DESIGN OF A FIRE DETECTION SYSTEM USING THE ARDUINO UNO PLATFORM

M.F. Preduș, *Babeș-Bolyai University, Reșița, ROMANIA*
C. Hațiegan*(corresponding author), *Babeș-Bolyai University, Reșița, ROMANIA*
M.D. Stroia, *Babes-Bolyai University, Faculty of Engineering, Resita, ROMANIA*
C. Popescu, *"Constantin Brâncuşi" University of Târgu Jiu, ROMANIA*
Lenuța Cîndea, Babeș-Bolyai University of Cluj-Napoca, ROMÂNIA

ABSTRACT: Fires represent one of the most critical threats across various environments, from residential and industrial facilities to public institutions and critical infrastructures. In this context, it is imperative to develop technological solutions capable of preventing fire outbreaks, continuously monitoring the environment, and responding promptly in hazardous situations. The efficiency of alarm and prevention systems can often determine whether an event remains a minor incident or escalates into a large-scale disaster. The main motivation of this study is to contribute to the advancement of automated fire detection systems by introducing additional advantages for risk-area protection through the implementation of a sensor-based solution featuring real-time user notification. This paper aims to design and implement an integrated fire detection system using cost-effective and reliable technologies, including flame and smoke sensors, a microcontroller-based control unit, and a data monitoring and management platform such as Node-RED.

KEY WORDS: fire detection, fire suppression, system testing.

1. INTRODUCTION

The history of fire suppression systems reflects continuous technological progress aimed at protecting human life and property from the destructive effects of fires [1]. Major advances in electrical and electronic engineering led to the development of automatic fire detection systems [2], introducing the first smoke and heat detectors capable of signaling environmental changes. These early systems used electrical circuits to detect smoke or temperature increases and transmitted alerts to control centers or local authorities.

Simultaneously, the first automatic suppression systems emerged, employing electric sprinklers or chemical agents to prevent fire propagation. Modern detection technologies have since become more accurate and efficient, integrating with alarms, mobile notifications, and other communication systems to ensure rapid response.

Today's fire suppression systems are complex networks composed of smoke and flame detectors, alarms, sprinklers, water pumps, and chemical extinguishing agents [3]. They form part of continuously monitored infrastructures that enable fast and coordinated intervention. Over time, these systems have evolved from mechanical solutions to intelligent platforms capable of interacting with automation and Internet-based technologies [3].

An intelligent detection system integrates sensors that identify key fire indicators—smoke, temperature, and flames—while centralized controllers analyze data in real time to trigger alerts or suppression mechanisms [3,4]. Advances in sensor technology have introduced optical detectors that can identify minimal smoke concentrations and reduce false alarms through signal interpretation algorithms.

Modern systems are increasingly interconnected, combining sensors, data-

processing modules, and remote management capabilities through mobile or web interfaces, providing real-time status updates and alerts [4,5].

2. EXPERIMENTAL FIRE DETECTION SYSTEM

By comparing existing market solutions, it becomes possible to better understand how to integrate and adapt innovative technologies to the specific requirements of the environments in which they are implemented. The developed system combines state-of-the-art detection technologies with automated suppression and alert functions. Analyzing similar systems is therefore essential for assessing their performance and efficiency in real-world conditions.

A major issue identified in existing systems is the precision and response speed during early-stage fire detection, which can be affected by external factors such as dust, humidity, or other environmental conditions. These influences may lead to false alarms or, more critically, to missed fire detections, endangering occupant safety [6,7,8]. Moreover, sensor calibration is often complex and difficult to adjust for location-specific conditions. The system proposed in this paper introduces an automatic sensor calibration strategy that continuously adjusts sensitivity based on ambient conditions. The implemented algorithm filters false signals, ensuring higher detection accuracy for real fire indicators.

