Design of a Cost-Effective Wireless Forest Fire Detection System using STM32 and Zigbee Technology

1st Ahana Khan (2410110030)
Dept. of Electrical Engineering
SHIV NADAR University, Delhi NCR
Email:ak168@snu.edu.in

2nd J Meher Krithi (2410110200)
Dept. of Electrical Engineering
SHIV NADAR University, Delhi NCR
Email: mj427@snu.edu

3rd Kshitij Khandelwal (2410110179)
Dept. of Electrical Engineering
SHIV NADAR University, Delhi NCR
Email: kk931@snu.edu

4th Purabee Mahapatra (2410110254)
Dept. of Electrical Engineering
SHIV NADAR University, Delhi NCR
Email: pm639@snu.edu

5th Rishabh Joshi (2410110269)
Dept. of Electrical Engineering
SHIV NADAR University, Delhi NCR
Email: rj910@snu.edu

6th Rajdeep Hiteshkumar Mistry (2410110260)
Dept. of Electrical Engineering
SHIV NADAR University, Delhi NCR
Email: rm395@snu.edu

7th Saahil Gupta (2410110282)
Dept. of Electrical Engineering
SHIV NADAR University, Delhi NCR
Email: sg790@snu.edu

Abstract - This paper details the development of an **autonomous and cost-effective Internet of Things (IoT) system** designed for the early detection and immediate alerting of forest fire incidents. The system is engineered to function as a crucial node within a **Wireless Sensor Network (WSN)**, leveraging low-power components optimized for remote deployment. The architecture integrates a suite of low-cost sensors to continuously monitor critical environmental conditions conducive to combustion. The primary sensing unit employs the **MQ-2 gas sensor** for the detection of smoke and flammable gases, while the **DHT22 temperature sensor** provides real-time ambient heat monitoring. Data integrity and rapid processing are handled by an **STM32 microcontroller**, selected for its high-performance, 32-bit architecture suitable for concurrent sensor polling and complex control logic.

To mitigate the critical issue of false alarms, a robust **dual-verification logic** is implemented: a fire is confirmed only when both the temperature reading surpasses a predefined safety threshold and the smoke/gas concentration exceeds its danger limit. This layered sensing approach significantly enhances system reliability. Upon confirmed detection, the system initiates a rapid, two-tier alert protocol: locally, immediate visual and auditory warnings are provided by activating a **buzzer and an LED**; concurrently, the critical alert status is transmitted wirelessly using **Zigbee modules**. The choice of Zigbee ensures **reliable, low-power communication** to a central receiver unit, circumventing the need for infrastructure-dependent networks. At the monitoring station, the receiver node displays the status and raw sensor data on a **20x4 LCD**, guaranteeing the rapid identification of the threat and minimizing emergency response time. The resulting integrated solution offers a scalable and effective strategy for resource protection against wildfires.

Index Terms - STM32, Forest Fire Detection, Zigbee, Wireless Sensor Network (WSN), MQ-2, DH22.

INTRODUCTION

I. Forest fires represent one of the most critical environmental challenges globally, leading to catastrophic **loss of biodiversity, significant economic damage, and increased atmospheric carbon emissions**. The rapid onset and spread of wildfires necessitate the development of highly reliable and low-latency detection systems. Traditional methods currently deployed for forest surveillance, such as **satellite monitoring** and fixed **human patrolling**, often suffer from significant limitations. Satellite systems typically have a poor temporal resolution and can be obscured by weather conditions, leading to substantial detection latency. Conversely, human patrolling is labour-intensive, costly, and prone to human error, making it ineffective for monitoring vast, inaccessible forest tracts.

II. This project addresses these shortcomings by proposing a robust, localized, and energy-efficient **Wireless Sensor Network (WSN) node** designed for the continuous and autonomous monitoring of hazardous environmental parameters. The core philosophy is **early-stage detection**, ensuring that small blazes are identified before they escalate into uncontrollable wildfires.

III. The proposed system significantly improves upon simpler, commercially available or academic prototype systems (e.g., those using basic microcontrollers and Wi-Fi communication) through the incorporation of three key technological features:

IV. **STM32 Processing:** The system utilizes an **STM32 micro-controller**, leveraging its 32-bit **ARM Cortex** architecture. This choice is critical for ensuring the computational capacity needed for **efficient real-time data processing**, interrupt handling, and the concurrent management of multiple sensors and communication protocols without experiencing delays that are common in less powerful micro-controllers.

V. **Dual-Verification Logic:** To achieve high accuracy and drastically minimize false positives—a frequent issue with single-sensor setups—the system employs a **dual-verification logic**. Fire threats are validated only when both the **MQ-2 gas sensor** (detecting smoke/combustible gases) and the **DHT22 temperature sensor** (detecting thermal anomalies) exceed their respective predefined safety thresholds.

VI. **Zigbee Communication:** For reliable data transmission in a dense forest environment often lacking cellular infrastructure, the project incorporates **Zigbee modules**. Zigbee is preferred over Wi-Fi for its **low power consumption**, resilience in creating self-healing mesh networks, and extended operational range, providing a robust communication link between the remote forest node and the central monitoring station.

VII. The remainder of this paper is structured as follows: Section II details the overall **System Architecture**, describing the Transmitter and Receiver Nodes. Section III outlines the specific **Hardware Implementation** and component selection. Section IV presents the **Methodology and Logic** for the fire detection algorithm. Section V discusses the **Results and Performance**, and finally, Section VI provides the **Conclusion and Future Scope** of the project.

