https://doi.org/10.70844/aest.2025.1.1

Applied Engineering Solutions and Technologies

Research Article

AEST-25-002

ESP32-Based Intelligent Fire Detection System Utilizing Machine Learning and Computer Vision for Enhanced Safety in Indian Micro and Small Enterprises

Mehtab Khan*, Piyush Mishra, Rehan Khan and Vishal Maurya

Department Computer Engineering, St. John College of Engineering and Management, Palghar, India

Corresponding Author: Mehtab Khan, Department Computer Engineering, St. John College of Engineering and Management, Palghar, India, E-mail: mehtabasmoon@gmail.com

Received date: 03 June, 2025, **Accepted date:** 07 August, 2025, **Published date:** 21 August, 2025

Citation: Khan M, Mishra P, Khan R, Maurya V (2025) ESP32-Based Intelligent Fire Detection System Utilizing Machine Learning and Computer Vision for Enhanced Safety in Indian Micro and Small Enterprises. Appl Eng Sol Technol 1(1): 1.

Abstract

Protection against fire hazards remains a critical concern for small to medium businesses with limited financial resources. This research introduces an intelligent fire detection system leveraging machine learning and real-time image processing to address conventional fire detection limitations. The proposed system integrates an ESP32 microcontroller with an Arduino-based camera module, creating an affordable solution for detecting smoke, flames and heat through video stream analysis. By implementing advanced algorithms, the system provides efficient and timely fire alarms while minimizing false positive rates and enhancing operational safety. The modular design enables seamless integration with existing security infrastructures, requiring minimal reconfiguration. Extensive testing under diverse conditions demonstrates the model's adaptability, achieving high accuracy and reliability. Specifically developed to address fire safety gaps in the Indian industrial context, the solution targets micro and small enterprises, promoting technological advancement, industrial safety and infrastructure development at a cost-effective price point. The research contributes to bridging critical safety technology gaps for resource-constrained businesses.

Keywords: Computer vision, ESP32, Fire alarm system, Image processing, Industrial safety, Machine learning, Real-time system, Small business solution

Introduction

Fire hazards pose a persistent and critical challenge, particularly for Micro and Small Enterprises (MSEs) in resource-constrained settings. Such businesses are often unequipped to invest in advanced safety systems, leaving them vulnerable to severe financial and human losses in the event of fire outbreaks. According to the International Labor Organization, the lack of efficient fire safety measures disproportionately affects small enterprises, underscoring the urgent need for affordable and reliable solutions tailored to their operational contexts.

In response to these challenges, our research aims to bridge the existing gaps in fire safety technology by proposing a scalable, intelligent fire detection system built around the versatile ESP32 microcontroller. By integrating multiple sensors—including temperature, humidity, smoke and flame detection modules, our system enhances detection accuracy while maintaining affordability. The project leverages machine learning algorithms to analyze sensor data and video streams, ensuring proactive hazard detection and significantly reducing false alarm rates.

The increasing frequency of fire incidents in industrial areas and the inability of conventional detection systems to provide timely alerts served as key motivators for this work. Traditional fire detection systems rely heavily on either visual inspection or rudimentary alarm mechanisms, which often fail to deliver actionable data or real-time situational awareness. As noted in “Hawk-Eye: An AI-Powered Threat Detector” by Ahmed Abdel Moamen and Mathias Ecchi, the integration of AI with existing surveillance systems can drastically improve safety outcomes by reducing the time required for anomaly detection and alert generation [1].

Building on this premise, our project adopts a holistic approach, combining computer vision, environmental monitoring and cloud-based analytics to provide small enterprises with an edge in emergency preparedness. Unlike high-end commercial systems, which are cost-prohibitive for MSMEs, this project focuses on delivering a lightweight, modular and efficient solution that addresses real-world constraints like unreliable internet connectivity and low budgets.

Motivation For the Project

The driving force behind this project stems from the growing number of fire incidents and the limitations of existing fire safety measures, particularly in the context of resource-constrained Micro and Small Enterprises (MSEs). Traditional fire detection systems are often too expensive and complex, making them inaccessible to small businesses. Moreover, conventional systems rely heavily on singular

modes of detection, such as visual smoke identification or localized heat alarms, which can result in delayed response times and an increased risk of damage.

Our project addresses these issues by integrating multiple detection methodologies and we are adopting the latest advancements in IoT and machine learning. Specifically, it builds upon findings from prior studies, such as "Hawk-Eye: An AI-Powered Threat Detector" by Ahmed Abdel Moamen and Mathias Ecchi, which demonstrated the efficiency of AI in accelerating detection and reducing false positives [1]. Similarly, the role of real-time environmental analysis highlighted in "An AI-Based Early Fire Detection System Utilizing HD Cameras" by Leendert Remmelzwaal serves as a foundation for enhancing the timeliness and precision of fire alerts in our system [2].

By adopting a multi-sensory approach, combining sound detection with gas, temperature and flame sensors, the proposed system ensures that no critical indicators of fire hazards are overlooked. Additionally, the real-time processing capabilities of ESP32 devices, as emphasized in "Gas Detection Using ESP32", further validate the suitability of the hardware for this application [3]. The emphasis on affordability, ease of deployment and scalability sets this project apart, making it a viable solution for MSEs to safeguard their operations against fire hazards.

Objectives

The primary goal of this research is to develop an intelligent fire detection system specifically designed for resource-constrained Micro and Small Enterprises (MSEs) in developing regions like India. Recognizing the unique challenges faced by these enterprises—including financial constraints, limited access to advanced technology and inadequate safety protocols—the proposed system aims to provide a robust, cost-effective solution that can detect fire hazards with speed and accuracy. The following are the detailed objectives:

1. **Cost-effective fire detection:** Design and implement a fire detection solution that is financially accessible to MSEs while maintaining high detection efficiency. The use of the ESP32 microcontroller ensures affordability without compromising on functionality [1].
2. **Integration of multi-sensor technology:** Develop a comprehensive system incorporating DHT22 for temperature and humidity sensing, MQ-2 for detecting smoke and combustible gases, infrared flame sensors, sound detection modules and Buzzer. This multi-sensor integration ensures the system can identify diverse indicators of fire hazards [2].
3. **Real-time data acquisition and processing:** Leverage IoT capabilities of the ESP32 to facilitate real-time monitoring and processing of environmental data. The system will immediately notify users of potential threats, enabling rapid emergency response [3].
4. **Deployment of machine learning algorithms:** Integrate pre-trained machine learning models for pattern recognition and anomaly detection, significantly enhancing system accuracy while minimizing false positives. The system employs TinyML to run algorithms locally on the ESP32, ensuring functionality in areas with unreliable internet connectivity [4].
5. **Seamless cloud and local integration:** Enable seamless integration with Firebase or alternative cloud services for remote monitoring while ensuring the system remains functional through local processing when cloud connectivity is unavailable [5].
6. **Scalable and modular architecture:** Design the system with scalability in mind, allowing easy customization and adaptation to a variety of industrial contexts. The modular approach ensures compatibility with existing safety infrastructures, reducing deployment complexity and costs [6].
7. **Empowerment of indian MSMEs:** Address the specific needs of Indian industrial settings by creating a localized, context-aware solution that meets international safety standards while being adapted to regional requirements [7].
8. **Reduction in fire-related losses:** Quantify and demonstrate the system's ability to reduce fire-related damages, ensuring sustainable and safer growth for MSEs through early detection and efficient response mechanisms [8].
9. **Integration of fire alarm sound:** Incorporate an audible fire alarm sound feature within the system to alert individuals immediately upon fire detection, enhancing the responsiveness and safety measures, ultimately ensuring the protection of assets and people in real-time for MSEs [9].

These objectives align with findings from various studies, including "Smart Fire Detection and Surveillance Using IoT" by Shanle Yao et al., which emphasize the potential of real-time monitoring and cloud analytics and "Gas Detection Using ESP32" by Shan R., which highlights the practical advantages of ESP32 in hazardous environments (Figure 1) [5,3].

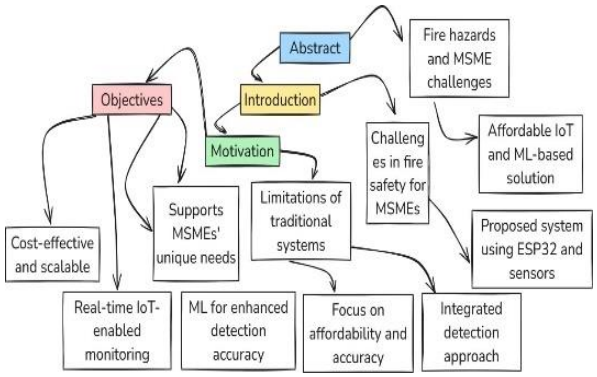

Figure 1: Representation for incorporating abstract, introduction, motivation and objectives.

Literature Review/Survey

IoT-based smart fire detection system using ESP32

Summary: This research describes an IoT-based fire detection system using an ESP32 microcontroller and sensors like the LM393 flame sensor, MQ-2 gas sensor and DHT11 temperature and humidity sensor. The system offers real-time monitoring, alarm activation and Blynk IoT app notifications, designed primarily for indoor

environments such as homes and offices. It emphasizes improving fire safety and minimizing damage through early detection.

Limitations: Sensor precision decreases in open areas and the system lacks integration with automatic fire suppression mechanisms. Additionally, it does not focus on improving the compactness or user-friendliness of the design.

Our advantage: Our project addresses open-area sensor limitations and integrates multiple sensors to ensure versatile fire detection. Furthermore, it includes robust fire suppression system capabilities and a compact, user-friendly design.

An efficient IoT-based novel approach for fire detection through ESP32 Microcontroller in forest areas

Summary: This study demonstrates a fire detection system using an ESP32 microcontroller with MQ6 gas sensors, DHT11 sensors and a GPS module for precise forest fire detection. Real-time sensor data is uploaded to Google Firebase and notifications are sent *via* a Kodular mobile app, allowing timely responses to fire threats.

Limitations: Network connectivity issues in remote areas, environmental interferences like vegetation and power constraints affect scalability. The system also lacks comprehensive emergency response mechanisms.