The performance of existing systems also depends on the communication between system components and external responses such as email notifications or the activation of visual and audible alarms [9]. Inefficient or delayed communication between sensors and control units may cause alarm delays and hinder emergency response [9,10]. In addition, most current systems lack flexibility, being designed for specific applications and difficult to reconfigure or expand to meet new requirements. Implementing new functionalities can therefore be complex and costly [10,11,12].

2.1. Proposed System Design

This work proposes an innovative Arduino-based fire prevention system that integrates smoke and infrared (IR) flame sensors to monitor and respond in real time to potential fire hazards. The system alerts users through a web interface and automatically sends email notifications when a fire is detected, ensuring a prompt and effective response.

The detection subsystem relies on two main sensors: a smoke sensor and an IR flame sensor. The smoke sensor, typically analog, measures the concentration of airborne particles and outputs a proportional analog signal. The IR flame sensor detects infrared radiation emitted by flames, providing a digital signal that allows the system to identify active fires or flames instantly. Both sensors are connected to the Arduino board, which interprets their inputs to determine the level of fire risk.

When both sensors indicate danger, the activation subsystem triggers the water pump through a relay. Arduino sends a control signal to close the relay circuit, starting the pump to extinguish the fire. This subsystem, together with a buzzer alarm, provides immediate intervention and user notification.

The monitoring and alert subsystem, implemented via Node-RED, complements the hardware by processing and visualizing sensor data through a web dashboard [2,4]. It allows users to observe system status in real time. When an anomaly is detected, Node-RED automatically sends email notifications to designated personnel. Users can check smoke and flame sensor readings and review the system's automated responses at any time [4].

2.2. Program Logic Overview

The Arduino-based fire prevention system is organized into three main subsystems, each performing a specific function while maintaining interconnection for rapid and reliable fire response:

Detection Subsystem: Includes the smoke and IR flame sensors connected to the Arduino board. The smoke sensor generates an analog signal representing airborne particle density, while the IR sensor outputs a digital signal

corresponding to infrared radiation from flames.

Actuation Subsystem: Activates when one or both sensors detect a threat. It controls the water pump and audible alarm via a relay module. When triggered, the relay powers the pump, enabling fire suppression.

Monitoring and Alert Subsystem: Manages sensor data and sends alerts to users through Node-RED. The platform collects and processes data from Arduino, displays sensor values on a web dashboard, and automatically emails alerts upon fire detection [9,10].

The development board serves as the core of the system, coordinating communication among all subsystems and ensuring synchronized operation [8].

This integrated architecture enables the detection, suppression, and notification processes to work together efficiently, ensuring both rapid fire response and remote supervision.

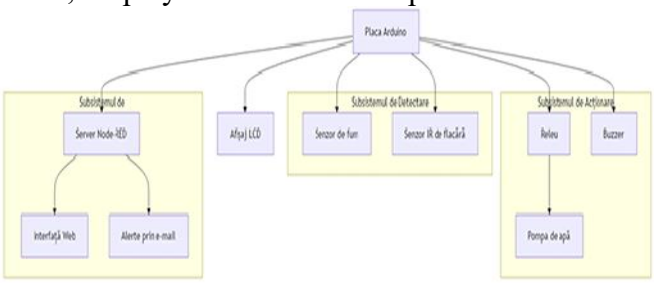

Figure 1. Block diagram

Development boards are equipped with a microcontroller and full software support. The Arduino Nano uses the ATmega328P, a widely adopted and easy-to-program microcontroller with an extensive library ecosystem and numerous development resources. It allows seamless integration of serial communication functions for interfacing with the Node-RED server [6,7,8].

The ATmega328P features an 8-bit processor operating at 16 MHz—sufficient for running small to medium-sized programs. The board provides 22 input/output (I/O) pins, including 8 analog inputs for reading analog sensor signals and 14 digital pins ideal for controlling external devices, such as triggering a relay or activating a buzzer alarm. Each hardware component is interconnected to create a functional and efficient system (Fig. 2). Proper power distribution to all components is essential for stable operation. The circuit employs two main power sources, each serving a specific purpose in supplying energy to different parts of the system. The first source is a USB cable that powers the Arduino board and other electronic components, while the second is a 9V battery dedicated to driving the water pump in case of a fire hazard.

