != 125){direc = 'L';}
  else{direc = 'S';}
}

else if(direc == 'L')
{
  FFA[yp+1][xp] = Maze[yp+1][xp];
  FFA[yp][xp+1] = Maze[yp][xp+1];
  FFA[yp][xp-1] = Maze[yp][xp-1];
  FFA[yp-1][xp] = Maze[yp-1][xp];

  if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
  if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
  if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
  if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}

  if(L_value <= S_value && L_value <= D_value && L_value != 125){direc = 'L';}
  else if(S_value <= L_value && S_value <= D_value && S_value != 125){direc = 'S';}
  else if(D_value <= L_value && D_value <= S_value && D_value != 125){direc = 'D';}
  else{direc = 'R';}
}

else if(direc == 'R')
{
  FFA[yp+1][xp] = Maze[yp+1][xp];
  FFA[yp][xp+1] = Maze[yp][xp+1];
  FFA[yp][xp-1] = Maze[yp][xp-1];
  FFA[yp-1][xp] = Maze[yp-1][xp];

  if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
  if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
  if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
  if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}

  if(R_value <= S_value && R_value <= D_value && R_value != 125){direc = 'R';}
  else if(D_value <= R_value && D_value <= S_value && D_value != 125){direc = 'D';}
  else if(S_value <= R_value && S_value <= D_value && S_value != 125){direc = 'S';}
  else{direc = 'L';}
}

            //PrintFFA();      //Call Print FFA Function
            //if (xp == 3 && yp == 3){i = 50;}

if(i != 50)
{
  if(direc == 'S')
```

```
        {
          //FFA[yp][xp] = 'X';
          yp = yp - 2;
        }

        else if(direc == 'D')
        {
          //FFA[yp][xp] = 'X';
          yp = yp + 2;
        }

        else if(direc == 'L')
        {
          //FFA[yp][xp] = 'X';
          xp = xp - 2;
        }

        else if(direc == 'R')
        {
          //FFA[yp][xp] = 'X';
          xp = xp + 2;
        }

        //current_value = FFA[yp][xp];
        FFA[yp][xp] = 'X';
        //PrintFFA();      //Call Print FFA Function
        i++;

        PrintFFA();      //Call Print FFA Function
        if (xp == 5 && yp == 5){i = 50;}
      }
   }
  i = 0;
  //delay(500);
  PrintFFA();      //Call Print FFA Function
  //delay(500);
  PrintFFA();      //Call Print FFA Function
  while(1);
}
```

Appendix-8: Robot code.

```
//////////the cod might have some modification in order to be reduced. //Headers And Libraries
#include <IRremote.h>

#define receiver 3                //Receiver pin
#define pwm_L            5               //Left Motor Speed Pin
#define pwm_R            10      //Right Motor Speed Pin
#define dir_L            13              //Left Motor Direction Pin
#define dir_R            12              //Right Motor Direction Pin

//=====Arrays Declarations=====
//Fill Flood Maze
char FFA[11][11] = {  {'.', '-', '.', '-', '.', '-', '.'},
                      {'|', 'c', ' ', 'b', ' ', 'c', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', 'b', ' ', 'a', ' ', 'b', '|'},
                      {'.', ' ', '.', ' ', '.', ' ', '.'},
                      {'|', 'c', ' ', 'b', ' ', 'c', '|'},
                      {'.', '-', '.', '-', '.', '-', '.'}};

//=====Declarations & Inital values=====
int x = 0;                               //X axis used for printing the maze
int y = 0;                               //Y axis used for printing the maze
int xp = 1;                              //X position of the Robot
int yp = 5;                              //Y position of the Robot
int i = 0;                               //conter
char direc = 'S';                //direction of the robot
char S_value;                    //straigt cell value
char D_value;                    //down cell value
char L_value;                    //left cell value
char R_value;                    //right cell value
char current_value;              //current cell value
int U_turn;                              //when robot needs take a u turn
int total;                               //calcultated result

int wall_threshold=800;  //wall detaction
int active = 400;                //active readings for the wall
int speed_L;                     //left motor speed
int speed_R;                     //right motor speed
int move;                                //integer for moving foward
int RC;                                  //Remote Control

//sensors values
int left_sensor;
int front_sensor;
int right_sensor;

//Reciver functions
IRrecv irrecv(receiver);
decode_results results;

