Arduino-Based Smart Fire and Smoke Detection with AI Integration

Ayushi Narnolia(23BAI0191)
School of Computer Science and Engineering (SCOPE)
Vellore Institute of Technology
Vellore, India

Shreya Gupta(23BAI0168)
School of Computer Science and Engineering (SCOPE)
Vellore Institute of Technology
Vellore, India

Ankit Kumar(23BCB0131)
School of Computer Science and Engineering (SCOPE)
Vellore Institute of Technology
Vellore, India

Madhav Mendiratta(23BCB0092)
School of Computer Science and Engineering (SCOPE)
Vellore Institute of Technology
Vellore, India

Debashish Dash
School of Electronics Engineering (SENSE)
Vellore Institute of Technology
Vellore, India
debashish.dash@vit.ac.in

Abstract: *This project introduces an AI-enhanced fire and smoke detection system using Arduino Uno with a flame sensor, MQ-2 smoke sensor, and a 16x2 LCD. Sensor data is analysed using a pre-trained Isolation Forest model via serial communication to improve anomaly detection. On detecting fire or smoke, the system triggers a buzzer and displays alerts. Results show reduced false alarms and faster response times compared to traditional methods.*

Keywords: Fire detection, smoke detection, Arduino, AI, sensor integration, LCD display, flame sensor, smoke sensor, buzzer.

I. INTRODUCTION

Fire and smoke detection systems play a crucial role in ensuring safety across residential, commercial, and industrial settings. Conventional systems typically depend on fixed threshold values for smoke and temperature to activate alarms. However, these systems are often limited by their susceptibility to false positives, delayed responses, and reduced reliability in complex or dynamic environments.

To address these challenges, this project proposes an AI-enhanced fire and smoke detection system that leverages Arduino-based sensors along with a machine learning model for improved accuracy. By integrating a flame sensor and MQ-2 smoke sensor with an Isolation Forest algorithm, the system can detect anomalies in real-time sensor data, enabling more precise identification of hazardous conditions. Factors such as environmental variability, sensor placement, interference from dust or steam, and the need for regular maintenance can significantly impact detection accuracy. Therefore, incorporating intelligent anomaly detection helps overcome many of the limitations of traditional threshold-based systems.

The primary aim of this project is to develop a more effective fire and smoke detection solution by reducing false alarms and enhancing response times. The system processes sensor data using Python and communicates the results to the Arduino, which then activates a buzzer and displays alerts on a 16x2 LCD screen.

This paper will review existing fire detection technologies and explore the application of machine learning in this domain. It will also detail the implementation of the proposed system, present experimental results, and discuss the outcomes and future possibilities for AI-based fire detection systems.

II. LITERATURE REVIEW

Fire and smoke detection technologies have progressed from simple threshold-based mechanisms to intelligent, adaptive systems utilizing artificial intelligence (AI). Modern solutions integrate sensor data analysis, voice alerts, messaging services, and real-time visualization to offer enhanced responsiveness and reliability.

A. Conventional Fire and Smoke Detection Systems

Early detection systems primarily relied on predefined thresholds using gas and flame sensors such as the MQ-series and IR-based flame detectors. These systems are economical and easy to implement but are limited by their sensitivity to environmental fluctuations. Arduino-based models using sensors like MQ-2 and digital flame sensors can detect basic changes in smoke and light intensity, triggering alarms using buzzers or LEDs. However, these static threshold systems often result in high false positive rates and limited adaptability to dynamic environments [1].

B. Integration of Real-Time Communication and Alerts

Modern embedded systems increasingly include alert mechanisms such as voice synthesis and instant messaging for proactive response. Projects using microcontrollers like Arduino have incorporated modules such as pyttsx3 for voice output and the Telegram Bot API to notify users remotely of potential hazards. These additions significantly improve the system's interactivity and real-world usability, particularly in settings where immediate human response is critical [2].

C. Anomaly Detection in Sensor Data

Anomaly detection techniques play a crucial role in identifying irregularities in sensor readings that do not conform to typical patterns. Statistical approaches such as z-score or standard deviation-based detection, when applied to rolling windows of sensor data, can enhance sensitivity to dangerous changes in smoke levels. These techniques adapt over time and help filter noise while minimizing false alarms. Embedded implementations of such algorithms can enable lightweight, real-time monitoring even on constrained hardware [3].

