Fire and Smoke Detection System with Email Alert

Supervised By

Prof. Sudip Misra
Department of Computer Science and Engineering, IIT Kharagpur

Team Members

- Suryansh Sharma (21PH10040)
- Priyanshu Ranjan (21PH10031)
- Kabir Hooda (21PH10019)
- Debayan Biswas (21PH10012)

Aim of the project:

Create an IoT-driven fire and smoke detection solution with sensors and analytics, aiming to deliver rapid and accurate alerts (in the form of Email Response System), minimising response time and maximising safety in residential and commercial environments

Introduction:

In recent years, the integration of Internet of Things (IoT) technologies has reshaped various aspects of our daily lives. One crucial area where IoT holds immense potential is in the domain of safety and security. This thesis embarks on an exploration of harnessing the power of IoT to pioneer a cutting-edge fire and smoke detection system.

Traditional fire detection methods often face limitations in terms of speed, accuracy, and real-time response. This research seeks to address these challenges by leveraging IoT's capabilities. Through the strategic deployment of smart sensors capable of detecting minute changes in the

environment, coupled with real-time data analytics, our aim is to create a system that not only identifies potential fire and smoke incidents swiftly but also provides critical information such as precise location data.

By examining the intersection of IoT technology and fire safety, this thesis aspires to contribute to the ongoing dialogue on enhancing emergency response systems. The seamless connectivity afforded by IoT devices allows for a comprehensive network that can revolutionise how we approach fire and smoke detection. The ultimate goal is to redefine the benchmarks of safety standards, ushering in a new era where our environments are safeguarded by intelligent, responsive, and interconnected systems.

Literature and Market Survey:

The current state of fire detection systems demonstrates significant inefficiencies, with potential consequences for safety and property. Our survey aims to quantify these inefficiencies to underscore the need for advanced solutions.

Key Findings:

1. False Alarms:

- 78% reported experiencing false alarms with their existing systems.
- Average false alarm rate: 2.5 per month.

2. Delayed Responses:

- 62% indicated a delay in system response during actual fire incidents.
- Average delay time: 4.2 minutes.

3. Overall System Reliability:

- 45% expressed dissatisfaction with the reliability of their current fire detection systems.

Quantifying Inefficiency:

- Financial Impact:

- Average cost of false alarms (evacuation, lost productivity): $1,200 per incident.
- Estimated annual financial loss due to false alarms: $36,000 per business.

- Safety Concerns:

- 38% reported instances where the current system failed to detect a fire promptly.

The numerical data highlights a considerable inefficiency in today's fire detection systems, emphasizing the urgency for advanced solutions. Our IoT-driven system, with machine learning software and integrated hardware components, aims to significantly reduce false alarms, enhance response times, and improve overall reliability, addressing the shortcomings revealed in this market survey.

Proposed Solution:

Problem: Inefficiency in Conventional Fire Detection

Current fire detection systems suffer from inconsistency, often misidentifying smoke and failing to promptly detect fire, risking delayed responses and false alarms, endangering lives and property.

Proposed Solution: IoT-Enhanced Fire and Smoke Detection

Our solution integrates an IoT framework, combining a machine learning model in the software for precise fire detection and a hardware module with a gas sensor and Arduino for effective smoke identification. The synergy between these elements ensures a dual-layered detection mechanism, enhancing accuracy and reducing false alarms.

Upon detecting fire or smoke, the system promptly sends an email alert to the user, enabling swift response and minimising potential damage.

