**MIN PROJECT**

(2020-2021)

**IOT based Fire Fighting Robot**

**FINAL REPORT**



**Institute of Engineering & Technology**

**Department of Computer Engineering & Application**

**Team Members**

NITISH KUMAR

(181500437)

VIKAS YADAV

(181500791)

RANJAN KUMAR

(181500557)

**Supervised By**

Mr. Mandeep Singh (Technical trainer)

**DEPARTMENT OF COMPUTER ENGINEERING AND APPLICATIONS, GLA UNIVERSITY, MATHURA**

**Contents**

1. Acknowledgment

2. Introduction

2.1 Motivation and overview

2.2 Objective

2.3 Scope

3. Component

4. Working model and result analysis

5. Pin Diagram

6.Problems

7. Layout

8.Program/Algorithm

9.Conclusion

10. References

**Chapter 1**

**ACKNOWLEDGEMENT**

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. On the completion of this project, I would like to extend my sincere thanks to all of them. I am highly indebted to this project guide **Mr Mandeep Singh** , **Technical trainer** for their guidance and constant supervision as well as for providing necessary information regarding the project. I wish to extend my sincere gratitude to **Prof. Anand Singh Jalal, Head ofDepartment of Computer Engineering and Applications** and faculty of CEA Department of **GLA University** for their guidance, encouragement and give this opportunity and valuable suggestion which prove extremely useful and helpful in completion of this project. I would also like to thank all those who directly or indirectly supported or helped me. I would like to express my gratitude towards my parents and member of my college for their kind cooperation and encouragement which helped me in completion of this project. All of them have willingly helped me out with their abilities.

**Chapter 2**

**INTRODUCTION**

**2.1Motivation and Overview**

Internet of Things is a field of technology that describes the network of various “things” that is physical objects that are all embedded with different sensors, that are destined to measure various things or parameters such as distance and presence of flame. It also has the ability to transfer over a network using technologies like Cloud computing and fog computing. Now days firing activity are much unpredictable and very difficult to control so, this type of iot based robot can make thing easy.

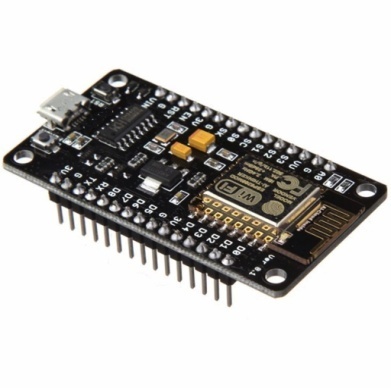
* 1. **Objective**

The fire fighting Robot is an intelligent safety management and monitoring system. The basic objective of this project is made to develop a system model. It monitors and keep a look over the place in which it is installed using sensors.

* 1. **Scope**
* Design a fire fighting robot using numerical approach.
* Select the suitable material to develop the robot.
* The developing of programming is necessary to develop a mechanism of the robot.

**Chapter 3**

**Component:-**

Node mcu esp8266 Ultrasonic sensor

Flame sensorServo motor

****

Mini water pump Bo motor

****

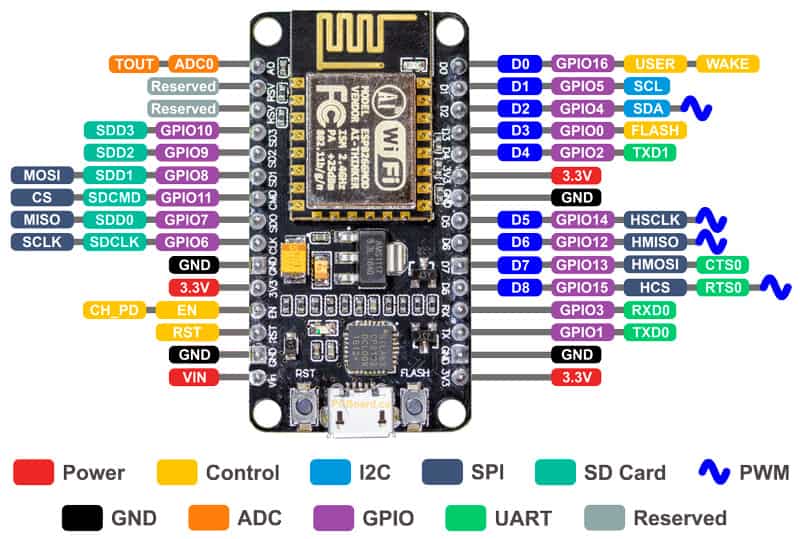
L293d motor driver

****

**Chapter 4**

**Working model and result analysis**

Node mcu esp8266

****

Power Pins There are four power pins. **VIN** pin and three **3.3V** pins.

