Fire Fighting Car

Graduation Project

By:

Mohamed Sherif Khalifa

19-00079

Khalil Elsenosy Khalil

20-01201

Maria Amgad Anwer

20-00707

Ahmed Ramadan Ahmed

19-00251

Menna Allah Yasser

20-00952

Abd-Elaziz Anan Mohamed

19-00512

Ahmed Mahmoud Fekry Mahmoud

20-01227

Project Graduation Document Submitted in Partial Fulfillment of the Requirements of the B.SC. Degree

Fire Fighting Car

Under Supervision of

Dr. Bassem

Lecturer, Department of Information Technology at Egyptian E-Learning University (EELU)

Eng. Safinaz Aalaa Eldin

Teacher Assistant, Department of Information Technology at Egyptian E-Learning University (EELU)

Project Graduation Document Submitted in Partial Fulfillment of the Requirements of the B.SC. Degree

Acknowledgement

First and foremost, we thank God Almighty for the blessing of reaching this advanced stage in our education to achieve this result of success. We put effort into this project. However, this would not have been possible without the support and assistance of many individuals and organizations. I would like to express my sincere thanks to all of them.

Thanks to the Egyptian University for E-Learning and especially the Asyut Center for helping us reach this level of awareness.

We would like to express our gratitude to our Project Supervisor **Dr. Bassem**for giving us the opportunity to lead us and provide invaluable guidance during this project. It was a great honor and privilege for us to work under her supervision.

We are especially indebted to express our heartfelt thanks to **Eng. Safinaz Aalaa Eldin** for the guidance, constant supervision, patience, friendship and great sense of humor for all his efforts with us.

Finally, we would like to express our gratitude to everyone who helped us during the graduation project.

Abstract

Self-driving cars were invented to increase the safety of transportation users, but we have added some fire suppression methods to prevent firefighters from getting hurt or to assist them in their work.

These cars can sense their environment and make decisions without any external assistance in choosing the best way to reach a destination or extinguish a fire.

The primary goal of a fire fighter robot car is to extinguish fires and protect lives and property in emergency situations.

Fire fighter robot cars are often remote-controlled or can operate autonomously.

Firefighting robot cars are designed to work in tandem with traditional fire trucks and other firefighting apparatus.

Although the idea seems futuristic, and if implemented successfully, many current problems related to transportation, firefighting, and citizen safety will be solved, caution must be taken before implementing the solution.

This type of car relies heavily on its sensors, and any tampering or tampering with the data that is generated and transmitted through these devices can have serious consequences, and may threaten people's lives.

The Fire Fighting Car uses ultrasonic sensors for its movements.

The ultrasonic sensor is mounted in front of the robot. When the robot walks the desired path, the ultrasonic sensor continuously transmits ultrasonic waves from its sensor head.

Whenever an obstacle appears in front of it, ultrasound waves are reflected from an object and that information is passed to the microcontroller.

The microcontroller controls the left, right, reverse and forward motors, based on ultrasonic signals. To control the speed of each motor, motor pulse width modulation (PW) is used.

It also uses a flame sensor, so that when a fire occurs, this information is sent to the microcontroller to deal with it.

When the flame sensor senses some heat and at a certain temperature it releases some fire extinguishing agents such as water, foam, carbon dioxide or halon.

Table of Contents

Contents :

CHAPTER 1: Project Description

This chapter will provide an introduction to this project. These following topics are described in this chapter, general information about fire fighting vehicle, problem statement, project objective, project organization and thesis outline. Self-driving firefighting vehicles rely on sensors, actuators, complex algorithms, machine learning systems, and powerful processors to execute the software.

1.1 introduction :

Nowadays, autonomous driving is an important field of robotics. It depends on one of the most important areas of the robot, which is the "decision-making algorithm," because this robot will be placed in an unknown place, and this requires that it have good decision-making ability.

This project consists of an autonomous driving and firefighting system using ultrasonic sensors to avoid obstacles to detect walls and a flame sensor to detect fires.

In order for the robot to be able to drive itself, this robot will apply wall tracking algorithms Like the left or right hand rule, it will also follow the flood filling algorithm to find the shortest path.

1.2 Problem Statement :

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

1.3 Objectives :

- Understand and implement the wall follower and Flood fill algorithm.
- Design and build ultrasonic and flame sensors array.
- Design and build Arduino based hardware.
- Programmed the robot to self-drive and fire fighting.

- Make a small maze.

1.4 Thesis Overview :

This thesis is divided into 9 chapters. Each chapter discuss ion the different issues which related to this project .The outline of each chapter is stated in the paragraphs below.

Project statement and objectives of the project have also been described in this chapter. Chapter2 covers the important background information and history about Self-Drive. Many important theories, method and algorithms on maze solving problem is also given there.

This section will start with providing some research about Wall Following Algorithm and Fill Flood Algorithm. Chapter 3 is going to give some details for each component including features, specifications and how it operates.

Important Algorithms which is applied in this project is described in Chapter 4. Methodology, Software development, Hardware implementation is described in Chapter 5. In Chapter 6 simulated result and important discussion is described .Conclusion and Further development is described in chapter 8.

CHAPTER 2: LITERATURE REVIEW

This chapter is going to provide important background information and history about Self-Drive robot. Many important theories, methods and algorithms also given there.

2.1 Background History :

In the mid-20th century, solving autonomous driving or maze problems became an important field as the maze was the unknown path for the robot.

In 1972, the editors of IEEE Spectrum came up with the concept of the tiny mouse, a small microprocessor-controlled vehicle with independent intelligence and the ability to navigate a critical maze.

Then in May 1977, the American Little Mouse Maze Contest was announced, dubbed the "Amazing Little Mouse Maze Contest" by IEEE Spectrum. Since then, this type of competition has become more popular, and several types of maze-solving robots are developed every year.

In the late 1970s, designs of maze-solving robots were used to obtain large physical forms containing many logic blocks and gates. Figure and show an example of early robots solving a maze (the little mouse). Due to technological development, the physical size of the robot has become smaller and the features of the robot have become modern.

2.2 Algorithms :

2.2.1 Wall Following :

In 1999, Michael James, Sonia Lenz and Dirk Becker of the University of East London developed a small mouse. They used a non-graph theoretical algorithm, the Next Wall algorithm. Their robot does this, but does not move intelligently in the map and cannot solve the maze and gets into the loop.

2.2.2 Flood fill :

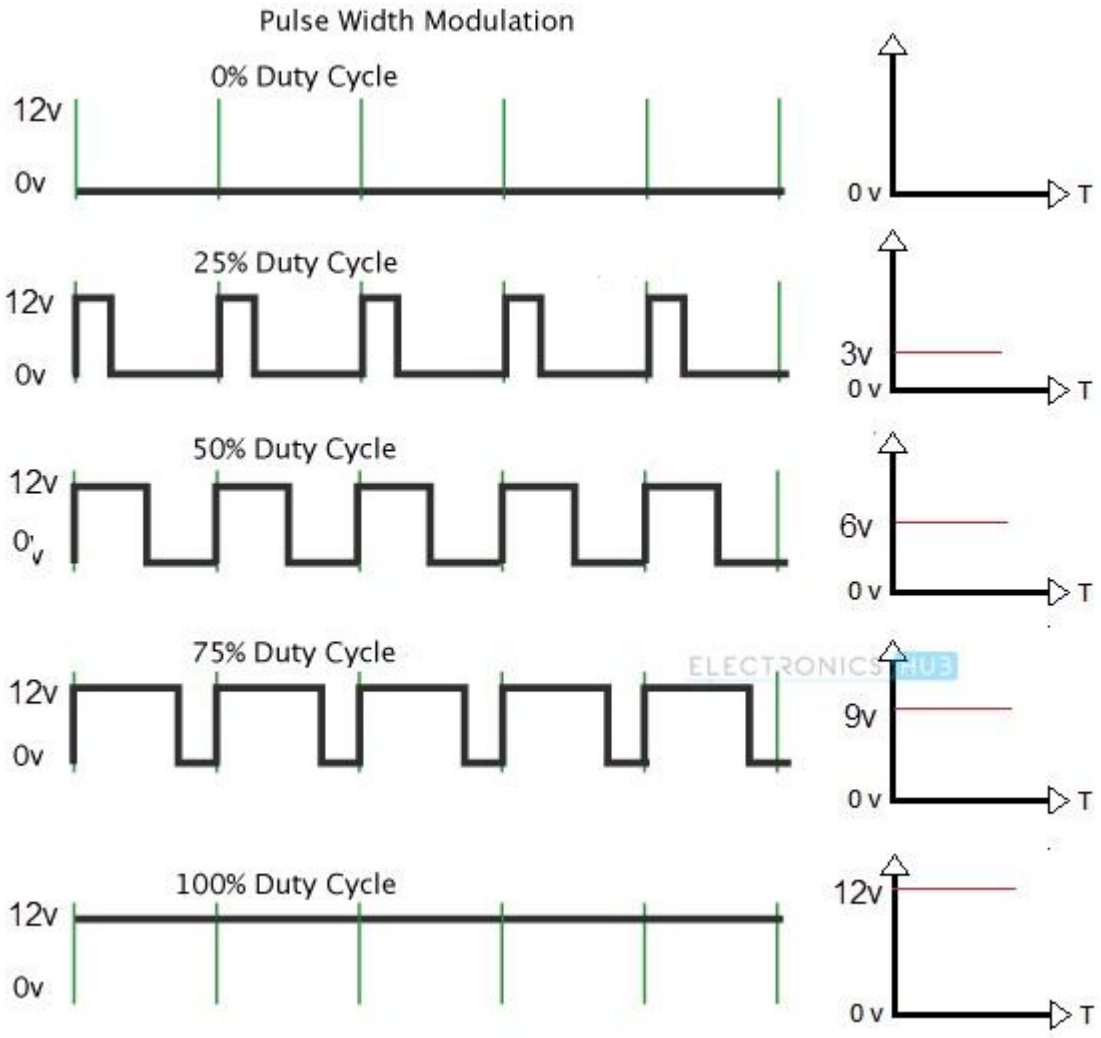
It is an algorithm with artificial intelligence that was developed to analyze and determine safe paths and verify the presence of drive able lanes using a camera.