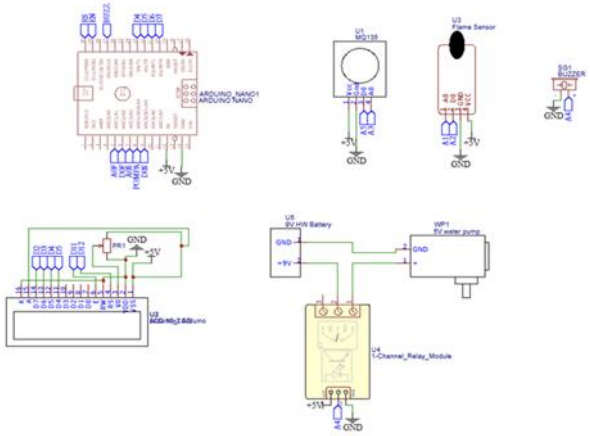

Figure 2. System electrical scheme

The Arduino board is powered via a USB cable connected to a laptop or power adapter. This cable provides a 5V supply used to energize the Arduino and its connected components.

The USB connection also enables data transfer between the Arduino and the laptop, allowing sensor data monitoring and management through the Node-RED server. The Arduino Nano consumes approximately 0.05 W per hour (50 mW) [4,12].

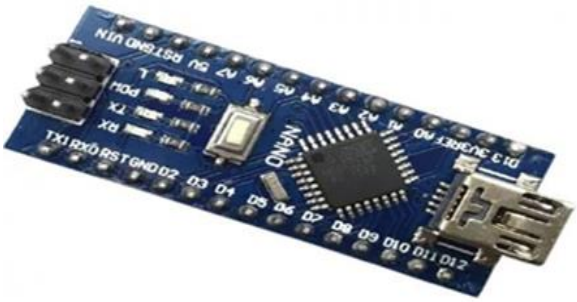

Figure 3. Development board

For testing purposes, the pump is powered by a 9V source and has a significantly higher energy demand compared to other components. The pump's power consumption depends on its type and activation duration; on average, it draws about 5 W per hour (5000 mW) when operating. It is important to note that the pump activates only when sensors detect danger, meaning energy consumption is not continuous throughout the day.

The total energy consumption of the experimental system is approximately 6.97 W, a factor that must be considered when sizing power sources—particularly when using batteries or external power supplies.

2.3. Software development

The software components include the programming of the Arduino board to collect, process, and respond to sensor signals, as well as the implementation of a Node-RED server, which enables information display through a web interface and automatic email notifications to responsible personnel.

The first step in developing the code for the fire prevention system was configuring an Integrated Development Environment (IDE) that provides all the necessary tools for writing, testing, and uploading the program to the Arduino Nano board. PlatformIO was selected for this purpose—a modern and highly flexible environment that operates as an extension within the Visual Studio Code (VS Code) editor.

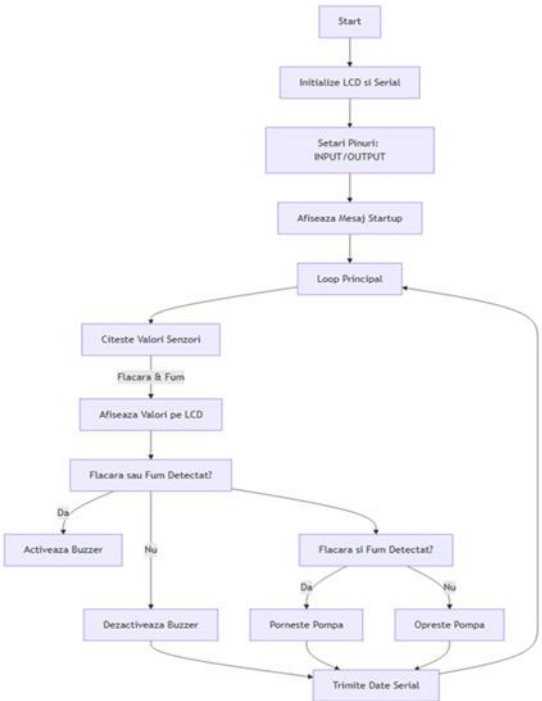

Figure 4. Software logical diagram

The program developed for the fire prevention system was designed in a modular manner, divided into several distinct logical sequences. This approach enabled a clear and efficient structure of the code, ensuring that each part corresponded to a specific system functionality (Fig. 4).

