University of Engineering & Technology, Taxila

Microprocessor & Interfacing

Final Project Report

Smart Fire Extinguisher System

Members:

Sameen Nadeem (23-CP-12)

Affan Yasir (23-CP-24)

Iqra Fatima (23-CP-62)

Section: Omega Ω

Semester: Fourth (4th)

Instructor Name: Engr. Sharoon Saleem

Date of Submission: 30-05-2025

Abstract

This project presents the development of an **intelligent fire extinguisher system** that integrates multiple sensors, microcontrollers, and IoT communication to provide **early fire detection** and **automatic suppression** capabilities. The system employs an MQ-2 smoke sensor and IR-based flame sensor for comprehensive hazard detection, coupled with an **ATmega328P microcontroller with built in ESP8266 Wi-Fi module** for processing and communication. Upon fire detection, the system automatically activates a **water pump** for fire suppression while simultaneously sending **email alerts** to the property owner. The dual-sensor approach with verification logic significantly reduces false positives. Testing results demonstrate reliable fire detection with response times under 5 seconds and successful email delivery within 10 seconds of hazard detection. This cost-effective solution addresses the critical need for **automated fire safety systems** in residential and small commercial applications.

Table of Contents

1. Introduction

Fire hazards remain one of the most devastating and unpredictable risks to both life and property. According to the **National Fire Protection Association (NFPA)**, residential fires alone account for over 300,000 incidents annually in the United States, with global statistics reflecting similar concerns. In developing countries, lack of smart infrastructure further magnifies the risks.

Globally, fire-related incidents cause billions of dollars in economic losses and result in thousands of injuries and fatalities each year. Recent high-profile fire incidents, such as the **Notre-Dame Cathedral fire in 2019** and frequent factory fires in developing regions, highlight the dire need for intelligent fire prevention and suppression systems.

Despite advancements in safety infrastructure, conventional fire detection systems often fall short in several critical areas. Most traditional systems rely on isolated smoke detectors and manual fire suppression methods that demand human intervention, often causing delays in emergency response. Moreover, these systems usually lack any real-time remote communication capabilities, leaving property owners uninformed during crucial moments especially when the premises are unattended.

Considering these limitations, the advancement of the **Internet of Things (IoT)** has opened new possibilities in the domain of intelligent safety and automation. By integrating IoT with embedded control systems and smart sensors, it is now possible to develop autonomous fire safety solutions that not only detect hazards early but also respond immediately and notify relevant stakeholders without requiring human involvement. This project leverages this technological advancement to design and implement a **Smart Fire Extinguisher System** that is capable of early detection, automated suppression, and remote alerting.

The system is built around the collaboration of two microcontrollers: an **ATmega328P** (commonly found in Arduino Uno boards) and the **ESP8266 Wi-Fi module**. The ATmega328P is responsible for monitoring fire-related environmental parameters through a **flame sensor** and an **MQ-series gas sensor**. Upon detecting abnormal conditions such as the presence of a flame or an unusually high concentration of combustible gases the system activates a **buzzer alarm** and triggers a **relay module** to power a water pump, thereby initiating fire suppression automatically.

In parallel, the ESP8266 microcontroller manages **wireless communication and cloud-based alerting**. When an alert condition is confirmed by the ATmega328P, a serial message is transmitted to ESP8266. This module then connects to a secure Wi-Fi network and sends an **email notification** to a predefined recipient using the **Gmail SMTP server**, providing real-time information about the detected hazard.

This smart fire extinguisher system has been designed with the following key features and goals in mind:

- **Early Detection:** Utilizes a **dual-sensor configuration** (flame + gas) to ensure rapid and reliable hazard identification.
- **Automated Response:** Controls a **water pump via relay** to extinguish fire upon confirmation.
- **Remote Monitoring:** Sends **real-time email notifications** to the user or property owner through the ESP8266.
- **False Positive Reduction:** Incorporates **verification logic** to minimize false alarms by rechecking sensor values before triggering the suppression system.
- **Cost-Effectiveness:** Employs **affordable components** and **open-source platforms**, making the system accessible for homes, small businesses, and academic use.

The motivation for this project is drawn from the growing demand for **intelligent, autonomous fire safety systems** that not only prevent major damage but also offer peace of mind to users. The need is particularly urgent in environments that remain unattended for extended periods such as warehouses, rural homes, or educational institutions after hours where traditional manual systems are not effective or fast enough. By combining **real-time sensing**, **automated actuation**, and **IoT-based communication**, this project aims to present a robust and scalable solution for modern fire safety challenges.

2. Related Work

Recent advances in IoT-based fire detection systems have demonstrated significant improvements in early fire detection and automated response capabilities. This section reviews key contributions in the field that inform the design and implementation of our system.

2.1. IoT-Based Fire Alarm System Using NodeMCU ESP8266

This project utilizes a NodeMCU ESP8266 microcontroller connected to flame and gas sensors to detect fire and smoke. Upon detection, it sends alerts via the Blynk app. [1]

Like our project, it employs Wi-Fi-based alerts but does not include an automated fire suppression mechanism like a water pump.

2.2. GSM-Based Fire Alert System Using Arduino

An Arduino UNO is connected to a flame sensor, gas sensor, and a GSM module (SIM800L) to send SMS alerts to emergency contacts upon detecting fire. [2]

It shares a similar detection method and uses SMS for alerts instead of Wi-Fi/email. However, it lacks an automated extinguishing feature.

2.3. Arduino Fire Fighting Robot with Auto Extinguisher

This project involves a mobile robot equipped with fire sensors and a water spraying module, controlled via Arduino. It detects fire, moves toward it, and extinguishes it. [3]

It includes an automated suppression feature like our system but is mobile and robot-based, not a fixed installation.

2.4. Flame Alert System with Blynk

This system uses a flame sensor to detect potential fires and sends the detected values to the Blynk app for remote monitoring. In case of a fire, Blynk promptly notifies the user via email. [4]

It offers real-time IoT alerts like our project but lacks a suppression feature like a water pump system.

2.5. Forest Fire Detection System Using Wireless Sensor Networks

This paper proposes a system that detects forest fires at the initial stage using a wireless sensor network. It employs a machine learning regression model for more accurate fire detection. [5]

Focused on fire detection and remote notification, though on a much larger scale without extinguishing capabilities.

2.6. Fire Detection System Using Flame Sensor & Arduino

This project involves building a fire detection system using a flame sensor and Arduino. It triggers a buzzer and LED upon detecting fire. [6]

It shares detection and buzzer alert features but lacks remote notification or suppression mechanisms.

2.7. Fire Alert Notification Using ESP8266 and Arduino IoT Cloud

There is a tutorial which demonstrates building a fire alert system using ESP8266 and a flame sensor. It sends real-time email notifications through the Arduino IoT Cloud upon detecting fire.[7]

It incorporates similar notification technique to our system; does not include extinguishing hardware like a relay or pump.

2.8. Research Gaps and Innovation in our System

While existing systems demonstrate various approaches to IoT-based fire detection, several gaps remain:

1. **Cost Barrier**: Many systems require expensive sensors or cameras
2. **False Positive Issues**: Single-sensor systems are prone to false activations
3. **Limited Accessibility**: Complex systems requiring specialized installation
4. **Integration Challenges**: Difficulty in combining detection, suppression, and notification

Our system addresses these gaps through:

- **Dual-sensor verification** to reduce false positives
- **Cost-effective component selection** using readily available sensors

- **Email-based notification system** for broader accessibility
- **Integrated design** combining all functions in a single, cohesive system

3. System Design and Methodology

3.1 System Architecture

The Smart Fire Extinguisher System employs a distributed architecture comprising two main processing units: an ATmega328P microcontroller for sensor management and local control, and an ESP8266 Wi-Fi module for internet connectivity and remote notifications. This dual-controller approach enables efficient task distribution and maintains system reliability through component isolation.