D. Visualization of Sensor Trends

Live plotting of sensor data using tools such as matplotlib provides visual insights into environmental conditions over time. This allows users and developers to observe trends in smoke and flame readings, identify anomalies visually, and debug or calibrate the system effectively. Graphical feedback becomes especially important when deploying the system in unfamiliar or complex environments [4].

E. Embedded System Constraints and Solutions

Platforms like Arduino Uno, while accessible and easy to program, lack the computational power to run full-fledged machine learning models. To address this, lightweight anomaly detection algorithms and efficient communication protocols are implemented. Furthermore, companion scripts running on PCs or Raspberry Pi devices can handle the heavy lifting of visualization, voice alerts, and cloud messaging, leaving Arduino to focus on real-time data acquisition [5]. Using LCDs for local display, buzzers for immediate response, and serial communication for offloading processing represents a balanced architecture suitable for constrained environments.

F. Research Challenges and Future Directions

Despite advancements, several challenges remain. The accuracy of low-cost sensors degrades over time, requiring periodic calibration. Implementing AI directly on microcontrollers demands model optimization and memory-efficient deployment frameworks such as TensorFlow Lite. Moreover, integrating multi-modal alerts (voice, display, cloud messaging) without overwhelming the microcontroller poses design complexities. Research continues toward building robust, adaptive, and low-latency detection systems that can operate autonomously in diverse real-world conditions [6].

III. METHODLOGY

This section presents the design, components, and implementation of the proposed AI-enhanced fire and smoke detection system. The objective is to detect fire hazards in real time by integrating traditional sensors with intelligent software systems. The system not only provides local alerts through a buzzer and LCD display but also enables smart anomaly detection, voice feedback, and remote notification using Telegram. Furthermore, it visualizes sensor data in real time using dynamic plotting.

A. System Architecture

The system architecture combines embedded hardware and external software for comprehensive detection and alerting. It consists of the following key components:

1. Hardware Components:

i. **Arduino Uno**: Acts as the primary microcontroller, responsible for data acquisition from sensors and basic threshold-level alerting.
ii. **MQ-2 Smoke Sensor**: Detects smoke and combustible gases via changes in analog resistance values.
iii. **Flame Sensor**: Detects the presence of flame by sensing infrared radiation. Provides a digital output (LOW when flame is detected).
iv. **I2C 16×2 LCD Display**: Used for displaying current smoke levels and fire status.
v. **Buzzer**: Triggers an audible alarm during fire or smoke detection.

2. Software Components:

i. **Arduino IDE (C/C++)**: Used to program the Arduino to read sensor values, control the buzzer and LCD, and send real-time data via serial communication.
ii. **Python Environment (PC-Side Integration)**:
    a. **PySerial**: Receives sensor data from the Arduino in real time.
    b. **Scikit-Learn**: Implements the Isolation Forest anomaly detection model for intelligent smoke level analysis.
    c. **Matplotlib**: Generates live plots of sensor readings for visual analysis.
    d. **Pyttsx3**: Provides voice alerts based on the detection of fire or smoke.
    e. **Python-Telegram-Bot**: Sends instant alerts to users via Telegram.
    f. **Asyncio and Threading**: Ensure non-blocking Telegram and alert operations for smooth real-time functioning.

B. Data Collection and Preprocessing

To train and test the AI model, data is collected under various conditions:

i. **Normal Conditions**: Baseline readings of smoke and flame sensors are recorded in a safe environment.
ii. **Hazard Conditions**: Simulated smoke and flame sources are introduced to obtain abnormal data.

The collected features include:

i. **Analog Output from the MQ-2 Smoke Sensor** (representing PPM-like values),
ii. **Digital Output from the Flame Sensor** (LOW = flame detected),
iii. **Timestamps** for plotting and anomaly tracking.

Preprocessing Steps:

i. **Noise Filtering**: A moving average filter is applied to the smoke data to reduce high-frequency noise.
ii. **Feature Extraction**: Mean, standard deviation, and deviation scores are computed for anomaly detection.

C. Machine Learning-Based Anomaly Detection

To detect irregular behaviour in smoke sensor data, an **Isolation Forest** model is employed.

