



PALESTINE POLYTECHNIC UNIVERSITY

College of IT and Computer Engineering

Department of Computer System Engineering

Graduation Project

Project Name

Maze solving robot

Project Team

Yasmeen Hunihen

Marah Karajeh

Supervisor

Dr. Alaa Halawani

Abstract

Maze solving robot aims to solve the maze by navigating from one point at the start to the other point at the end. And try to do it in the shortest path possible. We design and program a robot that depends on a Left-hand algorithm to solve the maze from the starting point to the end, and an LSRB algorithm to find the shortest path, by using C++ language to program the Arduino microcontroller. This robot is equipped with three ultrasonic sensors to check the path for right, left, and straight direction. The robot is able to move automatically in many mazes, find the shortest path between two points, calculate the traveled distance, and be able to solve the same maze in the second round in a shorter time.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION.....	9
OVERVIEW 1.1.....	9
MOTIVATIONS 1.2.....	9
OBJECTIVES 1.3.....	10
PURPOSE 1.4.....	10
PROJECT DESCRIPTION 1.5.....	10
PROJECT IMPORTANCE 1.6.....	10
REQUIREMENTS 1.7.....	11
CONSTRAINTS 1.8.....	11
EXPECTED RESULT 1.9.....	11
EXISTING WORK 1.10.....	11
CHAPTER 2: BACKGROUND.....	13
OVERVIEW 2.1.....	13
THEORETICAL BACKGROUND 2.2.....	13
PROBLEM ANALYSIS 2.3.....	13
DESIGN OPTIONS OF HARDWARE COMPONENT 2.4.....	14
Microcontroller 2.4.1.....	14
Detecting sensor 2.4.2.....	16
Motor driver 2.4.3.....	17
Raspberry Pi battery pack 2.4.4.....	19
Two wheeled robot car kit 2.4.5.....	19
SYSTEM SOFTWARE COMPONENT 2.5.....	20
CHAPTER 3: DESIGN.....	21
OVERVIEW 3.1.....	21
DETAILED SYSTEM DESCRIPTION 3.2.....	21
Software part 3.2.1.....	21
Hardware part 3.2.2.....	28
CHAPTER 4: HARDWARE AND SOFTWARE IMPLEMENTATION.....	34
OVERVIEW 4.1.....	34

ARDUINO MICROCONTROLLER 4.2.....	34
Hardware implementation 4.2.1.....	34
Software implementation 4.2.2.....	37
CHAPTER 5: VALIDATION AND TESTING.....	38
OVERVIEW 5.1.....	38
TEST 5.2.....	38
Motors test and connection 5.2.1.....	38
Ultrasonic test and connection 5.2.2.....	38
Speed Sensor Module test and connection 5.2.3.....	38
Infrared Sensor (IR) test and connection 5.2.4.....	38
IMPLEMENTING ISSUE 5.3.....	39
Hardware Issues 5.3.1.....	39
Software Issues 5.3.2.....	40
CHAPTER 6: CONCLUSION.....	41
OVERVIEW 6.1.....	41
CONCLUSION 6.2.....	41
FUTURE WORK 6.3.....	41
REFERENCES.....	42

LIST OF TABLES

- 1. *Table 2.1. Arduino Mega 2560 specification.....15***
- 2. *Table 2.3. Ultrasonic sensor specification.....17***
- 3. *Table 2.4. Motor driver specification.....18***

LIST OF FIGURES

1. <i>Figure 1.1: Simple maze</i>	9
2. <i>Figure 2.1: Arduino Mega 2560</i>	16
3. <i>Figure 2.2: Infrared Sensor</i>	16
4. <i>Figure 2.3: Ultrasonic sensor</i>	17
5. <i>Figure 2.4: Motor driver</i>	18
6. <i>Figure 2.5: Two wheeled robot car kit</i>	19
7. <i>Figure 2.6: Speed sensor module</i>	19
8. <i>Figure 3.1: Block diagram of maze solving robot</i>	21
9. <i>Figure 3.2: Flowchart of the project</i>	23
10. <i>Figure 3.4: Arduino Mega 2560 pins</i>	28
11. <i>Figure 3.5: Schematic diagram of Arduino Mega and ultrasonic interfacing</i>	29
12. <i>Figure 3.6: Schematic diagram of Arduino Mega and motor driver interfacing</i>	30
13. <i>Figure 3.7: Schematic diagram of connecting two speed sensor modules</i>	31
14. <i>Figure 3.8: Schematic diagram of connecting infrared sensor</i>	32
15. <i>Figure 3.9: Main schematic diagram</i>	33
16. <i>Figure 4.1: Connect ultrasonic sensors with microcontroller</i>	34
17. <i>Figure 4.2: Connect DC motors using a motor driver with microcontroller</i>	35
18. <i>Figure 4.3: Connect IR sensor with microcontroller</i>	36
19. <i>Figure 4.4: Connect speed sensor module with microcontroller</i>	36
20. <i>Figure 4.5: The window of Arduino program</i>	37

Abbreviations

PIC	Peripheral interface controller
EV	Evolution
RAM	Random-access memory
USB	Universal Serial Bus
GPIO	general-purpose input/output
PoE	power over ethernet
CPU	Central Processing Unit
I/O	Input/Output
GND	Ground
A	ampere
ENA	Enable pin
IN	Input

PWM	Pulse Width Modulation
IC	Integrated circuit
DC	Direct current
V	Volt
Vc	Voltage Common
VCC	Voltage Common Collector
IDE	integrated development environment
USB	Universal Serial Bus

Chapter 1 :Introduction

1.1 Overview

A maze solving robot is a universal theme that has a great importance, not only as a game, but also it has many uses in a practical life like factories and dangerous places. It is based on designing a robot able to determine its own direction of motion, to find the shortest path between two points A and B in any maze, as shown in Figure 1.1.

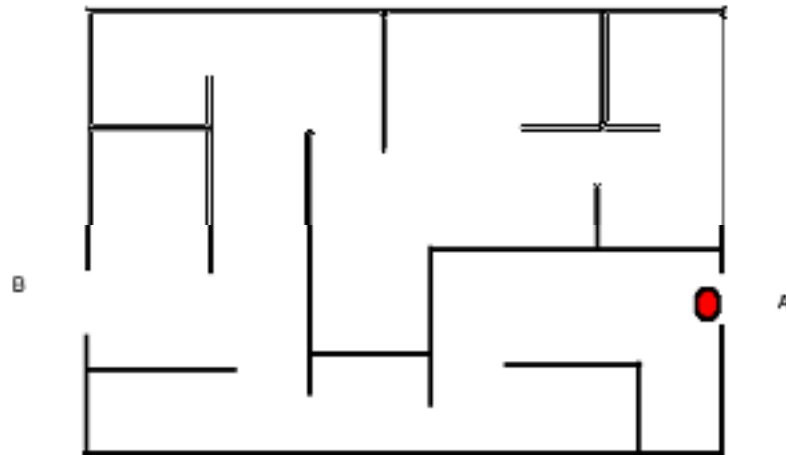


Figure 1.1: Simple maze

1.2 Motivations

The main idea in this project is to design and program a robot capable of determining the direction of its movement in different mazes to find the shortest path between the starting point A and the end point B in the first round, calculating the traveled distance, so that it can quickly go through the maze in the second round.

1.3 Objectives

The project aims to build a microcontroller-based system that is the brain of the robot, it is use to collect information from various input devices such as sensors to solve the maze, by achieving the following objectives:

- 1- Design a small scale robot that is able to move inside a maze and solve it.
- 2- Assemble the hardware pieces like sensors, motor driver, and motor encoder.
- 3- Convert the approved algorithm into a code to control the robot.
- 4- Make a small maze.

1.4 Purpose

Developments in the field of artificial intelligence have led to access a high level of time saving and performance speed, so we designed an effective robot that can move in a given maze that achieves the desired constraints to find the path between two points A and B.

1.5 Project description

This project is designed to solve the maze game. The project methodology is based on designing and programming a robot capable of finding the path that connects the starting and ending points, reaching all areas of the maze, storing these paths on the map, and the robot saving it. Then the robot calculates the distance covered by each path and it has to determine the shortest distance to reach the end, so in the second round it goes directly through the shortest path. The mazes we used in this system will contain no loops.

1.6 Project importance

There are many International competitions on this subject, in addition to the possibility of its use in navigating hazardous areas or hard to reach places such as: military rescue or navigation in minefields, navigation in toxic environments, to find the human in the wreckage of the plane, etc.