2.3 Pulse Width Modulation :

For controlling the motors speed, pulse width modulation (PWM) is used. Pulse width modulation is a simple method of controlling analogue devices via a digital signal through changing or modulating the pulse width. An analogue device is become digital by powering it with a pulse signal switches between on and off (5V and 0V). This digital control is used to create a square wave.

The duty cycle is defined as the percentage of digital 'High' to digital 'Low' plus digital 'High' pulse-width during a PWM period. The average DC voltage value for 0% duty cycle is zero; with 25% duty cycle the average value is 1.25V (25% of 5V). With 50% duty cycle the average value is 2.5V, and if the duty cycle is 75%, the average voltage is 3.75V and so on. So, by varying the average voltage, the motor speed can be controlled.

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



Various Duty cycle with Arduino (motor speed) code

CHAPTER 3: Hardware Description

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

3.1 Hardware Description

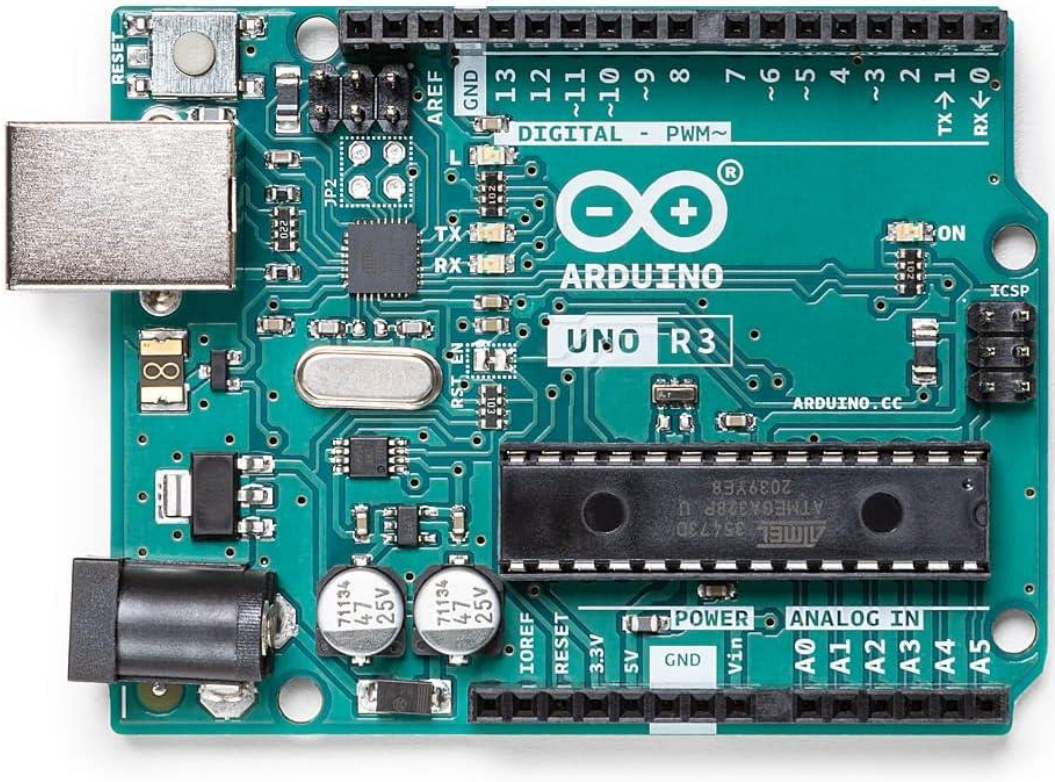
- Arduino Uno.
- Ultrasonic Sensor Hc-Sr04.
- L298 Motor Driver.
- Voltage Regulator.
- Motors.
- Batteries.
- Jumper Cables.
- Temperature or Flame Sensor.
- Water Pump.

3.1.1 Arduino :

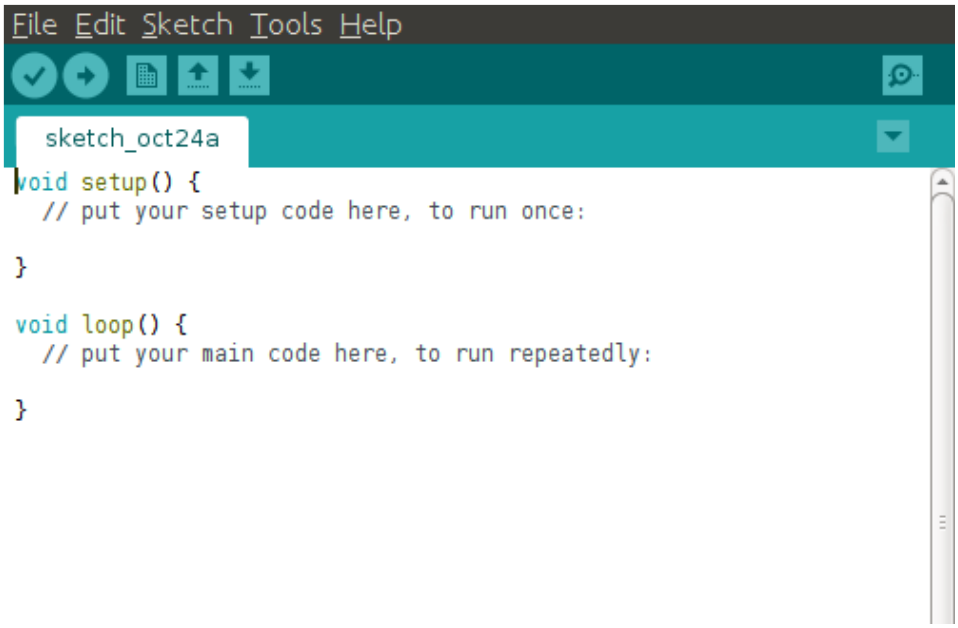
The heart of this project is Arduino. All program of this project is stored in its microprocessor . Arduino is an open source hardware development board. Arduino hardware consists of an open hardware design with an Atmel AVR processor . Arduino programming language is used to program the processor. There are many types of Arduino board available in the market. But, in this project Arduino UNO has been used.

- Arduino UNO R3 is used for controlling the whole the process of Fire Fighting Car.

- Arduino read these signals and send commands to driver circuit to driveline follower.



Arduino UNO R3



Arduino Software (IDE)

3.1.2 Ultrasonic Sensor :

- The ultrasonic sensor emits the short and high-frequency signal. These propagate in the air at the velocity of sound. If they hit any object, then they reflect an echo signal to the sensor. The ultrasonic sensor consists of a multivibrator, fixed to the base.
- The Multivibrator Is a Combination of a Resonator and A Vibrator.
- The Resonator Delivers Ultrasonic Wave Generated by The Vibration.
- The ultrasonic sensor consists of two parts; the emitter which produces a 40khz sound wave and the detector detects a 40khz sound wave and sends an electrical signal back to the microcontroller.
- If you need to measure the specific distance:

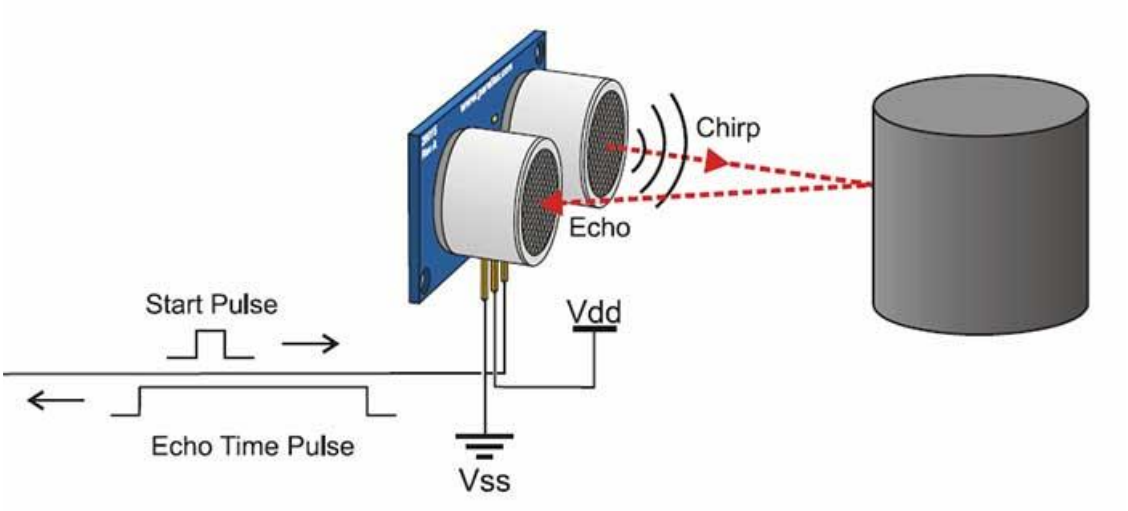
- Distance = ½ T x C (T = Time and C = the speed of sound).

Features :

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

Consisting of :

- The transmitter (emits sound using piezoelectric crystals).
- The receiver (which faces the sound beyond it Travel to and from the target).