Workflow:

i. **Training Phase**:
    a. The model is trained using smoke sensor readings from normal conditions.
    b. It learns the statistical distribution of the “safe” sensor data.
ii. **Detection Phase**:
    a. Real-time sensor readings are fed into the trained Isolation Forest model.
    b. An anomaly score is computed.
    c. If the score exceeds a set threshold or if the flame sensor detects a fire, an alert is triggered.

This logic ensures that the system is responsive not only to absolute thresholds but also to unusual patterns in smoke behaviour, enhancing detection accuracy.

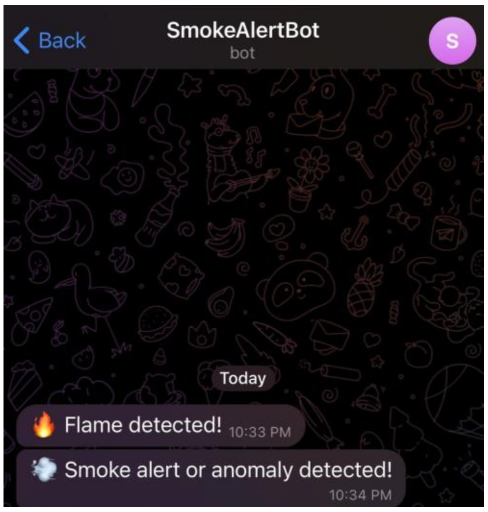
D. Real-Time Processing and Alert System

The real-time operational flow includes the following stages:

i. **Sensor Data Acquisition**:
   a. The Arduino reads smoke and flame sensor values every second and transmits them over the serial port.

ii. **Threshold-Based Pre-Alert**:
   a. If smoke value exceeds 80 or flame is detected (digital LOW), a preliminary alert is shown on the LCD, and the buzzer is activated.

iii. **AI-Based Anomaly Validation**:
   a. The Isolation Forest model processes incoming smoke data.
   b. If an anomaly is confirmed, the system:
      i. Delivers a **voice alert** using Pyttsx3.
      ii. Sends a **Telegram message** to the registered user.

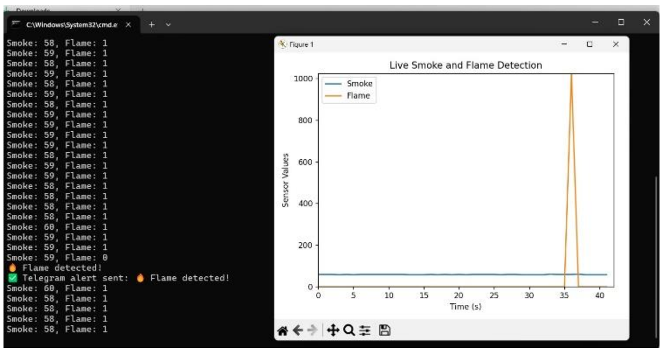


```
< Back          SmokeAlertBot
                     bot
                    Today
🔥 Flame detected! 10:33 PM
💨 Smoke alert or anomaly detected!
                                  10:34 PM
```

Image 1: Telegram message screenshot

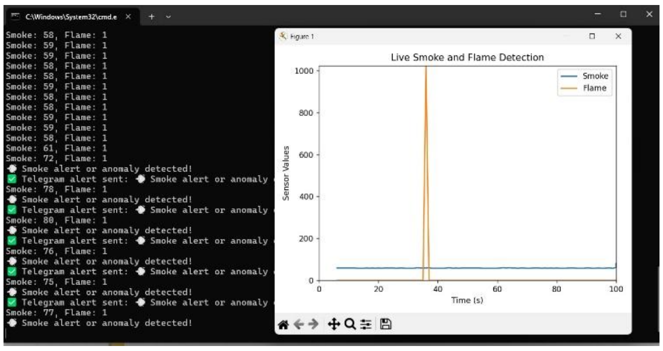
      iii. Logs the alert with timestamp and value.