* + - **VIN** can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on **VIN** is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the **VIN** pin
    - **3.3V** pins are the output of the onboard voltage regulator and can be used to supply power to external components.

* + GND are the ground pins of NodeMCU/ESP8266
  + I2C Pins are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.
  + GPIO Pins NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

* + ADC Channel The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

* + UART Pins NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

* + SPI Pins NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:
    - 4 timing modes of the SPI format transfer
    - Up to 80 MHz and the divided clocks of 80 MHz
    - Up to 64-Byte FIFO

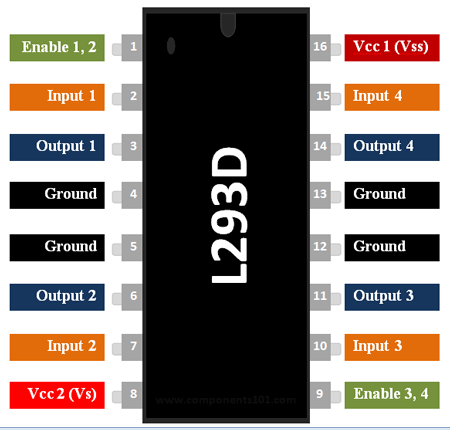
* + SDIO Pins NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

* + PWM Pins The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μs to 10000 μs (100 Hz and 1 kHz).

* + Control Pins are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
    - **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
    - **RST:** RST pin is used to reset the ESP8266 chip.
    - **WAKE:** Wake pin is used to wake the chip from deep-sleep.

* + Control Pins are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
    - **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
    - **RST:** RST pin is used to reset the ESP8266 chip.
    - **WAKE:** Wake pin is used to wake the chip from deep-sleep.

**L293d motor driver:-**

****

**L293D Pin Configuration**

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Name** | **Description** |
| 1 | Enable 1,2 | This pin enables the input pin Input 1(2) and Input 2(7) |
| 2 | Input 1 | Directly controls the Output 1 pin. Controlled by digital circuits |
| 3 | Output 1 | Connected to one end of  Motor 1 |
| 4 | Ground | Ground pins are connected to ground of circuit (0V) |
| 5 | Ground | Ground pins are connected to ground of circuit (0V) |
| 6 | Output 2 | Connected to another end of  Motor 1 |
| 7 | Input 2 | Directly controls the Output 2 pin. Controlled by digital circuits |
| 8 | Vcc2 (Vs) | Connected to Voltage pin for running motors (4.5V to 36V) |
| 9 | Enable 3,4 | This pin enables the input pin Input 3(10) and Input 4(15) |
| 10 | Input 3 | Directly controls the Output 3 pin. Controlled by digital circuits |
| 11 | Output 3 | Connected to one end of Motor 2 |
| 12 | Ground | Ground pins are connected to ground of circuit (0V) |
| 13 | Ground | Ground pins are connected to ground of circuit (0V) |
| 14 | Output 4 | Connected to another end of Motor 2 |
| 15 | Input 4 | Directly controls the Output 4 pin. Controlled by digital circuits |
| 16 | Vcc2 (Vss) | Connected to +5V to enable IC function |