3.1.1 Hardware Architecture

Core Components:

- **Main Controller**: ATmega328P + ESP8266 integrated development board
- **Sensors**: MQ-2 smoke/gas sensor and IR-based flame sensor (760-1100nm)
- **Actuators**: 5V single-channel relay module controlling 3-5V DC submersible water pump
- **Alert System**: Piezoelectric buzzer for local audio alerts
- **Power Management**: Separate power supplies for development board (USB) and pump system (3.7V Li-ion battery)

3.1.2 Software Architecture

ATmega328P Firmware:

- Sensor data acquisition and processing
- Fire detection logic with verification algorithms
- Local alert generation and pump control
- Serial communication with ESP8266 module

ESP8266 Firmware:

- Wi-Fi connectivity management
- SMTP email client implementation
- Alert message formatting and transmission
- Communication protocol handling with ATmega328P

3.2 Component Selection and Specifications

Component	Quantity	Description
Arduino UNO WiFi R3 (with ESP8266)	1	Microcontroller with integrated WiFi
MQ-2 Gas/Smoke Sensor	1	Detects smoke, propane, methane, LPG
IR Flame Sensor	1	Detects fire flames via IR radiation
Piezo Buzzer	1	Sounds like an alarm when hazard detected
1-Channel Relay Module	1	Controls high current DC pump
5V DC Mini Water Pump	1	Sprays water to extinguish fire
5V adapter or 18650 Li-Ion Batteries + Holder	1	Powers the system
Breadboard & Jumper Wires	As needed	For circuit connections
7.4V Battery + Holder (for pump)	1	Powers the DC pump separately

Table: Components Lists with Quantity

3.2.1 Specifications and Working of Components

1. Arduino UNO WiFi R3 (with ESP8266 8Mb Flash, CH340G)

Specifications:

- **Microcontroller:** ATmega328P
- **WiFi Module:** ESP8266 (integrated on board)
- **Digital I/O Pins:** 14 (D0–D13)
- **Analog Input Pins:** 6 (A0–A5)
- **Operating Voltage:** 5V
- **Input Voltage (recommended):** 7V–12V
- **Clock Speed:** 16 MHz
- **Flash Memory:** 32 KB (ATmega328P) + 8Mb (ESP8266)
- **USB-to-Serial Converter:** CH340G
- **ESP pins**
- **Switches to adjust between different modes**

Working:

This board acts as the **main controller** for the system. It:

- **Reads data** from the Flame sensor and MQ-2 gas sensor.

- **Controls the relay** to switch the water pump ON/OFF.
- **Activates a buzzer** on fire/smoke detection.
- **Uses the built-in ESP8266 WiFi module** to connect to WiFi and send an email alert using AT commands.
- **Operating Modes**:
    - Mode 1 (USB to AT): SW3, SW4 ON
    - Mode 2 (USB to ESP Prog): SW5, SW6, SW7 ON
    - Mode 3 (USB to ESP): SW5, SW6 ON
    - Mode 4 (AT to ESP): SW1, SW2 ON

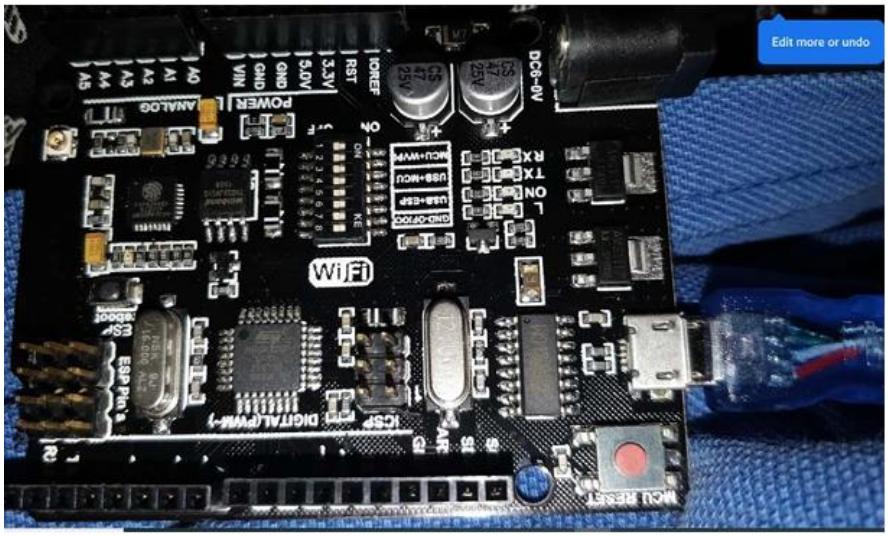

Figure: Arduino with esp8266(Atmega328p+Esp8266)

2. Flame Sensor

Specifications:

- **Detection Range:** 760nm–1100nm (Infrared light spectrum from flames)
- **Output:** Digital (0V or 5V)
- **Operating Voltage:** 3.3V–5V
- **Detection Distance:** 0–80 cm (depends on flame size)

Working:

The flame sensor detects infrared light emitted by a fire.

- **When a flame is present:**
  Sensor output = **LOW (0V)**
- **When there is no flame:**
  Sensor output = **HIGH (5V)**
  The Arduino constantly monitors this digital signal to detect a fire.

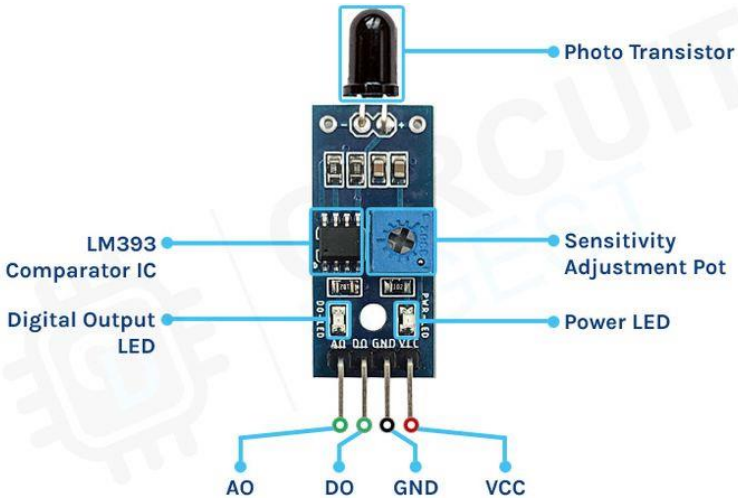

Figure: IR based Flame Sensor

3. MQ-2 Gas Sensor

Specifications:

- **Gas Types Detected:** LPG, Butane, Methane, Alcohol, Hydrogen, Smoke
- **Output:** Analog and Digital
- **Operating Voltage:** 5V
- **Detection Range:** 300–10000 ppm
- **Preheat Time:** 20 seconds
- **Sensitivity Adjustable:** Using onboard potentiometer (for digital output)

Working:

The MQ-2 contains a **sensitive material (SnO2)** whose conductivity increases with the presence of smoke or combustible gases.

- The sensor provides an **analog voltage** proportional to the gas concentration.
- Arduino reads this analog voltage through the **A0 pin**.
- If the value crosses a defined threshold, it indicates smoke presence.
- Optionally, its **digital pin can also be used**, but analog provides better accuracy.

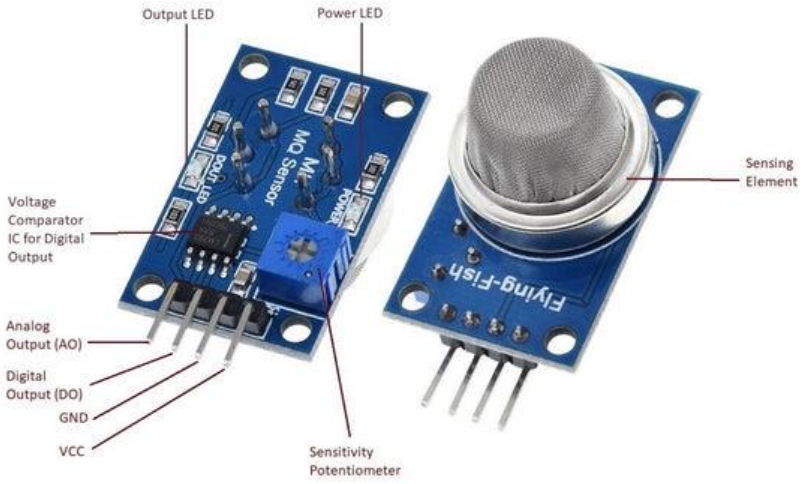

Figure: MQ-2 Smoke Sensor

4. Relay Module (5V)

Specifications:

- **Operating Voltage:** 5V DC
- **Control Signal Voltage:** 3.3V–5V (compatible with Arduino)
- **Maximum Switching Current:** 10A at 250V AC / 10A at 30V DC
- **Contacts:** Normally Open (NO), Normally Closed (NC), Common (COM)

Working:

A relay is an **electromechanical switch**.