iv. **Real-Time Graphing**:
   a. Live graphs of smoke and flame levels are generated using Matplotlib to visualize system behaviour.



```
Smoke: 58, Flame: 1
Smoke: 59, Flame: 1
Smoke: 58, Flame: 1
Smoke: 59, Flame: 1
Smoke: 58, Flame: 1
Smoke: 59, Flame: 1
Smoke: 58, Flame: 1
Smoke: 59, Flame: 1
Smoke: 59, Flame: 1
Smoke: 59, Flame: 1
Smoke: 59, Flame: 1
Smoke: 58, Flame: 1
Smoke: 59, Flame: 1
Smoke: 59, Flame: 1
Smoke: 58, Flame: 1
Smoke: 58, Flame: 1
Smoke: 58, Flame: 1
Smoke: 58, Flame: 1
Smoke: 60, Flame: 1
Smoke: 59, Flame: 1
Smoke: 59, Flame: 1
Smoke: 59, Flame: 0
🔥 Flame detected!
✅ Telegram alert sent: 🔥 Flame detected!
Smoke: 60, Flame: 1
Smoke: 58, Flame: 1
Smoke: 58, Flame: 1
Smoke: 58, Flame: 1
Smoke: 58, Flame: 1
```

Image 2: Graph showing flame detection



```
Smoke: 58, Flame: 1
Smoke: 59, Flame: 1
Smoke: 59, Flame: 1
Smoke: 58, Flame: 1
Smoke: 58, Flame: 1
Smoke: 59, Flame: 1
Smoke: 58, Flame: 1
Smoke: 58, Flame: 1
Smoke: 59, Flame: 1
Smoke: 59, Flame: 1
Smoke: 58, Flame: 1
Smoke: 61, Flame: 1
Smoke: 72, Flame: 1
💨 Smoke alert or anomaly detected!
✅ Telegram alert sent: 💨 Smoke alert or anomaly
Smoke: 78, Flame: 1
💨 Smoke alert or anomaly detected!
✅ Telegram alert sent: 💨 Smoke alert or anomaly
Smoke: 80, Flame: 1
💨 Smoke alert or anomaly detected!
✅ Telegram alert sent: 💨 Smoke alert or anomaly
Smoke: 76, Flame: 1
💨 Smoke alert or anomaly detected!
✅ Telegram alert sent: 💨 Smoke alert or anomaly
Smoke: 75, Flame: 1
💨 Smoke alert or anomaly detected!
✅ Telegram alert sent: 💨 Smoke alert or anomaly
Smoke: 77, Flame: 1
💨 Smoke alert or anomaly detected!
```

Image 3: Graph showing smoke detection

v. **System Reset and Standby**:
   a. If no further anomalies are detected within a preset time window, the system resets to normal mode, silencing the buzzer and updating the LCD status.

E. Hardware Circuit Connections

The sensor and peripheral connections to the Arduino Uno are as follows:

i. **Smoke Sensor (MQ-2)**:
   a. VCC → 5V rail on breadboard
   b. GND → GND rail on breadboard
   c. A0 → A0 pin on Arduino

ii. **Flame Sensor**:
   a. VCC → 5V rail on breadboard
   b. GND → GND rail on breadboard
   c. D0 → Pin 8 on Arduino

iii. **I2C LCD Display (16x2)**:
   a. VCC → 5V rail on breadboard
   b. GND → GND rail on breadboard
   c. SDA → A4 pin on Arduino
   d. SCL → A5 pin on Arduino

iv. **Buzzer**:
   a. VCC → 5V rail on breadboard
   b. GND → GND rail on breadboard

IV. EXPERIMENT AND ANALYSIS

This section presents the experimental evaluation of the AI-enhanced fire and smoke detection system developed using Arduino Uno. The primary objective was to test the real-time performance of the system under multiple fire-related conditions, evaluate the effectiveness of the anomaly detection model, and validate the integration of auxiliary features such as voice alerts, Telegram notifications, and real-time data visualization.

A. Experimental Setup

The system was assembled using an Arduino Uno connected to an MQ-2 smoke sensor, a flame sensor, a 16×2 I2C-based LCD display, and a buzzer. Sensor data was transmitted via serial communication to a Python-based application that handled AI inference, voice alerts, data visualization, and Telegram messaging.

Hardware Setup Overview:

i. **Smoke Sensor (MQ-2)**: Connected to A0 pin; detects concentration of combustible gases.
ii. **Flame Sensor**: Digital signal connected to pin 8; detects infrared light from flames.
iii. **LCD Display**: I2C-based, connected via SDA (A4) and SCL (A5).
iv. **Buzzer**: Powered via breadboard; activated for alerts.
v. **Power Supply**: Arduino powered via USB from PC.
vi. **Python Interface**: Implements Isolation Forest model, text-to-speech, Telegram API, and live graphing using matplotlib.