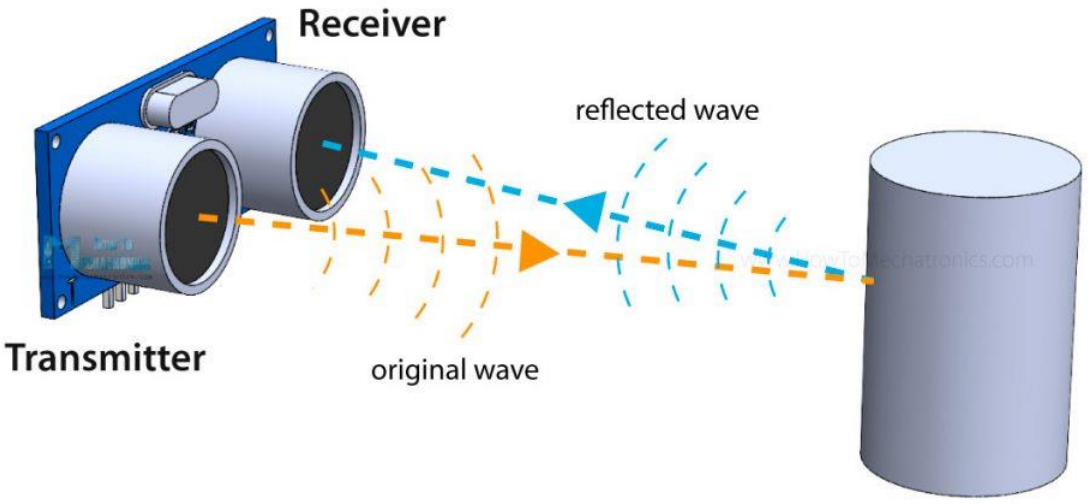


Working diagram of Ultrasonic sensor

How Does It Work (fire fighting with Obstacle Avoidance) :

- The fire fighting robotic vehicle uses ultrasonic sensors for its movements.
- The ultrasonic sensor is attached in front of the robot. Whenever the robot is going on -the transmitter the desired path the ultrasonic sensor transmits the ultrasonic waves continuously from its sensor head.
- The ultrasonic sensor transmits the ultrasonic waves from its sensor head and again receives the ultrasonic waves reflected from an object.
- Whenever an obstacle comes ahead of it the ultrasonic waves are reflected from an object and that information is passed to the microcontroller. The microcontroller controls the motors left, right, back, front, based on ultrasonic signals. To control the speed of each motor pulse width modulation is used (pw).
- Ultrasonic sensors work by emitting sound waves at a frequency too high for humans to hear. They then wait for the sound to be reflected back, calculating distance based on the time

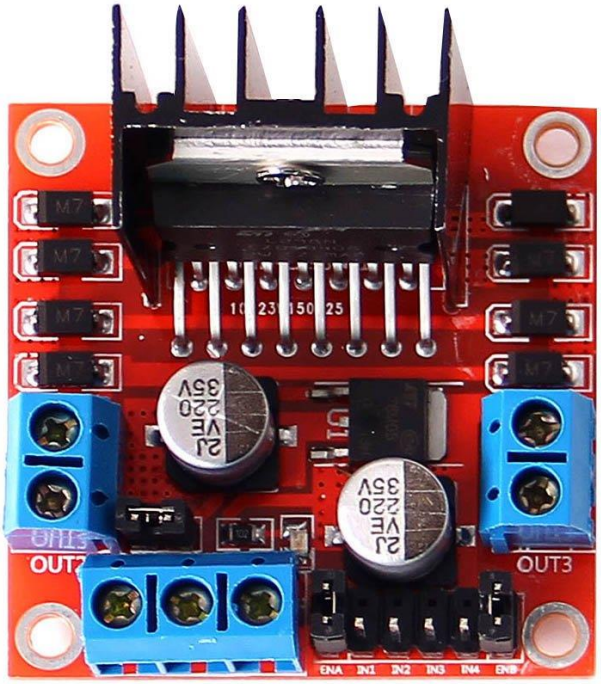
required.



3.1.3 MOTOR DRIVER (L298N) :

- **We used four dc motors at the bottom of the robot to control the rotation speed of the wheels.**

- **These motors provide more torque than normal motors and can be used for carrying some load as well.**



Features :

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

3.1.4 Motors :

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

- We have used two geared motors at the rear of the line follower robot.
- These motors provide more torque than normal motors and can be used for carrying some load as well.

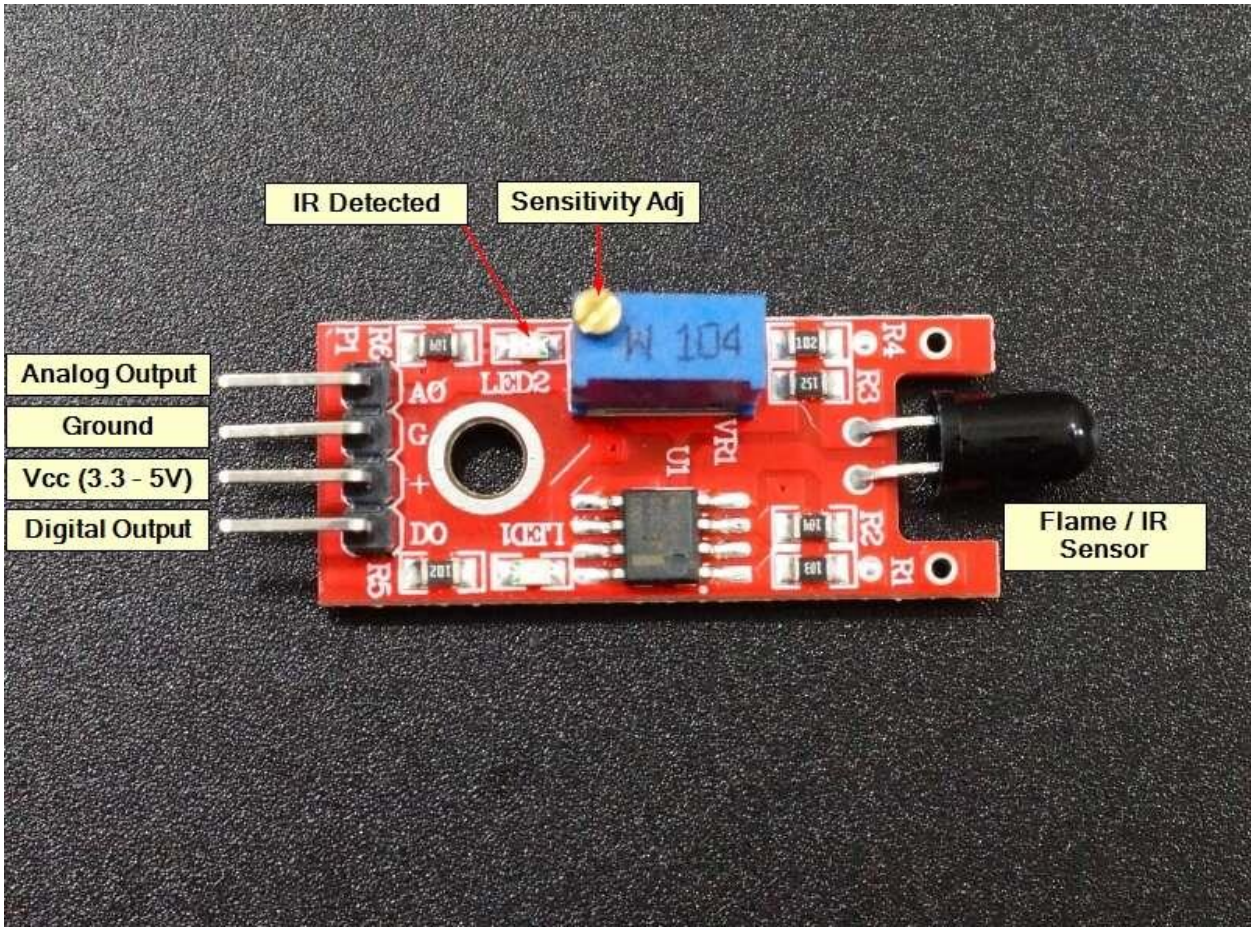
DC Motor

Features :

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

3.1.5 Flame sensor :

This sensor module is primarily used to detect an open flame. It does this by detecting light that is in the IR (infrared) spectrum which is emitted by a flame. That spectrum is typically in the range of 700-1100nm.



- Detects a flame or a light source of a wavelength in the range of 760nm-1100 nm.
- Detection range: up to 100 cm.
- Detection angle about 60 degrees, it is sensitive to the flame spectrum.