//=====Print Maze Function=====
void PrintMaze()
{
   //=====Print-out the maze=====
   x = 0;
   y = 0;
   while(x < 11)
   {
      Serial.print(Maze[y][x]);
      Serial.print(' ');
      x++;

      if(x > 10)
      {
         Serial.print("\n");
         y++;
         x = 0;
         if(y > 10)
         {
           x = 11;
         }
      }
   }
   Serial.print("\n");
   Serial.print("\n");
}

//=====Print LHW Function=====
void PrintFFA()
{
   //=====Print-out the FFA=====
   x = 0;
   y = 0;
   while((x < 11) && (i < 33))
   {
     Serial.print(FFA[y][x]);
     Serial.print(' ');
     x++;

     if(x > 10)
     {
        Serial.print("\n");
        y++;
        x = 0;
        if(y > 10)
        {
          x = 11;
        }
     }
   }
   Serial.print("\n");
   Serial.print("\n");
}

//=====stop Function=====
void stop()
{
  //set the speed
  speed_L = 0;
  speed_R = 0;
}

//=====move straight Function=====
void straight()
{
        move = 0;
        while(move < 9)
          //Read rensors
          left_sensor = analogRead(0);
          front_sensor = analogRead(1);
          right_sensor = analogRead(2);

          //set both motor high
          digitalWrite(dir_L, HIGH);
          digitalWrite(dir_R, HIGH);
          speed_L = 60;
          speed_R = 60;

          //move slightly to the left if the right sensor is near to the wall
          if(right_sensor > active)
          {
                speed_L = 0;
                speed_R = 60;
          }

          //move slightly to the right if the left sensor is near to the wall
          if(left_sensor > active)
          {
                speed_L = 60;
                speed_R = 0;
          }

          analogWrite(pwm_L, speed_L);
          analogWrite(pwm_R, speed_R);
          move++;
          delay(100);
```

```
}

//=====turn let Function=====
void turnLeft()
{
  //set the speed
  speed_L = 60;
  speed_R = 60;

  //set the direction
  digitalWrite(dir_L, LOW);
  digitalWrite(dir_R, HIGH);
  analogWrite(pwm_L, speed_L);
  analogWrite(pwm_R, speed_R);

  delay(600);
}

void turnRight()
{
  //set the speed
  speed_L = 60;
  speed_R = 60;

  //set the direction
  digitalWrite(dir_L, HIGH);
  digitalWrite(dir_R, LOW);
  analogWrite(pwm_L, speed_L);
  analogWrite(pwm_R, speed_R);

  delay(600);
}

//=====u turn Function=====
void uTurn()
{
  turnLeft();
  delay(1000);
}

//=====Remot control Function=====
void RemoteControl()
{
  RC = 0;
  while(RC == 0)
  {
    if (irrecv.decode(&results))                         //wate for result
    {
      Serial.println(results.value, HEX);                //print result
      irrecv.resume();

      if(results.value == 0xFFA25D){RC = 1;}     //if this kee pressed RC will be high
    }
  }
}

void setup()
{
   Serial.begin(9600);           // for serial monitor output
   irrecv.enableIRIn();          // enable reciver

   //set the outputs pins
   pinMode(pwm_L, OUTPUT);
   pinMode(pwm_R, OUTPUT);
   pinMode(dir_L, OUTPUT);
   pinMode(dir_R, OUTPUT);

   delay(500);
   Serial.print("\n");

   //=====MAZE Titel=====
   Serial.print("MAZE");
   Serial.print("\n");
   PrintMaze();                          //Call Print Maze Function
   //=====LHW Titel=====
   Serial.print("Fill Flood Algorithem");
   Serial.print("\n");
   current_value = FFA[yp][xp];
   PrintFFA();                           //Call Print FFA Function
   FFA[yp][xp] = 'M';
}

void loop()
{
   while(i < 40)
   {
          //Read rensors
      left_sensor = analogRead(0);
          front_sensor = analogRead(1);
      right_sensor = analogRead(2);

          //set all cells values to any high number, just to compare
          S_value = 125;
      L_value = 125;
      R_value = 125;
      D_value = 125;