1.7 Requirements

This system achieve the following:

- 1- Designing a robot that can be able to self-move.
- 2- The robot must be able to solve different mazes.
- 3- Being able to reach the end point in the given maze from the start point.
- 4- Determine the shortest path in the given maze.

1.8 Constraints

For the project to run successfully, there are many constraints!

- 1- There are no obstacles in the way it walks, like the streets.
- 2-The size of the maze is small.

1.9 The result

After we finish this project, we have a robot designed and programmed to enter different mazes, that can be able to move automatically to find the shortest path between the start point A and endpoint B and save the map of the given maze during its movement. Also, reach its target (the endpoint) in the second round in the same maze.

1.10 Existing work

In this section we will review previous work. Following are the details:

1-Modelling and Characterization of a Maze-Solving Mobile Robot Using Wall Follower Algorithm [1].

That project aimed to design and develop a working mobile robot that is able to solve a simply connected maze. They used the wall follower algorithm to solve the maze, use proximity sensors to detect the walls of the labyrinth, and they have used a PIC microcontroller.

2-Design and Implementation of a Robot for Maze-Solving using Wall Following Algorithm[2].

In this project a maze solving robot makes multiple runs in a maze, first it creates a map of the maze layout and stores it in its memory, then runs through a shortest path. The system design of the maze solving robot consists of obstacle avoidance ultrasonic sensors and then sensors will detect the wall. To solve the maze, this robot will apply wall following algorithms such as left or right hand rule.

The maze solving robot designed in this tutorial is built on Arduino UNO.

3-Experimentation on the motion of an obstacle avoiding robot [3] .

This paper implements an experiment on one of the important fields of AI – Searching Algorithms, to find the shortest possible solution by searching the produced tree. They have concentrated on Hill climbing algorithm, which is one of simplest searching algorithms in AI. This algorithm is one of most suitable searching methods to help expert systems to make decisions at every state, at every node.

The robot used is EV3 Lego Mindstorms, with native software for programming LabView, and it has 3 ultrasonic sensors, 1 in front to sensors obstacle in front, 2 by the sides, right left – they compare distance between the robot and the wall.

But our project will be concentrate on designing robot that finds the target point and calculates the shortest path using Left-Hand algorithm. What it means is that we will turn left whenever it is possible and turn right when we are at an intersection and no other path to follow, also when we reach at dead end, turn 180 and then follow the path back from where the robot came. With ultrasonic sensors to detect the walls in maze also use Arduino Mega 22560.

Chapter 2: Background

2.1 Overview

This chapter introduces a theoretical background of the project, some description of hardware and software components used in the system. Finally, discussion of specification and design constraints are presented.

2.2 Theoretical background

This project aims to develop a robot that can be used in some cases that are difficult for a person to deal with, by developing both hardware, and software side. The delay that is caused by the difficulty to reach the goal could lead to dangerous situations that affect people lives or even our personal life such as the military rescue or navigation in minefields, navigation in toxic environments also finds the human in the wreckage of the plane.

The proposed system is supposed to solve these cases by first designing a robot that can be able to move automatically, go in the right direction inside different mazes, using ultrasonic sensors, which are set in the robot. The sensors then send the data to a microcontroller in order to process with a specific algorithm, to solve the maze. Second, the system will determine the shortest path after it walks in a given maze between two points A and B, by using a motor encoder.

2.3 Problem analysis

Many accidents happen in our world, whether it is the result of nature, such as earthquakes and devastating storms, or perhaps caused by humans such as wars and disturbances, these accidents require people to search for the survivors and help get them out from the accident site. Consider the risks that are facing the rescuer that the search is done slowly and carefully, but this affects on time, to locate the survivors, which prompts us to use this robot to search in dangerous or large environments instead of humans. On the other hand there are many competitions that are based on this topic, which leads the person to design robots that can compete and win.

2.4 Design options of Hardware components

This section describes all of the hardware used in our project. It presents a figure for each one with a short description about its work principle and why it is used in the system.

2.4.1 Microcontroller

-First design option: Raspberry Pi.

A series of small single-board computers developed by the Raspberry Pi foundation, selling outside its target market for uses such as robotics. It can be used in many applications such as communications and networking, wireless, and consumer electronics. We can install many operating systems on it, such as Windows 10 IOT, Android things, and Ubuntu MATE. There are many types of it, such as RPi pi 3 A+, RPi 3 Model B, RPi 3 Model B and RPi 4 Model B.

-Second design option: Arduino.

It is a microcontroller, refers to an open-source electronics platform or board, it is made up of two types of pins: analog and digital, it can be attached to different expansion boards (shields). Arduino programming language is a simplified form of C/C++, there are many types of Arduino boards such as Uno Rev 3, Due, Mega and Leonardo. Processor used in the Arduino is from AVR family Atmega328P.

Chosen design option: Arduino Mega 2560

Arduino mega 2560, as shown in Figure 2.1: is a microcontroller board on the ATmega2560, that have 54 digital input/output pins, 16 of these pins analog input, 15 can be used as PWM outputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It is easily connected to a computer with a USB cable and also powers it with an AC-to-DC adapter or battery to start work.

It comes pre-programmed with a bootloader that allows you to upload new code to it without using an external hardware programmer.

There are many specifications of Arduino: It is very simple to interface sensors and other electronic components to Arduino. Available for a low cost. This is a plug-and-play device because if power is connected it starts running the program and if disconnected it simply stops. Arduino makes hardware projects simple. It has no OS which means easily using it, in addition to many specifications as shown in Table 2.1. Therefore, it is great for hardware projects in which you simply want things to respond to various sensor readings so we will choose it.

On the other hand, Raspberry Pi 3 is a Single Board Computer (SBC). This means that the board is a fully functional computer with its own dedicated processor, memory, and

can run an operating system (runs on Linux). The Raspberry Pi 3 B is a mini-computer with Raspbian OS which easily connects to a lot of actuators, it includes 1G RAM, 4 USB ports to connect different devices, 40 GPIO Pins, Ethernet port, provides an SD card port, audio output, and has a graphic driver for HDMI output. Raspberry pi requires a lot of complex tasks to start using it as installing libraries, and software to link sensors. It is more expensive than the Arduino. It does not have storage space on board. Provides an SD card port. This should be properly shut down if not, there is a risk of file corruption and software problems. It is difficult to power using a battery pack.

specification	Arduino Mega 2560
Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (Recommended)	7 -12V
Input Voltage (limit)	6 -20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
Flash Memory	256 KB of which 8 KB used by bootloader
DC Current per I/O Pin	20ma

Table 2.1: Arduino Mega 2560 specification[4].



Figure 2.1 :Arduino Mega 2560[4].

2.4.2 Detection sensor

We use two sensors, one for measuring the distance of an object and the other for distinguishing specific points.

Infrared Distance sensor

IR circuits are designed on the triangulation principle for distance measurement. A transmitter sends a pulse of IR signals which is detected by the receiver if there is an obstacle and based on the angle the signal is received, distance is calculated.

It is a small electronic apparatus as shown in Figure 2.2, that measures and reveals infrared rays, emitting in order to sense some aspects of the surrounding environment. There are two parts of IR sensor: The emitter (IR LED) and a receiver (IR photodiode). We use this piece to distinguish the start and end point.



Figure 2.2: Infrared Sensor[5].

Ultrasonic sensor

Is an electronic device as shown in Figure 2.3, that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. That has two main components: the transmitter and the receiver.

Ultrasonic sensors provide fast response times which allow robots to respond quickly to the changing environments around it. Also it has many specifications as presented in Table 2.3.

This sensor can detect a wall up to 300 cm from the robot's location by sending out a pulse. Because the pulse travels at the speed of sound, the time between the pulse being sent and received can be recorded, which allows the distance between the robot and the wall to be calculated. This allows the robot to find longer and more effective paths. On the other hand, there are limitations to only using the Infrared Distance sensor for mapping distances, that it can only detect walls in close proximity to itself, so ultrasonic sensors are more reliable than IR sensors in our project ,we will use three of those left ,right ,straight sensors to discover if there is a wall and detect the path.

Supply voltage	5 V
Global Current	15 mA
Ultrasonic Frequency	40 Hz
Maximal Range	400 cm
Minimal Range	3 cm
Resolution	1 cm
Trigger Pulse Width	10 μ s
Outline Dimension	43x20x15 mm

Table 2.3: Ultrasonic sensor specification[5].