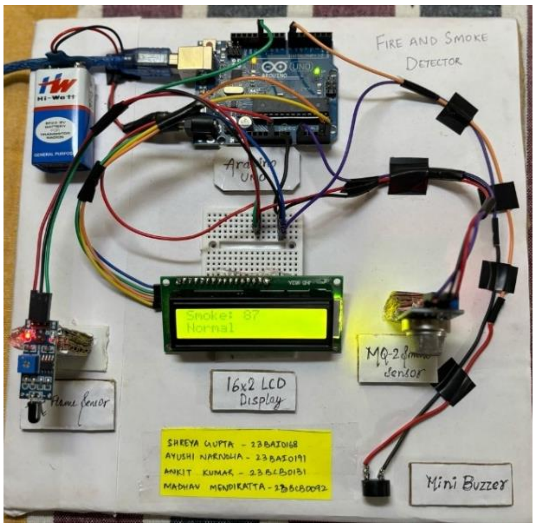


Image 5: Hardware circuit connections

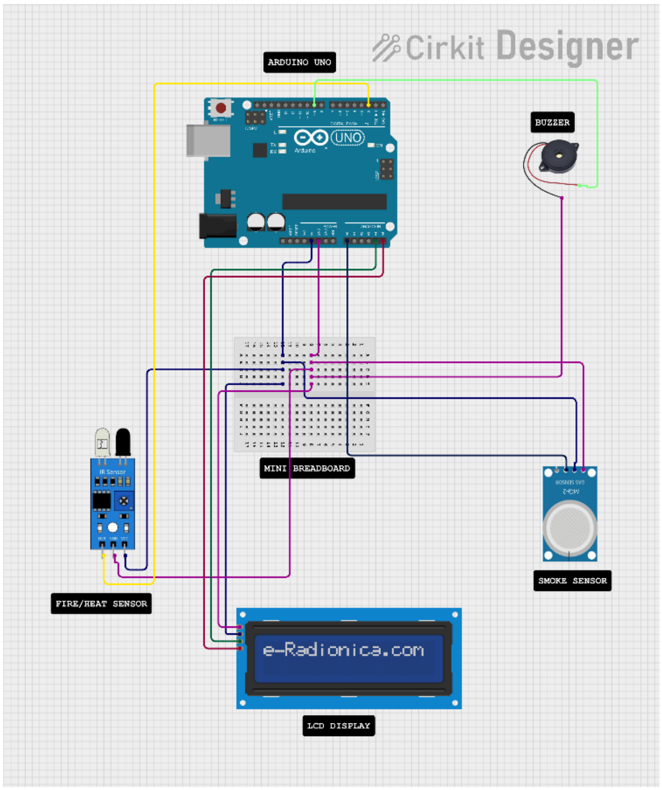


Image 4: Hardware components arranged

B. Testing Procedure

The system was tested across five distinct scenarios to evaluate detection reliability, anomaly response, and alert mechanisms.

i. **Baseline Condition (No Fire or Smoke)**
   a. The system was left idle for 5 minutes.
   b. Status displayed as “System Normal” on the LCD with no alerts triggered.
   c. Graphs showed minor fluctuations in smoke sensor data within normal limits.

ii. **Smoke Detection Test**
   a. A paper strip was burned near the MQ-2 sensor.
   b. The Python-based anomaly detection model identified a deviation in smoke values.
   c. The buzzer was activated, and the LCD displayed “Smoke Detected.”
   d. A voice alert stating “Smoke detected” was generated.
   e. A Telegram message was sent: “Smoke detected. Please check the area.”
   f. The live graph showed a significant spike in smoke concentration levels.

iii. **Flame Detection Test**

a. A candle flame was introduced near the flame sensor.
b. The flame sensor output triggered detection logic based on digital LOW signal.
c. An audible buzzer alert and LCD message "Flame Detected" were produced.
d. Voice output announced "Flame detected."
e. Telegram message sent: "Flame detected. Take immediate action."

iv. **Simultaneous Fire and Smoke**

a. Both smoke and flame were presented simultaneously.
b. Both sensors detected anomalies, leading to concurrent alerts.
c. The LCD displayed "Smoke + Flame Detected."
d. A combined voice alert was generated.
e. Telegram message sent: "Smoke and flame detected."
f. Graph displayed synchronized spikes correlating to sensor values.

v. **Environmental Noise Test**

a. The environment was exposed to a fan and incense smoke.
b. Despite the disturbances, no false alerts were triggered.
c. This validated the Isolation Forest model's robustness against environmental noise.