In the first section of the code, the fundamental components of the fire prevention system are initialized, preparing the program for proper operation. The first important step is the inclusion of the LiquidCrystal library, essential for controlling the LCD display. This library allows the Arduino to interact with the LCD, providing the functionality needed to visualize useful information such as sensor readings and system status messages. Without this library, communication with the LCD would be impossible, and no user interface could be displayed (Fig. 5.a).

```
#include <LiquidCrystal.h>

LiquidCrystal afisajLCD(12, 11, 5, 4, 3, 2);

const int pinFlacaraAnalogic = A1;
const int pinFlacaraDigital = A2;
const int pinFumAnalogic = A3;
const int pinFumDigital = A5;
const int pinAlarma = 9;
const int pinPompa = A4;
```

a) Defining variables

```
void setup() {
  lcd.begin(16, 2);
  lcd.print("Sistem Incendiu");
  lcd.setCursor(0, 1);
  lcd.print("Activ");
  Serial.begin(9600);

  pinMode(pinFlacaraDigital, INPUT);
  pinMode(pinFumDigital, INPUT);
  pinMode(pinAlarma, OUTPUT);
  pinMode(pinPompa, OUTPUT);
```

b) System initialization

```
  digitalWrite(pinAlarma, LOW);
  digitalWrite(pinPompa, LOW);

  delay(2000);
  lcd.clear();
}
```

c) Controlling the outcome

Figure 5. Software coding sequences 1

Next, variables are defined for the pins to which the sensors and other system components are connected—an essential step in configuring the hardware. For example, the variable pinFlacaraAnalogic is defined as A1, and pinFlacaraDigital as A2. These correspond to the flame sensor, which provides both analog and digital signals. The analog signal can be used to evaluate light or flame intensity, while the digital signal indicates simply the presence or absence of a flame, allowing rapid identification of a potential hazard. Similarly, for the smoke sensor, pinFumAnalogic is defined as A3 and pinFumDigital as A5. These pins allow reading of both analog and digital signals from the smoke sensor, which alerts the system in case smoke is detected—another indicator of a possible fire.

The program also initializes pins for output components such as pinAlarma and pinPompa. pinAlarma is connected to the system buzzer, which emits a loud audible signal when hazardous conditions are detected. When both the smoke and flame sensors detect danger, the pump is activated to extinguish the fire—an essential process to prevent fire spread.

In the second part of the code, the setup() function plays a crucial role in initializing the system, ensuring that all hardware components are properly configured to operate within the program. The first step within setup() is initializing the LCD screen using the command lcd.begin(16, 2), which configures the display to show 16 characters across two lines. This enables compact, clear, and easy-to-read display of relevant information. After configuration, a welcome message—lcd.print("Sistem Incendiu")—appears, indicating that the system is active and ready to begin monitoring. The next step sets the cursor to the second line using lcd.setCursor(0, 1), positioning it at the left corner of the display to show an additional status message. Here, the message "Activ" appears, confirming that the system is operational.

Therefore, the second part of the code plays a vital role in establishing the initial configuration of the entire system. Without this initialization, proper functionality would not be possible. The commands for the LCD screen, serial communication, and pin settings are all fundamental for building a stable operational foundation. Once each component is properly configured, the fire prevention system can accurately detect and respond to potential fire hazards.

