Contents

1 Aim and Objective **2**

2 Working principle **2**

2.1 Mechanical Design 3

3 Control Block Diagram **3**

4 Sensors Used **3**

4.1 Flame sensor module 4

4.2 Ultrasonic sensor 4

5 Other Components used **5**

6 Controller/Actuator used **6**

6.1 Arduino Uno 6

6.2 Motor Driver 6

6.3 Servo motor 6

6.4 Water pump 10

7 Programming **10**

8 DAQ **13**

9 Modifications Proposed **14**

ME2400 Project Report - ARDUINO FR

Shwetha V, Anjhana M, Priyanka, Varshitha, Shakthi G, Aiman

April 2025

1 Aim and Objective

With the advancement of technology where every process is being automated, the world progressively also moves towards automating tasks which require precise human skill, self-driving cars for example. Firefighters continuously run the risk of dying while they set out on the deadly task of finding and extinguishing fires, all in a very short span of time, as, if a fire is not put out it spreads quickly. Therefore, our automated fire extinguishing robot system steps in to solve this problem and protect the life of our heroes.

We use Arduino Uno as our microcontroller, which is equipped with flame sensors and a spray mechanism to detect and put out fires and ultrasonic sensors for obstacle avoidance as it navigates its way towards the fire.

2 Working principle

When the car is started it starts to move in a programmed random direction, while the flame sensor module scans the surrounding region for the presence of fire. The car moves around, covering the region bounded by walls randomly till it detects flame. Once flame is detected, the fire sensor outputs a signal to the Arduino which sends a signal to the motor driver, making the bot move towards the location of the flame. Upon reaching a particular distance from the flame, the motor driver stops and the Arduino activates the pump, which pumps up water through a nozzle. The Arduino also activates the servo motor which provides a good angular range for the nozzle. Water is sprayed from the nozzle till the flame is put out and the flame sensor can no longer detect the flame. All the while when moving, the ultrasonic sensors scan and detect obstacles in the environment, helping in safe navigation. Once the fire is extinguished the robot continues to move again in a random direction searching for more fire if any, to extinguish.

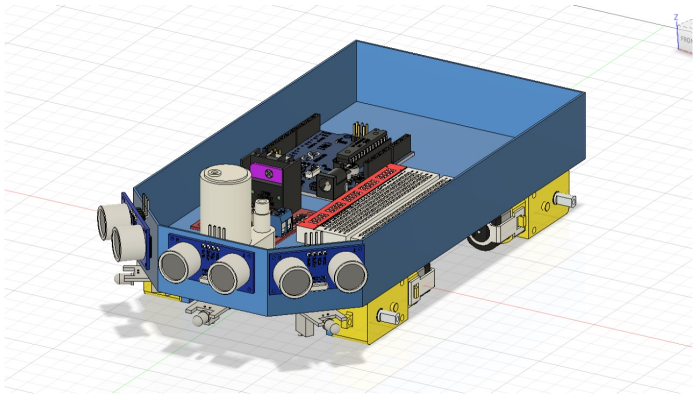


Figure 1: Chassis Design

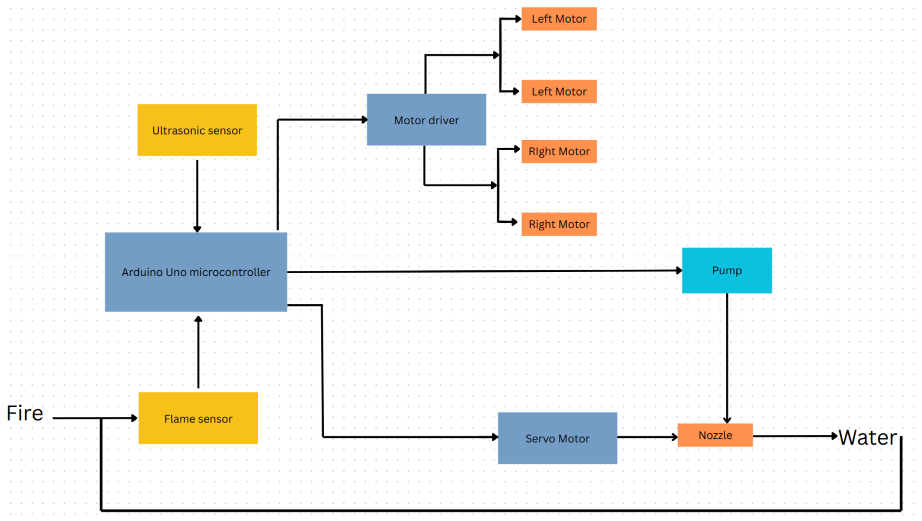


Figure 2: Control Block Diagram

2.1 Mechanical Design

3 Control Block Diagram

4 Sensors Used

A sensor is a device that detects and responds to some type of input from the physical environment. The input can be light, heat, motion, moisture, pressure etc. and the output is an electrical signal. Here our robot should detect flame and obstacles in the environment. So we use 2 sensors: a Flame sensor and an