- Arduino sends a **HIGH signal to the relay IN pin**.

- The relay coil is energized, closing the **NO (Normally Open)** contact.
- This allows current to flow from a **7.4V Li-ion battery to the water pump**, turning it ON.
- When the signal goes LOW, the relay turns OFF, cutting off power to the pump.

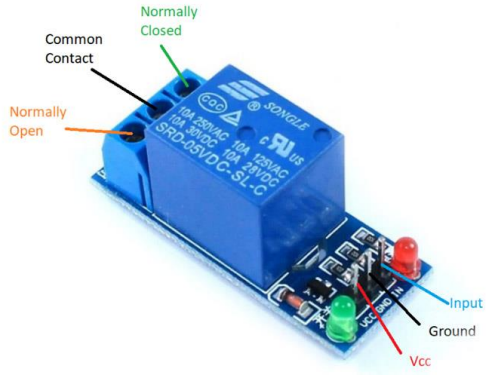

Figure: Relay Module

5. DC Water Pump (5V–12V)

Specifications:

- **Operating Voltage:** 5V–12V
- **Flow Rate:** 80–120 liters per hour (varies by model)
- **Current:** 0.3A–0.7A
- **Max Water Lift:** 0.6m–1.2m

Working:

- The water pump draws power from a **7.4V battery through the relay**.
- When the relay closes, the pump turns ON and sprays water towards the fire.
- Once the fire is controlled and sensors stop detecting danger, the relay opens, stopping the pump.

Figure: Mini submersible Water pump

6. Piezo Buzzer

Specifications:

- **Operating Voltage:** 3V–5V
- **Sound Output:** 85 dB–100 dB
- **Current Consumption:** 20mA–30mA
- **Frequency:** 2300Hz–5000Hz

Working:

- Controlled by Arduino on **digital pin D7**.
- When fire/smoke is detected:
  - Arduino sends a **HIGH signal to D7**.
  - Buzzer sounds an alarm to immediately alert people nearby.
- When no danger:
  - Arduino sends a **LOW signal**, turning it OFF.

Figure: Piezzo Buzzer

7. 7.4V Li-ion Battery Pack (for Pump)

Specifications:

- **Voltage:** 7.4V (2x 18650 Li-ion cells in series)
- **Capacity:** 1200mAh–2600mAh per cell (depends on cells)
- **Rechargeable:** Yes

Working:

- Provides sufficient voltage and current to operate the **water pump via relay**.
- Isolated from the Arduino's 5V supply for safety.
- Rechargeable and portable.

8. Battery Holder

Figure: Cell Holder

Specifications:

- Holds 1x 3.7V lithium-ion cell
- With two wires for connecting

Working:

Will be used to hold cell to power up pump.

9. Breadboard & Jumper Wires

Specifications:

- Standard 400/830 tie-point breadboards
- Male-to-Male jumper wires

Working:

- Used to connect sensors, buzzer, relay, and modules to the Arduino in a **temporary and flexible lawet** for testing and demonstration.

3.2.2 Summary Table

Component	Purpose	Control/Output Type
Arduino UNO WiFi R3	System controller, WiFi comm.	Digital / Analog
Flame Sensor	Detects fire flame	Digital
MQ-2 Gas Sensor	Detects smoke, gases	Analog
Relay Module	Switches water pump	Digital
Water Pump	Extinguishes fire	Controlled by Relay
Piezo Buzzer	Local sound alarm	Digital
ESP8266 WiFi Module	Sends Email alert	Serial (AT Commands)
7.4V Li-ion Battery	Powers pump	Power Supply
18650 Battery Pack	Powers Arduino and sensors	Power Supply

4. Circuit Implementation

4.1 Pin Configuration

First we did the following connections for Arduino pins:

- **Digital Pin 2**: Flame sensor input (D0)
- **Analog Pin A0**: MQ-2 gas sensor analog output (A0)
- **Digital Pin 7**: Relay control (pump activation)
- **Digital Pin 8**: Buzzer control
- **Serial Pins** : Communication between atmega328p and ESP8266 is done through serial pins. If you are:
    - **Using atmega328p with built in esp8266**

      No connections needed. We will use DIP switches to use ESP module.
    - **Using separate esp8266 and Arduino**

      Connect Arduino pin 1 (TXD) with RX of esp8266

      Connect Arduino pin 0 (RXD) with TX of esp8266
- Connected 5V and GND to VCC and GND respectively of all these components through bread board horizontal rail:
    a) Buzzer
    b) Relay
    c) Flame Sensor
    d) Smoke Sensor
- Connected 3.7 V battery using a holder with power terminals of relay module.
- Then we can Powerup using Battery or a micro-USB.

4.2 Circuit Diagram:

Below is the circuit diagram of the components with pin connections used in our project. We used separate esp8266 and Arduino instead of combined one to show the internal connection of esp8266 with atmega328p in our built in atmega328p+esp8266 development board. It would also be helpful for the one who want to do a similar project using a separate esp8266 module.