In the third part of the code, key actions are implemented to manage alarms and the pump,

ensuring correct system operation during emergencies. This section includes control commands for output components such as pinAlarma and pinPompa, which activate the buzzer and the extinguishing pump. Initially, digitalWrite(pinAlarma, LOW) is used to deactivate the buzzer, signaling that no alert is active. If no flame or smoke is detected, the buzzer remains silent, preventing unnecessary noise or confusion. Similarly, digitalWrite(pinPompa, LOW) deactivates the water pump, ensuring it remains off in the absence of danger.

```
void loop() {
  int valoareFlacaraAnalogic = analogRead(pinFlacaraAnalogic);
  int valoareFumAnalogic = analogRead(pinFumAnalogic);
  int valoareFlacaraDigital = !digitalRead(pinFlacaraDigital);
  int valoareFumDigital = !digitalRead(pinFumDigital);
```

a) Monitoring variables

```
lcd.setCursor(0, 0);
lcd.print("Flacara:");
lcd.print(valoareFlacaraAnalogic);
if (valoareFlacaraDigital == HIGH) {
  lcd.print("!");
} else {
  lcd.print(" ");
}
```

b) Fire signaling

```
lcd.setCursor(0, 1);
lcd.print("Fum:");
lcd.print(valoareFumAnalogic);
if (valoareFumDigital == HIGH) {
  lcd.print("!");
} else {
  lcd.print(" ");
}
```

c) Smoke signaling

Figure 6. Software coding sequences 2

The main program logic is implemented within the loop() function, which runs continuously during system operation. This is where sensor data are read and appropriate actions are taken based on the detected conditions. The purpose of this section is to continuously monitor the environment, evaluate readings from the flame and smoke sensors, and respond accordingly by activating or deactivating the alarm and pump.

The first step in loop() involves reading analog values from the sensors using analogRead(pinFlacaraAnalogic) and analogRead(pinFumAnalogic). These commands capture analog signals and convert them into numerical values stored in variables such as valoareFlacaraAnalogic and valoareFumAnalogic. These values are proportional to the intensity of the signals, reflecting smoke concentration or flame presence in the monitored environment.

Next, digital signals from the flame and smoke sensors are read using digitalRead(pinFlacaraDigital) and digitalRead(pinFumDigital). The digital outputs are used to determine whether a hazard is detected. Their logic is inverted—meaning a digital HIGH indicates danger—using the operator !, which flips the logical state of the read value.

The LCD interface provides real-time feedback to the user. The command lcd.setCursor(0, 0) sets the cursor at the top-left corner, followed by lcd.print("Flacara:"), displaying the flame label. The subsequent command lcd.print(valoareFlacaraAnalogic) shows the corresponding flame intensity value. When if (valoareFlacaraDigital == HIGH) evaluates true, an exclamation mark "!" is printed to indicate detected danger; otherwise, a blank space is displayed, signaling normal conditions.

The next section mirrors the previous one but focuses on the smoke readings. The command lcd.setCursor(0, 1) moves the cursor to the second LCD line, where lcd.print("Fum:") displays the smoke label and the corresponding analog value follows.

```
if (valoareFlacaraDigital == HIGH || valoareFumDigital == HIGH) {
  digitalWrite(pinAlarma, HIGH);
} else {
  digitalWrite(pinAlarma, LOW);
}
```

a) Alarm activation

```
if (valoareFlacaraDigital == HIGH && valoareFumDigital == HIGH) {
  digitalWrite(pinPompa, HIGH);
} else {
  digitalWrite(pinPompa, LOW);
}
```

b) Pump activation

```
Serial.println(String(valoareFlacaraAnalogic) + "$" +
               String(valoareFumAnalogic) + "$" +
               String(valoareFlacaraDigital) + "$" +
               String(valoareFumDigital));

delay(500);
}
```

c) Printing results

Figure 7. Software coding sequences 3

The system's logic for triggering alarms and activating the pump is then applied. If either digital sensor (flame or smoke) detects danger, the condition (valoareFlacaraDigital == HIGH || valoareFumDigital == HIGH) evaluates true, and digitalWrite(pinAlarma, HIGH) activates the buzzer to alert users. If no danger is detected, the buzzer is deactivated with digitalWrite(pinAlarma, LOW).