Ultrasonic sensor.

4.1 Flame sensor module

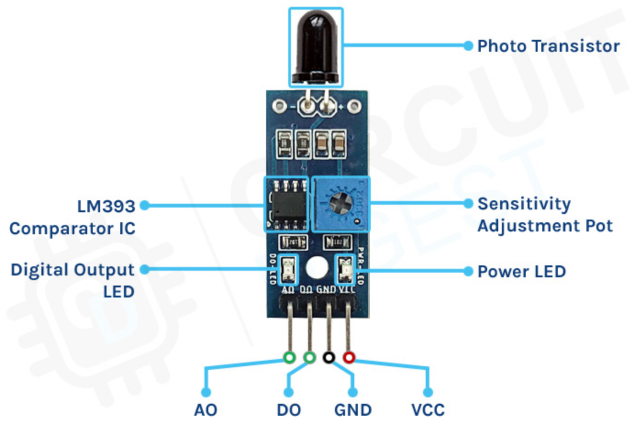


Figure 3: Flame sensor

The flame sensor identifies fire using UV(Ultraviolet)-IR(Infra-red) technology. Most fires produce UV radiation near the point of ignition and therefore in the event of a fire, at a good proximity distance set by the user, the sensor can sense the flame and send a series of pulses to the controller(Arduino). It can be easily damaged by high temperature and hence should be placed at a certain distance from the flame. It consists of 4 pins

- Pin1 VCC: Voltage supply range from 3.3-5.3V
- Pin2 GND: This is a ground pin
- Pin3: This is the Digital output pin.
- Pin4: This is the Analog output pin

The sensor uses an IC LM393, and an IR photodiode/phototransistor to detect sensitive infrared light, and is connected to 2 indicator LEDs which glow red if fire is detected and green otherwise.

4.2 Ultrasonic sensor

Ultrasonic sensors emit ultrasonic waves that are of higher frequencies than what can be detected by the human ear(20KHz). On encountering an obstacle the wave gets reflected and received back by the sensor, and the distance of how far the object is in front of it is calculated. The sensor has 4 pins VCC and GND go to 5V and GND pins on the Arduino, Trig to any digital pin and Echo to the PWM pins. Using the Trig pin, an ultrasonic wave is sent from

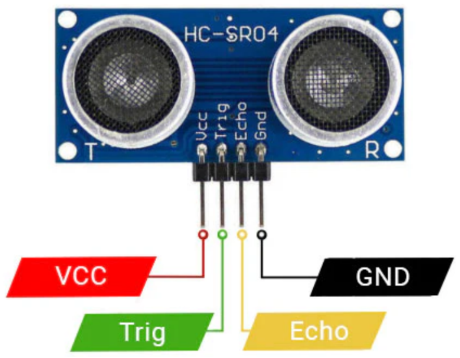


Figure 4: Ultrasonic sensor

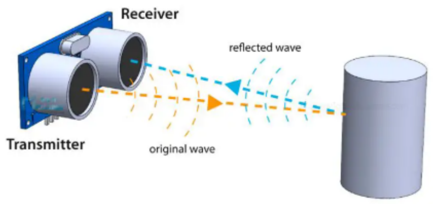


Figure 5: Working principle

the transmitter and with the Echo pin the reflected signal is listened for. Based on the time duration of the pulse received by echo, the Arduino calculates the distance to the object.

5 Other Components used

1. Battery - 3.7x3V
2. TIP-122 Transistor
3. Battery - 3.7x3V
4. Capacitor
5. Resistor
6. IN5408 Diode
7. Jumper wires
8. Signal Conditioning

The output signal from the sensor of a measurement system has to be processed for the next stage of operation and it can involve a combination of many processes like amplification, conversion from one type digital/analog to another etc. This process is called Signal Conditioning.