Figure 2.3: Ultrasonic sensor[6].

2.4.3 Motor driver

The motor driver L298 as shown in Figure 2.4, a type of H-Bridge, is an integrated circuit chip used as a motor controlling device in autonomous robots and embedded circuits it connects with the DC motors, used to control the direction of the motors depending on the commands it receives from the controller. Motor driver receives signals from the microprocessor and eventually, it transmits the converted signal to the motors.

Microprocessor output is low, and it cannot give enough power from its I/O pin to drive a motor. To supply this voltage/current from microprocessor to the motor, we need this Motor driver IC in between our motor and controller. Also it has some specifications shown in Table 2.4.

Motor channels	2
Maximum operating voltage	46 V
max drive current	2 A
Minimum logic voltage	4.5 V
Maximum logic voltage:	7 V
Package	15 Multi Watt

Table 2.4:Motor driver specification[7].



Figure 2.4: Motor driver[7].

2.4.4 Two wheeled robot car kit

This car kit is the ideal base for the robotic project on an Arduino or Raspberry basis. The chassis is made of acrylic and has many holes to mount components tightly, as shown in Figure 2.5. It works with two motors which need a voltage between 3 to 9 Volt. At the front is a caster wheel which is the third wheel. Through this design, it will be possible for the robots to spin on the spot. Moreover, it has a battery compartment with a switch which hugely supports the mobility of the robot.



Figure 2.5: Two wheeled robot car kit[8].

2.4.5 Speed sensor module

An electronic piece as shown in Fig.2.6, used to calculate the distance when the robot moves, it is connected with a wheels encoder, to count the holes found on the wheels encoder, through the movement of the wheels. In addition, the module can be used in association with a microcontroller for motor speed detection, pulse count, and position limit.



Figure 2.6: Speed sensor module[9].

2.5 System Software Component

The challenge in this robot and other robots is that it is impossible to know the state of the real environment. Only robot control programs can guess the state of the real world based on the measurements returned by its sensors, it can only try to change the state of the real world control signals, so in real-world robots, the software that generates the control signals is required to run at a very high speed and make complex computations. This affects the choice of which robot programming languages are best to use, using Arduino language that supports C/C++ is a very good compromise between execution speed and ease of development and testing. C/C++ is the more popular basic robot programming languages also its syntax is very clear, there are many programming languages such as python, which is a high-level, general-purpose programming language and available for many operating systems, but because we need simple and quick development tests or proof of concepts, so in our project, we will use Arduino language to control and manipulate Arduino Mega 2560 microcontroller.

Chapter 3: Design

3.1 Overview

This chapter discusses the conceptual design of the system; it shows a block diagram of the system, structural diagram, flow chart, detailed design, and schematic diagrams.

3.2 Detailed system description

3.2.1 Software part

Methodology

Figure 3.1 shows the block diagram of the maze solving robot.

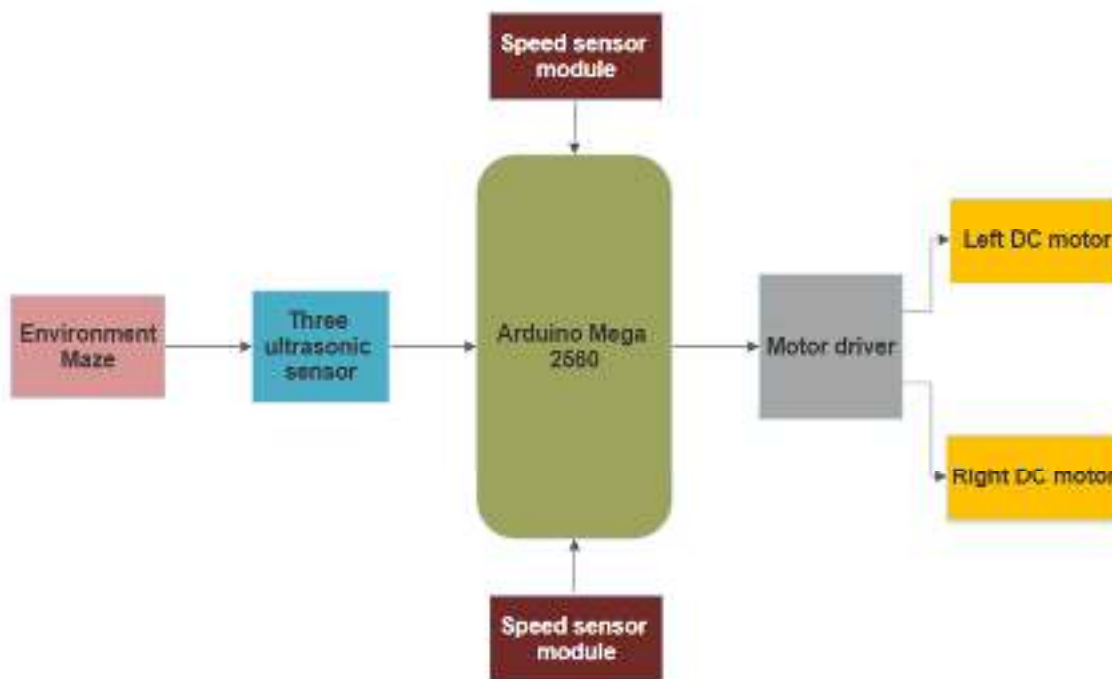


Figure 3.1: Block diagram of maze solving robot.

There are basically two steps. The first is to drive through the maze and find the end of it. The second is to optimize (Find the shortest path) that path so the robot can travel back through the maze.

The robot is programmed to drive in the maze and use ultrasonic sensors on the robot to track the path. As it travels along, the program we are using will determine which path the robot goes through in the given maze, by using a method called the "Left-Hand Rule" or sometimes called the "Left Hand on Wall" method.

Left-hand algorithm

At first, the robot will enter the maze to start, it will detect by its left sensors if this side is open to go through. If not, then the front sensor will search for an opening in front of it to complete its movement in this direction. If this direction is closed it will use the right sensor to move in this direction. Else the robot must turn around because it recognizes that it came to a dead end. So this algorithm can be simplified into these simple conditions:

- 1-Always turn left if you can.
- 2-If you cannot turn left, go straight.
- 3-If you cannot turn left, or go straight, turn right.
- 4-If you cannot turn left, go straight, or turn right, turn around.

The algorithm is shown in the flowchart illustrated in figure 3.2

As the Figure 3.2 shown (Flowchart of the project)