To enable automatic intervention, the system activates the water pump when both sensors detect danger simultaneously. The condition (valoareFlacaraDigital == HIGH && valoareFumDigital == HIGH) ensures the

pump operates only under significant fire risk, allowing the system to take active measures to mitigate or extinguish flames.
Sensor data are also transmitted externally for real-time monitoring and further analysis using the Serial.println() function. This function sends the sensor readings to the microcontroller's serial port, where they can be viewed on a serial terminal or processed by an external system. The expression
String(valoareFlacaraAnalogic) + "$" + String(valoareFumAnalogic) + "$" + String(valoareFlacaraDigital) + "$" + String(valoareFumDigital)
constructs a formatted string containing all sensor values, separated by the "$" delimiter. This structured format facilitates organized data transfer and straightforward integration with monitoring systems such as Node-RED.
Continuous transmission ensures uninterrupted tracking of environmental conditions.

3. TESTING SYSTEM FUNCTIONALITIES

Testing represents a critical stage in the development of any engineering system, ensuring correct and reliable operation under real-world conditions. The designed prototype (figure 8) was evaluated in terms of both sensor performance—flame and smoke detection—and software functionality, particularly the Node-RED data processing flows.

3.1. Testing the Flame and Smoke Sensors

During initial tests, two main issues were observed: the smoke sensor exhibited delayed reactions to smoke presence, while the flame sensor occasionally produced false positives in the absence of visible flames. These anomalies were primarily caused by signal instability due to electromagnetic interference (EMI) affecting the test circuit. The interference led to fluctuations in input signals, generating erratic digital readings.
To mitigate these effects, a signal filter was implemented on the analog sensor inputs using a filtering capacitor, and the sensitivity thresholds of the digital sensors were recalibrated to react only to significant input variations. Additionally, the test circuit was physically isolated from EMI sources through shielded connection cables and protective casing for both sensors and circuitry.
Following these improvements, sensor readings became stable and consistent. The sensors responded promptly to actual flame or smoke presence, and prolonged monitoring confirmed reliable performance under normal operating conditions. The system consistently detected and reported environmental changes, accurately signaling potential fire sources.

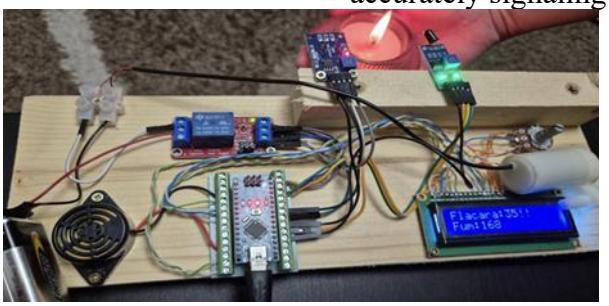

Figure 8. Testing the flame and smoke sensors

3.2. Testing the Alarm and Display Modules

Testing the LCD display, buzzer, and water pump was an essential step in validating the operational functionality of the fire detection and response system (Fig. 14). This stage aimed not only to verify LCD message display but also to assess the system's reaction during simulated fire scenarios.

The system was configured to trigger the buzzer alarm when either sensor detected flame or smoke, and to activate the pump when both sensors indicated danger simultaneously. The LCD correctly displayed messages, though initial tests revealed issues with message refresh rate—text either disappeared too quickly or failed to update in real time. This was resolved by synchronizing the LCD

refresh routine, introducing a timed "clear" function, and minimizing display delays to update only the necessary values at fixed intervals.

Further measurements identified slight activation delays in the buzzer and pump. These were traced to asynchronous digital signal reads occurring during rapid environmental changes. To address this, a software-level signal filtering mechanism was implemented to ensure accurate interpretation of sensor outputs and to prevent false triggers caused by transient noise or minor signal fluctuations.