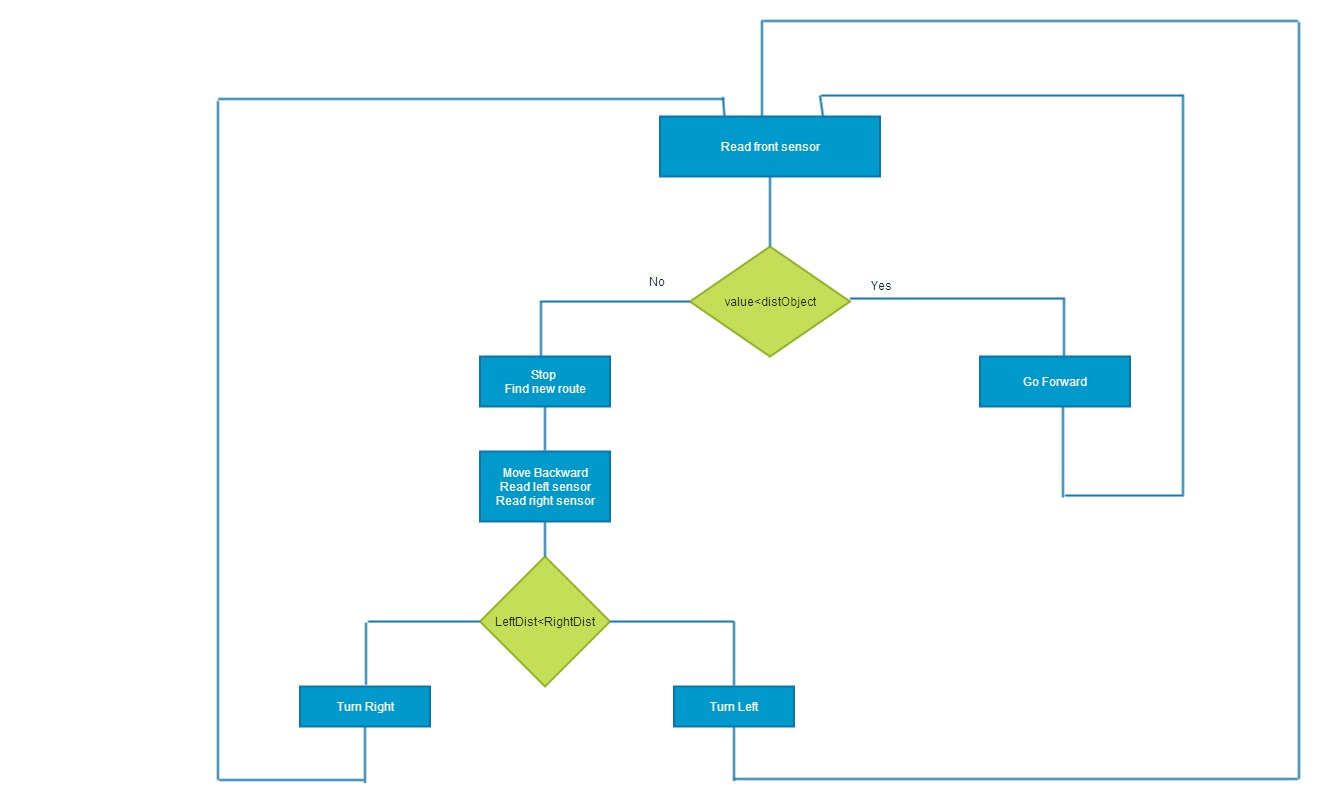
**Features**

* Can be used to run Two DC motors with the same IC.
* Speed and Direction control is possible
* Motor voltage Vcc2 (Vs): 4.5V to 36V
* Maximum Peak motor current: 1.2A
* Maximum Continuous Motor Current: 600mA
* Supply Voltage to Vcc1(vss): 4.5V to 7V
* Transition time: 300ns (at 5Vand 24V)
* Automatic Thermal shutdown is available
* Available in 16-pin DIP, TSSOP, SOIC packages

