Project Topic: Intelligent Fire and Smoke Detection using Arduino

Shreya Gupta(23BAI0168)
Ayushi Narolia(23BAI0191)
Ankit Kumar(23BCB0131)
Madhav Mendiratta(23BCB0092)

Introduction and Problem Statement

Fires are among the most dangerous hazards, often causing devastating loss of life and property. Traditional fire detection systems, while useful, often lack the responsiveness and intelligence required for modern safety needs.
 This project leverages **Arduino-based sensors and AI technology** to detect smoke and flames in real-time, providing instant alerts and enhancing fire safety measures.

Problem Statement:

- ⚠️ **Delayed fire detection** leads to significant damage and risk to human life.
- ⚠️ **Conventional systems** often lack **portability, intelligence, and real-time alerts**.
- ⚠️ **Limited integration** with smart safety systems restricts proactive prevention.

Why is Fire and Smoke Detection Important?

- Fires can spread rapidly, causing devastating damage within minutes.
- Early detection helps in preventing loss of life and property.
- Essential for residential, commercial, and industrial safety.

Real-Life Scenarios Where This System Can Be Useful

- **Homes & Apartments:** Detects fire in kitchens, electrical short circuits, or unattended candles.
- **Offices & Warehouses:** Prevents fire outbreaks due to overheating machines or flammable materials.
- **Factories & Industries:** Ensures safety in areas where fire hazards are common.

Objective of Our Project

To **design and implement an AI-enhanced Smoke & Flame Detection System** using Arduino that can:

- **Detect smoke and flames** in real-time using MQ-2 and Flame sensors.
- **Alert users immediately** through a buzzer and LCD display.
- **Enhance safety** by providing early fire hazard warnings.
- **Integrate AI for anomaly detection**, improving accuracy and responsiveness.
- Lay the foundation for **smart, automated fire prevention systems** in residential and industrial environments.

Components Used

- ❑ **Arduino UNO** – Microcontroller to process sensor data
- ❑ **Flame Sensor Module** – Detects fire and flame presence
- ❑ **Smoke Sensor Module (MQ-2 or MQ-135)** – Detects smoke and harmful gases
- ❑ **16x2 LCD Display** – Displays alerts and messages
- ❑ **Buzzer** – Generates an alarm sound when fire/smoke is detected
- ❑ **Mini Breadboard & Jumper Wires** – For circuit connections

Arduino UNO

- The **Arduino UNO** is the brain of our project.
- It is an **open-source microcontroller board** based on the ATmega328P.
- It processes input from the **flame and smoke sensors** and triggers an alert when necessary.
- Features:
    1. 14 digital I/O pins, 6 analog inputs
    2. USB interface for programming
    3. Supports external power supply

Flame Sensor Module

- The **Flame Sensor** detects the presence of fire or high heat sources by sensing infrared (IR) light emitted by flames.
- It works by detecting IR radiation in the **760-1100nm** wavelength range.
- Outputs **digital (HIGH/LOW)** or **analog signals** depending on flame intensity.
- Sensitivity can be adjusted using the onboard potentiometer.

Smoke Sensor Module

- The **MQ-2 or MQ-135 Smoke Sensor** detects harmful gases and smoke in the air.
- Detects **LPG, methane, carbon monoxide (CO), and other combustible gases**.
- Outputs an **analog signal** based on gas concentration.
- Sensitivity can be adjusted using the onboard potentiometer.

16x2 LCD Display

- The **16x2 LCD Display** is used to show system status and warnings.
- Displays messages like **"Fire Detected!"** or **"Smoke Detected!"**.
- Uses the **I2C module** or parallel pins for communication with Arduino.
- Can display **16 characters per row, 2 rows in total**.

Buzzer

- The **buzzer** is used to generate an **alarm sound** when fire or smoke is detected.
- Works as an **audio alert system** to warn users of potential hazards.
- Can be programmed to beep in different patterns for **different warning levels**.

Mini Breadboard

- ❑ A **breadboard** allows for easy, temporary circuit connections without soldering.
- ❑ Used to **connect sensors, LCD, and buzzer to the Arduino UNO**.
- ❑ Helps in rapid prototyping and circuit testing.

Jumper wires and Hookup Wires

- **Jumper Wires** (Male-to-Male, Male-to-Female) are used to connect components on the breadboard.
- **Hookup Wires** provide **stable electrical connections** for power and signal transmission.
- Essential for connecting **sensors, LCD, buzzer, and Arduino** in a structured manner.

Arduino Smoke & Flame Detection System Setup

1. Secure the Components:

- Place the **Arduino** and **mini breadboard** on a hard surface.
- Use **glue** to fix them securely in place.

2. Powering the Breadboard:

- Connect a **jumper wire**:
  - One end to the **5V pin** of the Arduino.
  - The other end to any point on the breadboard.
- Connect another **jumper wire**:
  - One end to the **GND pin** of the Arduino.
  - The other end to a different point on the breadboard.