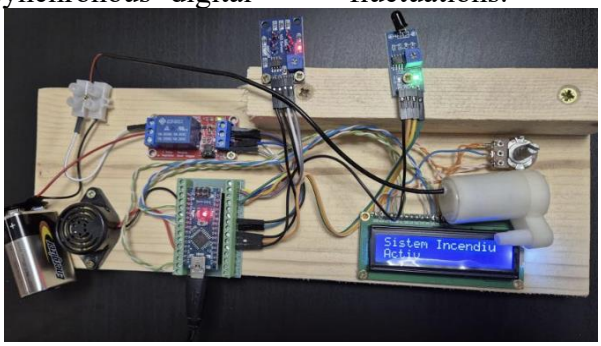

Figure 9. testing the alarm and display modules

Under final testing conditions, when only one sensor detected danger, the buzzer emitted a continuous sound, indicating the need for immediate attention. When both sensors were simultaneously active, the system automatically engaged the water pump to suppress potential flames. The LCD display confirmed these actions by presenting both sensor readings and real-time system status updates.

4. CONCLUSION

The primary objective of this work was the development of an integrated monitoring and control system for fire detection, using accessible and efficient technologies such as the Node-RED platform, flame and smoke sensors, and associated hardware components. The system was designed to enable real-time safety monitoring, issue alerts in critical situations, and ensure prompt response actions such as activating a water pump or an alarm system.

Throughout the research, several significant objectives were achieved, each contributing to the design of a functional and reliable prototype. The first goal was to develop a platform capable of integrating flame and smoke sensors, processing the collected data, and transmitting it in an interpretable format—both through a graphical interface and automated email notifications. Another key accomplishment was the integration of the Node-RED platform, which provided a flexible and efficient data flow architecture, facilitating sensor management and automated control operations.

The implementation of the email alerting mechanism ensured rapid and effective notification of the user in emergency scenarios, while continuous testing allowed optimization of both hardware and software components.

Performance testing proved essential in identifying potential weaknesses and implementing corrective measures to enhance overall reliability. In conclusion, the study successfully developed a fully functional fire detection and response prototype that leverages flame and smoke sensors within an automated monitoring framework. The system can detect hazards, alert users through multiple channels, and initiate protective actions, thus contributing to the prevention of potential fire-related disasters.

REFERENCES

[1] Johnson, R., & Sorenson, R., Automatic Fire Detection Systems, 2010

[2] Meyer, H., Designing Fire Safety Systems: Principles, Practices and Procedures, Routledge, London, 2014

[3] Linguo, Li, Hui Liu, Shujing, Li, Intelligent Fire Monitoring System Based on the Information

Fusion Algorithm, American Scientific Publishers, 2016
[4] Lin, Q., & Wang, L., Real-Time Fire Detection Using Machine Learning and Smart Sensors. Sensors and Actuators A: Physical, 2020
[5] Crosbie, T., & Watson, T., Innovative Fire Detection Methods for Smart Buildings. Journal of Fire Safety Engineering, 41(2), 2019
[6] Fernandez, A., & Roque, R., Smart Building Technologies for Fire Safety: A Review. Journal of Smart Buildings, 5(1), 2018
[7] Gilbert, M., & Roberts, S., Intelligent Fire Detection and Alarm Systems. IEEE Transactions on Industrial Electronics, 61(12), 2014
[8] Cai, S., & Wang, Y., Development and Application of Fire Detection and Alarm Systems, International Journal of Fire Science and Technology, 2014
[9] Bashir, M., & Awan, H., The Role of Internet of Things in Fire Detection and Prevention Systems, IEEE Sensors Journal, 19(2), 2019
[10] Mansfield, K., & Stone, R., Advanced Methods in Fire Detection and Safety, Journal of Safety Research, 43(5), 2012
[11] Herzog, A., Fire Detection Systems and IoT-based Approaches, Journal of Fire Safety Science, 30(2), 2017
[12] Mullins, R., Fire Safety Engineering: Principles and Practice, CRC Press, 2014