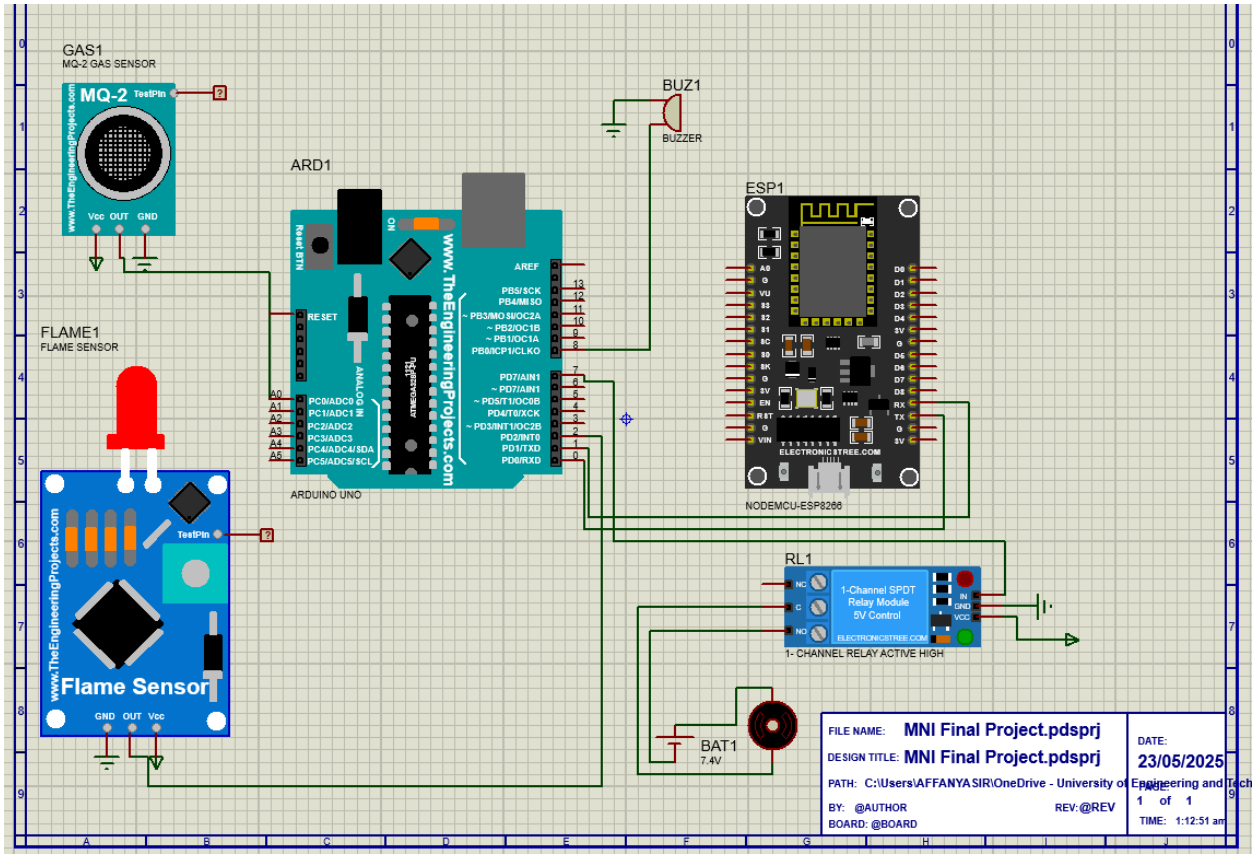

Figure: Connection diagram

4.3 Physical Circuit:

Now we will attach the physical circuit according to the circuit diagram shown above except RX, TX pins of ESP8266 because our Development board has these connections built in with the ESP8266. So, we will leave those connections and do remaining connections accordingly. If you are using a separate esp8266 module you would have to make these connections too.

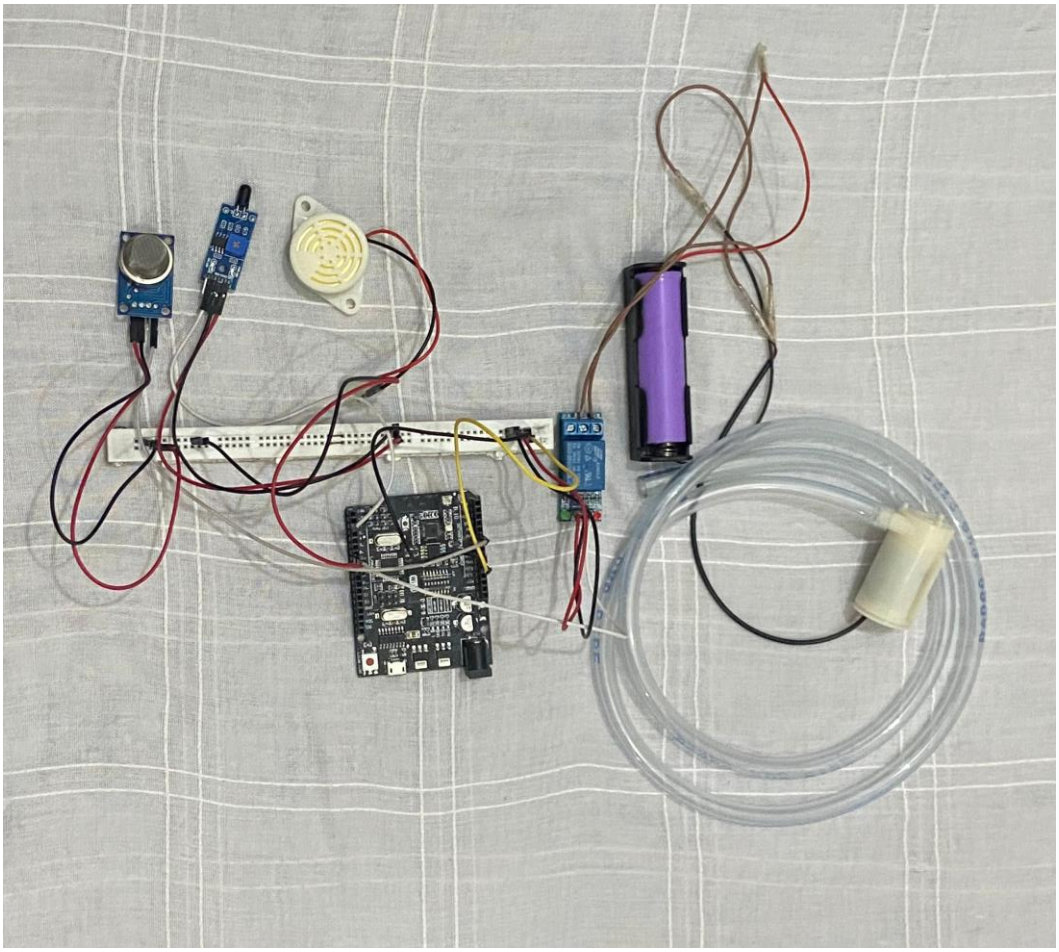

Figure: Circuit after connecting all the components together

5. Software Implementation

5.1 Libraries Required:

Before uploading code to atmega328p or esp8266 we have to first download these libraries:

a) Esp8266
b) Esp Mail Client

5.2 Source Code:

We created two .ino files having the source code provided below using Arduino ide.

ATmega328P Firmware (fire_detection.ino)

```
#include <SoftwareSerial.h>


// Pin Definitions
#define FLAME_SENSOR 2
#define GAS_SENSOR A0  // Now using analog pin A0
#define BUZZER 8
#define RELAY 7


bool alertSent = false;
int gasThreshold = 0;


void setup() {
  pinMode(FLAME_SENSOR, INPUT);
  pinMode(BUZZER, OUTPUT);
  pinMode(RELAY, OUTPUT);


  digitalWrite(BUZZER, LOW);
  digitalWrite(RELAY, HIGH);
```

```
  Serial.begin(9600);

  delay(2000);  // Give time to initialize

  // Set gas threshold after warm-up
  int baseline = analogRead(GAS_SENSOR);
  gasThreshold = baseline + 40;  // Adjust as needed

  Serial.println("Smart Fire Extinguisher System Ready");
  Serial.print("Gas baseline: "); Serial.println(baseline);
  Serial.print("Gas threshold: "); Serial.println(gasThreshold);
}

void loop() {
  int flameDetected = digitalRead(FLAME_SENSOR);
  int gasValue = analogRead(GAS_SENSOR);
  bool gasDetected = gasValue > gasThreshold;

  Serial.print("Flame: "); Serial.print(flameDetected);
  Serial.print(" | Gas Value: "); Serial.print(gasValue);
  Serial.print(" | Threshold: "); Serial.println(gasThreshold);

  // 🔔 Buzzer if any danger is detected
  if (flameDetected == LOW || gasDetected) {
    digitalWrite(BUZZER, HIGH);
     delay(5000);
```

```
} else {
  digitalWrite(BUZZER, LOW);
  delay(5000);
}

// 🔥 Confirmed flame detection
if (flameDetected == LOW && !alertSent) {
  Serial.println("🔥 Flame detected. Verifying...");
  delay(2000);
  if (digitalRead(FLAME_SENSOR) == LOW) {
    digitalWrite(RELAY, LOW);
    alertSent = true;
    Serial.println("☑️ Fire confirmed. Pump activated.");
  } else {
    Serial.println("❌ False flame alert.");
  }
}

// 💨 Confirmed gas detection
if (gasDetected && !alertSent) {
  Serial.println("💨 Gas detected. Verifying...");
  delay(2000);
  int recheckGas = analogRead(GAS_SENSOR);
  if (recheckGas > gasThreshold) {
    digitalWrite(RELAY, LOW);
    alertSent = true;
    Serial.println("☑️ Gas confirmed. Pump activated.");
```

```
    } else {
      Serial.println("❌ False gas alert.");
    }
  }

  // 🔁 Reset pump if both danger signals are gone
  if (flameDetected != LOW && analogRead(GAS_SENSOR) <= gasThreshold && alertSent) {
     delay(5000);
    if (digitalRead(FLAME_SENSOR) != LOW && analogRead(GAS_SENSOR) <= gasThreshold) {
      digitalWrite(RELAY, HIGH);
      alertSent = false;
      Serial.println("ℹ No danger. Pump deactivated.");
    }
  }

  // Communication with ESP8266
  if (gasDetected || flameDetected == LOW) {
    Serial.println("ALERT_TRIGGERED");
    delay(1000); // Prevent spamming
  }

  delay(100); // Stability delay
}
```