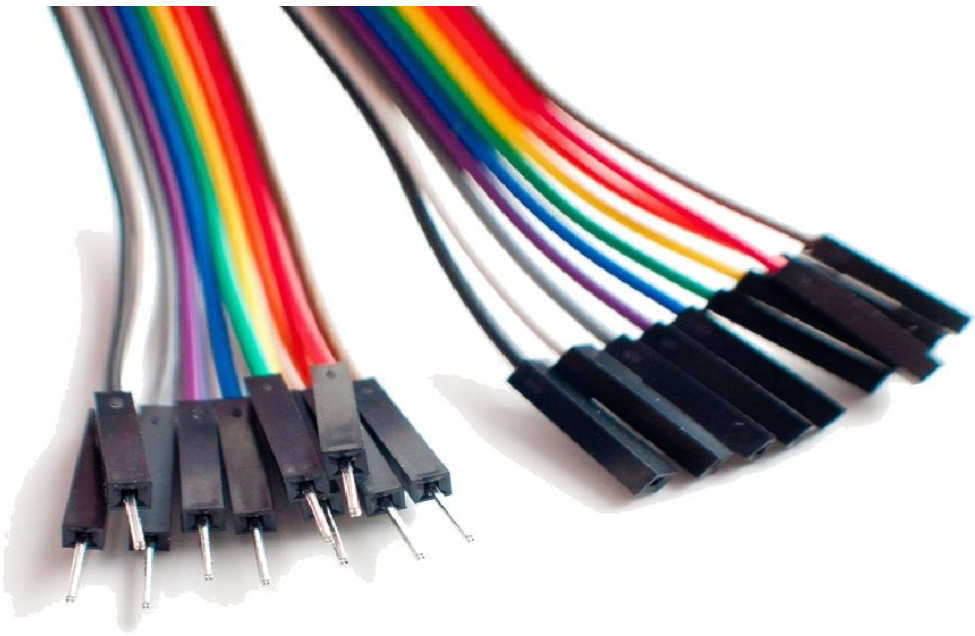
3.1.6 Batteries :

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

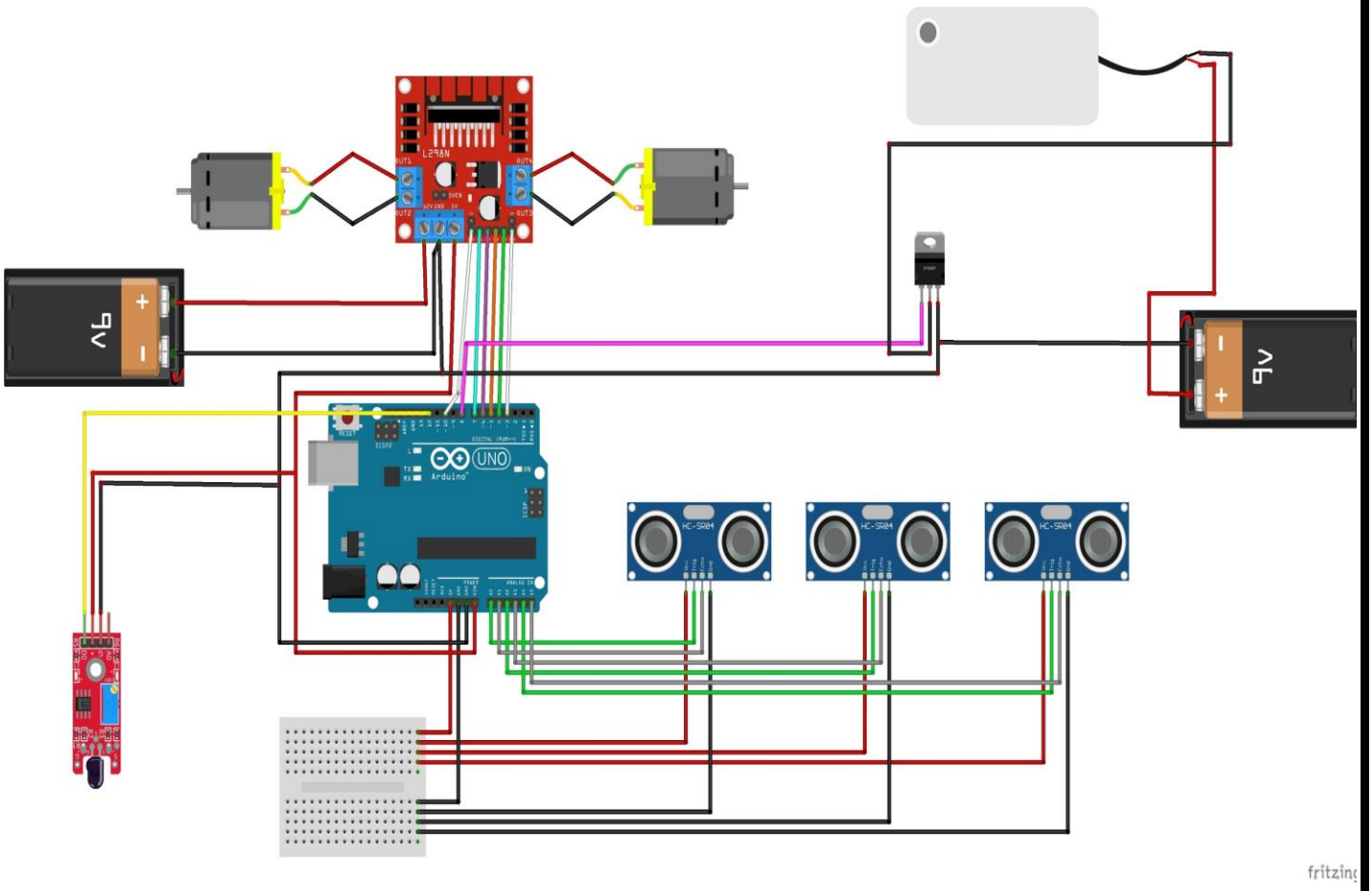
- Rechargeable Source of Energy to The Power Supply Circuit.
- It Gives Voltage 3.7V.

3.1.7 Jumper Cables :

- Jumper cables are insulated wires.
- They come in pairs and have alligator clips that are used to connect a car battery to another energy source.
- These sources can either be other vehicles, or batteries that have the same voltage as the car that needs to jump start.



3.2 Hardware Components :



CHAPTER 4: Methodolog

This Chapter Will Give the Details about The Algorithms. Mainly, Wall Following Algorithm and Flood Fill Algorithms are in this Project.

Implementation :

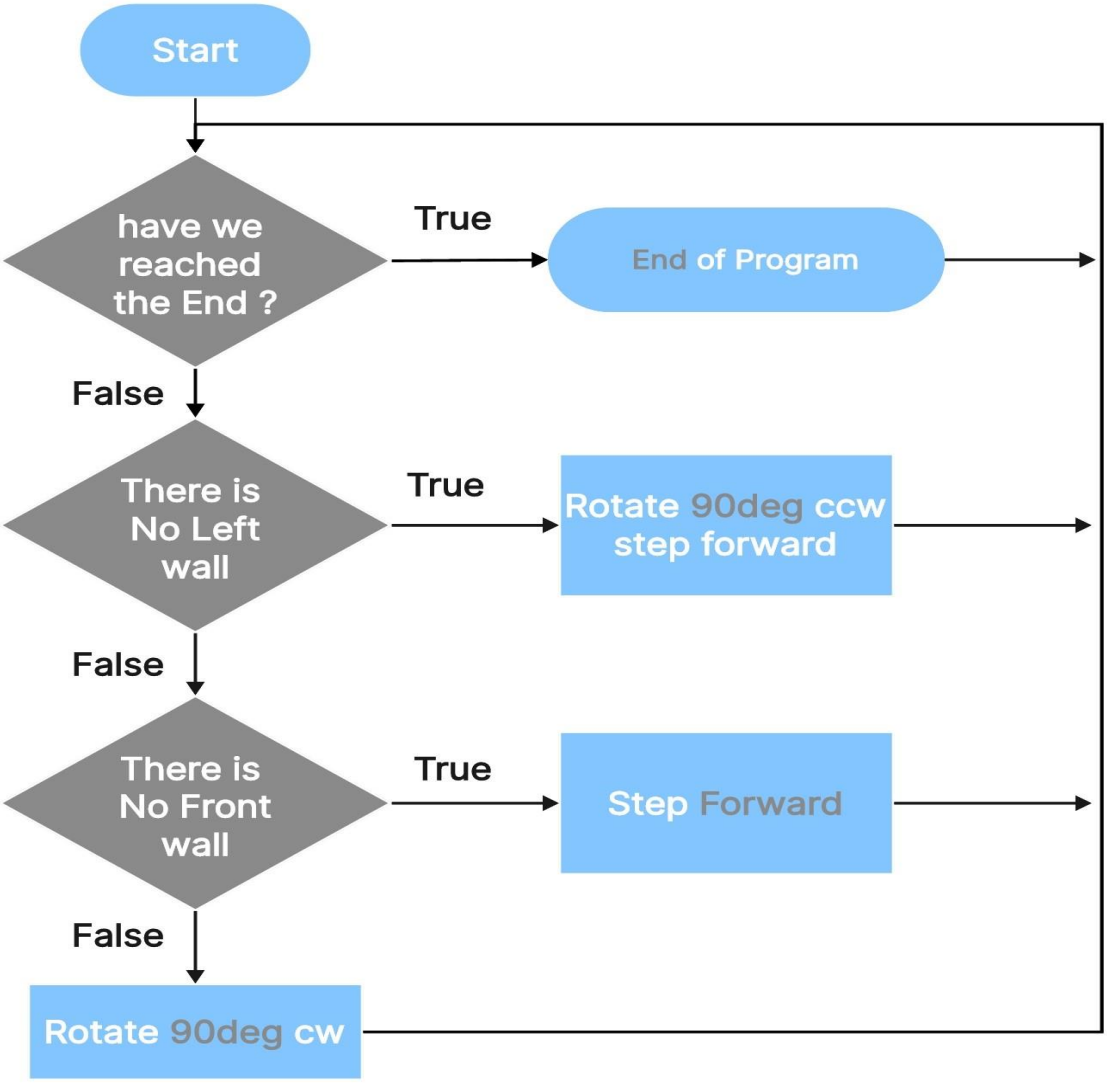
- ❑Wall Following Algorithm.
- ❑Flood Fill Algorithm.
- ❑Source Code

4.1 Wall following algorithm :

~ The most common algorithm for self-drive car robot is wall following algorithm. the robot will take its direction by following either left or right wall. This algorithm also called : left hand-right hand rules.

~ whenever the robot reaches a junction, it will sense for the opening walls and select it direction giving the priority to the selected wall.

~ by taking the walls as guide, this strategy is capable to make the robot reaches the finish point of the maze without actually solving it. But, this algorithm is not efficient method to solve a maze. Cause, the wall follower algorithm will fail to solve some maze construction, such as a maze with a closed loop region.