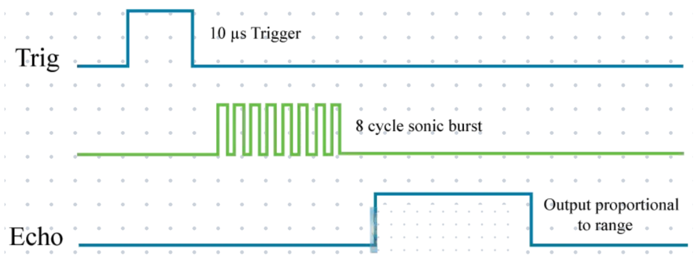


Figure 6: Output given by the ultrasonic sensor

6 Controller/Actuator used

6.1 Arduino Uno

This is our microcontroller board, which is based on the ATmega328P. It has 14 digital input/output pins(of which 6 can be used as PWM outputs), 6 analog inputs, a 16MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button.

6.2 Motor Driver

:

It receives a low current signal from the controller(here Arduino Uno) and converts it into a high current signal that is used to drive the motors. Within the motor driver, 4 transistors are used to control the speed and direction of the motors. Depending on whether a HIGH or LOW input is given by the microprocessor to the motor driver, the driver will rotate the motor in one direction, keeping one pin as HIGH and the other as LOW. The Enable A and Enable B pins are used for enabling and controlling the speed of the motor.

6.3 Servo motor

: It provides precise control over rotation, allowing it to move to and hold a specific angle. Servos are controlled by sending an electrical pulse of variable width or pulse width modulation(PWM) through the control wire. Based on the duration of the pulse, the rotor will turn to the desired position and hold. Here a servo motor is used on top of a motor driver for optimal coverage, as the water spraying nozzle is attached to the servo motor.