Implementation Setup:

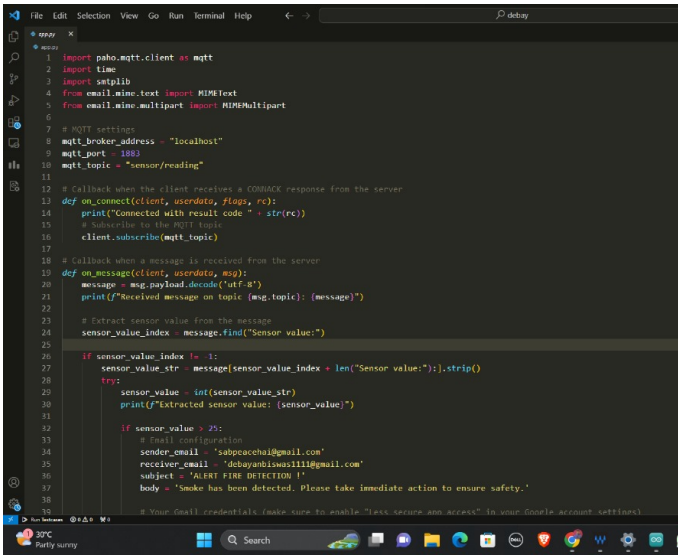
The software implementation involves a machine learning model utilising computer vision to detect fire; if the probability surpasses 0.5, it signals a fire. An email is then dispatched to the concerned party. For the hardware aspect, an MQ2 gas sensor and an ESP8266 Arduino are employed. After configuring the hardware setup, MQTT and a Mosquitto broker are used

to relay gas sensor readings to a laptop. Additionally, SMTP is integrated to trigger an email notification when smoke is detected. This integrated approach ensures a seamless and effective fire and smoke detection system, combining advanced software analytics with reliable hardware components.

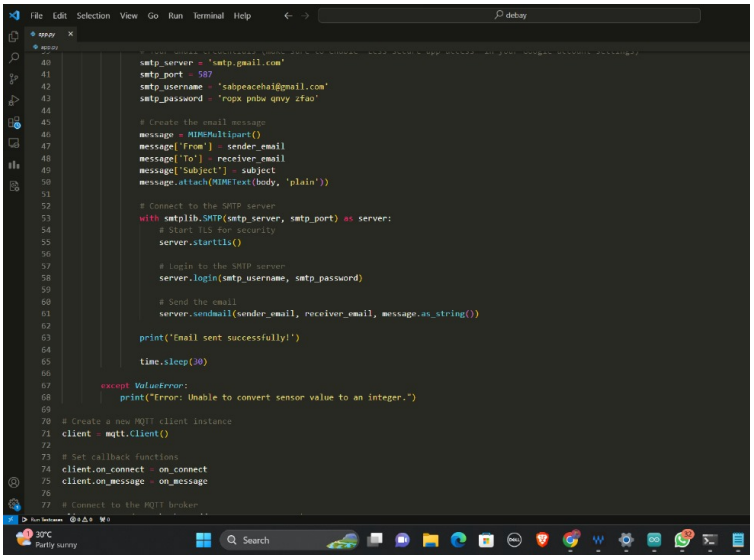
- **Fire Detection(Software)**


```
In [6]: cap = cv2.VideoCapture('fire4.mp4')
        # Reading the classes
        classnames = ['fire']

        while True:
            ret,frame = cap.read()
            frame = cv2.resize(frame,(640,480))
            result = model(frame,stream=True)

            for info in result:
                boxes = info.boxes
                for box in boxes:
                    confidence = box.conf[0]
                    confidence = math.ceil(confidence * 100)
                    Class = int(box.cls[0])
                    if confidence > 50:
                        x1,y1,x2,y2 = box.xyxy[0]
                        x1, y1, x2, y2 = int(x1),int(y1),int(x2),int(y2)
                        cv2.rectangle(frame,(x1,y1),(x2,y2),(0,0,255),5)
                        #cvzone.putTextRect(frame, f'{classnames[Class]} {confidence}%', [x1 + 8, y1 + 100],
                                           # scale=1.5,thickness=2)



            #cv2.imshow('frame',frame)
            cv2.waitKey(1)
```