          //if the current direction is straight
      if(direc == 'S')
      {
                if(front_sensor > wall_threshold){FFA[yp-1][xp] == '-'}          //check
front sensor
                else{FFA[yp-1][xp] == ' '}

                if(left_sensor > wall_threshold){FFA[yp][xp-1] == '|'}           //check left
sensor
                else{FFA[yp][xp-1] == ' '}

                if(right_sensor > wall_threshold){FFA[yp][xp+1] == '|'}          //check
right sensor
                else{FFA[yp][xp+1] == ' '}

                //only if there is no wall the cells values will be change to the fill flood
value
                //else the cell value will be high 125
                if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}

                //see if the robot needs to make a turn, and set the direction
        if(S_value <= L_value && S_value <= R_value && S_value != 125){direc = 'S';}
        else if(L_value <= S_value && L_value <= R_value && L_value != 125){direc = 'L';
turnLeft();}    //turn Left
        else if(R_value <= S_value && R_value <= L_value && R_value != 125){direc = 'R';
turnRight();}   //turn Right
        else{direc = 'D'; uTurn();}      //u turn
      }

          //if the current direction is down
      else if(direc == 'D')
      {
        if(front_sensor > wall_threshold){FFA[yp+1][xp] == '-'}           //check front sensor
                else{FFA[yp+1][xp] == ' '}

                if(left_sensor > wall_threshold){FFA[yp][xp+1] == '|'}           //check left
sensor
                else{FFA[yp][xp+1] == ' '}

                if(right_sensor > wall_threshold){FFA[yp][xp-1] == '|'}          //check
right sensor
                else{FFA[yp][xp-1] == ' '}

                //only if there is no wall the cells values will be change to the fill flood
value
                //else the cell value will be high 125
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}

                //see if the robot needs to make a turn, and set the direction
        if(D_value <= L_value && D_value <= R_value && D_value != 125){direc = 'D';}
        else if(R_value <= D_value && R_value <= L_value && R_value != 125){direc = 'R';
turnLeft();}    //Turn Left
        else if(L_value <= D_value && L_value <= R_value && L_value != 125){direc = 'L';
turnRight();}   //Turn Right
        else{direc = 'S'; uTurn();}      //u turn
      }

          //if the current direction is left
      else if(direc == 'L')
      {
        if(front_sensor > wall_threshold){FFA[yp][xp-1] == '-'}          //check front sensor
                else{FFA[yp][xp-1] == ' '}

                if(left_sensor > wall_threshold){FFA[yp+1][xp] == '|'}          //check left
sensor
                else{FFA[yp+1][xp] == ' '}

                if(right_sensor > wall_threshold){FFA[yp-1][xp] == '|'}          //check
right sensor
                else{FFA[yp-1][xp] == ' '}

                //only if there is no wall the cells values will be change to the fill flood
value
                //else the cell value will be high 125
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}

                //see if the robot needs to make a turn, and set the direction
        if(L_value <= S_value && L_value <= D_value && L_value != 125){direc = 'L';}
        else if(S_value <= L_value && S_value <= D_value && S_value != 125){direc = 'S';
turnRight();} //Turn Right
        else if(D_value <= L_value && D_value <= S_value && D_value != 125){direc = 'D';
turnLeft();} //Turn Left
        else{direc = 'R'; uTurn();} //u turn
      }

      else if(direc == 'R')
      {
        if(front_sensor > wall_threshold){FFA[yp][xp+1] == '-'}          //check front sensor
                else{FFA[yp][xp+1] == ' '}

                if(left_sensor > wall_threshold){FFA[yp-1][xp] == '|'}          //check left
sensor
                else{FFA[yp_1][xp] == ' '}

                if(right_sensor > wall_threshold){FFA[yp+1][xp] == '|'}          //check
right sensor
                else{FFA[yp+1][xp] == ' '}

                //only if there is no wall the cells values will be change to the fill flood
value
                //else the cell value will be high 125
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}

                //see if the robot needs to make a turn, and set the direction
        if(R_value <= S_value && R_value <= D_value && R_value != 125){direc = 'R';}
        else if(D_value <= R_value && D_value <= S_value && D_value != 125){direc = 'D';
turnRight();} //Turn Right
        else if(S_value <= R_value && S_value <= D_value && S_value != 125){direc = 'S';
turnLeft();} //Turn Left
        else{direc = 'L'; uTurn();} //u turn
      }
      PrintFFA();                                                         //Call Print FFA 
Function
      if (xp == 3 && yp == 3){i = 50;}           //stop if reach the goal

          if(i != 50)                                                     //if not reaching 
the goal continue
      {
                //if the direction is straight, check if the value needs to be updated
        if(direc == 'S')
        {
          if(S_value > current_value){current_value = S_value + 1;}
          FFA[yp][xp] = current_value;
          yp = yp - 2;
        }