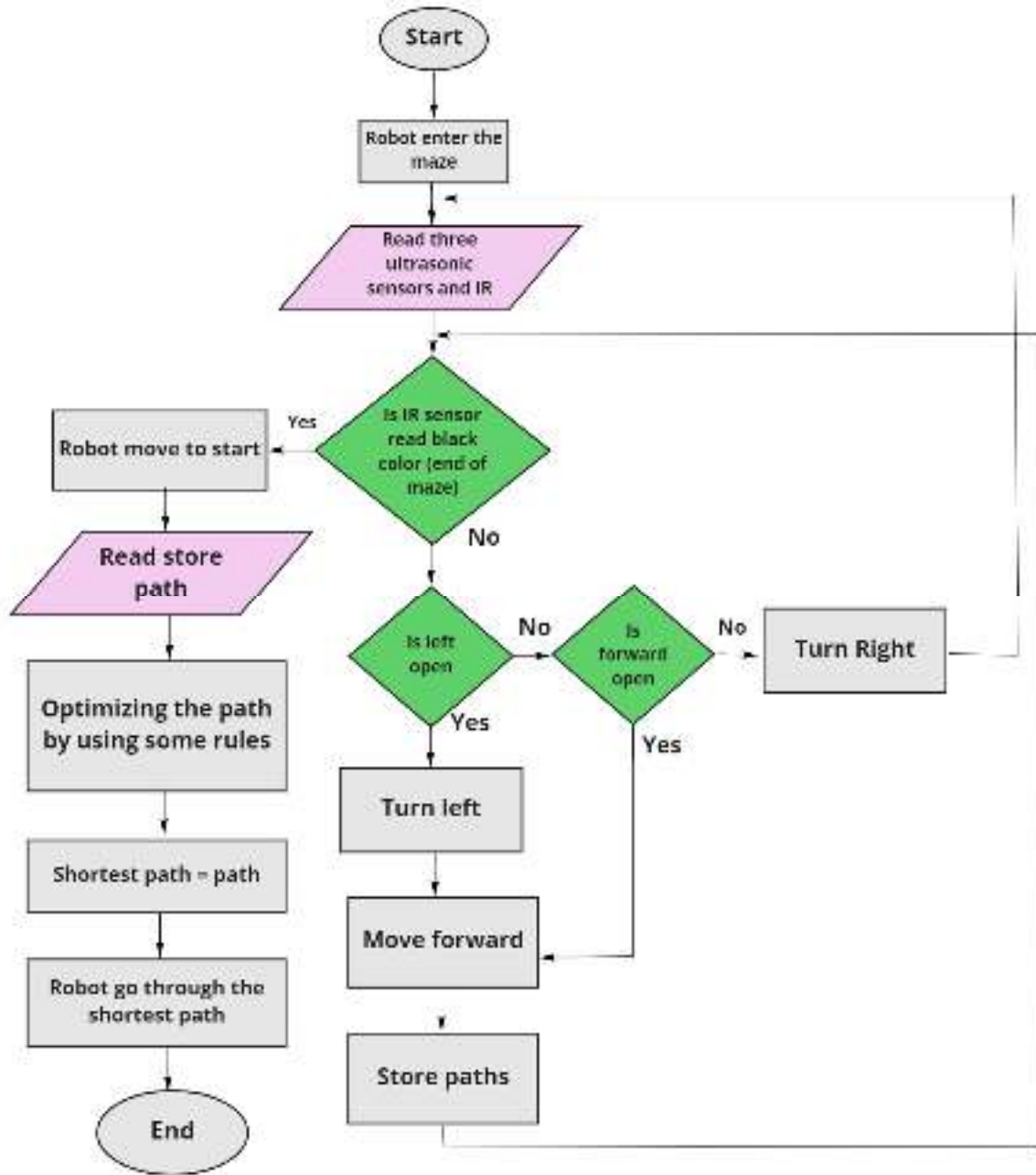


Figure 3.2: Flowchart of the project.

The robot has to make these decisions when at an intersection. An intersection is any point on the maze where you have the opportunity to turn. If the robot comes across an

opportunity to turn and does not turn then this is considered going straight. Each move was taken at an intersection or when turning around has to be stored. When the robot reaches an open region (there are no walls) it determines that it is at the endpoint of the maze.

Here are some symbols and their means:

L = left turn

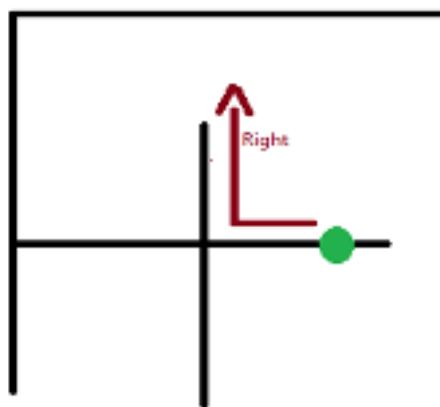
R= right turn

S= going straight past a turn

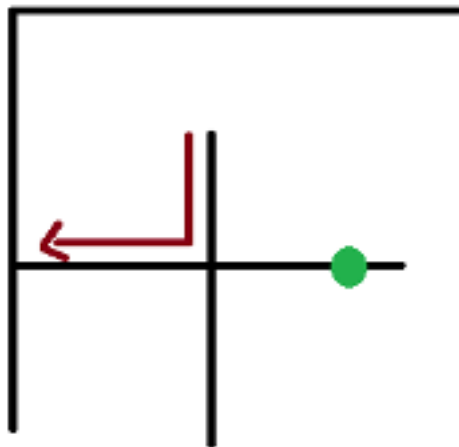
B= turning around

So let us apply this method to a simple maze and see if you can follow it. View the photos to see this method in action.

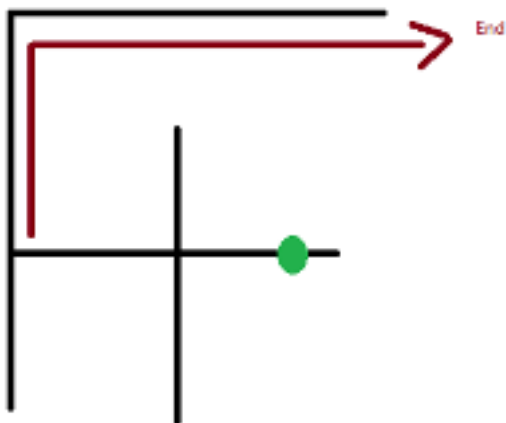
The green circle will be the robot.



Step1: S then turn R



Step2: L then S



Step3: R then R

The method of solving the maze for a shortest path is by using a **LSRB Algorithm[10]**.

These condition simplifies the LSRB algorithm:

- 1.If (the robot) can turn left then go on and turn left.
- 2.Else if it can keep driving straight then drive straight.
- 3.Else if it can turn right then turn right.
- 4.If it is at a dead end then turn around.

Based on this algorithm, the robot makes a decision to turn in a specific direction when there is an intersection. Every step taken at an intersection or when turning around has to be stored.

The LSRB algorithm simplifies the value stored in the register using the following equation shorted and follow the value of the register, which are basic rules for this algorithm, as shown:

- LBR = B
- LBS = R
- RBL = B
- SBL = R
- SBS = B
- LBL = S

If we take the full path we reduce it as these example:

path = [**L**BLLLBSBLLBSLL] → LBL = S

path = [SLL**B**SBLLBSLL] → LBS = R

path = [SLR**B**LLBSLL] → RBL = B

path = [S**L**BLBSLL] → LBL = S

path = [S**S**BSLL] → SBS = B

path = [S**B**LL] → SBL = R

path = RL

In the example we shorten the long path by using the previous equation to reach the end from the shortest path, if the robot takes Right at first intersection and after that a Left, the robot will reach the end of the maze in the shortest path.

The path is stored in an array and it goes to store a new move every time, it checks to see if the previous move was a "B", if it was then it optimizes the path. to optimize the path we need to know 3 moves at least: The move before and after the turn around (and the turn around itself).

We search in the full path about one of the possibilities that were found previously then optimize it to one path (direction). We noticed that a lot of energy and time would be saved.

If we have a:

Path = LBLLBSR

LBL = S so the path would be → SLBSR

LBS = R from the previous equation, so the new path would be → SRR

We reach the shortest path.

3.2.2 Hardware part

Detailed and Schematic diagrams

A powerful feature of arduino mega is the big I/O system design with inbuilt 16 analog pin 54 digital pin that supports a communication mode as shown in Figure 3.4, also it has more than 5 pins for ground and Vcc.

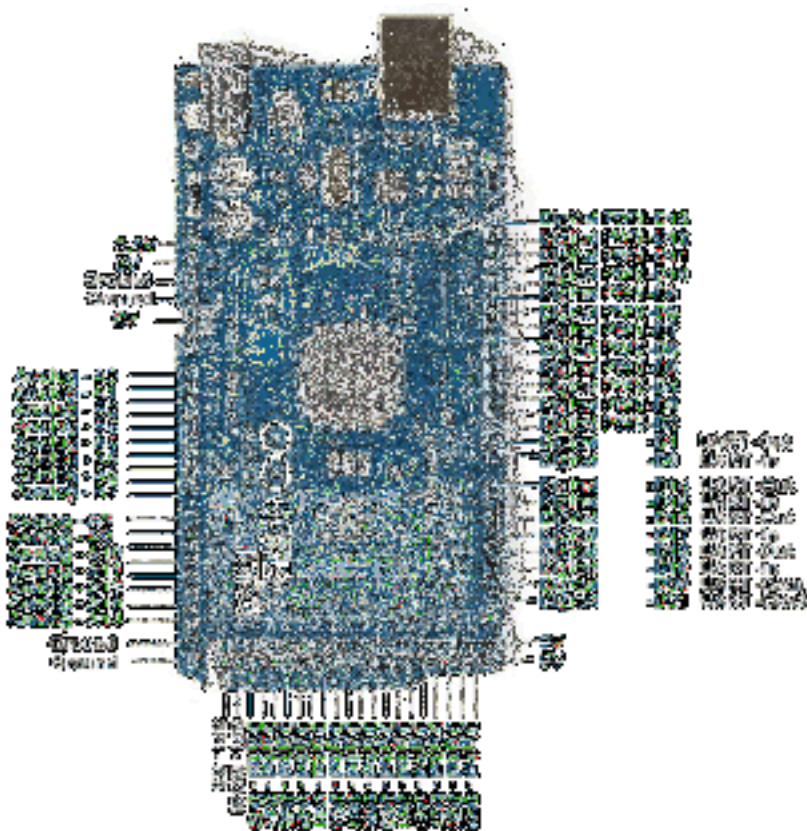


Figure 3.4: Arduino Mega 2560 pins[11].