Wall Follower Algorithms

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

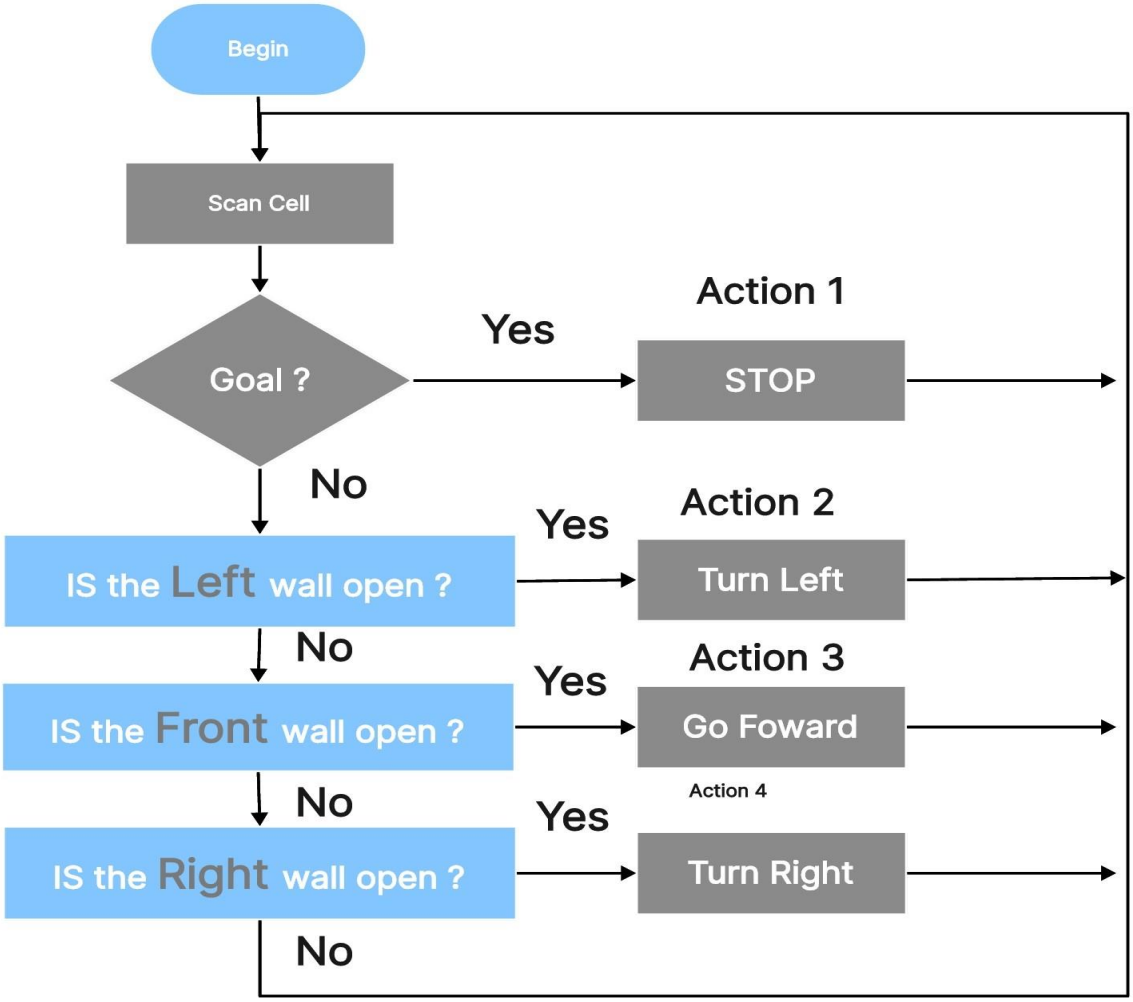
Right wall following routine	Left wall following routine
if there is no wall at right,	if there is no wall at left
turn right	turn left
else	else
if there is no wall at straight	if there is no wall at straight
keep straight	keep straight
else	else
if there is no wall at left	if there is no wall at right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

Table: Left-Right wall following routine

4.2 Flood fill algorithm :

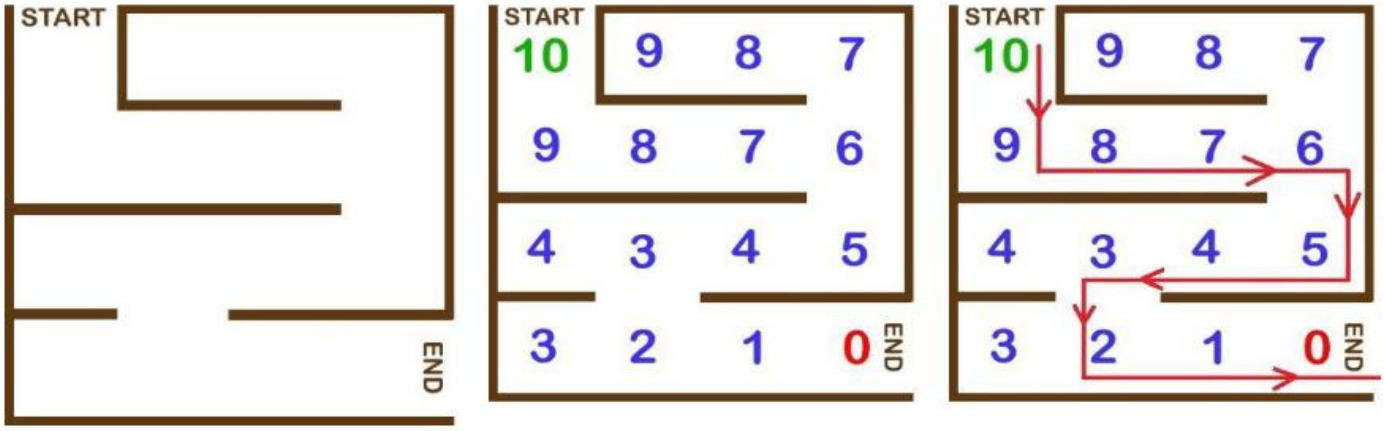
~The most efficient self-drive car algorithm is flood fill algorithm. It is derived from “Bellman Ford Algorithm “.

~ 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. The first array is holding the walls map values, while the other one is storing the distance values.



~ In every cell, robot will follow the following steps :

- Update the wall map.
- Flood the maze with the new distance values.
- Decide which neighboring cell has the lowest distance value.



The Maze **Apply Flood Fill** **Shortest pathfinder**

- Move to the neighboring cell with the lowest distance value.

4.3 Source Code :

```
#include <NewPing.h>

#define S1Trig A4
#define S1Echo A5
#define S2Trig A2
#define S2Echo A3
#define S3Trig A0
#define S3Echo A1

//Create objects for the motors
#define motor1F 4
#define motor1B 5
#define motor2F 6
#define motor2B 7

#define ENBa 3
#define ENBb 11

#define Bump 13
#define Flame 2

int distance=0, t=0;

int count = 0;
bool fireDetected = false;

void setup() {
  Serial.begin(9600);
  //Set the Trig pins as output pins
  pinMode(S1Trig, OUTPUT);
```

```
  pinMode(S2Trig, OUTPUT);
  pinMode(S3Trig, OUTPUT);
  //Set the Echo pins as input pins
  pinMode(S1Echo, INPUT);
  pinMode(S2Echo, INPUT);
  pinMode(S3Echo, INPUT);

  pinMode(Bump, OUTPUT);
  pinMode(Flame, INPUT);

  analogWrite (ENBa, 100) ;
  analogWrite (ENBb, 100) ;
}

void loop() {

    offFire();

}
//Get the sensor values
int sensorOne() {
  //pulse output
  digitalWrite(S1Trig, LOW);
  delayMicroseconds(5);
  digitalWrite(S1Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(S1Trig, LOW);

  t = pulseIn(S1Echo, HIGH);//Get the pulse
  distance = t /57; // Distance = (Speed of Sound * Time/ 2) = t / (1/(350 * 0.0001/2 ))   0.0001
---> conver to cm  //50 --> (1/(361.6 * 0.0001/2 )) distance = t /55.31;
  return distance; // Return the values from the sensor
}
```

```
//Get the sensor values
int sensorTwo() {
  //pulse output
  digitalWrite(S2Trig, LOW);
  delayMicroseconds(5);
  digitalWrite(S2Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(S2Trig, LOW);

  t = pulseIn(S2Echo, HIGH);
  distance = t /57;
  return distance;
}


int sensorThree() {
  //pulse output
  digitalWrite(S3Trig, LOW);
  delayMicroseconds(5);
  digitalWrite(S3Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(S3Trig, LOW);

  t = pulseIn(S3Echo, HIGH);
  distance = t /57;
  return distance;
}

/*******************Motor functions**********************/
void forward() {
analogWrite (ENBa, 150) ;
analogWrite (ENBb, 150) ;
```

```
digitalWrite (motor1F, HIGH) ;
digitalWrite (motor1B, LOW) ;
digitalWrite (motor2F, HIGH) ;
digitalWrite (motor2B, LOW) ;
}
void back() {
analogWrite (ENBa, 130) ;
analogWrite (ENBb, 130) ;
digitalWrite (motor1F, LOW) ;
digitalWrite (motor1B, HIGH) ;
digitalWrite (motor2F, LOW) ;
digitalWrite (motor2B, HIGH);
}
void left() {
analogWrite (ENBa, 130) ;
analogWrite (ENBb, 110) ;
digitalWrite (motor1F, HIGH) ;
digitalWrite (motor1B, LOW) ;
digitalWrite (motor2F, LOW) ;
digitalWrite (motor2B, HIGH) ;
}
void right() {
analogWrite (ENBa, 110) ;
analogWrite (ENBb, 130) ;
digitalWrite (motor1F, LOW) ;
digitalWrite (motor1B, HIGH) ;
digitalWrite (motor2F, HIGH) ;
digitalWrite (motor2B, LOW) ;
}
void Stop() {
analogWrite (ENBa, 0) ;
analogWrite (ENBb, 0) ;
digitalWrite (motor1F, LOW) ;
```

```
digitalWrite (motor1B, LOW) ;
digitalWrite (motor2F, LOW) ;
digitalWrite (motor2B, LOW) ;
}

void offFire(){
   int flameValue = digitalRead(Flame);

   if (flameValue == 1) {
      fireDetected = true;

     if (fireDetected) {
        count ++;
        Serial.println(flameValue);
     }
     if (count >= 50) { // If count reaches 20, activate water pump
      Stop();
      //MOVE();
      //analogWrite (ENBa, 50) ;
      //analogWrite (ENBb, 50) ;
      activatePump();
      count = 0; // Reset count after pumping water
      fireDetected = false; // Reset fire detection
     } }
   else{
     count = 0;
     MOVE();
     }}
void activatePump() {
  Serial.println("Fire !!! Fire !!! Fire !!!");
  digitalWrite(Bump, HIGH);
  delay(3000); // Run pump for 5 seconds
  digitalWrite(Bump, LOW);
```

```
}
void MOVE(){
  int centerSensor = sensorTwo();
  int leftSensor = sensorOne();
  int rightSensor = sensorThree();

if (centerSensor > 30 && leftSensor > 30 && rightSensor > 30) {
    forward();
    Serial.println("forward");
  }else {
  if ( centerSensor <= 15) {
    Stop();
    Serial.println("Stop");
    delay(50);
    back();
    Serial.println("BACK");
    delay(100);
    if (leftSensor > rightSensor) {
      left();
      Serial.println("Left");
      delay(300);
    } else {
      right();
      Serial.println("Right");
      delay(300);
    }
  }else if (leftSensor <= 15) {
      right();
    Serial.println("Right");
    } else if (rightSensor <= 15) {
      left();
    Serial.println("Left");
    }}}
```