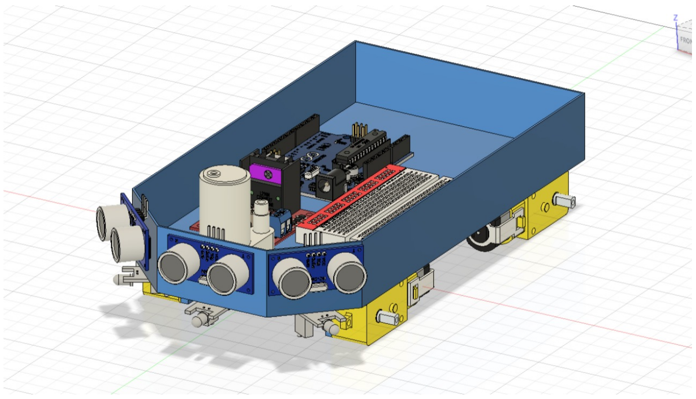


Figure 7: Arduino Uno

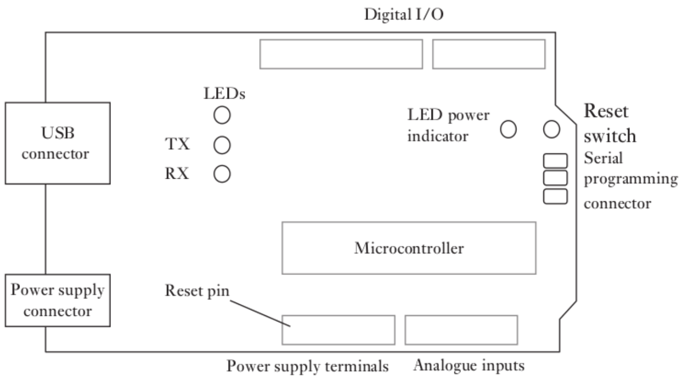


Figure 8: Basic elements of an Arduino Board

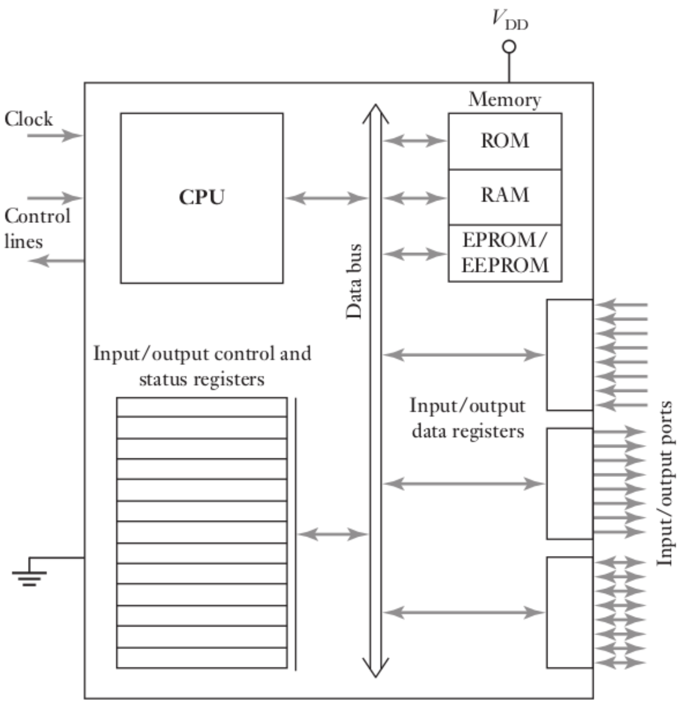


Figure 9: Block diagram of a microcontroller

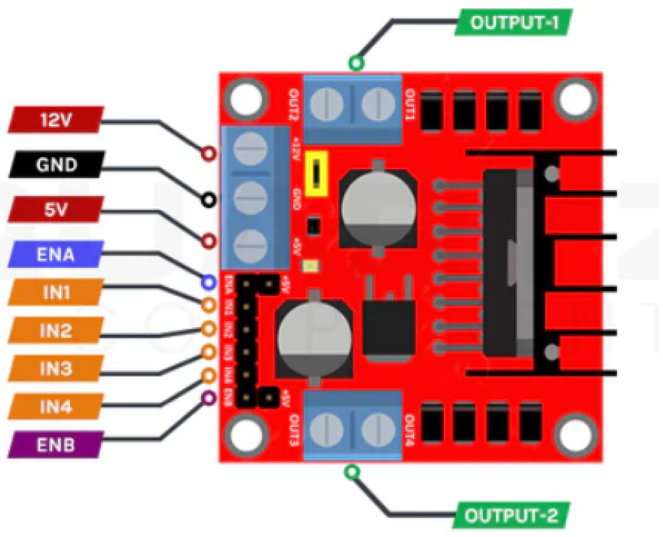


Figure 10: L298 Motor Driver

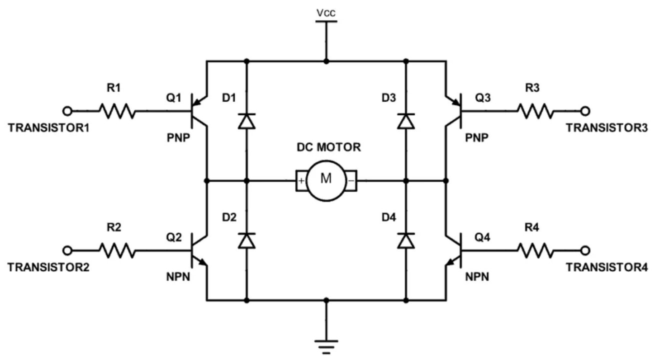


Figure 11: Circuit Diagram - The motor driver makes use of a H-bridge based circuit with transistors to control the motors with PWM signals