ESP8266 Firmware (email_notification.ino)

```
#include <ESP8266WiFi.h>

#include <ESP_Mail_Client.h>


// WiFi configuration for eduroam

#define WIFI_SSID "eduroam"

#define WIFI_PASSWORD ""


// SMTP configuration for Gmail

#define SMTP_HOST "smtp.gmail.com"

#define SMTP_PORT 465


// Email configuration

#define AUTHOR_EMAIL "razanidrive87442@gmail.com"

#define AUTHOR_PASSWORD "umwy dcvd vprv unif"  // Use App Password if 2FA
enabled

#define RECIPIENT_EMAIL "affanyasir16@gmail.com"


SMTPSession smtp;


void setup() {

  Serial.begin(9600);


  // Connect to eduroam WiFi

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);


  Serial.print("Connecting to eduroam");
```

```
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println("Waiting for Arduino alerts...");
}

void loop() {
  if (Serial.available() > 0) {
    String message = Serial.readStringUntil('\n');
    message.trim();

    if (message == "ALERT_TRIGGERED") {
      Serial.println("Fire/Gas alert detected! Sending email...");
      sendEmailAlert();
      delay(5000); // Prevent multiple emails for same alert
    }
  }
}

void sendEmailAlert() {
  smtp.debug(1);  // Enable debug output

  ESP_Mail_Session session;
```

```
  session.server.host_name = SMTP_HOST;
  session.server.port = SMTP_PORT;
  session.login.email = AUTHOR_EMAIL;
  session.login.password = AUTHOR_PASSWORD;
  session.login.user_domain = "";

  SMTP_Message msg;
  msg.sender.name = "Fire Safety System";
  msg.sender.email = AUTHOR_EMAIL;
  msg.subject = "URGENT: Fire/Gas Detected!";
  msg.addRecipient("Owner", RECIPIENT_EMAIL);

  // HTML Email Content
  String htmlMsg = R"(
  <div style="font-family: Arial, sans-serif; color: #d32f2f;">
    <h2 style="color: #b71c1c;">🚨 EMERGENCY ALERT 🚨</h2>
    <p>The fire safety system has detected a potential hazard!</p>

    <h3>⚠ Possible Hazards:</h3>
    <ul>
      <li>Fire detected by flame sensor</li>
      <li>Dangerous gas concentration detected</li>
    </ul>

    <p><strong>Action Required:</strong> Please check the premises
immediately!</p>
```

```
    <div style="margin-top: 20px; padding: 10px; background-color:
#ffebee; border-left: 4px solid #f44336;">

      <p>System alert triggered at: )";

  htmlMsg += String(millis() / 1000);

  htmlMsg += R"( seconds after startup</p>

    </div>

  </div>

  )";


  msg.html.content = htmlMsg.c_str();

  msg.text.content = "EMERGENCY: Fire/Gas detected! Check premises
immediately.\n\nTrigger time: " + String(millis() / 1000) + "s after
system start";

  msg.priority = esp_mail_smtp_priority::esp_mail_smtp_priority_high;


  if (!smtp.connect(&session)) {

    Serial.println("Connection error!");

    return;

  }


  if (!MailClient.sendMail(&smtp, &msg)) {

    Serial.print("Error sending email: ");

    Serial.println(smtp.errorReason());

  }

}


void smtpCallback(SMTP_Status status) {

  Serial.println("----------------");
```

```
  Serial.println("Email Status:");
  Serial.print("Info: ");
  Serial.println(status.info());
  if (status.success()) {
    Serial.println("Result: Success");
    Serial.print("Messages sent: ");
    Serial.println(status.completedCount());
    Serial.print("Messages failed: ");
    Serial.println(status.failedCount());
  } else {
    Serial.println("Result: Failed");
  }
  Serial.println("----------------"); }
```

Explanation:

1. fire_detection.ino (ATmega328P Firmware)

This firmware is responsible for detecting flame and gas presence using relevant sensors, controlling the output devices (buzzer and water pump via relay), and communicating with the ESP8266 for alert notification.

Key Functionalities:

- **Sensor Initialization:** The flame sensor (digital) and gas sensor (analog) are initialized in the setup() function. The gas sensor is calibrated to a dynamic threshold based on initial ambient readings.
- **Continuous Monitoring:** The loop() function continuously reads sensor values to detect fire or hazardous gas concentration.
- **Buzzer Activation:** The buzzer is turned on for 5 seconds whenever flame or gas is detected.
- **Pump Activation Logic:** Upon confirmed detection (verified through a second reading), the system activates a water pump by switching the relay.
- **Danger Resolution:** If the fire and gas levels return to normal, the system deactivates the pump and resets the alert state.

- **Serial Communication:** The Arduino sends an "ALERT_TRIGGERED" message to ESP8266 via the serial interface when danger is detected.

2. email_notification.ino (ESP8266 Firmware)

This firmware enables ESP8266 to handle wireless connectivity and send emergency email notifications upon receiving an alert signal from the Arduino.

Key Functionalities:

- **Wi-Fi Connectivity:** Connects the ESP8266 to the eduroam network using the configured SSID.
- **Serial Listening:** Continuously listens to the Serial port for the "ALERT_TRIGGERED" signal sent by the ATmega328P.
- **Email Sending:** Once an alert is received, the ESP8266 constructs and sends a formal emergency email using Gmail's SMTP server. The email includes:
    - A brief description of the detected danger (flame/gas)
    - A stylized HTML alert message with red emergency indicators
    - A timestamp showing how many seconds after system startup the alert occurred
- **Security:** An app-specific password is used to authenticate the Gmail account, which is essential when using 2-Step Verification (2FA).

This dual-microcontroller approach provides a reliable and responsive method for fire and gas detection, immediate local response, and remote notification, making it suitable for use in smart safety systems in residential or industrial environments.

5.3 Uploading Code:

First we switched to atmega328p by turning on switches 3,4 of our development board. Make sure to always reset MCU after changing switches. Then we selected the Arduino UNO and corresponding ports and finally uploaded the code after compiling in Arduino.

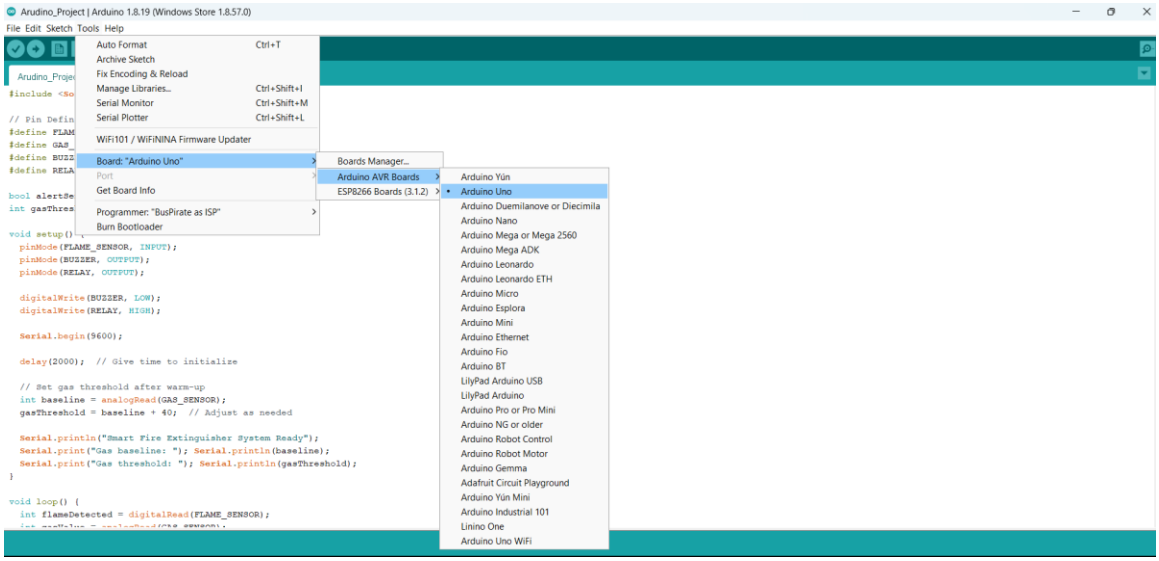

Figure: Selecting arduino uno from boards

Then we turned to ESP upload mode by turning on switches 5,6 & 7. Then we reset the MCU by pressing button. Then we compiled and uploaded the code.