Our advantage: Our system ensures connectivity in remote areas using efficient networking solutions and optimizes power usage. It combines sensor data with GPS tracking, along with a more efficient emergency alert system, to ensure timely response in forest fire scenarios [10].

Smart fire detection and surveillance system using IoT

Summary: This research describes an IoT-based system for real-time fire and smoke monitoring using the ESP32 microcontroller and various sensors. Alerts and images are sent *via* Telegram to notify users remotely. The focus is on efficient and effective detection.

Limitations: The system lacks an integrated approach that combines fire detection with comprehensive surveillance capabilities. Most systems analyzed address either monitoring or detection separately.

Our advantage: Our project integrates fire detection with advanced surveillance features, including a real-time camera for continuous monitoring. This holistic approach enhances safety by bridging the gap between monitoring and detection [5].

Fire department alerting system

Summary: This study presents a system that integrates gas and fire sensors with the ESP32 microcontroller for real-time hazard detection. It features a buzzer alert mechanism and transmits sensor data to the Blynk IoT platform for continuous monitoring and analysis.

Limitations: The system lacks real-time monitoring and immediate response capabilities, critical for proactive fire management.

Our advantage: Our system builds on this by incorporating immediate response mechanisms and integrating real-time monitoring with multiple sensors for a more proactive and efficient risk mitigation strategy [11].

Image-based fire detection in industrial environments with YOLOv4

Summary: This research explores the use of AI and YOLOv4 for fire detection in industrial settings, such as warehouses. The model demonstrates superior performance over traditional smoke detectors in high-ceiling environments, achieving an F1 score of 98%.

Limitations: The system focuses exclusively on image-based detection, lacking multi-sensor integration and hardware adaptability.

Our advantage: Our project enhances image-based detection with YOLOv4 by integrating it with multi-sensor inputs for smoke, temperature and flame detection. This hybrid approach ensures high accuracy and robustness, making it suitable for diverse industrial applications [12].

Innovating fire detection system using artificial intelligence by image processing

Summary: This study reviews current methods in fire detection utilizing deep learning techniques, particularly Convolutional Neural Networks (CNNs), to identify fires through image processing. The authors discuss various datasets and the accuracy achieved by these methods.

Limitations: The paper primarily focuses on reviewing existing methods without proposing a novel implementation. It lacks practical application details and does not address integration with hardware components for real-time detection.

Our advantage: Our project not only leverages deep learning for image-based fire detection but also integrates multiple sensors (temperature, smoke, flame and sound) with the ESP32 microcontroller. This multi-sensor approach enhances detection accuracy and provides real-time alerts, making it more suitable for practical applications in small to medium-sized enterprises [13].

Fire detection in images using AI

Summary: This paper introduces a novel approach using deep learning models trained on extensive datasets of annotated fire and non-fire images. The system aims to automatically detect and localize fire instances with high accuracy and reliability.

Limitations: While the approach demonstrates high accuracy in image-based fire detection, it relies solely on visual data, which may be insufficient in scenarios where visibility is compromised, such as dense smoke or obstructions.

Our Advantage: By incorporating additional sensors like the MQ-2 for smoke detection and the DHT22 for temperature and humidity monitoring, our system can detect fires even in low-visibility conditions, providing a more robust solution [14].

Fire detection with deep learning: A comprehensive review

Summary: This comprehensive review covers advancements in fire detection using deep learning from 1990 to 2023. It discusses various deep learning architectures, sensor systems and the geographical distribution of studies in this field.

Limitations: The paper provides an extensive overview but does not delve into specific implementation details or practical applications of the discussed technologies.

Our advantage: Our research builds upon the insights from this review by implementing a practical, cost-effective fire detection system tailored for small and medium-sized businesses, addressing the gap between theoretical research and real-world application [15].

Recent advances and emerging directions in fire detection

Summary: This review paper presents recent studies and perspectives on machine learning algorithms applied to fire detection systems, focusing on performance metrics, datasets and misclassification issues.

Limitations: The study emphasizes the importance of dataset quality and addresses misclassification but does not propose specific solutions to mitigate false positives and negatives in practical scenarios.

Our advantage: Our system employs advanced algorithms and a multi-sensor approach to minimize false positives, enhancing detection reliability and operational safety in diverse conditions [16].

Fire detection using AI

Summary: This paper presents a novel approach for fire detection leveraging artificial intelligence and machine learning techniques, aiming to improve detection accuracy and response times.

Limitations: The approach is primarily theoretical, lacking details on hardware integration and real-world deployment challenges.

Our advantage: Our project details the integration of AI algorithms with specific hardware components like the ESP32 microcontroller and various sensors, providing a comprehensive solution ready for deployment in real-world settings [17].

Deep learning method for real-time fire detection system for surveillance

Summary: This study proposes a method employing a YOLOv5s network with a squeeze-and-excitation module for image filtering and classification to detect fires in real-time.

Limitations: While effective in image-based detection, the system may face challenges in environments with poor visibility or where the camera's field of view is obstructed.

Our advantage: Our system's integration of multiple sensors ensures that fire detection is not solely reliant on visual data, allowing for effective operation even in challenging environmental conditions [18].

A review on fire and smoke detection with intelligent control for enhanced safety using machine learning and internet of things

Summary: This paper provides a thorough review of fire and smoke detection features, advantages and innovative contributions to fire safety challenges, emphasizing the role of machine learning and IoT.

Limitations: The review highlights various technologies but does not offer a concrete implementation strategy or address cost considerations for small to medium-sized enterprises.

Our advantage: Our research offers a cost-effective implementation strategy, utilizing affordable hardware and open-source software, making advanced fire detection accessible to resource-constrained businesses [19].

Image-based fire detection in industrial environments with YOLOv4

Summary: This research explores the potential of AI to detect and recognize fires using object detection on image streams, specifically implementing the YOLOv4 model in industrial warehouse settings.

Limitations: The system's reliance on image streams may limit its effectiveness in scenarios where visual data is insufficient or compromised.

Our advantage: By combining image-based detection with data from various environmental sensors, our system ensures comprehensive monitoring and detection capabilities, even in less-than-ideal visual conditions [12].

Integrated fire detection system using machine learning and IoT

Summary: This paper discusses an integrated fire detection system that combines smoke, thermal and image analysis using machine learning to provide real-time fire detection and reduce false alarms.

Limitations: The study provides a conceptual framework but lacks detailed information on hardware selection, system architecture and implementation challenges.

Our advantage: Our research offers detailed insights into hardware components, system architecture and practical implementation, providing a ready-to-deploy [20].

Methodology

The methodology section outlines the systematic approach adopted to design, develop and implement the proposed system for detecting fire, smoke and other environmental conditions. This project integrates hardware and software components to create an efficient IoT-based monitoring system capable of processing real-time data. The hardware includes an ESP32 microcontroller, Buzzer, DHT22 temperature and humidity sensor, MQ-2 smoke sensor, IR flame sensor and KY-038 sound sensor. These components were strategically selected for their affordability, compatibility and ability to fulfill the project’s functional requirements. Each module was carefully connected to ensure accurate data collection and seamless communication with the central processing unit. In addition to hardware assembly, the software development process focused on programming, sensor calibration and implementing data processing logic to ensure reliable detection of critical conditions. The system uses the ESP32 as the main controller to gather data from multiple sensors, analyze the input in real time and trigger appropriate actions. The data collection process was structured to prioritize efficiency, leveraging GPIO pins and standardized protocols like I2C and UART for sensor communication. This methodology is designed to ensure

replicability and scalability, providing a solid framework for implementing IoT-based environmental monitoring solutions.

Hardware parts

The hardware components form the foundation of the proposed system, ensuring reliable data collection and processing. The setup includes the ESP32 microcontroller as the central processing unit, responsible for handling data from sensors and facilitating communication. Sensors like the DHT22 provide temperature and humidity readings, the MQ-2 detects smoke and gases, the IR Flame Sensor detects flame intensity and the KY-038 sound sensor monitors sound levels for anomalies. These components were carefully chosen for their accuracy, compatibility and ease of integration, enabling a cost-effective and efficient solution for real-time monitoring (Table 1).

Component	Description and features	Pins/connections
DOIT DEVKIT V1 ESP32-WRO OM-32	High-performance microcontroller with Wi-Fi and Bluetooth capabilities. Features dual-core Xtensa® processors.	30 pins (GPIO, VIN, 3.3V, GND, RX, TX, ADC, DAC, SPI, I2C, PWM, etc.). GPIO used for sensors: D4, D5, D14, D15 and GND.
Active buzzer module (3.3V-5V)	Buzzer with ESP32: An active buzzer is connected to ESP32's GPIO4 to generate sound alerts when triggered	VCC (Buzzer) 3.3V (ESP32), GND (Buzzer) GND (ESP32), I/O (Signal) GPIO4 (ESP32)
ESP32-CAM-MB USB Programmer	Micro USB interface for easy uploading of code to ESP32-CAM. Also powers the board.	Micro-USB port connects to PC. Pins: 5V, GND, RX, TX. Connect RX-TX between ESP32-CAM and CAM-MB for programming.
Breadboard (840 points)	Solderless breadboard for prototyping. Durable nickel-plated contacts for temporary hardware connections.	Multiple tie-points. Power rails for 3.3V and GND connections to ESP32 and sensors. Fits male jumper wires or modules.
DHT22 sensor (AM2302)	Digital temperature and humidity sensor. Highly accurate with ±0.5°C temp and ±2% humidity tolerances.	3 pins: OUT (data), VCC (3.3–6V), GND. Uses digital communication. Equation: Humidity = ADC counts × linear coefficient.
IR flame sensor module	Detects flame or infrared light (760-1100 nm wavelength). Useful for early fire detection.	3 pins: AO (Analog Out), DO (Digital Out), VCC (3.3V or 5V) and GND. Adjustable sensitivity *via* potentiometer.
Sound detection module (KY-038)	Detects sound intensity changes with an integrated microphone. Suitable for threshold-based alerts.	3 pins: DO (digital signal output), GND, VCC (3.3–5V). Potentiometer to adjust sensitivity. Threshold = Signal × Gain factor.
MQ-2 gas sensor	Detects flammable and hazardous gases like LPG, CO, H2. Offers both analog and digital output.	4 pins: VCC (5V), GND, AO (analog output), DO (digital output). Resistance variation is processed to detect gas concentration: RS = VO × RL / (Vc - VO).
Male-to-male jumper wires	Connect hardware components on the breadboard and ESP32 for easy prototyping.	Standard 20cm wires. Supports breadboards and module connections.
Female-to-Female jumper wires	For linking modules and breadboard pins or directly interconnecting sensors.	Standard 10cm and 20cm wires. Firm grip on module headers like the DHT22 or IR Flame Sensor.