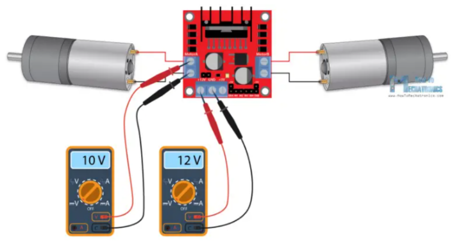


Figure 12: Motors connected to Motor driver

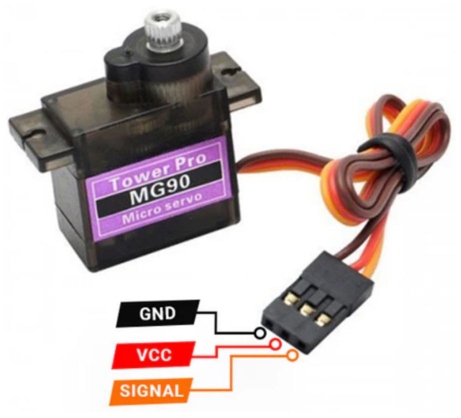


Figure 13: Servo motor

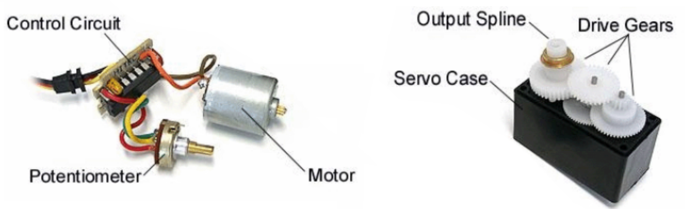


Figure 14: Components of servo motor

Figure 15: Water pump

6.4 Water pump

Once the bot reaches to a specified distance from the flame, the motors stop and the pump is turned on by the motor driver, to which it is connected. The servo motor is also turned on such that as it starts pumping water up, the nozzle rotates about covering maximum area to put out the fire.

7 Programming

Our Working Code

We first define the pins on the Arduino for the connections of each component.

The void setup() consists of the setup code which is run once, where the pinMode is set as INPUT/OUTPUT.

void loop() contains the program which runs continuously as the robot is operated.

analogRead() is used to read analogue input from the flame sensors.

Using the analogWrite() or digitalWrite() function we send the PWM signal to the enable pin of the L298 board, which actually drives the motor. We have user defined custom functions that define the movement forward, backward, left and right of the motors.

```
#define enA 10//Enable1 L298 Pin enA
#define in1 9 //Motor1 L298 Pin in1
#define in2 8 //Motor1 L298 Pin in2
#define in3 7 //Motor2 L298 Pin in3
#define in4 6 //Motor2 L298 Pin in4
#define enB 5 //Enable2 L298 Pin enB
```

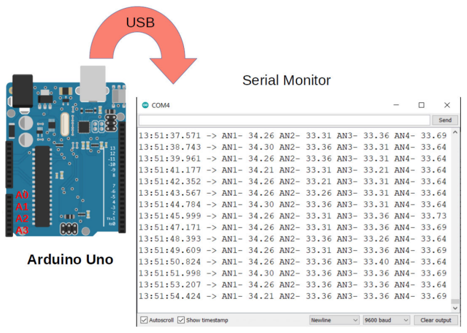
```
#define ir_R A0
#define ir_F A1
#define ir_L A2
#define servo A4
#define pump A5
int Speed = 160; // Write The Duty Cycle 0 to 255 Enable for Motor
    Speed
int s1, s2, s3;
void setup(){ //setup code here, to run once
Serial.begin(9600); // start serial communication at 9600bps
pinMode(ir_R, INPUT);// declare fire sensor pin as input
pinMode(ir_F, INPUT);
pinMode(ir_L, INPUT);
pinMode(enA, OUTPUT); // declare as output for L298 Pin enA
pinMode(in1, OUTPUT); // declare as output for L298 Pin in1
pinMode(in2, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
pinMode(enB, OUTPUT);
pinMode(servo, OUTPUT);
pinMode(pump, OUTPUT);
for (int angle = 90; angle <= 140; angle += 5) {
servoPulse(servo, angle); }
for (int angle = 140; angle >= 40; angle -= 5) {
servoPulse(servo, angle); }
for (int angle = 40; angle <= 95; angle += 5) {
servoPulse(servo, angle); }
analogWrite(enA, Speed); // Write The Duty Cycle 0 to 255 Enable
    Pin A for Motor1 Speed
analogWrite(enB, Speed); // Write The Duty Cycle 0 to 255 Enable
    Pin B for Motor2 Speed
delay(500);
}
void loop(){
s1 = analogRead(ir_R);
s2 = analogRead(ir_F);
s3 = analogRead(ir_L);
//===============================================================
// Auto Control
//===============================================================
Serial.print(s1);
Serial.print("\t");
Serial.print(s2);
Serial.print("\t");
Serial.println(s3);
delay(50);
if (s1 < 50 || s2 < 50 || s3 < 50) {  // adjust values if needed
Stop();
digitalWrite(pump, 1);
for (int angle = 90; angle >= 40; angle -= 3) {
servoPulse(servo, angle);
}
for (int angle = 40; angle <= 90; angle += 3) {
servoPulse(servo, angle);
}
}
else if(s1<250){
```

```
Stop();
digitalWrite(pump, 1);
for(int angle = 90; angle >= 40; angle -= 3){
servoPulse(servo, angle);
}
for(int angle = 40; angle <= 90; angle += 3){
servoPulse(servo, angle);
}
}
else if(s2<350){
Stop();
digitalWrite(pump, 1);
for(int angle = 90; angle <= 140; angle += 3){
servoPulse(servo, angle);
}
for(int angle = 140; angle >= 40; angle -= 3){
servoPulse(servo, angle);
}
for(int angle = 40; angle <= 90; angle += 3){
servoPulse(servo, angle);
}
}
else if(s3<250){
Stop();
digitalWrite(pump, 1);
for(int angle = 90; angle <= 140; angle += 3){
servoPulse(servo, angle);
}
for(int angle = 140; angle >= 90; angle -= 3){
servoPulse(servo, angle);
}
}
else if(s1>=251 && s1<=700){
digitalWrite(pump, 0);
backword();
delay(100);
turnRight();
delay(200);
}
else if(s2>=251 && s2<=800){
digitalWrite(pump, 0);
forword();
}
else if(s3>=251 && s3<=700){
digitalWrite(pump, 0);
backword();
delay(100);
turnLeft();
delay(200);
}else{
digitalWrite(pump, 0);
Stop();
}
delay(10);
}
void servoPulse (int pin, int angle){
int pwm = (angle*11) + 500; // Convert angle to microseconds
```

```
digitalWrite(pin, HIGH);
delayMicroseconds(pwm);
digitalWrite(pin, LOW);
delay(50); // Refresh cycle of servo
}
void forword(){ // move forward
  digitalWrite(in1, LOW);  // Reverse logic
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}

void backword(){ // move backward
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}

void turnRight(){ // turn right
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}

void turnLeft(){ // turn left
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}
void Stop(){ //stop
digitalWrite(in1, LOW); //Right Motor forword Pin
digitalWrite(in2, LOW); //Right Motor backword Pin
digitalWrite(in3, LOW); //Left Motor backword Pin
digitalWrite(in4, LOW); //Left Motor forword Pin
}
```