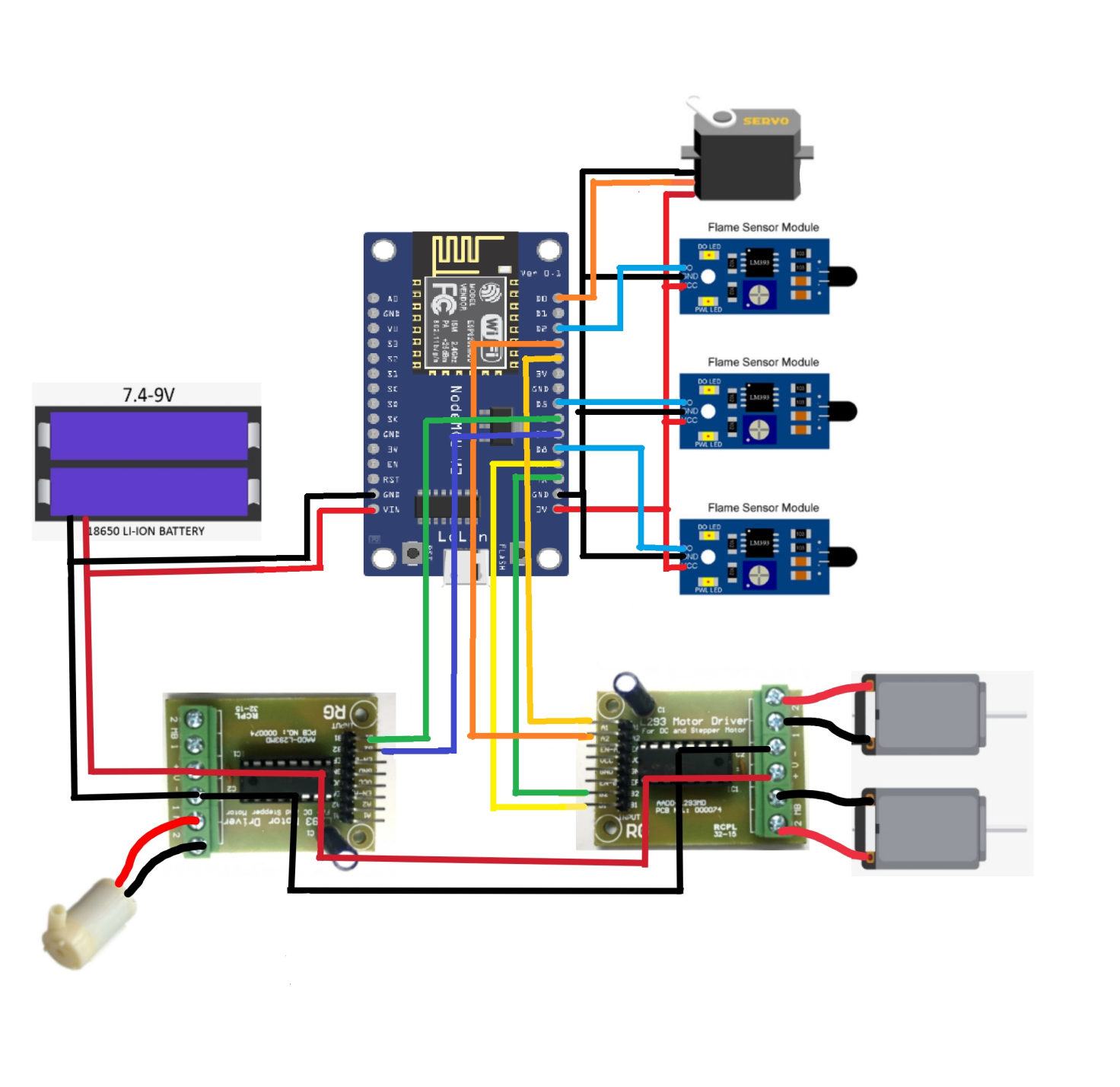
### ****How to use a L293D Motor Driver IC****