Figure 3.5 shows the interface between three ultrasonic sensors and Arduino Mega. We connect Vcc of ultrasonic with DC power (5V) in Arduino Mega, then we connect the Trig and Echo pins of ultrasonic sensors with digital pins, and the ground of ultrasonic with the ground pin in Arduino.

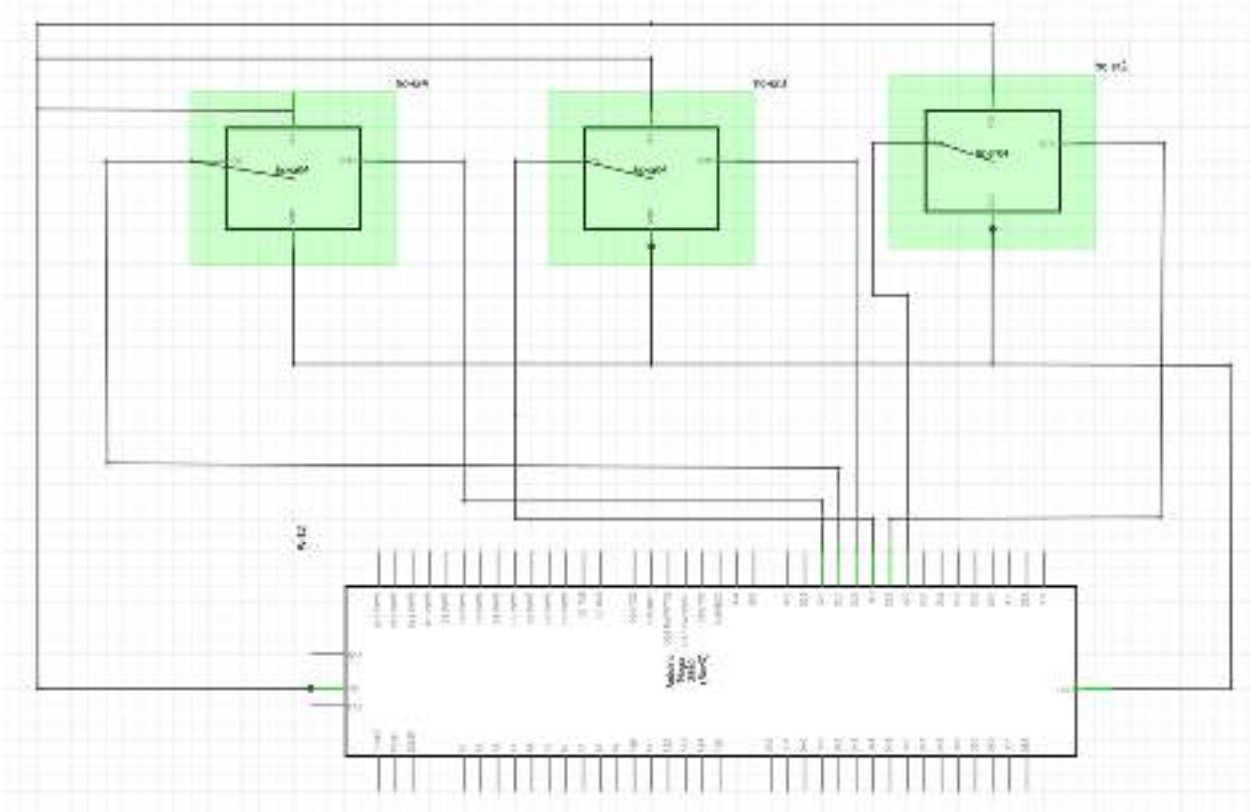


Figure 3.5: Schematic diagram of Arduino Mega and ultrasonic interfacing.

Figure 3.6 shows the Interfacing between Arduino Mega and motor driver L298N. Vc of motor driver connects with 12V battery, and the other pin with 5V from arduino board, then we connect IN1, IN2, IN3, IN4 each one with different digital pins, the GND of motor driver connected with the GND of the arduino mega and GND of battery. Each tow pin output of the motor driver connected to DC motors.

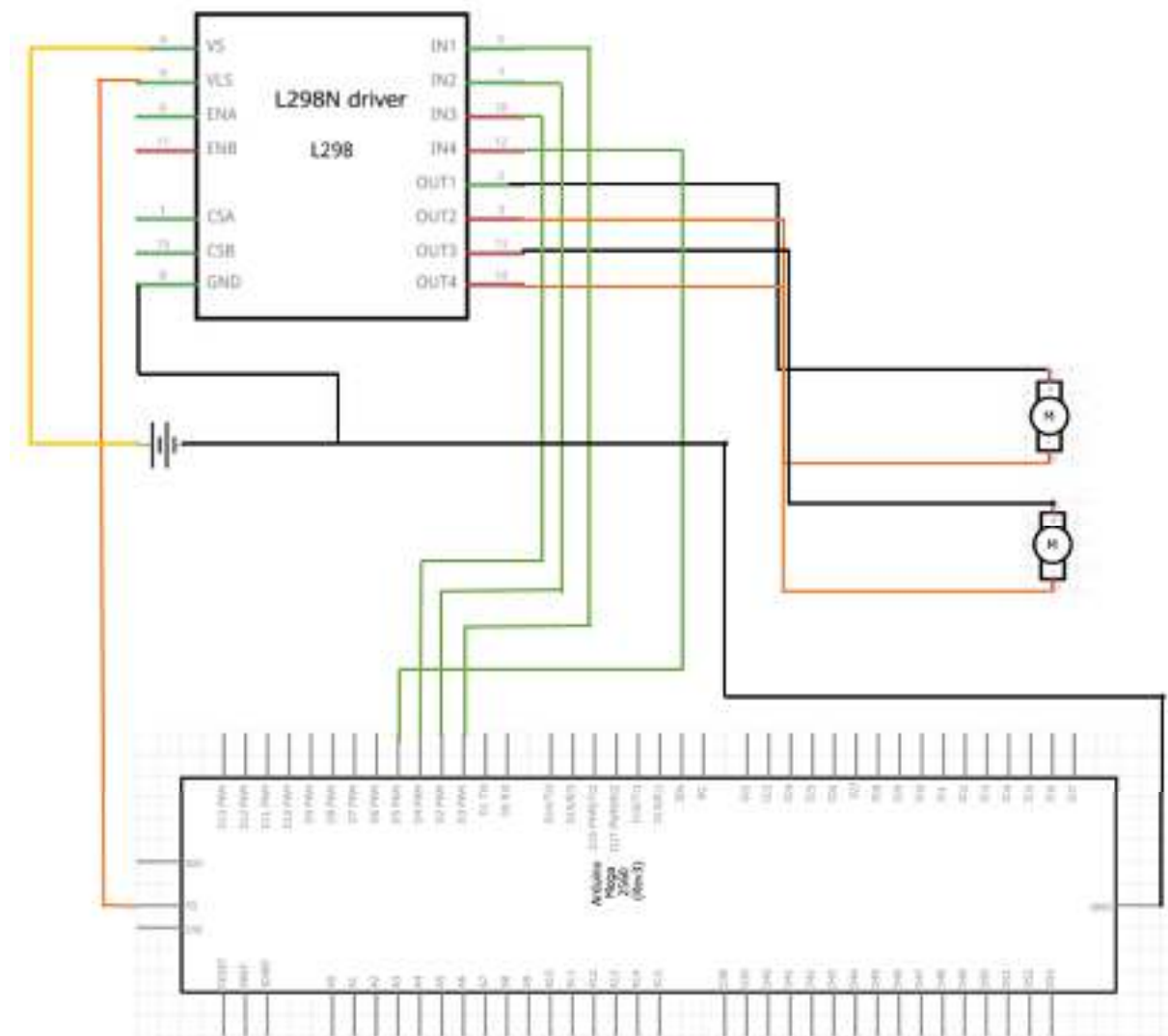


Figure 3.6: Schematic diagram of Arduino Mega and motor driver interfacing.