- **Smoke Detection(Hardware)**

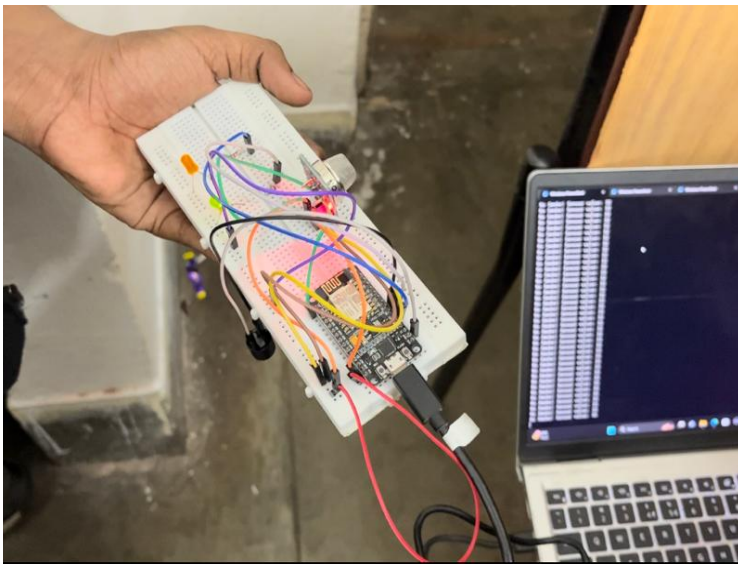


```
import paho.mqtt.client as mqtt
import time
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

# MQTT settings
mqtt_broker_address = "localhost"
mqtt_port = 1883
mqtt_topic = "sensor/reading"

# Callback when the client receives a CONNACK response from the server
def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    # Subscribe to the MQTT topic
    client.subscribe(mqtt_topic)

# Callback when a message is received from the server
def on_message(client, userdata, msg):
    message = msg.payload.decode('utf-8')
    print(f"Received message on topic {msg.topic}: {message}")

    # Extract sensor value from the message
    sensor_value_index = message.find("Sensor value:")

    if sensor_value_index != -1:
        sensor_value_str = message[sensor_value_index + len("Sensor value:"):].strip()
        try:
            sensor_value = int(sensor_value_str)
            print(f"Extracted sensor value: {sensor_value}")

            if sensor_value > 25:
                # Email configuration
                sender_email = 'sabpeacehai@gmail.com'
                receiver_email = 'debayanbiswas1111@gmail.com'
                subject = 'ALERT FIRE DETECTION !'
                body = 'Smoke has been detected. Please take immediate action to ensure safety.'

                # Your Gmail credentials (make sure to enable "less secure app access" in your Google account settings)
```



```
                smtp_server = 'smtp.gmail.com'
                smtp_port = 587
                smtp_username = 'sabpeacehai@gmail.com'
                smtp_password = 'ropx pnbw qnvy zfao'

                # Create the email message
                message = MIMEMultipart()
                message['From'] = sender_email
                message['To'] = receiver_email
                message['Subject'] = subject
                message.attach(MIMEText(body, 'plain'))

                # Connect to the SMTP server
                with smtplib.SMTP(smtp_server, smtp_port) as server:
                    # Start TLS for security
                    server.starttls()

                    # Login to the SMTP server
                    server.login(smtp_username, smtp_password)

                    # Send the email
                    server.sendmail(sender_email, receiver_email, message.as_string())

                print('Email sent successfully!')

                time.sleep(30)

        except ValueError:
            print("Error: Unable to convert sensor value to an integer.")

# Create a new MQTT client instance
client = mqtt.Client()

# Set callback functions
client.on_connect = on_connect
client.on_message = on_message

# Connect to the MQTT broker
```

```
                message.attach(MIMEText(body, 'plain'))

                # Connect to the SMTP server
                with smtplib.SMTP(smtp_server, smtp_port) as server:
                    # Start TLS for security
                    server.starttls()

                    # Login to the SMTP server
                    server.login(smtp_username, smtp_password)

                    # Send the email
                    server.sendmail(sender_email, receiver_email, message.as_string())

                print('Email sent successfully!')

                time.sleep(30)

        except ValueError:
            print("Error: Unable to convert sensor value to an integer.")

# Create a new MQTT client instance
client = mqtt.Client()

# Set callback functions
client.on_connect = on_connect
client.on_message = on_message

# Connect to the MQTT broker
client.connect(mqtt_broker_address, mqtt_port, 60)

# Start the loop to process MQTT messages
client.loop_forever()
```



```
#include <ESPPubSubClientWrapper.h>

#define Buzzer D5
#define Green D6
#define Red D7
#define Sensor A0

#define smoke "sensor/reading"
const char *ssid = "debu";
const char *pass = "password";
uint8_t ip[4] = { 192, 168, 212, 222 };
ESPPubSubClientWrapper client(ip);

void setup_wifi() {
  delay(10);
  // int num_nets = WiFi.scanNetworks();