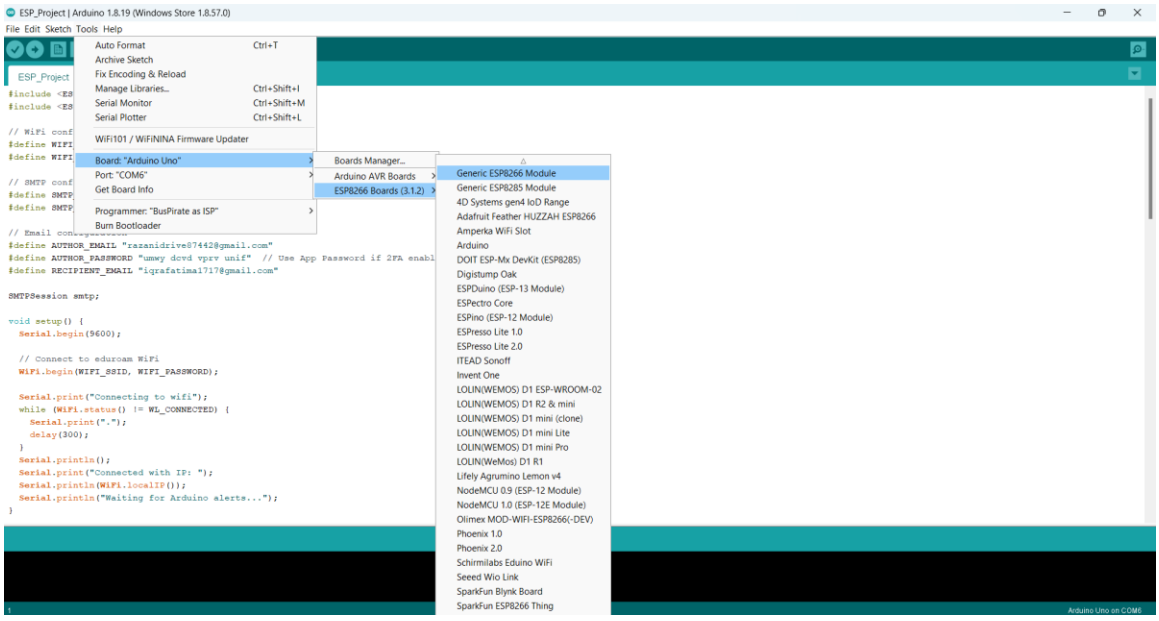

Figure: Selecting esp8266 from boards

6. System Integration and Testing

6.1 ATmega328P sensor reading validation

First we moved to atmega328p to USB mode by turning ON switches 3,4 of development board. Selected the corresponding port and board. The following screen would display:

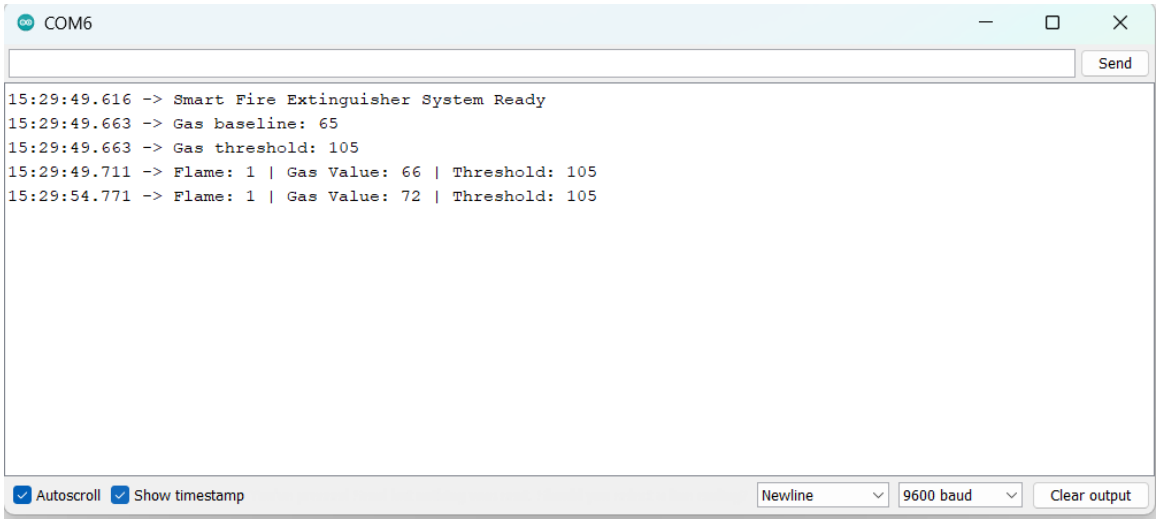

Figure: Initializing the values and readings displayed

Then we brought flame near the sensor for few seconds. Buzzer was triggered but not the alarm as we removed it before limit. So, it detected it as a false alert.

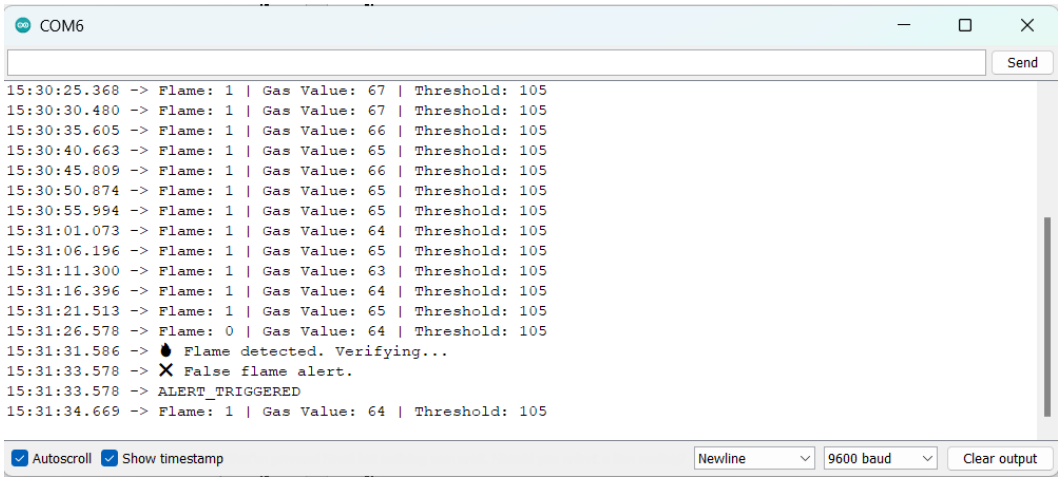

Figure: False flame detected

Then we hold the flame near the sensor for a bit longer, so it detected the flame as original and turned the pump ON. Later the pump deactivated when no flame was found.