In Figure 3.7, we connect two speed sensor modules to the arduino, the Vcc pin of each speed sensor module connects to the 5V pin in the arduino, also the GND of each one connects to the arduino GND pin, and the digital pins of speed sensor module with a digital pins of the microcontroller.

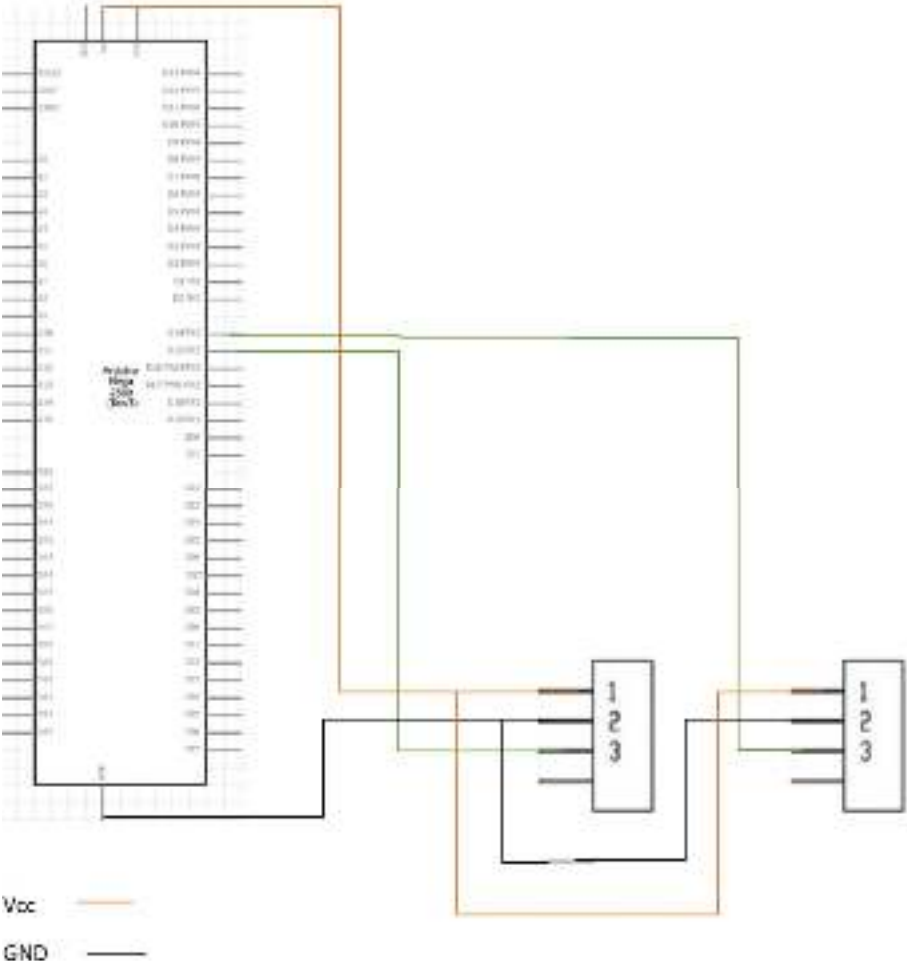


Figure 3.7: Schematic diagram of connecting two speed sensor modules.

In Figure 3.8, we connect the infrared sensor, the Vcc pin of the IR sensor connects to the 5V pin in the Arduino, also the GND to Arduino GND pin and data pin of IR sensor connect with pin digital pin in the Arduino.

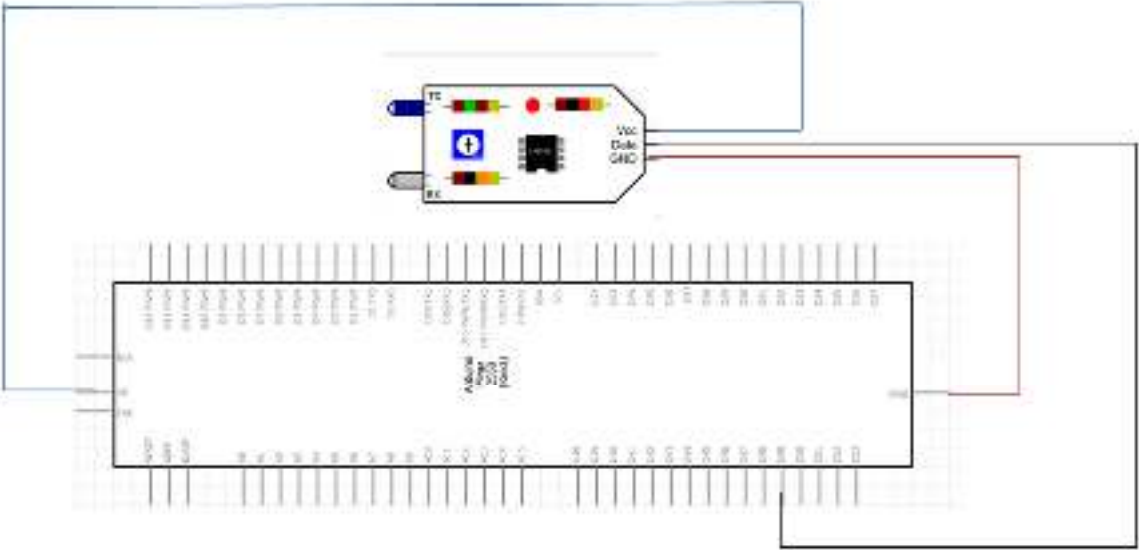


Figure 3.8: Schematic diagram of connecting infrared sensor(IR).

Figure 3.9 shows the schematic diagram of connecting all hardware components to the Arduino Mega.

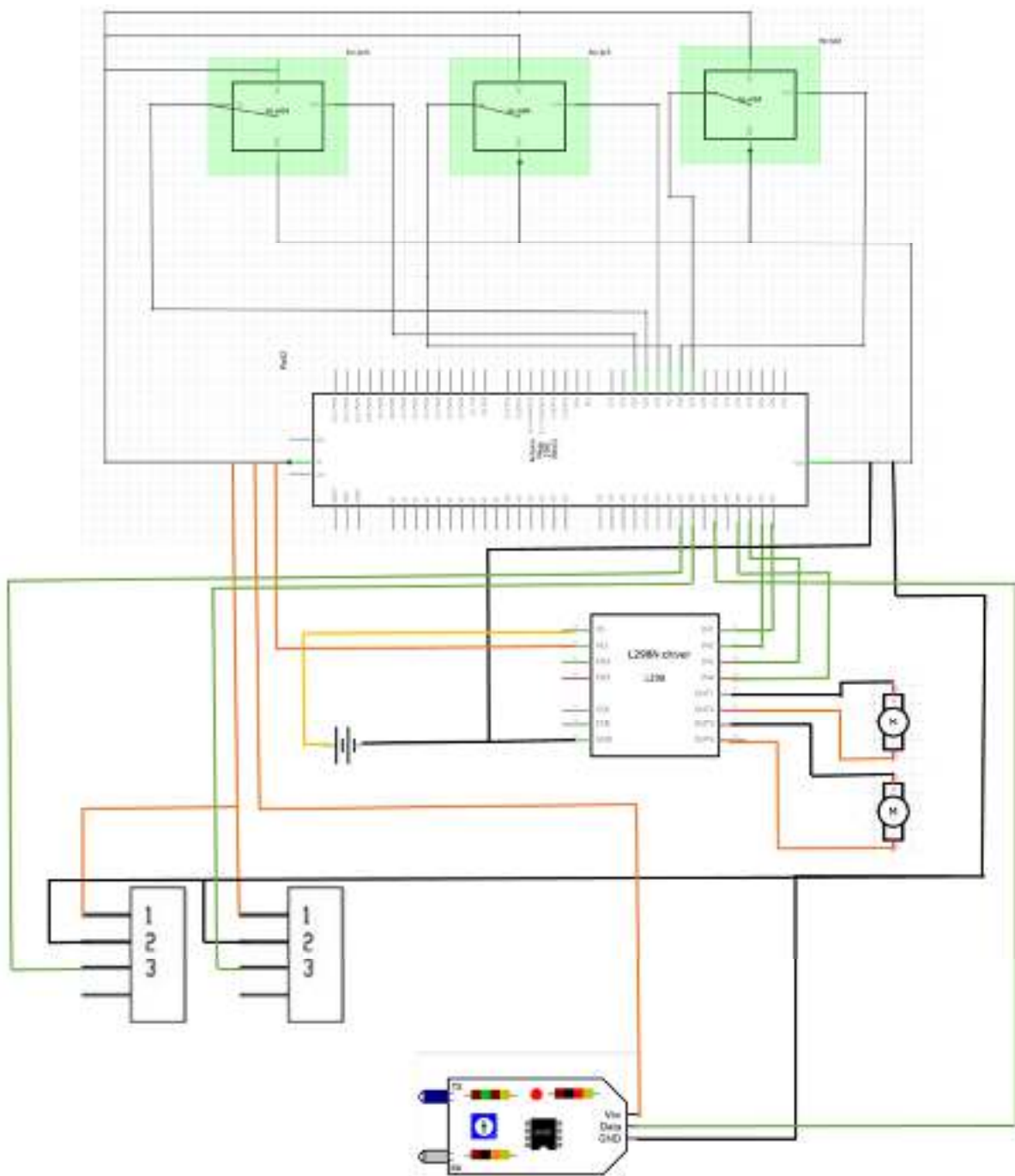


Figure 3.9: Main schematic diagram.