8 DAQ

The Arduino board can be used for data acquisition (DAQ) by sending the data from sensors to a computer or other devices for processing or storage. Outputs can be viewed in the Arduino IDE itself through the serial monitor, and this output can also be logged onto an excel sheet in a computer, which acts as a data acquisition system. It can be used to store time taken for executing fire extinguishing missions and data of regions prone to fire based on the distance computed by the sensing elements.



```
13:51:37.571 -> AN1- 34.26 AN2- 33.31 AN3- 33.36 AN4- 33.69
13:51:38.743 -> AN1- 34.30 AN2- 33.36 AN3- 33.31 AN4- 33.64
13:51:39.961 -> AN1- 34.26 AN2- 33.36 AN3- 33.31 AN4- 33.64
13:51:41.177 -> AN1- 34.21 AN2- 33.31 AN3- 33.21 AN4- 33.64
13:51:42.352 -> AN1- 34.26 AN2- 33.21 AN3- 33.31 AN4- 33.64
13:51:43.567 -> AN1- 34.26 AN2- 33.26 AN3- 33.31 AN4- 33.64
13:51:44.784 -> AN1- 34.30 AN2- 33.36 AN3- 33.31 AN4- 33.64
13:51:45.999 -> AN1- 34.26 AN2- 33.31 AN3- 33.36 AN4- 33.73
13:51:47.171 -> AN1- 34.26 AN2- 33.31 AN3- 33.36 AN4- 33.69
13:51:48.393 -> AN1- 34.26 AN2- 33.36 AN3- 33.26 AN4- 33.64
13:51:49.609 -> AN1- 34.26 AN2- 33.31 AN3- 33.36 AN4- 33.69
13:51:50.824 -> AN1- 34.26 AN2- 33.36 AN3- 33.40 AN4- 33.64
13:51:51.998 -> AN1- 34.30 AN2- 33.36 AN3- 33.36 AN4- 33.69
13:51:53.207 -> AN1- 34.26 AN2- 33.36 AN3- 33.36 AN4- 33.64
13:51:54.424 -> AN1- 34.21 AN2- 33.36 AN3- 33.36 AN4- 33.69
```

Figure 16: Arduino Serial Monitor

9 Modifications Proposed

The firefighting robot can be made completely autonomous by integrating it with an IOT device that receives information about the location where there's fire, so that it autonomously navigates to that location using algorithms like SLAM (Simultaneous Localization and Mapping), and on reaching there it switches to a more cautious mode and detects flame using the sensor and puts it off.