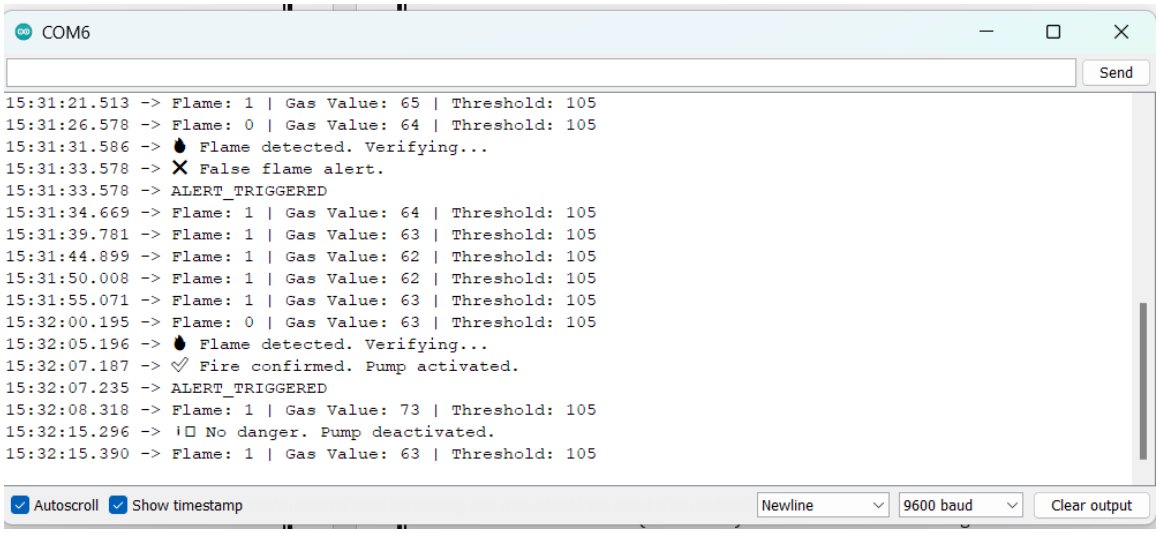

Figure: Pump Activation Message

Then We brough smoke near the gas sensor but for less time than the limit of 2 second. So, It detected it as false alert.

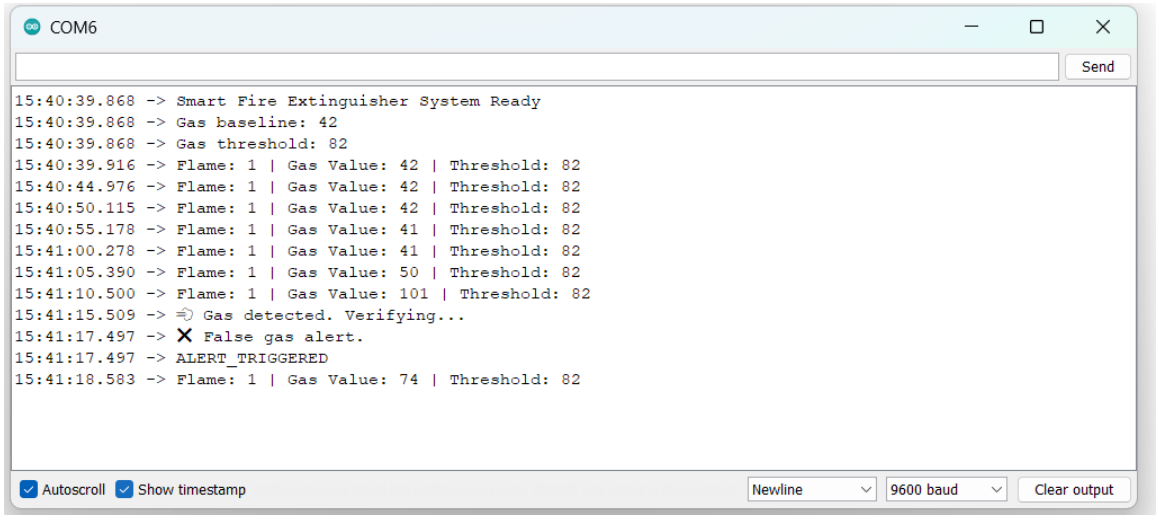

Figure: False Gas Detection Alert

But when we keep it for longer it was detected as a true alert and so it turned the relay ON due to which pump got activated. It would stay on untill flame/gas does not stop being detected by the sensors.

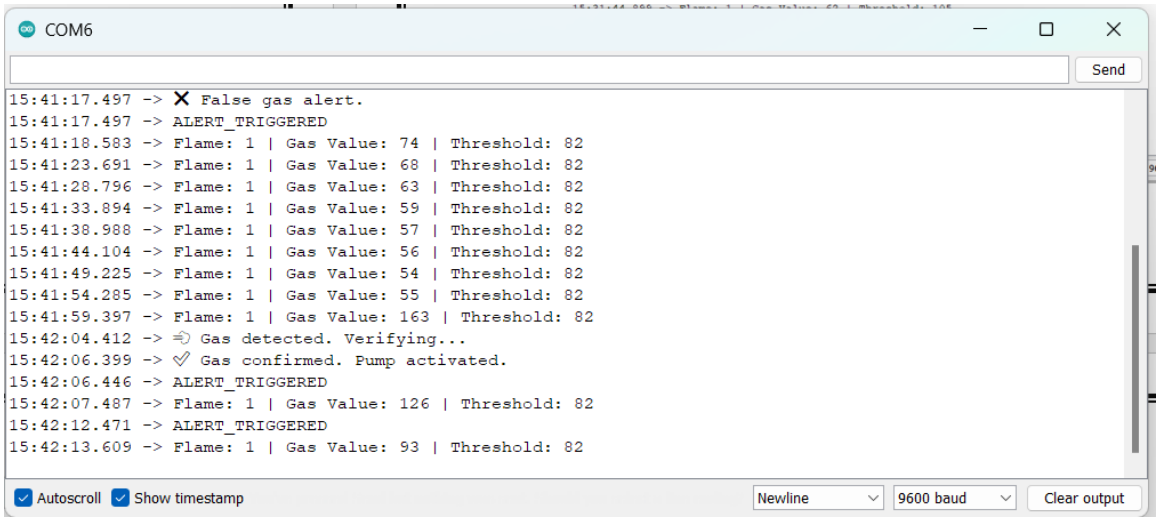

Figure: Pump Activation Message

6.2 ESP8266 Wi-Fi connectivity verification

Then we turned ESP mode by turning on 5,6 switches. Wi-Fi was connected successfully.

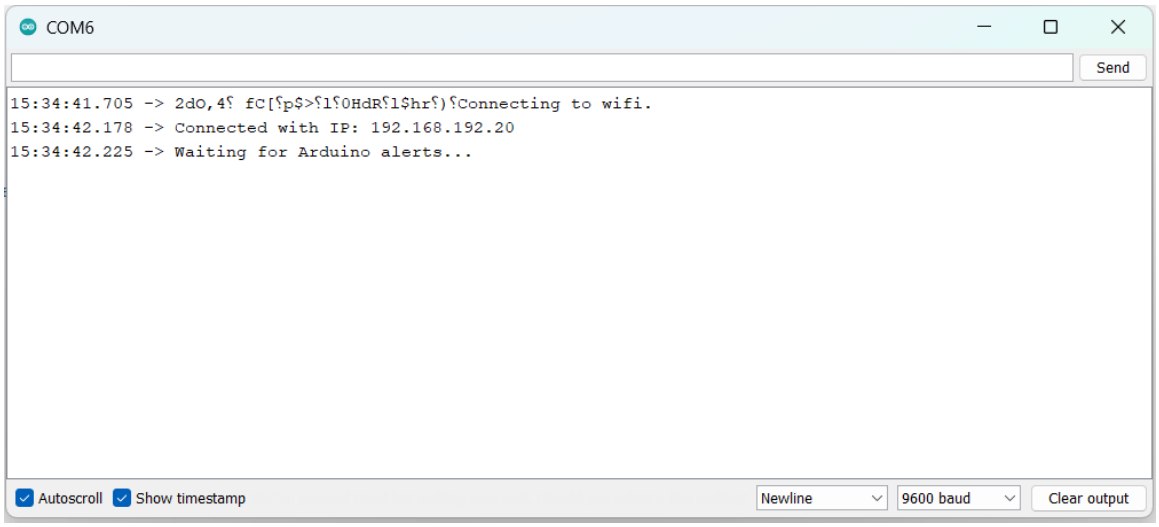

Figure: Wi-Fi-Connection confirmation message

6.3 ESP8266 Wi-Fi and Atmega328p interconnectivity verification

Then we turned 1,2 and 5,6 on so that Arduino and esp8266 can communicate and we can see esp8266 details on serial monitor. Then we brought flame near the flame sensor, It triggered buzzer, relay and an alert was received on the corresponding email. This all connection and procedure was also shown on the serial monitor as shown in the figure,