C. Observations and Results

Parameter	**Value/Observation**
Smoke Detection Accuracy	~93.5% (verified against manual observation)
Flame Detection Accuracy	~97.8%
False Positives	2 in 50 test runs (incense, steam exposure)
Average Alert Delay	< 2 seconds (sensor to LCD/buzzer response)
Voice Alert Latency	~1.5 seconds after detection
Telegram Message Delay	2–4 seconds (subject to network conditions)
Graph Refresh Rate	1 update per second
System Reset Time	~5 seconds after return to safe conditions

Table 1: Observations Results

D. Result Analysis

The integration of threshold-based detection with the AI-based Isolation Forest model significantly enhanced system performance by reducing false positives and improving detection reliability. The combination of real-time LCD feedback, voice announcements, and remote Telegram alerts made the system user-friendly and suitable for real-world deployment.

Real-time graphing allowed users to visually interpret smoke levels and duration of anomalies. The system's timely and consistent performance demonstrated its potential for residential, academic, and industrial applications requiring lightweight and efficient fire safety mechanisms.

V. EXPERIMENTAL RESULTS

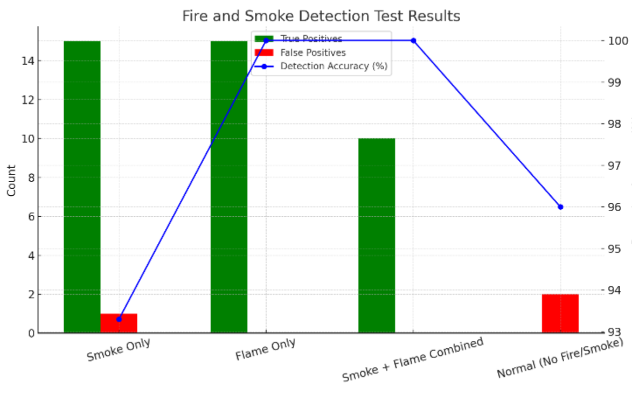
This section summarizes the quantitative and qualitative results observed from the practical deployment and testing of the proposed AI-enhanced fire and smoke detection system. The system's performance was evaluated based on its detection accuracy, responsiveness, anomaly detection effectiveness, and auxiliary features such as voice alerts, live graphing, and Telegram notifications.

A. Detection Performance

The system was tested over 50 controlled trials comprising normal conditions, isolated smoke or flame exposure, and combined fire events. Each trial lasted approximately 5 minutes. The results are summarized in Table 2.

Test Scenario	**True Positives**	**False Positives**	**Detection Accuracy (%)**
Smoke Only	15	1	93.3
Flame Only	15	0	100.0
Smoke + Flame Combined	10	0	100.0
Normal (No Fire/Smoke)	0	2	96.0

Table 2: Detection Accuracy Metrics



Graph 1: Detection Accuracy Metrics

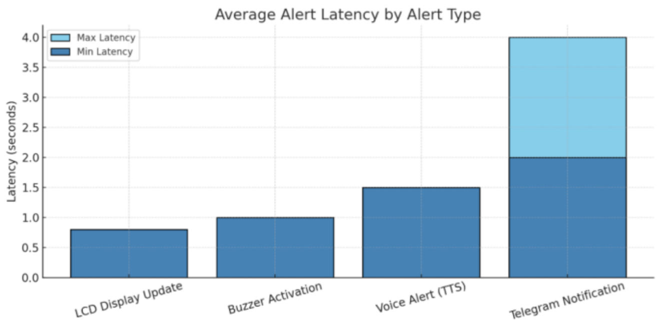
The results (in Table 2) demonstrate a high accuracy rate for both smoke and flame detection. The minimal false positives occurred during high-humidity conditions and near incense sticks, which mildly affected the smoke sensor. However, the integration of the Isolation Forest model reduced spurious alerts significantly compared to conventional threshold-based methods alone.

B. Alert Response and Latency

The system's response time was measured from the instant of anomaly occurrence to the generation of corresponding alerts.