Chapter4: Hardware and Software Implementation

4.1 Overview

This chapter describes the project Software and Hardware implementation, and the various components and tools used to create our project.

4.2 Arduino microcontroller

Before using the arduino microcontroller we download Arduino IDE to write arduino code, then we connect the arduino microcontroller with the computer using USB connection to upload the code on it.

4.2.1 Hardware implementation:

Power configuration

The arduino mega needed a power to run the elements connected to it, so we used a 8V lithium battery.

Arduino mega microcontroller with ultrasonic sensor:

We connect the microcontroller with three ultrasonic sensors to avoid hitting the wall while the robot is walking in the maze and find the distance between the robot and wall . We put one in the forward position, the second one in the left position, and the last one in the right position. These sensors help the robot to detect which direction the robot will go through. We connect the ultrasonic sensors Trig, Echo, Ground, Vcc with the arduino mega pins, as shown in the figure 4.1.

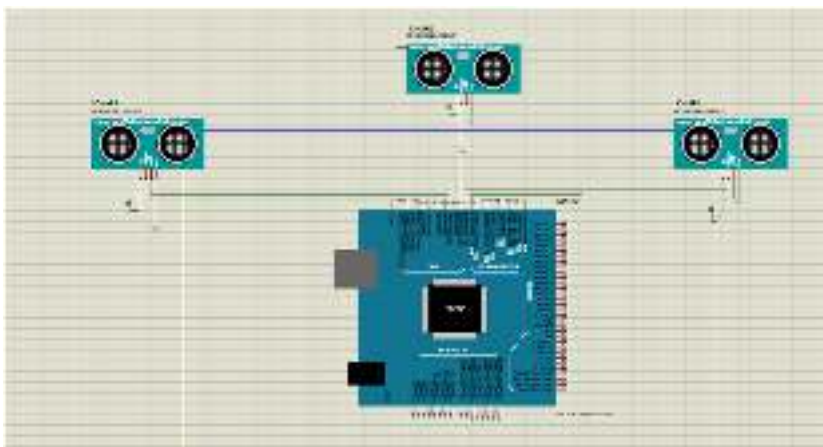


Figure 4.1: Connect ultrasonic sensors with microcontroller.

Arduino mega microcontroller with a motor driver:

we connected the microcontroller with a motor driver and connected it with 12v power as shown in the figure 4.2, and powered from a 5 volt pin in the microcontroller, it controls direction of the robot so it helps the DC motor to work, through connecting the DC motor to it.

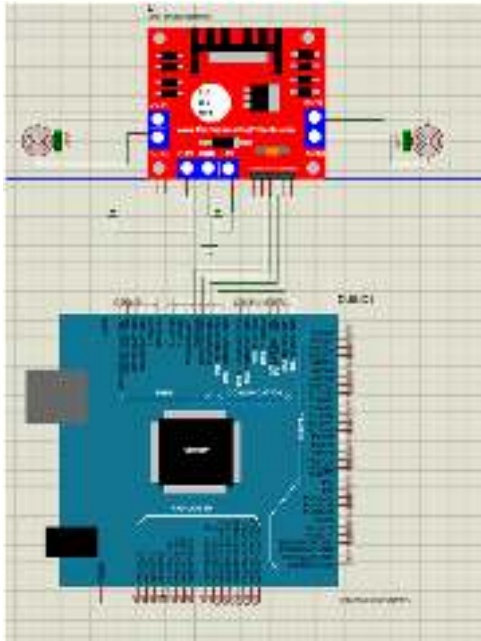


Figure 4.2: Connect DC motors using a motor driver with microcontroller.

Arduino mega microcontroller with IR sensor

We connect the infrared sensor with the microcontroller as shown in the figure 4.3, to distinguish the start point through the white color and the end point through the black color in the maze. IR sensor has three pins, the OUT pin can connect to digital or analog pin but, we connected it with digital pin of the microcontroller.

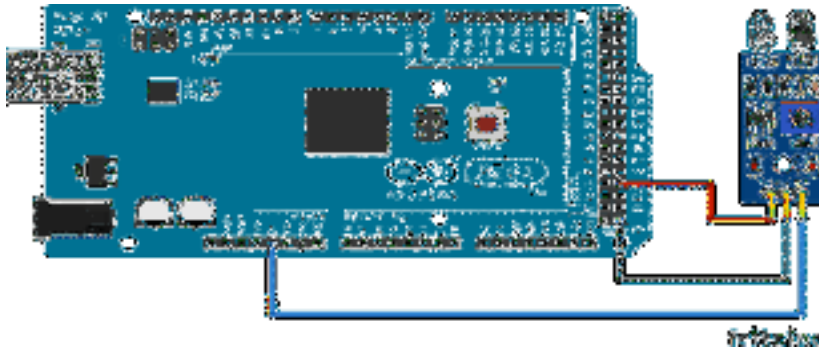


Figure 4.3: Connect IR sensor with microcontroller[12].

Arduino mega microcontroller with Speed sensor module

This module was connected to the microcontroller as shown in the figure 4.4, to calculate the distance inside the maze through the movement of the wheels with counting the slots of the wheel encoder. It includes 4 pins digital, analog, GND, Vcc pins.

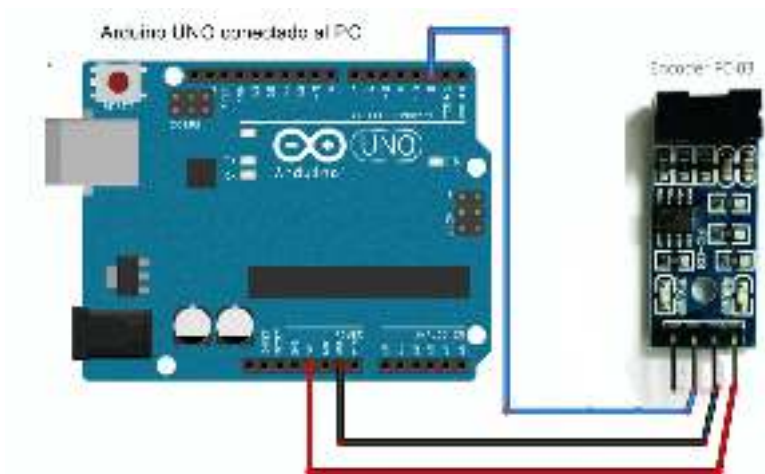


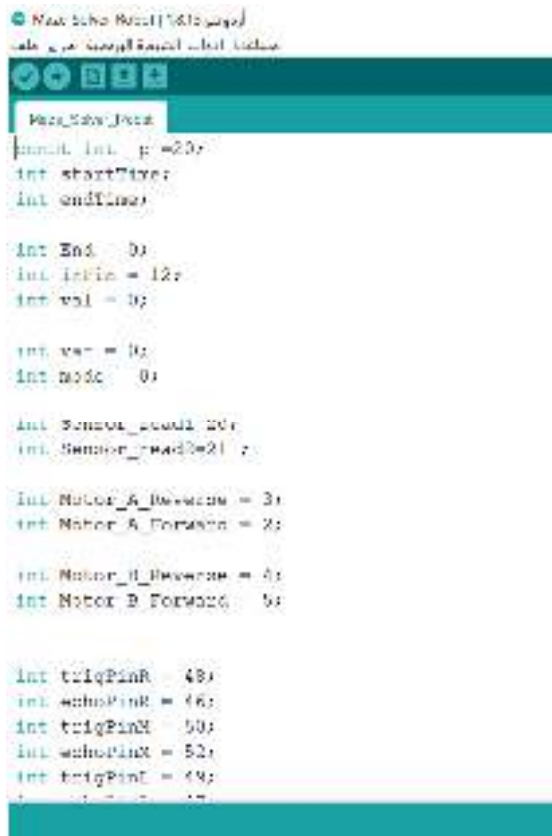
Figure 4.4: Connect speed sensor module with microcontroller[13].