Table 1: Name and descriptions of hardware parts used in the project.

Integration of sensors with EPS32

The integration of sensors with the ESP32 microcontroller involves precise connection and configuration to ensure seamless communication between the hardware components. Each sensor, such as the DHT22 for temperature and humidity monitoring, MQ-2 for gas detection and the IR Flame Sensor for flame detection, is connected to specific GPIO pins of the ESP32, based on their power and signal requirements. Digital sensors like the DHT22 and IR Flame Sensor use GPIO pins for signal transmission, while analog sensors like MQ-2 leverage the ADC capabilities of the ESP32 for data interpretation. Power considerations, including voltage levels (3.3V or 5V), are managed to ensure compatibility, while jumper wires and a breadboard facilitate a solderless and modular approach for prototyping. This systematic hardware integration forms the foundation for reliable data collection and processing within the system.

DOIT DEVIT V1 ESP32-WROOM-32 development board

The DOIT DEVIT V1 ESP32-WROOM-32 development board is a robust platform designed for Internet of Things (IoT) applications, featuring the powerful ESP32 microcontroller. Operating at frequencies up to 240 MHz, it provides dual-core processing capabilities that enhance multitasking and performance. The board is equipped with Wi-Fi (802.11 b/g/n) connectivity, enabling seamless integration into wireless networks for data transmission and device control. It offers 30 GPIO pins, which support various interfaces such as I2C, SPI and PWM, allowing for extensive peripheral connections. Additionally, its low-power modes optimize energy consumption, making it ideal for battery-operated devices in smart home and industrial applications (Figure 2, 3).

Figure 2: DOIT DEVIT V1 ESP32-WROOM-32.

https://doi.org/10.70844/aest.2025.1.1

Applied Engineering Solutions and Technologies

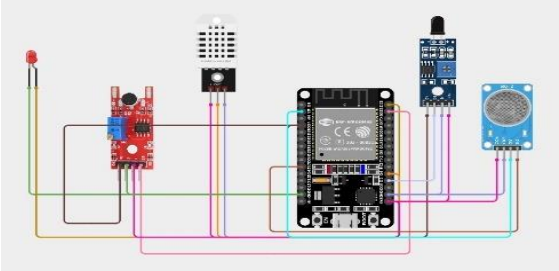

Figure 3: Wiring diagram of DOIT DEVIT V1 ESP32-WROOM-32 with various sensors and modules.

ESP32 CAM Wi-Fi module Bluetooth with OV2640 camera module 2MP

The ESP32 CAM Wi-Fi Module integrates a powerful ESP32 microcontroller with Bluetooth capabilities and an OV2640 camera module, featuring a 2MP resolution. This compact module is designed for applications such as face recognition and image processing, leveraging its dual-core architecture that operates up to 240 MHz The ESP32 CAM supports various wireless communication protocols, enabling seamless connectivity for IoT projects. Its built-in JPEG compression allows efficient image transmission over Wi-Fi, making it ideal for real-time surveillance (Figure 4).

Figure 4: ESP32 CAM Wi-Fi module Bluetooth with OV2640 camera module 2MP.

ESP32-CAM-MB MICRO USB download module for ESP32 CAM development board

The ESP32-CAM-MB Micro USB Download Module is designed to facilitate the programming and debugging of the ESP32-CAM development board. This module simplifies the process of uploading code to the ESP32-CAM by providing a USB interface, eliminating the need for additional hardware like USB-to-serial converters (Figure 5).

Figure 5: ESP32-CAM-MB MICRO USB download module for ESP32 CAM development board.

REES52 elementz nickel plated 840 points bread board or solderless piecesb circuit test board,white,battery powered

The REES52 Elementz Nickel Plated 840 Points Breadboard is a versatile, solderless circuit test board designed for prototyping electronic circuits. With 840 connection points, it provides ample space for assembling complex projects without the need for soldering, making it ideal for both beginners and experienced engineers. (We have used 2 board with connected together) (Figure 6).

Figure 6: REES52 Elementz nickel plated 840 points bread board or solderless piecesb circuit test board, white, battery powered.

DHT22 digital temperature and humidity sensor module AM230

The DHT22 Digital Temperature and Humidity Sensor Module, also known as AM2302, is a highly accurate sensor widely used in various environmental monitoring applications. It measures temperature in the range of -40°C to 80°C with an accuracy of ±0.5°C and humidity from 0% to 100% RH with an accuracy of ±2-5% RH (Figure 7, 8).

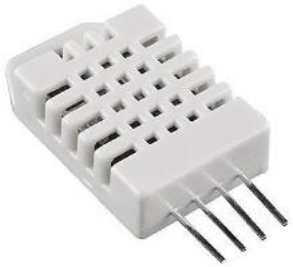

Figure 7: DHT22 Digital temperature and humidity sensor module AM230.

The provided image illustrates the connection of a DHT22 (AM2302) temperature and humidity sensor with the ESP32 microcontroller (Figure 8). Here's a detailed explanation of each connection step:

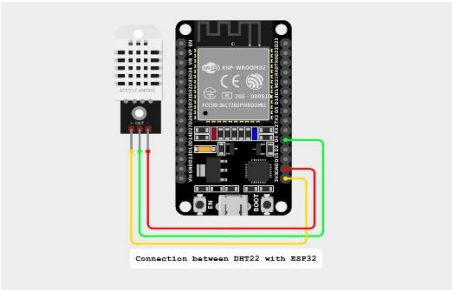

Figure 8: Connection between DHT22 with ESP32.

https://doi.org/10.70844/aest.2025.1.1

Applied Engineering Solutions and Technologies

DHT22 power supply: The red wire from the + (VCC) pin of the DHT22 is connected to the 3V3 pin of the ESP32. This provides a stable 3.3V power supply to the sensor.

Ground connection: The black or green wire from the - (GND) pin of the DHT22 is connected to the GND pin of the ESP32, ensuring a common ground between the sensor and the ESP32.

Data signal line: The yellow wire from the OUT (Data) pin of the DHT22 is connected to the GPIO D4 pin of the ESP32. This connection allows the ESP32 to read temperature and humidity data from the sensor.

Infrared IR flame sensor module

The Infrared IR Flame Sensor Module is a crucial component for fire detection systems, particularly when interfaced with Arduino platforms. Utilizing the LM393 comparator chip, this sensor effectively detects infrared light emitted by flames, specifically within the wavelength range of 760 nm to 1100 nm (Figure 9, 10).

Figure 9: Infrared IR flame sensor module.

The image demonstrates the connections of an Infrared IR sensor module with the ESP32 microcontroller. Here's a detailed step-by-step explanation of the connections:

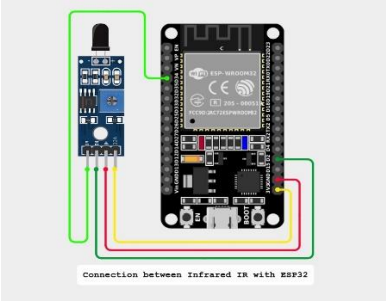

Figure 10: Connection between infrared IR with ESP32.

Power supply (VCC): The VCC pin of the IR sensor (red wire) is connected to the 3V3 pin on the ESP32. This supplies a stable 3.3V power to the IR sensor module.

Ground (GND): The GND pin of the IR sensor (black or green wire) is connected to the GND pin of the ESP32. This establishes a common ground between the IR sensor and the ESP32 for proper operation.

Digital output (DO): The DO (Digital Output) pin of the IR sensor (yellow wire) is connected to the GPIO D4 pin on the ESP32. This allows the ESP32 to read the digital signal generated by the sensor when it detects an obstacle.

Analog output (AO) (Optional): The AO (Analog Output) pin is left unconnected in the illustration, as it is not being utilized for digital IR detection. If required for analog sensing, it can be connected to an ADC pin on the ESP32.

Active buzzer module 3.3V-5V

The buzzer alarm is a simple electronic component used to produce sound for alerting or signaling purposes. It can be connected to a microcontroller, such as the ESP32, to activate the sound alarm when certain conditions are met, such as detecting a temperature threshold, motion, or other events (Figure 11, 12).

Figure 11: Active buzzer module 3.3V-5V.

The image shows the connection between an active buzzer module 3.3V-5V and the ESP32 microcontroller, with the following wiring details:

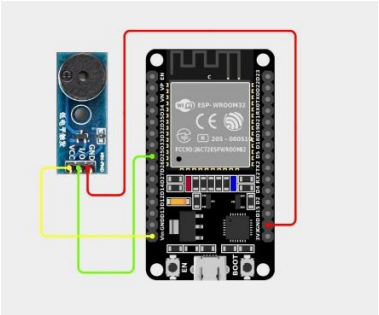

Figure 12: Connection between active buzzer module with ESP32.

Buzzer power supply: The yellow wire from the + (VCC) pin of the buzzer is connected to the 5V pin of the ESP32. This provides a stable 5V power supply to the buzzer for it to function correctly.

Ground connection: The red wire from the - (GND) pin of the buzzer is connected to the GND pin of the ESP32, ensuring a common ground between the buzzer and the ESP32.

Control signal line: The green wire from the control pin (or sometimes marked as the signal pin) of the buzzer is connected to a GPIO pin (such as GPIO 25 or another available pin) of the ESP32. This connection allows the ESP32 to send signals to the buzzer to activate or deactivate it as needed.