Arduino Smoke & Flame Detection System Setup

3. Connecting the MQ-2 Smoke Sensor Module:

- Insert a **red wire** into the **VCC pin** of the MQ-2 sensor and connect it to the **breadboard's 5V connection**.
- Insert a **black wire** into the **GND pin** of the MQ-2 sensor and connect it to the **breadboard's GND connection**.
- Take another jumper wire and connect:
  - One end to the **A0 pin** of the MQ-2 sensor.
  - The other end to the **A0 pin** of the Arduino.

4. Connecting the Flame Sensor Module:

- Connect a **red wire** to the **VCC pin** of the Flame Sensor.
- Connect a **black wire** to the **GND pin** of the Flame Sensor.
- Insert the **red wire** into the **breadboard's 5V connection**.
- Insert the **black wire** into the **breadboard's GND connection**.
- Take a jumper wire and connect:
  - One end to the **D0 pin** of the Flame Sensor.
  - The other end to **Digital Pin 8** on the Arduino.

.

Arduino Smoke & Flame Detection System Setup

5. Connecting the I2C-Based 16x2 LCD Display:

- Connect the **GND pin** of the LCD to the **breadboard's GND connection**.
- Connect the **VCC pin** of the LCD to the **breadboard's 5V connection**.
- Connect the **SDA pin** of the LCD to **A4** on the Arduino.
- Connect the **SCL pin** of the LCD to **A5** on the Arduino.

6. Connecting the Buzzer:

- Attach **male jumper wires** to the buzzer.
- Connect the **buzzer's GND wire** to the **breadboard's GND connection**.
- Connect the **buzzer's VCC wire** to **Digital Pin 9** on the Arduino.

7. Securing the Setup:

- Use glue to secure all components on the flat surface.
- Use electrical tape to neatly bundle and secure the wires.

.

Arduino Smoke & Flame Detection System Setup

8. Connecting the Arduino to a PC:

1. **Open Arduino IDE:**
   - Download and install the **Arduino IDE**.
   - Open the IDE on your laptop or PC.
2. **Select the Board & Port:**
   - Go to **Tools** → **Board** and select **Arduino Uno (or your board model)**.
   - Go to **Tools** → **Port** and select the COM port where your Arduino is connected.
3. **Install Required Libraries:**
   - If you haven’t installed the **LiquidCrystal_I2C** library for the LCD, follow these steps:
     - Open **Arduino IDE** → **Sketch** → **Include Library** → **Manage Libraries**.
     - Search for **LiquidCrystal_I2C** and install it.
4. **Upload the Code:**
   - Copy and paste the following code into the **Arduino IDE**.
   - Click the **Upload (Arrow) Button** to send the code to the Arduino.

Arduino Smoke & Flame Detection System Setup

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C LCD address, 16 chars, 2 lines

#define SMOKE_SENSOR A0

#define FLAME_SENSOR 12

#define BUZZER 7

void setup() {

  pinMode(SMOKE_SENSOR, INPUT);

  pinMode(FLAME_SENSOR, INPUT);

  pinMode(BUZZER, OUTPUT);
```

Arduino Smoke & Flame Detection System Setup

```
Serial.begin(9600); // Send data to Python
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Flame & Smoke");
  lcd.setCursor(0, 1);
  lcd.print("Detector Ready");
  delay(2000);
  lcd.clear();
}
```

Arduino Smoke & Flame Detection System Setup

```
void loop() {
int smokeValue = analogRead(SMOKE_SENSOR);
int flameStatus = digitalRead(FLAME_SENSOR);
lcd.setCursor(0, 0);
lcd.print("Smoke: ");
lcd.print(smokeValue);
lcd.print("   ");
lcd.setCursor(0, 1);
lcd.print("                ");
lcd.setCursor(0, 1);
```

Arduino Smoke & Flame Detection System Setup

```
if (flameStatus == LOW) {
  lcd.print("Flame Detected!");
  digitalWrite(BUZZER, HIGH);
} else if (smokeValue > 80) {
  lcd.print("Smoke Alert!");
  digitalWrite(BUZZER, HIGH);
} else {
  lcd.print("Normal");
  digitalWrite(BUZZER, LOW);
}
```

Arduino Smoke & Flame Detection System Setup

```
 // Send data to Python
  Serial.print("SMOKE:");
  Serial.print(smokeValue);
  Serial.print(",FLAME:");
  Serial.println(flameStatus);

  delay(1000);
}
```

Python Code

```
import serial
import time
import pyttsx3
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from telegram import Bot
import asyncio
import threading
```

Python Code

```
# ==== CONFIGURATION ====

PORT = 'COM11'                # Change this to your Arduino port

BAUD_RATE = 9600

TELEGRAM_TOKEN = '7623805118:AAHTGbcVVTDO2aA9BTr4OULTnqPDPP5TaA0'

CHAT_ID = '1012611597'