We installed the arduino mega above the car kit, and installed the ultrasonic sensor in the front, right and left of the car kit. At the bottom the motor driver was installed.

4.2.2 Software implementation

The programming language used in this project is similar to C++ language it is called Arduino and we downloaded the Arduino integrated development environment(IDE) program and wrote the code.

The Figure 4.5 below shows the window of arduino program:



```
Max_Say_Pest
#include <math.h>
int startTime;
int endTime;

int End = 0;
int left = 12;
int val = 0;

int var = 0;
int mode = 0;

int Sensor_read1 = 0;
int Sensor_read2 = 1;

int Motor_A_Forward = 3;
int Motor_A_Backward = 2;

int Motor_B_Forward = 4;
int Motor_B_Backward = 5;

int trigPinR = 48;
int echoPinR = 46;
int trigPinL = 50;
int echoPinL = 52;
int trigPinE = 49;
int echoPinE = 47;
```

Figure 4.5 The window of Arduino program

The algorithm of the robot is the Left hand algorithm which does the following: if it found the intersection it is going left, else it goes forward, else that goes right. If not any one of the previous cases the robot will turn and back, after that repeat the previous steps until reaching the endpoint.

In the second round, the robot goes to the endpoint through the shortest path by using an LSRB algorithm, after doing that if there are many paths it simplifies it to only one, in order to go through the shortest one.

Chapter 5: Validation and Testing

5.1 Overview:

This chapter explains and describes the problems we faced and the results of the implementation of the system of our project.

5.2.1 Motors test and connection

For testing the motor driver and the dc motor we connect it to the ground of the Arduino microcontroller and with 9volt power supply, and connect the input1, input2 with the pins of the Arduino microcontroller, then connect the dc motor with the output channel of it, then we write Arduino code and run it. At first, the motor driver(L298N) lights up then the dc motor starts rotating in one direction, when reversing the DC motor poles it rotates in the other direction.

5.2.2 Ultrasonic test and connection

To test the ultrasonic sensor we connect the Echo and Trig pin with two pins of the microcontroller, connect the ground and Vcc of the ultrasonic with the ground and 5-volt pins of the microcontroller, and we write Arduino code and run it. We put an object in front of each one of the ultrasonic, then measure the distance between them through the sound wave. We found that the ultrasonic sensors help to keep a distance between the robot and the other objects as walls.

5.2.3 Speed Sensor Module test and connection

To test the speed sensor module we connect the D0(Digital 0) pin with the pin of the microcontroller, connect the ground and Vcc of the speed sensor module with ground and 5-volt pins of the microcontroller, then we write Arduino code to run it. When we run the robot, the wheels rotate then the speed sensor module counts the holes of the wheel encoder that is put on the wheel, then calculates the traveled distance.

5.2.4 Infrared Sensor (IR) test and connection

According to the test (IR) sensor, we connect the OUT pin with the pin of the microcontroller, connect the ground and Vcc of the Infrared sensor with the ground and 5-volt pins of the microcontroller, after that, we write Arduino code to test it. When we run the IR sensor it distinguishes the white color used for the start point and the black color for the endpoint in the maze.

5.3 Implementing issue

5.3.1 Hardware Issues

1. There are many microcontrollers that we encountered including Raspberry Pi and Arduino to choose one of them, at first we used raspberry pi We encountered difficulty in dealing with it, and then we used Arduino Uno but memory was not enough to store the path. Because of that we use Arduino Mega as it has good specifications for our project at the present time.
2. We had a problem in our testing and operation on the robot. We start to connect pieces of our robot and discover that DC motors cannot run the robot we found it needed to use an H-bridge(L298), the motor driver that controls the direction of the motor depending on the instructions coming from the microcontroller and by using it, we solved this problem.
3. One of the problems was that the robot was walking in the wrong direction, unlike what is in the code. At first, we thought that the error was from the code, so we wrote more than one code, but the result was not changing, then we checked the ultrasonic sensors and found that the error was in one of them. We brought in a new sensor to solve the problem.
Another thing, we know the maximum range of this sensor up to 300cm but we change the range of the ultrasonic sensors to 25cm, in order to detect the walls around the robot in our simple maze.
4. One of the biggest problems is how the robot should distinguish the starting and ending points. We discovered that we need an IR sensor to do this through color detection where it sends information from an infrared remote control to another device by receiving and decoding signals. In general, the receiver outputs a code to uniquely identify the infrared signal that it receives.
5. When we turn on the robot at first we use a power bank and power adapter, but they hinder the robot's movement in the maze, so we decide to use a lithium battery that can be charged.

5.3.2 Software Issues

1. To solve the maze there are many algorithms to use, we need an algorithm that makes the robot reach all points in the maze at the same time save a lot of energy and time, so we found the Left-hand algorithm that depends on turning left when it is possible to reach the endpoint.
Also, there are many algorithms to find the shortest path. At first, we choose the Dijkstra algorithm, but we found another algorithm easier to find the shortest path named LSRB algorithm depending on optimizing the paths when found an intersection and store all steps. It saves a lot of energy and time.
2. We tested the pieces of our project after writing the code of each one, but we found small problems during connecting these codes, then we changed and added some functions to solve the problem.

Chapter 6: Conclusion

6.1 Overview

In this chapter, we conclude the challenges that we faced in our project and the final results, in addition to the main goals that have been achieved, as well as future actions that can be developed on this project.

6.2 Conclusion

As a conclusion, the Left-Hand algorithm and LSRB algorithm has been implemented on the robot in order to achieve the goals of this project which is designing and programming a robot to find the path that connects the starting and ending points using the Left-Hand algorithm, reaching all areas of the maze, storing these paths on the map, and the robot saving it after that the robot calculates the distance covered by each path and it has to determine the shortest distance to reach the end using LSRB algorithm, so in the second round it goes directly through the shortest path and we design on the condition that there is no loop in the maze. The proposed algorithms have low space complexity, high performance, and provide an optimal solution to the maze.

6.3 Future Work

In the future, we will develop this robot to perform more complex operations such as solving a maze that has a loop, in addition to participating in many competitions.

References

- [1] "Modelling and Characterization of a Maze-Solving Mobile Robot Using Wall Follower Algorithm ".<https://www.researchgate.net>
- [2] "Design and Implementation of a Robot for Maze-Solving using Wall Following Algorithm".<https://ijsret.com/wpcontent/uploads>
- [3] "Experimentation on the motion of an obstacle avoiding robot".<https://arxiv.org/ftp/arxiv/papers/1907/1907>
- [4] Arduino mega 2560 specification.<https://www.google.com/search?q=arduino+mega+2560>
Arduino Mega.<https://www.google.com/search?q=Arduino+Mega+2560&source>
- [5] Infrared sensor. <https://www.google.com/search?q=Infrared+Sensor>
- [6] Ultrasonic sensor.<https://components101.com/ultrasonic-sensor>
- [7] Motor driver specification. <https://www.pololu.com>
Motor driver. <https://www.amazon.com>
- [8] Two wheeled robot car kit. <https://joy-it.net/en/products/robot05>
- [9] Speed sensor module. <https://www.amazon.com/Optocoupler>
- [10] LSRB algorithm. <https://medium.com/@TowardInfinity/coding-a-line-follower-robot-using-lsrb-and-finding-the-shortest-path>
- [11] Arduino mega 2560 pins. <https://www.google.com/search?q=arduino+mega+2560+pins>
- [12] Connect IR sensor with microcontroller.[https://www.researchgate.net/figure/nfrared - Sensor-interface](https://www.researchgate.net/figure/nfrared-Sensor-interface)
- [13] Connect speed sensor module with microcontroller. <https://www.google.com/search?q=speed+sensor+module>