CHAPTER 5: Mechanism

This chapter will cover the flowchart of this project, Physical configuration, physical implementation, software development, software implementation etc.

- Physical configuration
- Sequence diagram
- UML class diagram
- Describes work Diagram

5.1 Physical configuration :

The mechanism of the car in project involves integrating these components together to create fire fighting car self-driving .

The microcontroller reads the sensor data, makes decisions about how to move the car, and sends signals to the motors to drive the wheels.

As the car moves , it continues to gather sensor data and adjust its movements accordingly, until it completes a specific task.

The mechanism of a fire fighter car in a project typically involves several components working together to navigate , extinguish the fire. Here are the main components and how they work:

5.1.1 Chassis:

The chassis is the basic structure of the car and provides a platform for mounting the other components. It is usually made from lightweight materials such as plastic or metal

5.1.2 Microcontroller:

The microcontroller is the brain of the car and controls its movements based on the sensor readings. It is usually a small computer that can be programmed to respond to different inputs and make decisions about how to navigate ,extinguish the fire .

5.1.3 Power source:

The power source provides the energy needed to drive the motors and power the microcontroller and sensors. It is usually a battery or set of batteries that can be recharged as needed.

5.1.4 *Sensors:*

- Sensors are used to flame/temperature measurement and detect obstacles, navigate.
- Use any type of temperature sensor to measure the temperature accurately, such as DHT11 or NTC thermistor sensor .
- Ultrasonic sensors are commonly used in self-driving projects, but other types of sensors such as infrared or lidar sensors can also be used.

5.1.5 Motors:

The motors are used to drive the car's wheels and provide motion. They are usually DC motors that can be controlled using a microcontroller.

5.1.6 Wheels :

The wheels provide traction and support for the car as it moves . They are usually made of rubber or plastic and can be driven by the motors using gears or belts.

5.2 physical implementation :

The first section of the code, if the right wall and left wall distance is equal or less than 10cm than both motors are in forward mode. So, the robot move straight. If, the right wall is more than 10cm, it is considered as open wall, than the robot turn into right. Same as, if the left wall is more than 10cm, it is considered as open wall, than the robot turn into left. And this function is continued in a loop till the maze is solved.

The robot can be designed right oriented or left oriented or even can be designed to follow either sides. A right or left oriented wall follower can be designed easily with the help of just two sensors. Though more sensors can be used in making such a robot which will ultimately improve the path accuracy of the robot. For making a wall follower which can side either ways.

at least three sensors are must to use and the program logic goes a little complex and sophisticated. If right oriented wall follower is designed, the obstacle detection sensors need to be mounted on front and right side of the robot.If Left oriented wall follower is designed, the obstacle detector sensors need to be mounted on front and left side of the robot. If the robot is designed to follow either sides, obstacle detector sensors need to be mounted on front, left and right side of the robot. In this project a left side wall following robot is designed.

The obstacle detection sensor used in the wall follower can be an ultrasonic sensor or an IR sensor, but we prefer ultrasonic sensor.

The ultrasonic sensor is used as an obstacle detector, and the robot can be designed to maintain a distance with the wall which not only improves the flexibility of the path but also improves its accuracy.

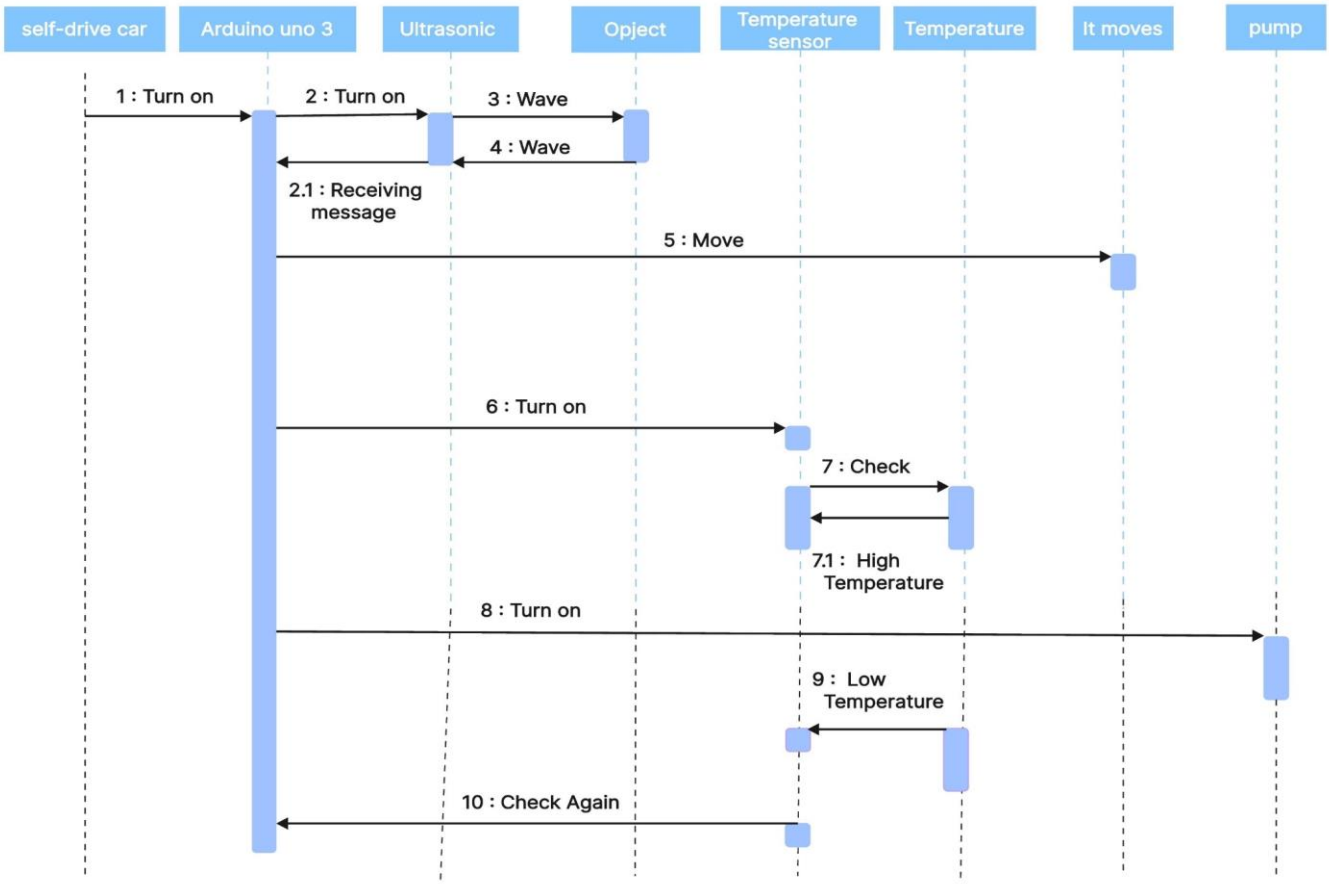
Since ultrasonic sensors work based on the reflection of ultrasonic (sound) waves, they can be relied upon in environments where there is sunlight or black obstacles in the path.

The software code developed for this robot is Arduino UNO and it will work fine if the communication between the sensors and the DC motor drive is done the same way the software does.

The program code is written and burned using the Arduino IDE.