Sound detection module sensor for intelligent vehicle compatible with Arduino

The Sound Detection Module Sensor is a vital component for intelligent vehicle systems, compatible with Arduino platforms. This sensor detects sound levels and frequencies, enabling applications such as emergency alert systems and voice command recognition. It features high sensitivity, allowing it to pick up a wide range of sounds, from sirens to specific audio signals. The module typically offers both analog and digital outputs for flexible integration, along with an adjustable sensitivity setting *via* a potentiometer. By

incorporating this sensor, developers can enhance vehicle safety and functionality, facilitating real-time responses to auditory cues in various driving environments (Figure 13, 14).

Figure 13: Sound detection module sensor for intelligent vehicle compatible with Arduino.

The image shows the connection between a Sound Detection Sensor and the ESP32 microcontroller.

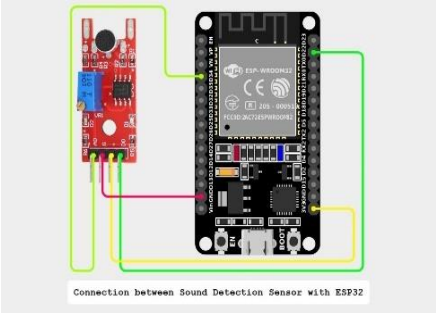

Figure 14: Connection between sound detection sensor with ESP32.

Power supply (VCC): The VCC pin of the sound sensor (red wire) is connected to the 3V3 pin of the ESP32. This provides the required 3.3V power supply to the sensor.

Ground (GND): The GND pin of the sound sensor (black wire) is connected to the GND pin on the ESP32, ensuring a common ground connection.

Digital output (D0): The D0 pin (yellow wire) of the sensor is connected to GPIO D2 on the ESP32, enabling the microcontroller to receive digital signals for sound detection.

Analog output (A0): The A0 pin (green wire) is connected to an ADC-capable pin GPIO D34 on the ESP32, allowing the sensor to transmit analog sound intensity data to the microcontroller. This setup ensures that the ESP32 can process both digital and analog data from the sound sensor for applications such as noise level monitoring or sound-based event detection. Ensure proper wiring and secure connections to avoid errors.

MQ-2 smoke LPG butane hydrogen gas sensor detector module

The MQ-2 Smoke, LPG, Butane and Hydrogen Gas Sensor Detector Module is a versatile semiconductor sensor designed for detecting various flammable gases and smoke. It operates effectively within a detection range of 300 to 10,000 ppm, making it suitable for applications such as gas leakage alarms and portable gas detection devices (Figure 15-16).

Figure 15: MQ-2 smoke LPG butane hydrogen gas sensor detector module.

The image illustrates the connection of the MQ-2 Smoke and Gas Sensor to the ESP32 microcontroller. Below are the connection details:

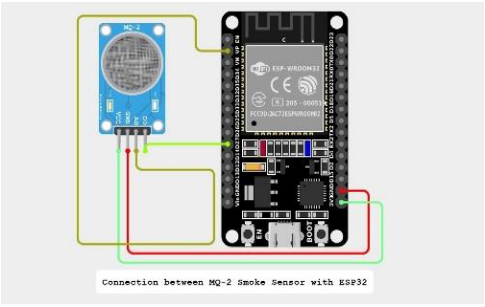

Figure 16: Connection between MQ-2 smoke sensor with ESP32.

Power supply (VCC): The VCC pin of the MQ-2 sensor (red wire) is connected to the 3V3 pin of the ESP32. This provides the required 3.3V power to the sensor.

Ground (GND): The GND pin of the MQ-2 sensor (black wire) is connected to the GND pin of the ESP32, establishing a common ground for operation.

Digital output (D0): The D0 pin (yellow wire) of the MQ-2 sensor is connected to GPIO D2 on the ESP32. This allows the microcontroller to receive a digital signal when smoke or gas levels exceed a certain threshold.

Analog output (A0): The A0 pin (green wire) of the MQ-2 sensor is connected to GPIO D34 of the ESP32. This enables the microcontroller to read analog data representing varying smoke or gas concentrations. This setup allows the ESP32 to process both digital and analog signals from the MQ-2 sensor, making it suitable for applications that monitor air quality or detect harmful gases. Ensure secure connections to avoid issues during operation.

Male to male jumper wires

These wires feature male connectors at both ends, allowing direct connections between male header pins on components or breadboards. Typically, available in packs of 40, they are designed for versatility and ease of use. With a standard 0.1" (2.54 mm) pitch, these wires are compatible with various development boards and sensors. They facilitate quick and reliable connections without the need for soldering, making them ideal for prototyping, testing and modifying circuits. The wires are often color-coded for easy identification, enhancing organization during assembly (Figure 17).

https://doi.org/10.70844/aest.2025.1.1

Applied Engineering Solutions and Technologies

Figure 17: Male to male jumper wires.

Female to female jumper wires

Conversely, Female to Female Jumper Wires have female connectors on both ends, enabling connections between male pins of different components. This configuration is particularly useful when interfacing with devices that have male headers. Similar to their male counterparts, these wires provide a solderless solution for connecting components, allowing for flexible circuit design and easy modifications. They are also available in various lengths and colors, promoting efficient organization and clarity in circuit setups (Figure 18).

Figure 18. Female to female jumper wires.

Data acquisition process

The data acquisition process forms the core of the proposed system, as it ensures the accurate collection and transmission of environmental data critical for fire detection, smoke monitoring and general environmental assessment. The system leverages a range of sensors, each specialized in capturing a unique parameter necessary for detecting dangerous conditions. The data acquisition process involves several steps: sensor activation, signal measurement, data transmission to the microcontroller and processing for real-time decision-making.

Sensors and their operating principles

The system includes multiple sensors, each with specific roles in capturing environmental data:

- **DHT22 temperature and humidity sensor:** This sensor operates based on a capacitive humidity sensor and a thermistor. It measures relative humidity with an accuracy of ±2% RH and temperature with a tolerance of ±0.5°C. The data from the DHT22 is captured using digital signals that represent humidity and temperature readings in real time.
- **MQ-2 smoke and gas sensor:** The MQ-2 sensor detects gases such as LPG, methane, carbon monoxide and smoke. It works by measuring the resistance of the sensor's heating element, which changes based on the concentration of gases. Higher resistance correlates with lower gas concentrations and vice versa. The sensor's output can be processed as either an analog or digital signal.
- **IR flame sensor:** The IR flame sensor detects the intensity of infrared light emitted by a fire. It is sensitive to infrared light in the range of 760–1100 nm, which is characteristic of a flame's radiation. The analog output from the sensor is processed to determine if flame detection conditions are met.
- **KY-038 sound detection module:** This sensor detects sound levels, typically used to identify noise from a flame or combustion process. The module includes a microphone and output pins to signal when the noise level surpasses a predefined threshold, indicating potential fire or anomaly situations.

Formula for temperature (in °C) from DHT22

- The temperature value is calculated using the raw digital signal provided by the sensor. A simple formula to convert the digital reading to temperature is:

$$T = \left(\frac{65536.0}{RH_{Raw}}\right) \times 200.0 - 50.0$$

Where:

- T = Temperature in °C.
- RH_{Raw} = Raw data from the sensor's temperature section.

For graphical representation:

- A sensor performance chart (graphs) showing how the DHT22 sensor's accuracy improves with calibration, especially in different temperature and humidity ranges, can help visualize its reliability.

For MQ-2 smoke and gas sensor, you can use:

- Gas sensitivity curve: It shows how the sensor's resistance varies with gas concentration. The general formula:

$$R = R_0 \times C^n$$

Where:

- R = Resistance at a given gas concentration C.
- R_0 = Resistance under normal (clean air) conditions.
- C = Gas concentration.
- n = Sensor's characteristic constant.

A graph of gas sensor characteristics versus resistance or ppm (parts per million) could be visually appealing to highlight how the resistance changes as the concentration of gas increases.

Signal measurement and acquisition

Each sensor measures its respective environmental parameter and converts the raw physical values into electronic signals that can be read by the ESP32 microcontroller. Digital sensors such as the DHT22 generate digital outputs corresponding to the temperature and humidity levels. These signals are transmitted directly to the GPIO pins configured to receive digital signals. On the other hand, analog sensors like the MQ-2 and IR flame sensor generate analog voltage levels corresponding to the intensity of gas concentration and flame detection, respectively. To measure these signals, the ESP32 makes use of its Analog-to-Digital Converter (ADC) to convert the analog

voltages into readable digital values. This process ensures that real-world continuous signals are accurately converted into data that can be processed by the microcontroller.

- ADC conversion formula:

$$V_{in} = \frac{ADC_{raw}}{ADC_{max}} \times V_{ref}$$

Where:

- V_{in} = Input voltage.
- ADC_{raw} = Raw ADC value.
- ADC_{max} = Maximum ADC resolution (4095 for a 12-bit ADC).
- V_{ref} = Reference voltage (3.3V for ESP32).

For graphical representation, show an ADC transfer function curve:

- A graph that compares ADC values with respect to the input voltage would make this visual and provide clarity about how analog values get scaled for digital processing.

Frequency of data collection

The sensors in the system are designed to continuously monitor their respective environmental parameters, with data acquisition taking place at specific intervals to ensure up-to-date information for fire detection. For instance, temperature and humidity readings from the DHT22 sensor are acquired every second, while gas detection readings from the MQ-2 sensor are collected every two seconds to ensure accurate monitoring. Similarly, the IR flame sensor and the sound sensor capture data continuously, allowing for a nearly real-time reaction to changes in their environmental conditions.

Nyquist-Shannon sampling theorem:

$$f_s \geq 2 \times f_{max}$$

Where:

- f_s = Sampling frequency.
- f_{max}= Maximum frequency of the signal.

Signal processing and data validation

To enhance the reliability and accuracy of the readings, the collected raw data is filtered and processed before being acted upon. The raw outputs from analog sensors are typically averaged or smoothed over a few readings to eliminate noise or signal fluctuations. This process ensures that only consistent, high-quality data is passed to the system's decision-making algorithms. Furthermore, for enhanced accuracy, sensor calibration may be performed where necessary to minimize errors due to manufacturing tolerances or environmental conditions. For instance, the DHT22 sensor is calibrated to ensure that its temperature and humidity readings are within the desired precision levels, reducing the possibility of inaccurate readings during extreme conditions.