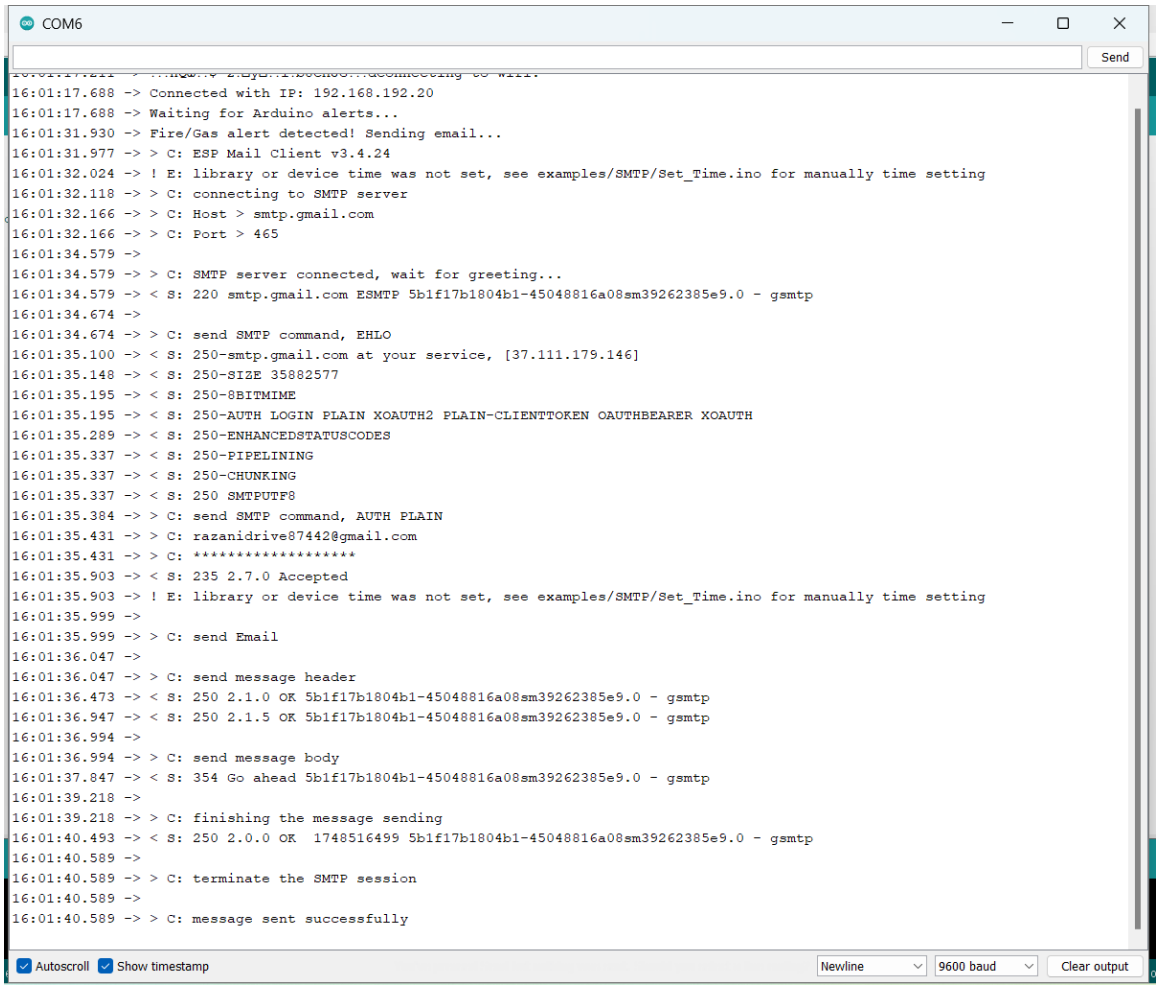

Figure: Email Alert Confirmation Message

6.4 Complete System Testing

We moved our circuit into a design project and tested everything again. We removed from serial-monitor. Switched On 1,2 switches. So, it will just communicate between esp8266 and Arduino.

Figure: Brought flame near the sensor

Then we observed how first buzzer was turned ON and after few seconds relay was turned ON which activated the water pump. At the same time we received email on our phone.

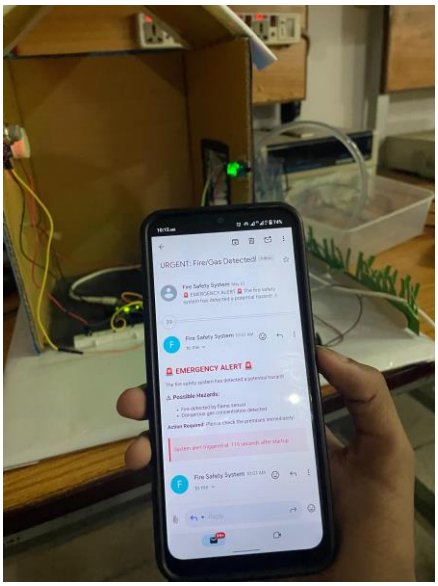

Figure: Email Received

6.5 Demonstration Videos:

The entire system is working as intended. We recorded videos which can be checked on the links below:
https://1drv.ms/f/c/17379a8d00e56608/Elkq8_bAppNKhEwiy_Wv5YoBSW8DZHRFnPpKoW_WTOMAiQ?e=JV9ACz

7. Conclusion:

In conclusion, this project successfully demonstrates the design and implementation of an Arduino-Based Smart Fire Extinguisher System that integrates both fire detection and suppression functionalities into a single, automated safety solution. Utilizing the MQ-2 gas sensor and a flame sensor, the system reliably detects the presence of smoke and open flames in its environment. Once a hazard is identified, the Arduino UNO WiFi R3 enhanced with the ESP8266 module immediately activates a local piezoelectric buzzer for audible alerts and triggers a relay-controlled motorized water pump to suppress the fire. Simultaneously, the ESP8266 sends real-time email notifications to remote users, ensuring timely awareness even when no one is present at the location. The system's design methodology emphasizes quick detection, immediate action, and remote alerting through affordable, accessible components, making it a practical and scalable solution for homes, offices, and industrial setups. Compared to related projects, this system not only alerts users through IoT connectivity but also incorporates an automated extinguishing response, adding a critical layer of initiative-taking fire safety. Through this project, we demonstrate how integrating IoT-based communication with simple, cost-effective hardware can result in a highly efficient, user-friendly, and reliable fire safety system.

8. References

[1] 1NextPCB, “IoT Based Fire Alarm System using NodeMCU ESP8266,” Hackster.io, 2020. [Online]. Available: https://www.hackster.io/1NextPCB/iot-based-fire-alarm-system-using-nodemcu-esp8266-226aa3

[2] JustDoElectronics, “Fire Alert System using Arduino and GSM Module,” JustDoElectronics, 2021. [Online]. Available: https://justdoelectronics.com/fire-alert-system-using-arduino-gsm/

[3] Instructables, “Arduino Fire Fighting Robot With Auto Extinguisher,” Instructables.com, 2020. [Online]. Available: https://www.instructables.com/Arduino-Fire-Fighting-Robot-With-Auto-Extinguisher/

[4] SunFounder, “IoT Flame Alert System,” Ultimate Sensor Kit Docs, 2022. [Online]. Available: https://docs.sunfounder.com/projects/ultimate-sensor-kit/en/latest/iot_project/01-iot_Flame_alert_system.html

[5] M. K. Jadhav et al., “Design and implementation of a smart fire detection and alert system,” Scientific Reports (Nature), vol. 11, no. 1, 2021. [Online]. Available: https://www.nature.com/articles/s41598-021-03882-9

[6] How2Electronics, “Fire Detection System using Flame Sensor and Arduino,” 2019. [Online]. Available: https://how2electronics.com/fire-detection-system-using-flame-sensor-arduino/

[7] YouTube, “Fire Alert Notification Using ESP8266 and Flame Sensor,” YouTube, 2021. [Online]. Available: https://www.youtube.com/watch?v=BSzSUzE_iU8