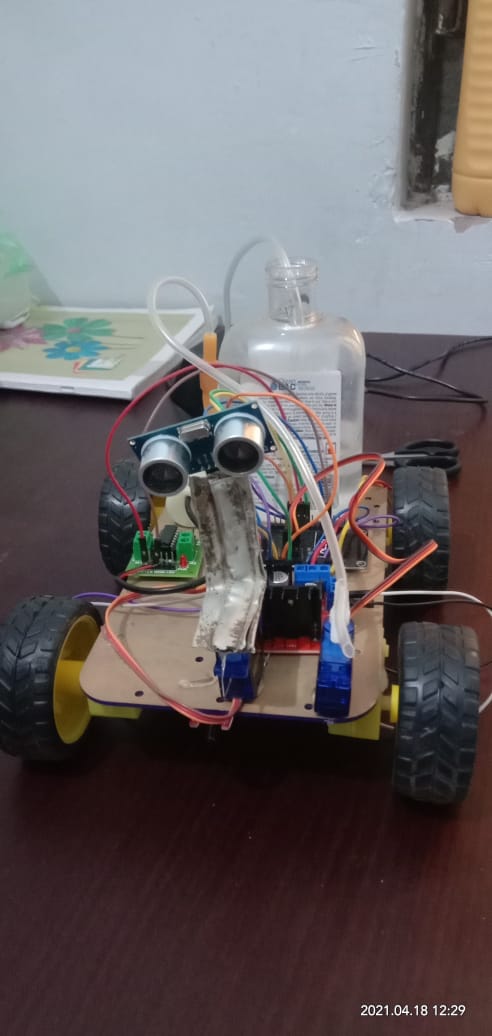
**Using this L293D motor driver IC** is very simple. The IC works on the principle of **Half H-Bridge**, let us not go too deep into what H-Bridge means, but for now just know that H bridge is a set up which is used to run motors both in clock wise and anti clockwise direction. As said earlier this IC is capable of running two motors at the any direction at the same time

**ALGORITHM**

****

**Pin Diagram**

****

****

Problems

|  |  |  |
| --- | --- | --- |
| **Component name** | **Quantity** | **Price** |
| Flame sensor | 1 | 40 |
| Distance sensor | 1 | 65 |
| nodemcu | 1 | 210 |
| Servo motor | 2 | 85\*2=170 |
| Water Pump | 1 | 45 |
| Relay | 1 | 6 |
| L293D driver | 1 | 60 |
| Bo-motor | 4 | 45 \*4=180 |

TOTAL COST= 1450rs.

PROGRAM/ALGORITHM

#include <Servo.h>

/\*------ Arduino Fire Fighting Robot Code by hobby project---- \*/

#include <Servo.h> //include servo.h library

Servo myservo;

int pos = 0;

boolean fire = false;

#define Left 9 // left sensor

#define Right 10 // right sensor

#define Forward 8 //front sensor

#define LM1 2 // left motor

#define LM2 3 // left motor

#define RM1 4 // right motor

#define RM2 5 // right motor

#define pump 6

void setup()

{

pinMode(Left, INPUT);

pinMode(Right, INPUT);

pinMode(Forward, INPUT);

pinMode(LM1, OUTPUT);

pinMode(LM2, OUTPUT);

pinMode(RM1, OUTPUT);

pinMode(RM2, OUTPUT);

pinMode(pump, OUTPUT);

myservo.attach(11);

myservo.write(90);

}

void put\_off\_fire()

{

delay (500);

digitalWrite(LM1, HIGH);

digitalWrite(LM2, HIGH);

digitalWrite(RM1, HIGH);

digitalWrite(RM2, HIGH);

digitalWrite(pump, HIGH);

delay(500);

for (pos = 50; pos <= 130; pos += 1) {

myservo.write(pos);

delay(10);

}

for (pos = 130; pos >= 50; pos -= 1) {

myservo.write(pos);

delay(10);

}

digitalWrite(pump,LOW);

myservo.write(90);

fire=false;

}

void loop()

{

myservo.write(90); //Sweep\_Servo();

if (digitalRead(Left) ==1 && digitalRead(Right)==1 && digitalRead(Forward) ==1)

{

digitalWrite(LM1, HIGH);

digitalWrite(LM2, HIGH);

digitalWrite(RM1, HIGH);

digitalWrite(RM2, HIGH);

}

else if (digitalRead(Forward) ==0)

{

digitalWrite(LM1, HIGH);

digitalWrite(LM2, LOW);

digitalWrite(RM1, HIGH);

digitalWrite(RM2, LOW);

fire = true;

}

else if (digitalRead(Left) ==0)

{

digitalWrite(LM1, HIGH);

digitalWrite(LM2, LOW);

digitalWrite(RM1, HIGH);

digitalWrite(RM2, HIGH);

}

else if (digitalRead(Right) ==0)

{

digitalWrite(LM1, HIGH);

digitalWrite(LM2, HIGH);

digitalWrite(RM1, HIGH);

digitalWrite(RM2, LOW);

}

delay(300);//change this value to increase the distance

while (fire == true)

{

put\_off\_fire();

}

}

****