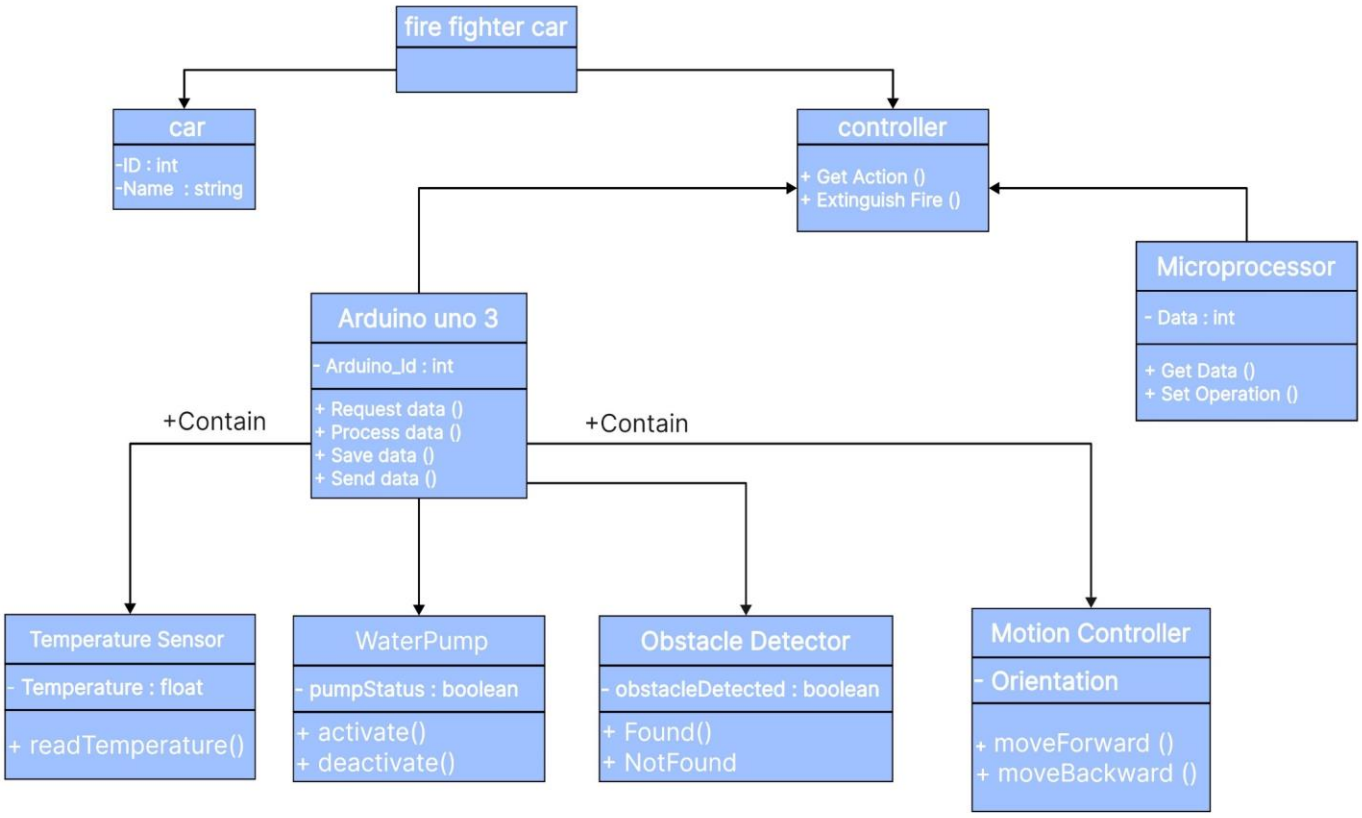
5.3 Sequence Diagram :

They are used to depict how different organisms in a given system interact over time. These diagrams show the order of messages exchanged between objects to achieve a particular function or scenario.



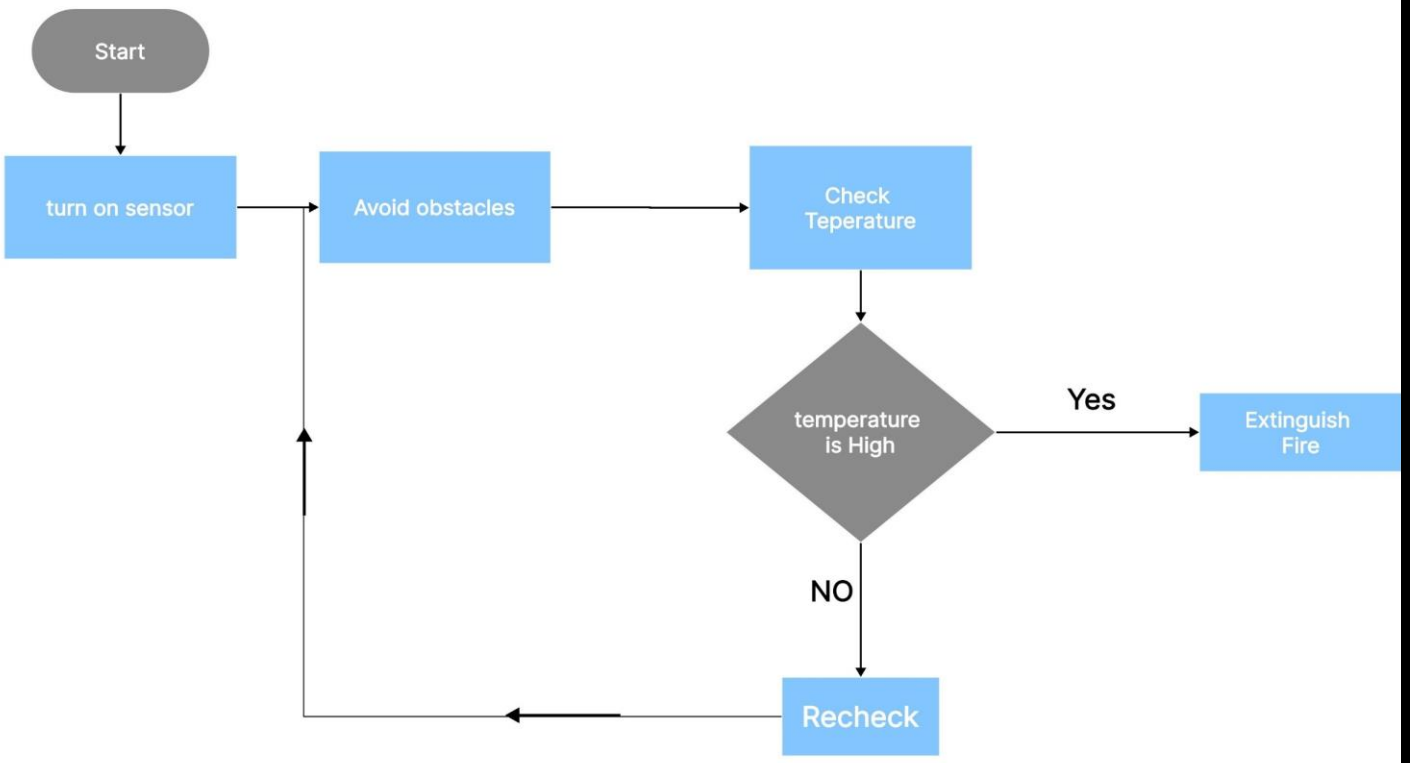
5.4 UML Class Diagram :

They are used to depict different aspects of software systems. They include classes (classes), harvesters (attributes), operations (operations), and relationships (relationships) between classes.

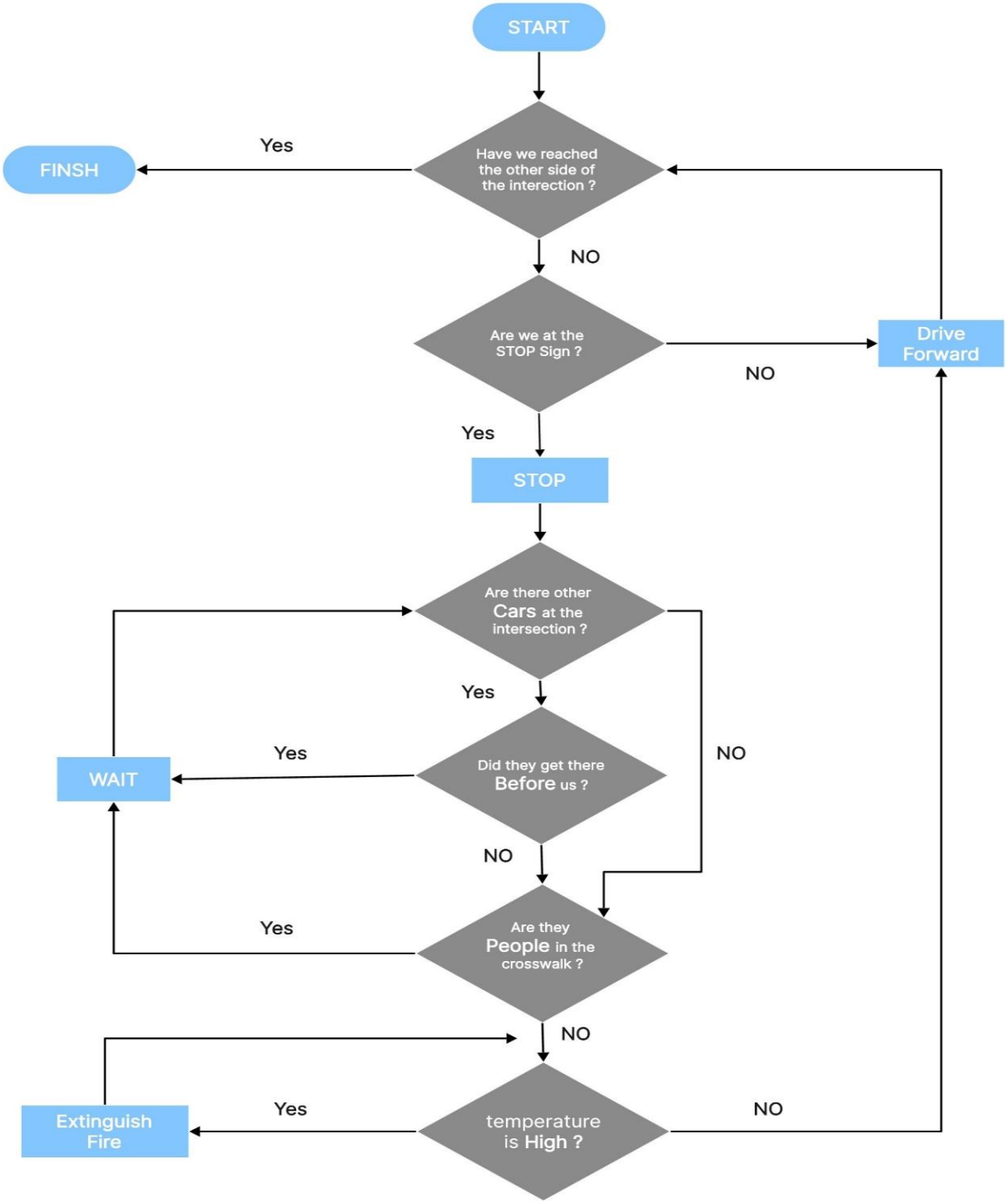


5.5 Describes work Diagram :

It depicts the workflow or different activities that occur within the system, and how the system moves to another, including various events. So this should be helpful to understand the whole thing.