Discuss data filtering techniques with visual charts:

- Moving average filter formula (for smoothing):

$$MA = \frac{\sum_{i=1}^{n} x_i}{n}$$

Where:

- MA = Moving average.
- x_i = Raw sensor data.
- n = Window size.

Provide a filtered data curve vs. unfiltered data curve graph to show the smoothing effect and the reduction in signal noise when using a moving average filter.

For Kalman filter (more advanced):

- The Kalman filter combines a series of predictions and measurements to estimate the state more accurately.

Example formula:

$$x_{k+1} = Fx_k + Bu_k + noise$$

You could show a Kalman Filter implementation chart comparing raw and filtered output for noisy data.

Communication with the microcontroller

Once the data from the sensors is gathered and processed, it is sent to the ESP32 microcontroller, which acts as the central unit in charge of monitoring and decision-making. The data is transmitted using direct GPIO communication for digital sensors (e.g., DHT22, KY-038) and analog-to-digital conversion for sensors providing analog output (e.g., MQ-2, IR Flame Sensor). The ESP32 reads data continuously from all sensors, allowing it to maintain real-time awareness of the monitored environment.

- Signal transmission formula:

$$Single\ period = \frac{1}{Baud\ rate}$$

Where:

- Signal period is the time for one full signal cycle at a specific baud rate (e.g., 9600 baud for simple UART). A time series plot of serial data transmission or an ISR (Interrupt Service Routine) flowchart would be ideal visuals. For example, show how the data reading from a DHT22 happens with interruptions in the signal transmission process.

Real-time data transmission to the cloud/server: In advanced IoT systems, it is crucial that data is accessible remotely for monitoring and analysis. To facilitate this, the system can be configured to send the acquired data to the cloud *via* an internet connection. The ESP32's Wi-Fi capability is used to connect to the internet, transmitting data to a remote server using protocols like MQTT or HTTP. The cloud-based server receives the data, which is processed for storage or used in further analysis, such as triggering alerts or controlling other hardware systems in response to the detected environmental hazards. You can include an MQTT Packet Transmission Graph, showing how data is transferred *via* MQTT protocol (if using IoT communication). Use the formula for message frequency:

- **Message frequency formula:**

$$f_m = \frac{Message\ sent}{Time\ interval}$$

Visualize this through a bar graph showing messages per second versus the operational state of the system.

- Additionally, show the communication latency and how it varies with sensor distance:

$$Latency = \frac{Distance}{Transmission\ speed}$$

Where the transmission speed could be represented with values in Mbps (megabits per second).

Software implementation

The software implementation plays a key role in the overall performance of the system, ensuring smooth communication between the hardware components (sensors, ESP32) and the user interface. This section outlines the tools and technologies used, the programming languages involved and the coding process to bring the project to life.

IDE and development tools

The project was developed using two primary tools:

1. **Arduino IDE:**

Arduino IDE is the most suitable integrated development environment for working with the ESP32 microcontroller. It allows easy writing, uploading and debugging of code on the ESP32.

- The Arduino IDE is flexible and supports libraries that simplify the process of working with different sensors (DHT22, MQ-2, etc.). This feature made it easier to interface the sensors with the ESP32 board.
- ESP32 boards were added to the Arduino IDE using the board manager URL (https://dl.espressif.com/dl/package_esp 32_index. json) in the preferences section to enable uploading to the board.

2. **VS Code (Visual Studio Code):**

- While Arduino IDE handles programming the microcontroller, VS Code was used for writing the overall design code in some other scripting languages such as HTML and CSS for web interface applications (where applicable). The ESP32 could communicate and post data to a web server interface through *vs* Code to make the system more user-friendly.
- The use of VS code enhanced productivity by providing access to various extensions and debugging.

Programming languages used

The following programming languages and frameworks were used in the software stack of this project:

C/C++ for ESP32 programming (Arduino C/C++): The Arduino C/C++ programming language was utilized for coding on the ESP32 microcontroller to read data from the sensors and communicate it *via* Wi-Fi to a remote server or display interface. The code is structured around a central loop that runs continuously, checking the sensor states, reading values and performing actions based on the readings.

Libraries/frameworks for sensor integration

Different libraries were used in conjunction with the Arduino IDE for integrating the sensors with the ESP32:

- DHT22 Library: For reading data from the DHT22 temperature and humidity sensor.
- MQ-2 Smoke Sensor Library: For detecting gases, such as methane (CH4) or carbon monoxide (CO), which contribute to fire risks.
- IR Flame Sensor: Custom code written based on the module's datasheet for flame detection.

Data sharing and communication

To ensure smooth data communication between the ESP32 and any external server or user interface:

- **Wi-Fi communication (ESP32):** The ESP32 uses WiFi for transmitting data to either a web server (*via* HTTP) or directly to a connected device.

Code for Wi-Fi connection:

```
WiFi.begin("SSID", "PASSWORD");
while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to
WiFi...");
}
Serial.println("Connected to WiFi!");
```

Serial communication: Data was also shared with an external monitoring system using Serial communication *via* USB to the laptop or external devices.

Programming logic & flow

The flow of operations in the code involves setting up hardware peripherals (sensors), collecting readings at intervals, processing those readings and executing actions based on the values:

- Initialize sensors: Each sensor is initialized with its specific pins and values are continuously monitored.
- Data Processing: Read the sensor outputs (e.g., temperature, smoke levels) and compare them to predefined thresholds (for alerts like fire).
- Communication: Send the processed data over the Wi-Fi network to the intended recipient device/server for real-time monitoring.
- UI Interface/Alerting: In some configurations, alerts might be generated through the web interface when values exceed critical thresholds (e.g., high smoke concentration or high temperature).

Debugging and error handling

The Arduino IDE's built-in Serial Monitor was used for debugging by displaying sensor values and error messages (if any), to ensure the system was working correctly during development. Appropriate error handling was incorporated in the code to handle sensor reading errors (e.g., a timeout or failed sensor readings).

https://doi.org/10.70844/aest.2025.1.1

Applied Engineering Solutions and Technologies

System architecture/workflow

The system architecture of the project is designed to facilitate seamless data acquisition, processing and communication. It integrates multiple hardware components, such as the ESP32 microcontroller, DHT22 temperature and humidity sensor, MQ-2 gas sensor, IR Flame sensor and others, to monitor environmental conditions. The system follows a modular approach where each sensor captures data related to specific environmental parameters (temperature, humidity, gas levels and potential fire hazards). The microcontroller, in this case, the ESP32, is at the core of the architecture. It reads data from the sensors, processes the input and communicates over WiFi to a server or a web interface, providing real-time monitoring. The software running on the ESP32 is designed to handle all necessary tasks, including sensor initialization, data acquisition, signal processing and error handling. In terms of workflow, the system operates in a continuous loop. The ESP32 microcontroller constantly monitors and processes the sensor readings. As the sensors detect changes in their respective parameters, this data is transferred to a cloud-based server or user interface for display and further action. Depending on the project setup, the system may trigger alerts if certain predefined thresholds (e.g., high temperature or gas concentration) are exceeded. This workflow ensures timely detection of potential hazards and supports decision-making in real-time. It can also be expanded with machine learning or AI for predictive analysis of trends in the sensor data. The overall architecture is designed for scalability, allowing for the inclusion of additional sensors and control mechanisms as the system evolves (Figure 19-23).

Use case diagram:

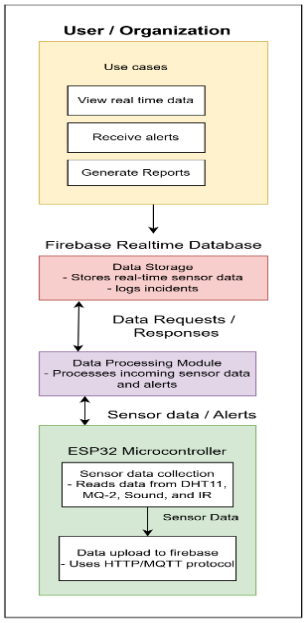

Figure 19: Use case diagram.

The diagram illustrates a system architecture for real-time sensor data monitoring, processing and alerting (Figure 20).

- Use case layer: End-users can monitor real-time data, receive alerts and generate reports for actionable insights.
- Firebase Realtime Database: Acts as centralized data storage, logging sensor data and incidents while enabling data requests/responses.
- Data Processing Module: Processes incoming sensor data, triggers alerts based on conditions and ensures consistent storage.
- ESP32 Microcontroller: Collects data from sensors (DHT11, MQ-2, sound, IR) and transmits it to Firebase using HTTP/MQTT protocols.
- Data Flow: Sensor data is collected by ESP32, sent to Firebase for processing and made available to users for real-time monitoring and reporting. This architecture is scalable for IoT applications like environmental monitoring or smart systems (Figure 20).

State diagram:

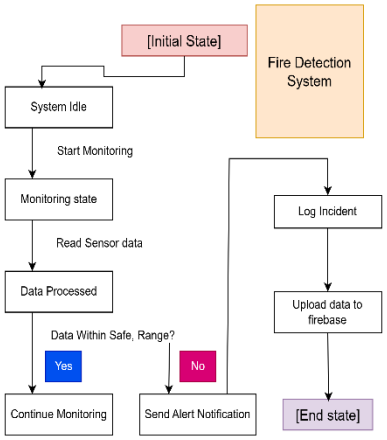

Figure 20: State diagram.

The state diagram represents the operational workflow of the Fire Detection System, detailing the sequential transitions between various states from initialization to termination.

- Initial State: The system begins in the System Idle state, where it awaits activation.
- Monitoring State: Upon activation, the system transitions to the Start Monitoring state. It proceeds to the Monitoring State, where sensors are actively reading environmental data (e.g., temperature, smoke, or gas levels).
- Data Processing: The sensor data is processed to evaluate if the observed values are within safe thresholds. If the data is within the safe range, the system returns to the Monitoring State, continuing its operation without interruptions. If the data exceeds the threshold, the system transitions to the Send Alert Notification state.
- Alert and Incident Logging: In case of unsafe readings, an alert notification is sent to relevant stakeholders or systems. Simultaneously, the Log Incident state records the event for traceability and further analysis.
- Data Upload: Processed data, including incident details, is uploaded to the Firebase Realtime Database for centralized storage and future reference.
- End State: The system concludes the cycle in the End State, signifying the completion of one monitoring process.