                //if the direction is down, check if the value needs to be updated
        else if(direc == 'D')
        {
          if(D_value > current_value){current_value = D_value + 1;}
          FFA[yp][xp] = current_value;
          yp = yp + 2;
        }

                //if the direction is left, check if the value needs to be updated
        else if(direc == 'L')
        {
          if(L_value > current_value){current_value = L_value + 1;}
          FFA[yp][xp] = current_value;
          xp = xp - 2;
        }

                //if the direction is right, check if the value needs to be updated
        else if(direc == 'R')
        {
          if(R_value > current_value){current_value = R_value + 1;}
          FFA[yp][xp] = current_value;
          xp = xp + 2;
        }

                //just to make the caracter M  visible  on the result
        current_value = FFA[yp][xp];     //store the current value in this integer
        FFA[yp][xp] = 'M';                               //replace it with "M"


                straight();                                               //move straight
                //delay(1000);
                stop()
                delay(50);

        i++;                                                      //continue
      }
  }

  //shorten path proccess

  Serial.println("==========================");

  //return the "a" value
  FFA[3][3] = 'a';
  delay(200);
  PrintFFA();      //Call Print FFA Function

  Serial.println("==========================");

  i = 0;        //counter set to 0
  //set the algoreithm to start from the centre
  xp = 3;
  yp = 3;

  while(i < 40)
   {
          /////////these will be used after, for making the robot return back to
          /////////start position by it self
          //Read rensors
      //left_sensor = analogRead(0);
          //front_sensor = analogRead(1);
      //right_sensor = analogRead(2);

          //set all cells values to any high number, just to compare
      S_value = 125;
      L_value = 125;
      R_value = 125;
      D_value = 125;

          //if the current direction is straight
      if(direc == 'S')
      {
                //only if there is no wall the cells values will be change to the fill flood
value
                //else the cell value will be high 125
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}

                //see if the robot needs to make a turn, and set the direction
        if(S_value <= L_value && S_value <= R_value && S_value != 125){direc = 'S';}
        else if(R_value <= S_value && R_value <= L_value && R_value != 125){direc = 'R';}
        else if(L_value <= S_value && L_value <= R_value && L_value != 125){direc = 'L';}
        else{direc = 'D'; U_turn = 1;}
      }

      else if(direc == 'D')
      {
                //only if there is no wall the cells values will be change to the fill flood
value
                //else the cell value will be high 125
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}

                //see if the robot needs to make a turn, and set the direction
        if(D_value <= L_value && D_value <= R_value && D_value != 125){direc = 'D';}
        else if(L_value <= D_value && L_value <= R_value && L_value != 125){direc = 'L';}
        else if(R_value <= D_value && R_value <= L_value && R_value != 125){direc = 'R';}
        else{direc = 'S'; U_turn = 1;}
      }

      else if(direc == 'L')
      {
                //only if there is no wall the cells values will be change to the fill flood
value
                //else the cell value will be high 125
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}

                //see if the robot needs to make a turn, and set the direction
        if(L_value <= S_value && L_value <= D_value && L_value != 125){direc = 'L';}
        else if(S_value <= L_value && S_value <= D_value && S_value != 125){direc = 'S';}
                else if(D_value <= R_value && D_value <= S_value && D_value != 125){direc =
'D';}
        else{direc = 'R'; U_turn = 1;}
      }

      else if(direc == 'R')
      {
                //only if there is no wall the cells values will be change to the fill flood
value
                //else the cell value will be high 125
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}

                //see if the robot needs to make a turn, and set the direction
        if(R_value <= S_value && R_value <= D_value && R_value != 125){direc = 'R';}
        else if(D_value <= R_value && D_value <= S_value && D_value != 125){direc = 'D';}
        else if(S_value <= R_value && S_value <= D_value && S_value != 125){direc = 'S';}
        else{direc = 'L'; U_turn = 1;}
      }