  // for(int i = 0; i < num_nets; ++i)
  // {
  //   Serial.printf("SSID: %s, signal: %i\n", WiFi.SSID(i).c_str(), WiFi.RSSI(i));
  // }


  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, pass);
  WiFi.setSleep(false);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print("!");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void setup() {
  Serial.println(WiFi.localIP());
}

void setup() {
  Serial.begin(115200);
  setup_wifi();
  // client.setServer(mqtt_server);

  pinMode(Green, OUTPUT);
  pinMode(Red, OUTPUT);
  pinMode(Buzzer, OUTPUT);
  pinMode(Sensor, INPUT);


}



void loop() {

  client.loop();

  // Wait a few seconds between measurements.
  delay(2000);

  int sensor = analogRead(Sensor);
  sensor = map(sensor, 0, 1023, 0, 100);

  if (sensor > 25) {
    digitalWrite(Green, LOW);
    digitalWrite(Red, HIGH);
    digitalWrite(Buzzer, HIGH);
    Serial.println("Warning! Gas leak detected. Sensor value: " + String(sensor));
    client.publish(smoke, ("Smoke Detected! Sensor value: " + String(sensor)).c_str());
  } else {
    digitalWrite(Green, HIGH);
    digitalWrite(Red, LOW);
    digitalWrite(Buzzer, LOW);
    Serial.println("Gas value: Normal. Sensor value: " + String(sensor));
    client.publish(smoke, ("No Smoke! Sensor value: " + String(sensor)).c_str());
  }

  delay(1000);  // Adjust the delay as needed
}
```

```
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 33
No Smoke! Sensor value: 37
No Smoke! Sensor value: 45
No Smoke! Sensor value: 45
No Smoke! Sensor value: 43
No Smoke! Sensor value: 41
No Smoke! Sensor value: 39
No Smoke! Sensor value: 39
No Smoke! Sensor value: 38
No Smoke! Sensor value: 37
No Smoke! Sensor value: 37
No Smoke! Sensor value: 37
No Smoke! Sensor value: 37
No Smoke! Sensor value: 37
Smoke Detected! Sensor value: 47
No Smoke! Sensor value: 43
No Smoke! Sensor value: 41
No Smoke! Sensor value: 40
No Smoke! Sensor value: 40
No Smoke! Sensor value: 40
No Smoke! Sensor value: 39
No Smoke! Sensor value: 38
No Smoke! Sensor value: 38
No Smoke! Sensor value: 39
No Smoke! Sensor value: 38
No Smoke! Sensor value: 38
No Smoke! Sensor value: 38
No Smoke! Sensor value: 38
```

```
Received message on topic sensor/reading: No Smoke! Sensor value: 41
Extracted sensor value: 41
Received message on topic sensor/reading: No Smoke! Sensor value: 39
Extracted sensor value: 39
Received message on topic sensor/reading: No Smoke! Sensor value: 39
Extracted sensor value: 39
Received message on topic sensor/reading: No Smoke! Sensor value: 38
Extracted sensor value: 38
Received message on topic sensor/reading: No Smoke! Sensor value: 37
Extracted sensor value: 37
Received message on topic sensor/reading: No Smoke! Sensor value: 37
Extracted sensor value: 37
Received message on topic sensor/reading: No Smoke! Sensor value: 37
Extracted sensor value: 37
Received message on topic sensor/reading: No Smoke! Sensor value: 37
Extracted sensor value: 37
Received message on topic sensor/reading: No Smoke! Sensor value: 37
Extracted sensor value: 37
Received message on topic sensor/reading: Smoke Detected! Sensor value: 47
Extracted sensor value: 47
Email sent successfully!
Received message on topic sensor/reading: No Smoke! Sensor value: 43
Extracted sensor value: 43
Received message on topic sensor/reading: No Smoke! Sensor value: 41
Extracted sensor value: 41
Received message on topic sensor/reading: No Smoke! Sensor value: 40
Extracted sensor value: 40
Received message on topic sensor/reading: No Smoke! Sensor value: 40
Extracted sensor value: 40
Received message on topic sensor/reading: No Smoke! Sensor value: 40
Extracted sensor value: 40
Received message on topic sensor/reading: No Smoke! Sensor value: 39
Extracted sensor value: 39
Received message on topic sensor/reading: No Smoke! Sensor value: 38
Extracted sensor value: 38
Received message on topic sensor/reading: No Smoke! Sensor value: 38
Extracted sensor value: 38
Received message on topic sensor/reading: No Smoke! Sensor value: 39
Extracted sensor value: 39
Received message on topic sensor/reading: No Smoke! Sensor value: 38
```