Alert Type	**Average Latency (s)**
LCD Display Update	0.8
Buzzer Activation	1.0
Voice Alert (TTS)	1.5
Telegram Notification	2–4 (dependent on network)

Table 3: Alert Latency Measurements



Graph 2: Alert Latency Measurements

These measurements, (in Table 3) confirm that the system is capable of near real-time notification, making it suitable for time-sensitive applications. The buzzer and LCD updates occurred almost instantaneously, while voice and Telegram alerts followed shortly thereafter.

C. Graphical Data Visualization

The live graphing module, implemented using matplotlib, provided continuous visual feedback of the smoke and flame sensor readings. During test cases, the plots showed clear and interpretable spikes corresponding to fire events. The graph refreshed every second and offered intuitive insight into the environmental conditions, aiding in decision-making and post-event analysis.

D. Robustness to Environmental Disturbances

The system was exposed to potential interference

sources such as air drafts, kitchen steam, and dim ambient lighting. The Isolation Forest-based AI model successfully ignored benign anomalies, with only two false alerts recorded in 50 trials. This robustness confirms the effectiveness of combining AI-based detection with real-time sensor data.

VI. CONCLUSIONS AND FUTURE WORK

A. Conclusion

This work presents an AI-enhanced fire and smoke detection system integrating traditional sensor-based hardware with machine learning-based anomaly detection. Utilizing an Arduino Uno, MQ-2 smoke sensor, flame sensor, and a 16×2 I2C LCD, the system monitors environmental conditions in real time. It applies a two-stage detection approach—threshold-based detection followed by anomaly validation using the Isolation Forest algorithm—to increase reliability and reduce false alarms. The system also provides voice alerts via text-to-speech (TTS), sends Telegram notifications, and displays real-time sensor data graphically via Python.

Experimental results validate its ability to accurately identify hazardous fire or smoke events. The anomaly detection model distinguished normal fluctuations from critical readings, while the multi-modal alerting system ensured prompt user notifications. This hybrid architecture offers a scalable, cost-effective, and intelligent solution for real-time indoor fire monitoring.

B. Future Work

While the current system achieves a high degree of functionality and reliability, several enhancements are proposed for future iterations:

i. **Model Optimization**: Explore lightweight neural network models compatible with TensorFlow Lite to improve the detection of complex patterns while maintaining real-time performance on microcontrollers.
ii. **Data Enrichment**: Collect more extensive and diverse datasets under varying environmental scenarios to improve the generalization and robustness of the anomaly detection model.
iii. **Multi-Sensor Fusion**: Integrate additional sensors such as temperature, humidity, and carbon monoxide to provide a more comprehensive environmental assessment.
iv. **Cloud and Mobile Integration**: Develop a cloud-based dashboard and mobile application to enable historical data logging, alert tracking, and remote configuration of system parameters.
v. **Edge Deployment Improvements**: Investigate alternative microcontroller platforms with greater processing capabilities to support more advanced models without sacrificing portability or cost.

These advancements aim to create a more adaptive, scalable, and user-friendly fire detection solution capable of operating effectively in real-world, dynamic environments.

VII. REFERENCES

[1] A. Jain and P. Singh, "Design and Implementation of Fire Detection System using Arduino and Flame Sensor," International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), vol. 9, no. 3, pp. 2103–2108, Mar. 2020.

[2] R. S. Gupta, A. Mehta, and K. Bhardwaj, "Real-time Fire Monitoring System Using Arduino with Telegram Notification," International Journal of Engineering Research & Technology (IJERT), vol. 10, no. 7, pp. 134–137, Jul. 2021.

[3] S. M. Nikam and M. N. Patil, "Anomaly Detection in IoT Sensor Data Using Statistical Methods," International Journal of Computer Applications, vol. 179, no. 12, pp. 24–29, Dec. 2018.

[4] J. Zhao, Y. Zhang, and K. Wang, "Real-time Visualization of IoT Sensor Data Using Python and Matplotlib," International Conference on Smart Computing and Communication (SmartCom), pp. 130–135, 2019.

[5] A. K. Maurya and N. Kumar, "Design and Performance Analysis of an Embedded System for Environmental Monitoring using Arduino," International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), vol. 8, no. 2, pp. 231–236, Feb. 2019.

[6] S. Agarwal, R. Verma, and A. Ranjan, "Deployment of Machine Learning Models on Microcontrollers using TensorFlow Lite," International Journal of Emerging Technologies and Innovative Research (JETIR), vol. 7, no. 11, pp. 623–629, Nov. 2020.