Flowchart diagram:

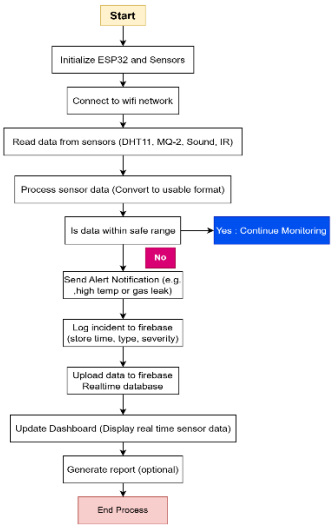

Figure 21: Flowchart diagram.

The flowchart demonstrates the logical workflow of the Fire Detection System, outlining each step from initialization to completion, with a focus on real-time monitoring, data processing and response mechanisms.

- System Initialization: The process begins with initializing the ESP32 microcontroller and sensors, including DHT11, MQ-2, sound and IR sensors, which are crucial for environmental data collection.
- Network Connection: The system connects to a Wi-Fi network to enable seamless communication with the Firebase Realtime Database and remote monitoring platforms.
- Data Collection: Sensors continuously read environmental parameters, such as temperature, gas levels, sound and infrared signals, providing critical inputs for fire detection.
- Data Processing: Collected data is processed and converted into a usable format, enabling effective decision-making and response actions.
- Threshold Evaluation: The system evaluates whether the processed data falls within predefined safety thresholds.
- Safe Data: If data values are within safe limits, the system loops back to continue monitoring.
- Unsafe Data: If data exceeds safety thresholds (e.g., high temperature or gas leaks), the system transitions to the alert mechanism.
- Alert Generation: Alerts are generated and sent to relevant stakeholders, indicating critical events such as fire hazards. These alerts can include parameters like temperature, gas concentration, or severity level.
- Incident Logging: The system logs the incident details, including time, type and severity, into the Firebase Realtime Database for future reference and analysis.
- Data Upload and Dashboard Update: Processed data, including incident logs, is uploaded to the Firebase Realtime Database. The system updates the dashboard in real-time, displaying sensor data and alert notifications for monitoring purposes.

ER diagram:

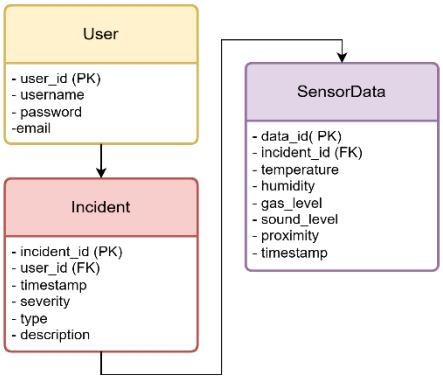

Figure 22: ER diagram.

The ER diagram illustrates the database schema of the fire detection system, highlighting the entities, their attributes and the relationships among them. The design ensures the efficient storage, retrieval and management of data related to users, incidents and sensor readings.

Entities and attributes:

a. User entity:

Attributes:

- $user_{id}$ (PK): A unique identifier for each user.
- username: The name associated with the user.
- password: The authentication credential for secure access.
- Purpose: The User entity stores details of the system users, enabling authentication and associating users with incidents.

b. Incident entity:

Attributes:

- $incident_{id}$ (PK): A unique identifier for each incident.
- $user_{id}$ (FK): A foreign key referencing the user_id in the User entity to associate incidents with specific users.
- timestamp: The date and time when the incident occurred.
- severity: The severity level of the incident (e.g., minor, major, critical).
- type: The type of incident detected (e.g., fire, gas leak).
- description: Additional details describing the incident.

Purpose: The Incident entity records information about detected hazards, their characteristics and the associated user.

c. Sensor data entity:

Attributes:

- $data_{id}$ (PK): A unique identifier for each data entry.
- $incident_{id}$ (FK): A foreign key referencing the incident_id in the Incident entity to associate sensor readings with specific incidents.

- temperature: The temperature data collected by the sensor.
- humidity: The humidity level detected.
- gas_{level}: The concentration of gases (e.g., CO, methane).
- $sound_{level}$: The sound intensity recorded.
- proximity: The proximity or motion data, if applicable.
- timestamp: The time the sensor data was recorded.

Purpose: The SensorData entity stores detailed environmental readings from various sensors, serving as the foundation for monitoring and hazard detection.

Relationships:

- User to Incident (1-to-Many): A single user ($user_{id}$) can be associated with multiple incidents ($incident_{id}$). This relationship ensures that incidents are traceable to the user monitoring the system.
- Incident to SensorData (1-to-Many): Each incident ($incident_{id}$) can have multiple associated sensor readings ($data_{id}$). This relationship links specific hazard events with the detailed sensor data collected during the event.

Deployment diagram:

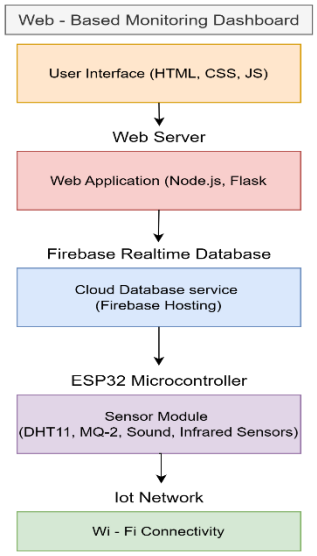

Figure 23: Deployment diagram.

The deployment diagram depicts the architectural layout and the interactions between various components of the fire detection system. It provides a comprehensive representation of the hardware and software infrastructure utilized to monitor and manage fire detection in real-time. The system comprises the following layers and components:

1. Web-based monitoring dashboard

User interface (UI):

- Built using standard web technologies such as HTML, CSS and JavaScript.
- Serves as the primary interface for users to visualize sensor data, monitor incidents and generate reports.
- Provides real-time updates, enhancing user experience with interactive data visualization.

2. Web server

Web application layer:

- Developed using Node.js and Flask, ensuring a scalable and efficient backend to handle requests from the dashboard.
- Acts as a middleware to process and route data between the user. interface and the Firebase database.
- Implements RESTful APIs for seamless communication.

3. Firebase Realtime database

Cloud database service:

- Hosted on Firebase, offering real-time data synchronization and secure storage.
- Stores critical information, such as sensor readings, incident logs and system updates.
- Ensures high availability and fault tolerance to support continuous monitoring.

4. ESP32 microcontroller

Processing unit:

- The ESP32 microcontroller is the core hardware component that integrates various sensors to collect environmental data.
- Features low power consumption and in-built Wi-Fi capability, enabling seamless communication with the IoT network.
- Sensor Module:
- Includes sensors such as DHT11 (for temperature and humidity), MQ-2 (for gas detection), sound and infrared sensors.
- Captures critical parameters related to fire incidents and transmits the data to the microcontroller for processing.

5. IoT network

Wi-Fi connectivity:

- Facilitates communication between the ESP32 microcontroller and the cloud database.
- Ensures continuous data flow for real-time monitoring and incident logging.

Implementation

Hardware prototype of the fire detection system

The image showcases the hardware prototype of an ESP32-based intelligent fire detection system, designed for real-time environmental monitoring. The circuit is assembled on a breadboard, integrating multiple sensors and electronic components essential for detecting fire hazards. The ESP32 microcontroller, positioned centrally, serves as the core processing unit, managing data acquisition, sensor readings and wireless communication.

https://doi.org/10.70844/aest.2025.1.1

Applied Engineering Solutions and Technologies

Key components and connections

The setup includes:

- Flame sensor: Detects infrared radiation emitted by flames and provides an alert if fire is detected. The red LED indicator is active, indicating operational status.
- Gas sensor (MQ-2): Monitors the presence of combustible gases, such as LPG and carbon monoxide, to identify potential leaks.
- Temperature and humidity sensor (DHT22): Measures atmospheric conditions, ensuring detection of abnormal temperature spikes or humidity changes linked to fire outbreaks.
- Sound sensor: Captures sudden loud noises that might indicate fire-related events such as explosions.
- Jumper wires and breadboard: Facilitate secure electrical connections between components, ensuring signal transmission to the ESP32.

Functionality and applications: The system is designed to detect fire hazards, smoke and gas leaks, transmitting alerts *via* wireless networks for remote monitoring and early warning notifications. The prototype demonstrates an affordable and scalable approach to fire safety, making it particularly suitable for micro and small enterprises (MSEs) and industrial environments. With further optimization, this setup can be enhanced by integrating machine learning algorithms for intelligent threat detection and automated emergency response mechanisms (Figure 24-29).

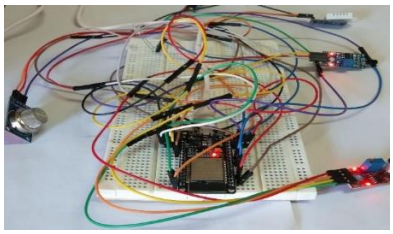

Figure 24: Hardware prototype of the ESP32-Based intelligent fire detection system.

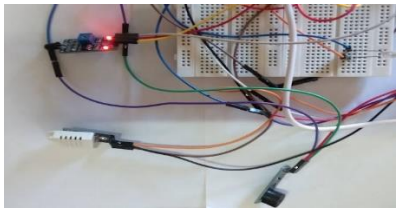

Figure 25: Detailed view of the flame sensor, DHT22 and buzzer in fire detection system prototype.

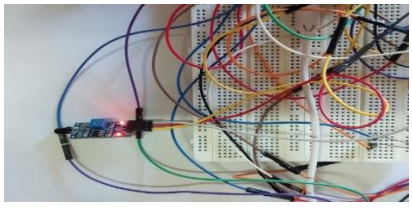

Figure 26: Close-Up of the flame sensor in the fire detection prototype.

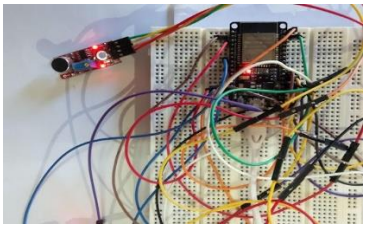

Figure 27: Close-up of the sound detection sensor in the fire detection prototype.