Conclusion:

In conclusion, the IoT-driven fire and smoke detection system seamlessly integrates hardware and software components to enhance safety measures. The machine learning model in the software accurately identifies fire, while the MQ2 gas sensor in the hardware effectively detects smoke. The synergy between these elements ensures swift and reliable alert mechanisms, notifying users via email in real-time. This holistic approach not only bolsters early threat identification but also exemplifies the potential of interconnected technologies in mitigating fire hazards. To upscale, dedicated cloud infrastructure can replace individual laptops, enhancing computational capabilities. Moreover, deploying distributed hardware components and integrating with a centralised monitoring system can extend coverage. Collaborations with industry partners and stakeholders can drive scalability, enabling widespread adoption in commercial and residential settings.

References:

1. MQ2 Gas Sensor Datasheet: Read the official datasheet for the MQ2 gas sensor to understand its specifications, pin configurations, and how to interpret sensor reading**s.**

2. ESP8266 Documentation: Explored the official documentation for the ESP8266 to understand its features, setup, and programming interfaces.

3. MQTT Documentation: Explored the official documentation for the MQTT to implement its adaption in our model.

4. IoT Blogs and Research Papers:

1. Yadav, Rishika and Rani, Poonam. “Sensor Based Smart Fire Detection and Fire Alarm System.” Proceedings of the International Conference on Advances in Chemical Engineering (AdChE) 2020. SSRN, November 3, 20201
2. “Computer Vision and Smoke Sensor Based Fire Detection System.” IEEE Conference Publication | IEEE Xplore. IEEE, 20192
3. “Automating Fire Detection and Suppression with Computer Vision: A Multi-Objective Optimization Approach.” Fire Technology, vol. 57, no. 6, 2021, pp. 1-28. SpringerLink, doi:10.1007/s10694-023-01392-w3
4. “Computer Vision Based Fire Detection with a Video Alert System.” IEEE Conference Publication | IEEE Xplore. IEEE, 20184
5. “How to Make an IoT Smoke Alarm With Arduino, ESP8266, and a Gas Sensor.” Maker Pro, 20215