CHAPTER 6: Flow chart



CHAPTER 7: Software Description

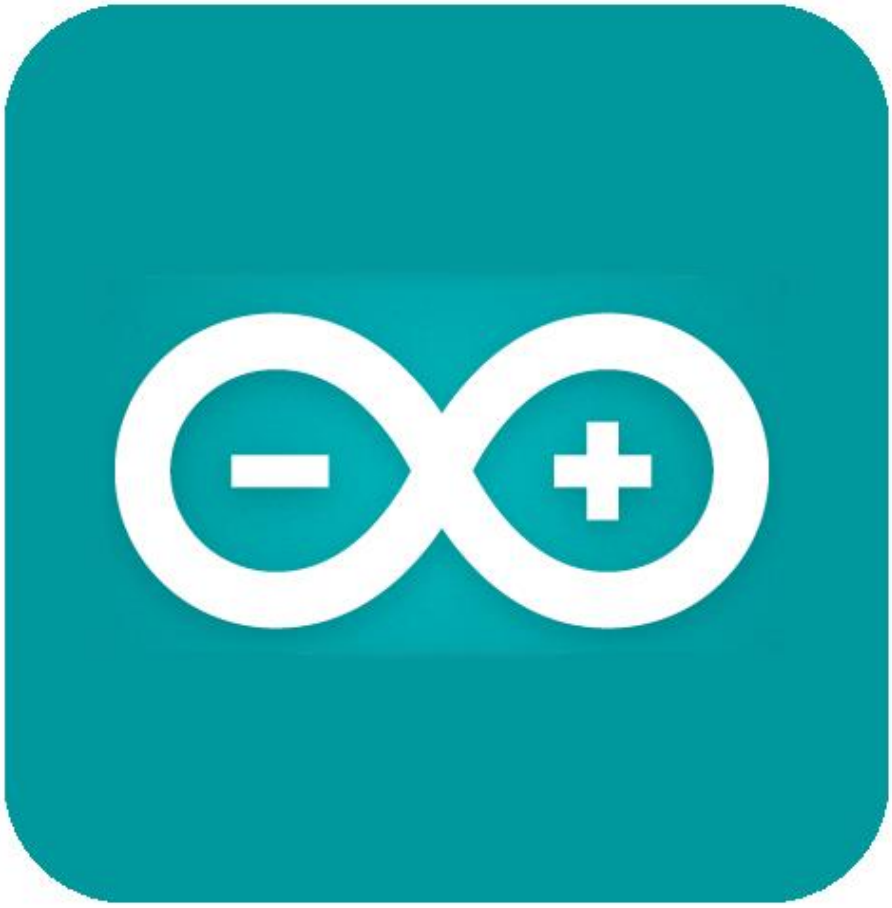
- Arduino IDE
- Proteus
- AVR DUDES

7.1 Arduino IDE :

• The Arduino Integrated Development Environment - or Arduino Software (IDE) .

• contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus.

• It connects to the Arduino hardware to upload programs and communicate with them.



7.2 Proteus :

- The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation.
- The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards.

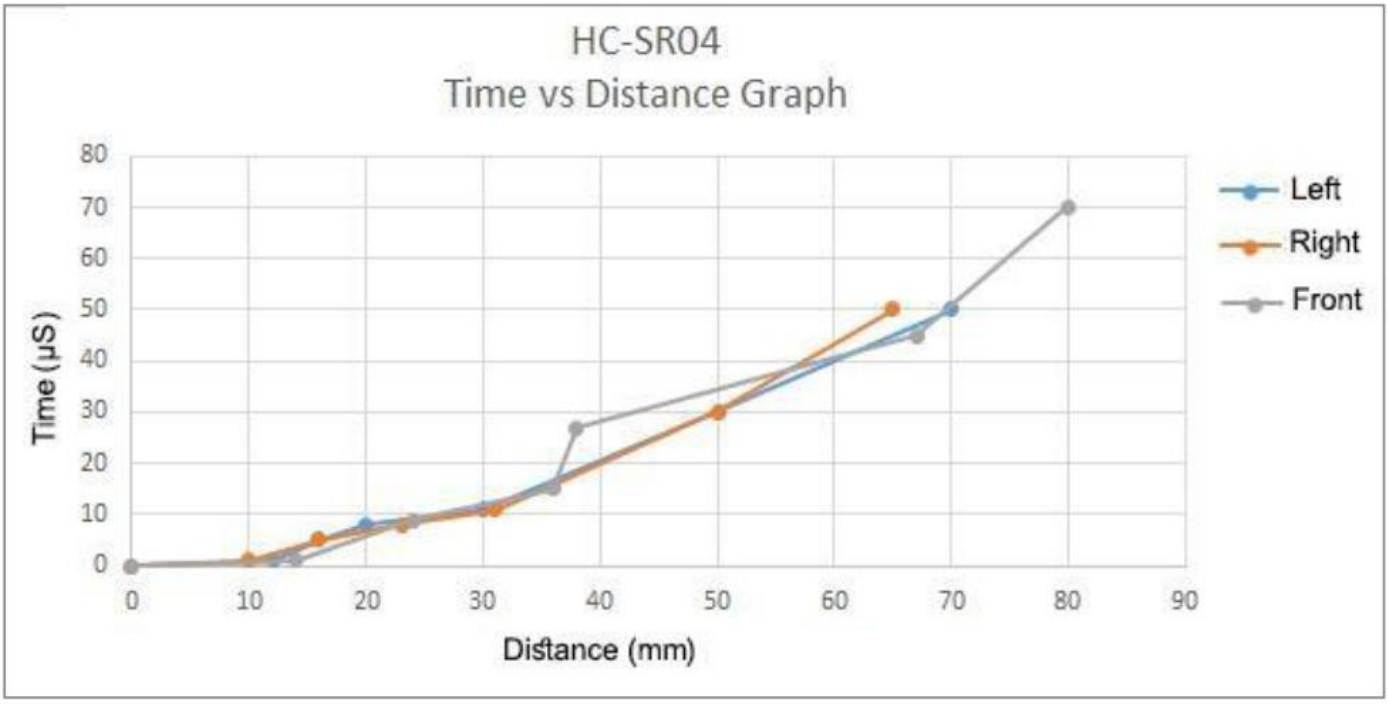
CHAPTER 8: Conclusions and Recommendations

8.1 Conclusions:

- In conclusion, the two-maze solving algorithm was successfully implemented in the robot and the project objectives were achieved. The first algorithm was the wall tracking algorithm.
- The basic method shows a good result for solving the maze. But due to his lack of self-intelligence, he failed to solve the maze by the shortest path. It cannot be solved to close the annular maze.
- Therefore, an effective method can be used to find the shortest path, which is the flood filling algorithm method and some cameras to transmit a better image of the ground.
- After applying all the methods, the robot was trained to be**(fire fighting car and self-drive)**Many tests have been conducted to ensure the best performance of the robot. This project helps to improve various important information about robots and knowledge about several decision-making algorithms. It is also useful to become familiar with many of the electronic components such as the motor drive, sensors, etc. This knowledge gained will have a significant impact on future work.

8.2 Ultrasonic sonar respons :

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



Time vs. Distance Graph of HC-SRO4

8.3 Further development :

• This robot is a bit slow, so a new method must be developed. The size of the robot can be reduced. In this project, an ultrasonic sensor is used to map the maze. Other active navigation sensors can be used, and instead of one flame sensor, more can be added for more accurate identification of fire locations. Finally, its wheels and body need to be upgraded to make it comfortable on rough surfaces.

CHAPTER 9: References

1. https://www.synopsys.com/automotive/what-is-autonomous-car.html

2. https://www.linkedin.com/pulse/what-challenges-driverlessautonomous-cars-patrick-mutabazi

3. https://ieeer8.org/category/committee/meetings/2021-march-online/

4. https://www.ennomotive.com/automated-vehicles-arduino-technology/

5. https://creately.com/diagram/example/ivmsk38t/process-chart-self-driving-car-classic

6. https://www.techtarget.com/searchenterpriseai/definition/driverless-car

7. https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

8. M. McRoberts, Beginning Arduino, 1st ed. [Berkeley, Calif.

9 Arduino - PWM", Arduino.cc, 2017. [Online]. Available: http://www.arduino.cc/en/Tutorial/PWM

10.https://theconversation.com/driverless-cars-could-be-a-revolution-for-people-with-disabilities-but-they-also-have-good-reason-to-be-worried-213314