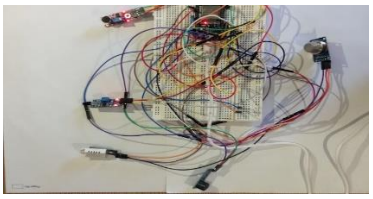

Figure 28: Comprehensive view of the ESP32-based fire detection system prototype on breadboard.

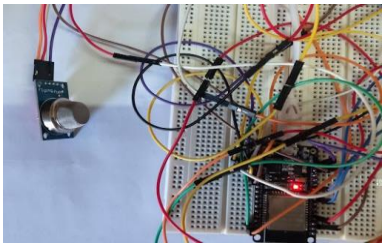

Figure 29: Integration of the MQ-2 gas sensor in the ESP32-based fire detection system.

Fire detection system dashboard

The screenshot displays the web-based monitoring dashboard of the FireX Fire Detection System, designed to provide real-time environmental monitoring. The interface presents key fire hazard indicators, including temperature, humidity, flame status and gas detection, ensuring continuous surveillance of critical safety parameters.

Key data metrics:

- Temperature: The dashboard shows a recorded temperature of 29.2°C, suggesting a stable indoor environment with no abnormal heat spikes.
- Humidity: Measured at 56.1%, indicating standard atmospheric conditions with no sudden fluctuations.
- Flame status and gas detection: Both indicators are marked with green dots, confirming the absence of fire hazards or gas leaks at the time of monitoring.

Graphical analysis and usability: The system features time-series graphs tracking temperature and humidity variations over time. These visual insights allow users to identify trends, predict anomalies and enhance fire prevention strategies. The ability to analyze historical sensor data enables a proactive safety approach rather than reactive responses.

Cloud-connected remote monitoring: The cloud-integrated nature of this dashboard ensures that fire detection data is accessible

remotely, allowing businesses, industrial facilities and small enterprises to monitor fire risks in real time. This feature is particularly beneficial for resource-constrained enterprises, enabling them to take data-driven safety measures with minimal infrastructure costs.

By integrating sensor-based monitoring, historical data visualization and remote access, this system enhances fire safety management, ensuring quick detection and timely preventive action (Figure 30, 31).

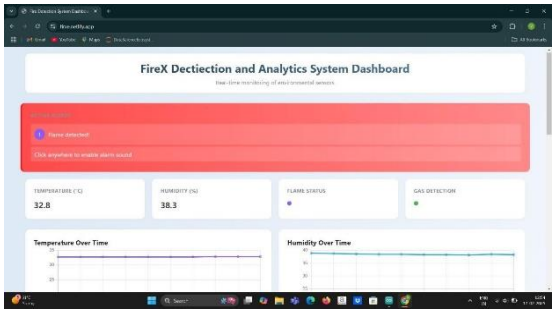

Figure 30: Fire detection system dashboard interface for real-time monitoring.

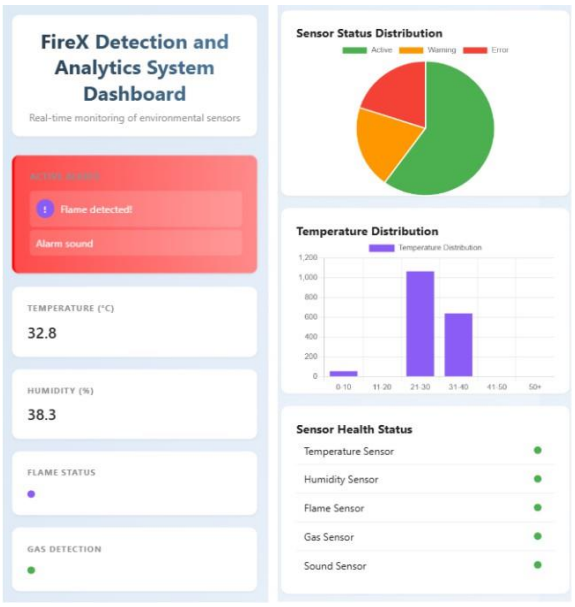

Figure 31: Fire detection system dashboard showing sensor status and environmental trends.

Fire detection system dashboard results

The screenshot displays the web-based monitoring dashboard of the AI-powered surveillance system. The interface provides real-time data visualization for critical environmental parameters, including temperature, humidity, sensor status and system health.

- Temperature and humidity monitoring: The dashboard features time-series graphs tracking temperature and humidity variations over time. The flat trend lines indicate stable environmental conditions, with no significant fluctuations detected.
- Sensor status distribution: A pie chart visually represents the operational status of the sensors, showing that the majority are functioning correctly, with only a small fraction marked as "Error." This allows for quick assessment of sensor reliability.
- Temperature distribution: The bar chart categorizes temperature readings into specific ranges, helping in identifying any abnormal patterns that might indicate fire hazards.
- Sensor health status: The list confirms the proper functioning of essential sensors, including temperature, humidity, flame, gas and sound sensors. The presence of green checkmarks signifies that all sensors are active and reporting accurate data.

This real-time cloud-connected dashboard enhances remote monitoring capabilities, enabling warehouse operators to proactively track environmental conditions, detect potential risks and ensure infrastructure safety (Figure 32).

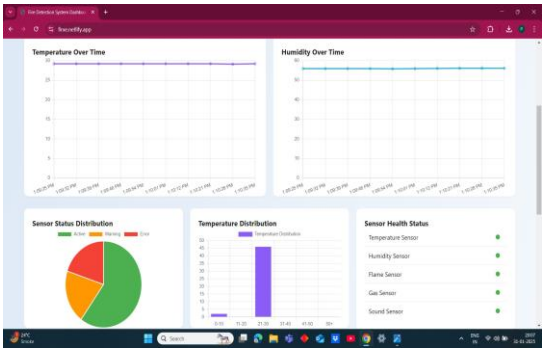

Figure 32: Fire detection system dashboard results showing sensor status and trends.

Sensor history dashboard

The screenshot displays the sensor history dashboard of the ESP32-Based Intelligent Fire Detection System, which monitors critical environmental parameters in real-time. The interface logs multiple sensor readings, including temperature, humidity, flame detection, gas detection and sound detection, providing a structured dataset for fire hazard assessment (Figure 33).

- Temperature and humidity Monitoring: The temperature readings remain consistent at 29.2°C, with minor variations over time. Similarly, humidity levels range between 55.9% and 56.1%, indicating stable environmental conditions. These parameters are essential for detecting sudden changes that might indicate fire hazards, such as rapid temperature spikes or abnormal humidity fluctuations.
- Sensor status analysis: The flame, gas and sound sensors consistently display active status indicators (green dots), confirming that no fire-related anomalies were detected during the recorded period. This uniform status suggests that the system is functioning correctly and continuously monitoring the environment.
- Data logging and export capabilities: The system provides timestamped records for each reading, ensuring a structured and time-sequenced dataset. Additionally, the interface includes options to export data in JSON or CSV formats,

allowing users to perform further analysis or integrate the data into external monitoring systems.

- System reliability and application: This cloud-connected monitoring dashboard enhances remote fire detection capabilities, making it ideal for micro and small enterprises (MSEs) that require cost-effective fire safety solutions.

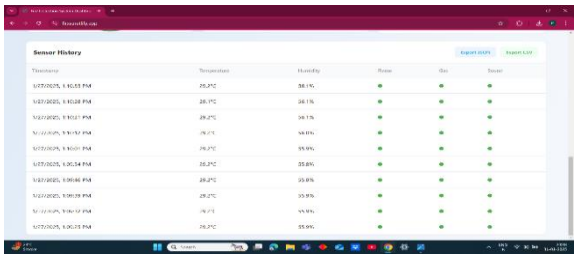

Figure 33: Sensor history dashboard of the ESP32-based fire detection system.

Flame detection ratio

This pie chart illustrates the flame detection ratio achieved by the fire detection model. It highlights that 95.5% of cases are accurately classified as flames, demonstrating the model's reliability in identifying fire hazards. Meanwhile, a minor proportion of 4.5% is classified as "No Flame", representing cases where fire is not detected. This distribution indicates the high precision and robustness of the implemented system in real-world scenarios, ensuring effective fire safety mechanisms (Figure 34).

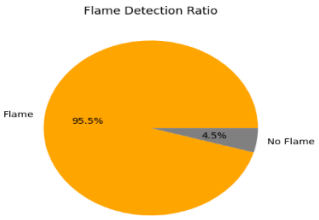

Figure 34: Flame Detection Ratio Pie Chart.

Temperature trends over time

The graph illustrates the variation in temperature readings over time, as measured by the fire detection system. The observed data appears largely stable, with temperatures consistently hovering around a specific range. However, there is a significant anomaly or sudden drop in temperature recorded at approximately 11:00 AM, potentially indicating a sensor error or environmental disturbance. Beyond this point, the system resumes recording normal temperature values. This analysis emphasizes the importance of sensor calibration and validation in ensuring accurate and reliable temperature monitoring for fire detection (Figure 35).

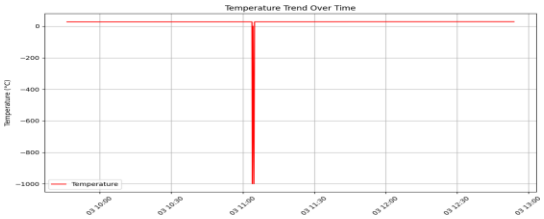

Figure 35: Temperature trends over time graph.

Humidity trends over time

The graph describes humidity variations over time as monitored by the fire detection system. The data shows consistent humidity levels throughout the recorded period, with a sudden and sharp drop around 11:00 AM, likely due to a sensor glitch or external disturbance. Post-anomaly, the readings stabilize and return to normal, indicating reliable system performance after the disruption. This highlights the importance of continuous system validation to ensure data integrity in real-time monitoring applications (Figure 36).