      PrintFFA();      //Call Print FFA Function
      if (xp == 1 && yp == 5){i = 50;}

      total = 0;
      if(S_value==125){total = total + 1;}
      if(D_value==125){total = total + 1;}
      if(L_value==125){total = total + 1;}
      if(R_value==125){total = total + 1;}

      if(total < 2)
      {
        U_turn = 0;
      }

      if(direc == 'S')
      {
        if(S_value < FFA[yp][xp] && U_turn==0)
        {
          FFA[yp-2][xp] = FFA[yp][xp] + 1;
        }

        yp = yp - 2;
      }

      else if(direc == 'D')
      {
        if(D_value < FFA[yp][xp] && U_turn==0)
        {
          FFA[yp+2][xp] = FFA[yp][xp] + 1;
        }

        yp = yp + 2;
      }

      else if(direc == 'L')
      {
        if(L_value < FFA[yp][xp] && U_turn==0)
        {
          FFA[yp][xp-2] = FFA[yp][xp] + 1;
        }

        xp = xp - 2;
      }

      else if(direc == 'R')
      {
        if(R_value < FFA[yp][xp] && U_turn==0)
        {
          FFA[yp][xp+2] = FFA[yp][xp] + 1;
        }

        xp = xp + 2;
      }

      i++;
  }

   i = 0;
   PrintFFA();
    Serial.println("==========================");
  i = 0;
  xp = 1;
  yp = 5;
  direc = 'S';
  FFA[yp][xp] = 'X';

  while(i < 40)
   {
```

```
          //Read rensors
      left_sensor = analogRead(0);
          front_sensor = analogRead(1);
      right_sensor = analogRead(2);

      S_value = 125;
      L_value = 125;
      R_value = 125;
      D_value = 125;

      if(direc == 'S')
      {
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}


        if(S_value <= L_value && S_value <= R_value && S_value != 125){direc = 'S';}
        else if(L_value <= S_value && L_value <= R_value && L_value != 125){direc = 'L';
turnLeft();}    //turn left
        else if(R_value <= S_value && R_value <= L_value && R_value != 125){direc = 'R';
turnRight();}   //turn right
        else{direc = 'D'; uTturn();}     //u turn
      }

      else if(direc == 'D')
      {
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}

        if(D_value <= L_value && D_value <= R_value && D_value != 125){direc = 'D';}
        else if(R_value <= D_value && R_value <= L_value && R_value != 125){direc = 'R';
turnLeft();}    //turn left
        else if(L_value <= D_value && L_value <= R_value && L_value != 125){direc = 'L';
turnRight();}   //turn right
        else{direc = 'S'; uTturn();}     //u turn
      }

      else if(direc == 'L')
      {
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}

        if(L_value <= S_value && L_value <= D_value && L_value != 125){direc = 'L';}
        else if(S_value <= L_value && S_value <= D_value && S_value != 125){direc = 'S';
turnRight();}   //turn right
        else if(D_value <= L_value && D_value <= S_value && D_value != 125){direc = 'D';
turnLeft();}    //turn left
        else{direc = 'R'; uTturn();}     //u turn
      }

      else if(direc == 'R')
      {
        if(FFA[yp][xp+1] == ' '){R_value = FFA[yp][xp+2];}
        if(FFA[yp-1][xp] == ' '){S_value = FFA[yp-2][xp];}
        if(FFA[yp+1][xp] == ' '){D_value = FFA[yp+2][xp];}
        if(FFA[yp][xp-1] == ' '){L_value = FFA[yp][xp-2];}

        if(R_value <= S_value && R_value <= D_value && R_value != 125){direc = 'R';}
        else if(D_value <= R_value && D_value <= S_value && D_value != 125){direc = 'D';
turnRight();}   //turn right
        else if(S_value <= R_value && S_value <= D_value && S_value != 125){direc = 'S';
turnLeft();}   //turn left
        else{direc = 'L'; uTturn();}     //u turn
      }

      if(i != 50)
      {
        if(direc == 'S')
        {
          yp = yp - 2;
        }

        else if(direc == 'D')
        {
          yp = yp + 2;
        }

        else if(direc == 'L')
        {
          xp = xp - 2;
        }

        else if(direc == 'R')
        {
          xp = xp + 2;
        }

        FFA[yp][xp] = 'X';
        i++;
        PrintFFA();      //Call Print FFA Function

                straight();                                              //move straight
                //delay(1000);
                stop()
                delay(50);

        if (xp == 3 && yp == 3){i = 50;}
      }
   }
  i = 0;

  PrintFFA();      //Call Print FFA Function
  while(1);
}
```