# ==== INITIALIZATION ====

ser = serial.Serial(PORT, BAUD_RATE)

engine = pyttsx3.init()

bot = Bot(token=TELEGRAM_TOKEN)

smoke_history = []

x_data, y_smoke, y_flame = [], [], []
```

Python Code

```
# ==== GRAPH SETUP ====

fig, ax = plt.subplots()

line_smoke, = ax.plot([], [], label='Smoke')

line_flame, = ax.plot([], [], label='Flame')

ax.set_xlim(0, 100)

ax.set_ylim(0, 1024)

ax.set_title("Flame & Smoke Detector")

ax.set_xlabel("Time")

ax.set_ylabel("Sensor Value")

ax.legend()
```

Python Code

```
# ==== TELEGRAM EVENT LOOP IN BACKGROUND ====

loop = asyncio.new_event_loop()

threading.Thread(target=loop.run_forever, daemon=True).start()

async def async_telegram_send(message):

    try:

        await bot.send_message(chat_id=CHAT_ID, text=message)

        print(f"✅ Telegram alert sent: {message}")

    except Exception as e:

        print("❌ Telegram Error:", e)

def send_telegram_alert(message):

    asyncio.run_coroutine_threadsafe(async_telegram_send(message), loop)
```

Python Code

```
# ==== ALERT FUNCTION ====

def alert_user(message):

    print(message)

    engine.say(message)

    engine.runAndWait()

    send_telegram_alert(message)
```

Python Code

```
# ==== ANOMALY DETECTION ====

def detect_anomaly(value, history, threshold=3):

    history.append(value)

    if len(history) < 20:

        return False

    recent = np.array(history[-20:])

    mean = np.mean(recent)

    std = np.std(recent)

    return abs(value - mean) > threshold * std
```

Python Code

```
# ==== ANIMATION FUNCTION ====
def update(frame):
    global x_data, y_smoke, y_flame
    try:
        line = ser.readline().decode().strip()
        if "SMOKE" in line and "FLAME" in line:
            parts = line.split(',')
            smoke_value = int(parts[0].split(':')[1])
            flame_value = int(parts[1].split(':')[1])
            print(f"Smoke: {smoke_value}, Flame: {flame_value}")
```

Python Code

```
# Update graph data
    x_data.append(len(x_data))
    y_smoke.append(smoke_value)
    y_flame.append(1023 if flame_value == 0 else 0)
    if len(x_data) > 100:
        x_data = x_data[-100:]
        y_smoke = y_smoke[-100:]
        y_flame = y_flame[-100:]
    line_smoke.set_data(x_data, y_smoke)
    line_flame.set_data(x_data, y_flame)
    ax.set_xlim(max(0, len(x_data) - 100), len(x_data))
```

Python Code

```
# Alert Conditions
            if flame_value == 0:
                alert_user("🔥 Flame detected!")
            elif smoke_value > 80 or detect_anomaly(smoke_value, smoke_history):
                alert_user("💨 Smoke alert or anomaly detected!")
        else:
            print("Invalid data:", line)
    except Exception as e:
        print("Error:", e)
    return line_smoke, line_flame
```

Python Code

```
# ==== START ANIMATION ====

ani = animation.FuncAnimation(fig, update, interval=1000)

plt.tight_layout()

plt.show()
```

Arduino Smoke & Flame Detection System Setup

9. Connecting the Battery to the Arduino

- Take an **HW Battery (9V or 12V)**.
- Connect the **HW Battery Clip** to the battery terminals (**Red to +, Black to -**).
- Attach the **DC Jack Pin Connector** to the battery clip.
- Insert the **DC Jack Pin** into the **Arduino's DC power port**.

10. Powering On the Project

- Once the **DC jack is connected**, the **Arduino will turn on** and start running the program.
- The **LCD display** should show `"System Ready..."`.
- Sensors will start monitoring for **smoke and flames**.
- If **smoke or fire is detected**, the **buzzer will activate** and warnings will appear on the LCD.

○

Arduino Smoke & Flame Detection System Setup

11. Setting Up AI Anomaly Detection on PC

1. Install Python and Required Libraries
2. Load the Pretrained Isolation Forest Model
3. Start Reading Sensor Data from Arduino
4. Preprocess the Incoming Sensor Data
5. Run Anomaly Detection
6. Trigger AI Alerts on Anomaly Detection
7. Send Real-Time Telegram Alerts
8. Visualize Sensor Data with Live Graphs
9. Shut Down the AI Monitoring Script Safely

Conclusion

- Successfully designed a **Flame and Smoke Detection System** using **Arduino Uno** with real-time alerts.
- Integrated **AI-based anomaly detection** using **Isolation Forest** for intelligent monitoring.
- Implemented **voice alerts**, **Telegram notifications**, and **live sensor visualization** for enhanced responsiveness.
- System offers **early warning**, **increased safety**, and can be extended for **smart buildings**, **hostels**, and **industrial environments**.
- A step toward combining **embedded systems** with **AI for real-world safety applications**.