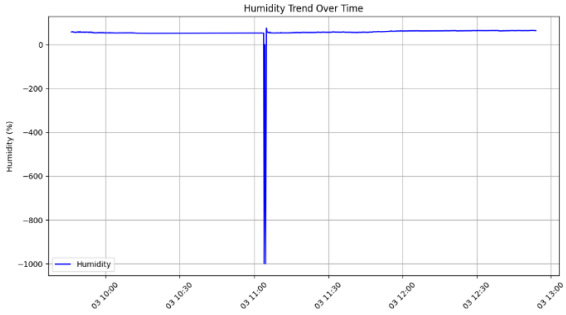

Figure 36. humidity trends over time graph.

Description of flame detection events (Temperature correlation)

The scatter plot illustrates the correlation between flame detection events and temperature over time. Each point represents an instance of flame detection, with temperature values ranging from 26.5°C to 29°C. The data shows sporadic occurrences of flame detection, with slight variations in temperature, suggesting a potential relationship between flame presence and localized temperature increases. This analysis underlines the system's sensitivity to detecting temperature fluctuations associated with flames, critical for early fire detection (Figure 37).

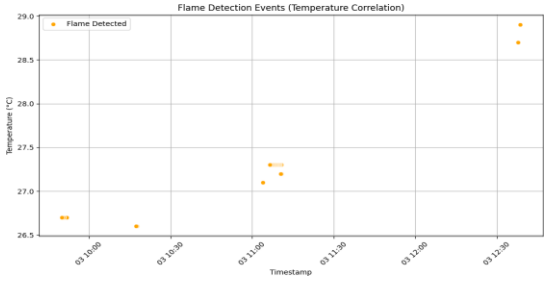

Figure 37: Scatter plot of flame detection events correlated with temperature.

Mean sensor readings

The bar chart describes the mean values of various sensor readings, including temperature, humidity, sound and gas levels. Among the parameters, humidity shows the highest average value, surpassing 50 units, followed by temperature, which averages slightly above 20 units. The readings for sound and gas sensors are negligible or minimal in comparison, indicating their relatively lower activity or sensitivity in this context. This analysis highlights the dominant role of humidity and temperature measurements in the dataset, reflecting their significance in the monitored environment (Figure 38).

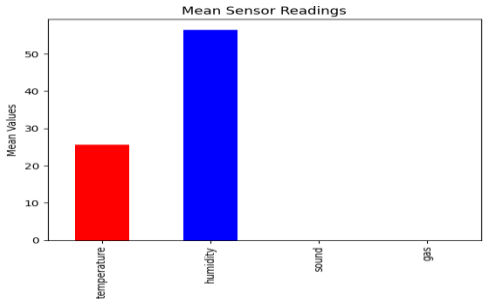

Figure 38: Bar chart of mean sensor readings.

Distribution of temperature and humidity

The graph illustrates the distribution of temperature and humidity sensor readings. Both variables exhibit a high frequency of values concentrated around a narrow range near zero, as indicated by the peak towards the right of the graph. The red curve represents the temperature distribution, while the blue curve denotes humidity. Despite minor variations, the two distributions largely overlap, suggesting a consistent relationship or similar range of values between the two sensors in the observed dataset. Outliers on the negative axis are minimal and may indicate rare anomalies or sensor calibration artifacts (Figure 39).

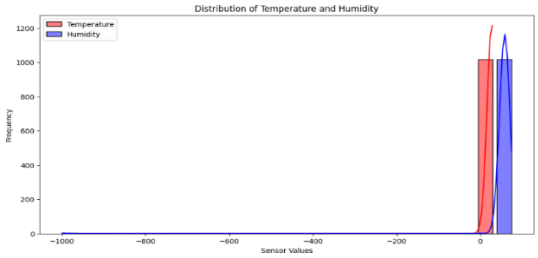

Figure 39: Distribution graph of temperature and humidity readings.

Correlation matrix

The correlation matrix illustrates the relationships between various sensor readings, with values ranging from -1 to 1. A perfect positive correlation (1.00) is observed between temperature and humidity, indicating a strong interdependence. Other features, such as sound, gas and flame, exhibit weak or negligible correlations with temperature and humidity, as their values are close to 0 (e.g., 0.01 or -0.01). The flame sensor shows a perfect self-correlation (1.00), as expected. Overall, the matrix reveals minimal linear relationships among most variables, except for the strong correlation between temperature and humidity (Figure 40).

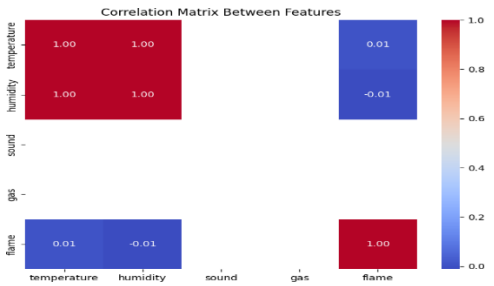

Figure 40. Correlation matrix of sensor readings.

Conclusion

This research paper highlights the successful implementation and integration of a comprehensive IoT-based monitoring and detection system for environmental factors such as temperature, smoke, flame and sound. Through the use of key components, including the ESP32 microcontroller, ESP32-CAM, various sensors (MQ-2, DHT22, KY-026) and programming platforms such as Arduino IDE and VS Code, the system demonstrated its potential as a valuable tool for real-time data collection, monitoring and alarm generation for fire detection and environmental analysis. The project leveraged cutting-edge technologies, such as machine learning, edge computing and cloud-based platforms, to offer both immediate alerts and long-term data analytics capabilities. By enabling remote monitoring *via* a web interface and analyzing sensor data, the system provides efficient, scalable solutions for fire safety, environmental awareness and potentially other areas like industrial automation or smart homes. The process also highlighted the challenges faced in system integration, calibration of sensors and real-time data processing but showed how such challenges could be overcome through careful planning, resource management and iterative testing. Moreover, the success of this project not only proves the feasibility of IoT-based systems in enhancing environmental awareness but also establishes the foundation for future innovations, including the application of AI for enhanced predictive analysis, expansion into more complex environments and the development of user-friendly platforms for mass-market use. The work presented serves as a stepping stone for further research, suggesting areas such as improving the accuracy of sensor data, incorporating additional environmental sensors and incorporating more sophisticated machine learning algorithms for fire prediction and prevention. Through this project, we have shown how intelligent systems using affordable hardware and software platforms can contribute significantly to real-time monitoring and emergency detection, while emphasizing sustainability, scalability and ease of deployment. Future work could explore the integration of advanced technologies, such as edge AI models for decision-making, sensor fusion techniques and the expansion of data-sharing capabilities for global use. In essence, the research not only demonstrates technical execution but also emphasizes its potential for practical applications that mitigate risks, provide proactive solutions and advance smart monitoring systems for societal safety and awareness.

Conflicts of Interest

The authors declare no conflicts of interest in this research.

References

1. Moamen AA, Ecchi M (2021) Hawk-Eye: An AI-powered threat detector for intelligent surveillance cameras. IEEE Access 6: 63283-63293. [Crossref] [GoogleScholar]
2. Remmelzwaal L (2023) An AI-based early fire detection system utilizing HD cameras and real-time image analysis. Artificial Intelligence and Applications 1-6. [Crossref] [GoogleScholar]
3. Shan R (2023) Gas detection using ESP32 and fire alarm project. ResearchGate. [Crossref]
4. Yao S, Ardabili BR, Pazho AD, Noghre GA, Neff C, et al. (2024) From lab to field: Real-world evaluation of an AI-driven smart video solution to enhance community safety. Internet of Things 33: 101716. [Crossref] [GoogleScholar]
5. Komalapati N (2021) Smart fire detection system using IoT in forest areas. IEEE Transactions.

6. Saketh M, Nandal N, Tanwar R, Reddy BP (2023) Intelligent surveillance support system. Discover internet of things 3(1): 9. [Crossref] [GoogleScholar]
7. Siddineni B (2022) IoT-based industrial fire detection using ESP32 and sensors.
8. Benjamin O (2023) Enhancing fire safety for SMEs using AI-based systems.
9. Rahimah RBA, Daud S (2025) IoT-based smart fire detection system using ESP32.
10. Subbarayudu Y, Reddy GV, Bhargavi J, Latha K (2024) An efficient iot-based novel approach for fire detection through Esp 32 microcontroller in forest areas. MATEC Web of Conferences 392. [Crossref] [GoogleScholar]
11. Logesh A, Sudha ARJ (2024) FIRE department alerting system. IARJSET 11.
12. Zell O, Pålsson J, Hernandez-Diaz K, Alonso-Fernandez F, Nilsson F (2022) Image-based fire detection in industrial environments with YOLOv4. arXiv:2212.04786. [Crossref] [GoogleScholar]
13. Ghadani A, Jayakumari S (2020) Innovating fire detection system fire using artificial intelligence by image processing. International Journal of Innovative Technology and Exploring Engineering, 2020.
14. Alam RY (2024) Fire detection in images using AI. [Crossref]
15. Vasconcelos RN, Franca Rocha WJ, Costa DP, Duverger SG, Santana MM, et al. (2024) Fire detection with deep learning: A comprehensive review. Land 13(10): 1696. [Crossref] [GoogleScholar]
16. Diaconu BM (2023) Recent advances and emerging directions in fire detection systems based on machine learning algorithms. Fire 6(11): 441. [Crossref] [GoogleScholar]
17. Surabhi KS, Babu A (2024) Fire Detection Using AI. International Journal for Multidisciplinary Research 6(5): 1-8. [Crossref] [GoogleScholar]
18. Yang W, Wu Y, Chow SK (2024) Deep learning method for real-time fire detection system for urban fire monitoring and control. International Journal of Computational Intelligence Systems 17(1): 216. [Crossref] [GoogleScholar]
19. Medewar AG, Sawarkar AD, Kshirsagar UV (2024) A review on fire and smoke detection with intelligent control for enhanced safety using Machine Learning (ML) and Internet of Things (IoT). Cureus Journals 1(1): 1-11. [Crossref] [GoogleScholar]
20. Sawant S, Kumbhar S, Chauhan B, Chaudhari G, Thakar P (2024) Integrated fire detection system using ML and IOT. International Journal for Research in Applied Science & Engineering Technology. [Crossref]