SYSTEM ARCHITECTURE

The proposed Forest Fire Detection System operates as a two-part interconnected network, partitioned into the remote sensing unit and the local monitoring station. This architecture maximizes coverage and minimizes response time by compartmentalizing data acquisition and alerting.

The system is fundamentally divided into two primary, wireless linked units: the **Transmitter Node (Forest Unit)** and the **Receiver Node (Monitoring Station)**.

A. Transmitter Node (Forest Unit)

The Transmitter Node is the autonomous, battery-operated module deployed directly within the forest environment. Its primary responsibilities are continuous environmental polling, decision-making, and initiating localized alerts.

1. **Core Processing:** The central element is the **STM32 micro-controller**. It executes a low-power firmware loop designed to minimize energy consumption while ensuring continuous sensor data acquisition. The STM32 utilizes its Analog-to-Digital Converter (ADC) channels for the MQ-2 sensor and a dedicated GPIO pin for the DHT22 data line.
2. **Data Acquisition and Logic:** The STM32 constantly polls data and compares the digitized sensor values against predefined safety thresholds. This is where the **dual-verification logic** is executed: a confirmed fire requires simultaneous readings above both the temperature and gas limits.
3. **Local Alerting:** If the fire condition is met, the STM32 immediately drives two output peripherals: the **Buzzer** is activated to provide an audible alarm, and the **LED** is illuminated to provide a visual warning to anyone nearby.
4. **Wireless Transmission:** Upon threat confirmation, the STM32 formats the alert data packet (e.g., status flag, current sensor values) and serializes it to the integrated **Zigbee transmitter module**. This module broadcasts the "FIRE DETECTED" packet across the WSN, serving as the essential communication link back to the central base.

B. Receiver Node (Monitoring Station)

The Receiver Node is situated at a secure, accessible location (e.g., a ranger station or base camp) and serves as the human-machine interface (HMI) for the entire system.

1. **Wireless Reception:** This unit consists of a matching **Zigbee receiver module** operating on the same channel and network ID as the transmitter. It constantly listens for incoming data packets.
2. **Data Parsing and Display:** The received Zigbee data, which is typically streamed serially, is ingested and parsed by its own micro-controller (which can also be an STM32 or a basic controller for simplicity). If the packet contains the critical fire status flag, the microcontroller immediately triggers the display.
3. **Visual Alert and Information:** The parsed data is output to the **20x4 LCD**. The display is programmed to clearly show the "FIRE DETECTED" alert in large text, ensuring immediate visibility. It can also be configured to display auxiliary information, such as

the current timestamp or the last received raw sensor values, allowing human observers to verify the severity and urgency of the threat and take immediate action.

HARDWARE IMPLEMENTATION

The system's functionality is realized through the careful selection and integration of specialized hardware components. The design prioritizes **efficiency, low-power operation, and reliability** to ensure autonomous operation in a remote forest environment.

A. Microcontroller (STM32F303RET6)

The STM32F303RET6 micro-controller (typically deployed on the Nucleo development board) serves as the core processing and control unit for the entire system. Its selection is based on its superior technical specifications for high-reliability, real-time control applications compared to standard 8-bit controllers:

1. 32-bit Architecture (Cortex-M4 with FPU): The integrated ARM Cortex-M4 core provides significantly faster instruction execution, which is critical for the real-time data processing loop and quickly executing the dual-verification logic. Crucially, it includes a dedicated Floating-Point Unit (FPU), enabling fast and precise sensor data calibration and filtering without compromising system latency.
2. Rich Peripheral Set: The STM32F303RET6 features a generous peripheral count, including up to 5 Hardware Universal Asynchronous Receiver-Transmitters (UARTs). This allows for simultaneous, non-blocking communication with the Zigbee module (on one UART) and easily accommodates potential future peripherals (such as a GPS or logging device) on separate ports. This capability eliminates the reliance on slower, inefficient software-based communication methods.
3. High-Resolution ADC: The micro-controller is equipped with high-speed, high-resolution 12-bit Analog-to-Digital Converters (ADCs). This precision is essential for reliably sampling the analog voltage output from the MQ-2 gas sensor, ensuring the concentration level is converted into accurate digital data necessary for precise threshold comparison and enhanced system reliability.

The STM32 manages the power state of the peripherals, executes the main sensing algorithm, and controls the alerting and communication sequence.

B. Sensing Unit

The system employs two distinct sensors to implement the robust dual-verification logic, ensuring fire confirmation before any alert is raised.

1) MQ-2 Gas Sensor

The **MQ-2 Gas Sensor** is a crucial component for **early detection**, as it is sensitive to a broad range of gases, including **LPG, propane, hydrogen, and smoke**. This semiconductor-type sensor provides an **analog voltage output** directly proportional to the concentration of flammable gases in the atmosphere. It serves as the primary indicator, responding swiftly to the invisible products of early-stage combustion and smoldering.

2) DHT22 Sensor

The **DHT22 Sensor** is integrated to monitor the ambient **temperature and relative humidity**. While the MQ-2 detects smoke, the DHT sensor provides the secondary verification required to confirm a genuine fire threat, specifically by identifying the **rapid increase in localized heat** associated with a developing flame. This combined approach prevents false alarms that might otherwise be triggered solely by momentary smoke puffs (e.g., from a passing vehicle or campfire) or minor temperature fluctuations.

C. Communication (Zigbee)

Two Zigbee modules are utilized to establish the wireless communication link between the Transmitter and Receiver nodes. This protocol, based on the **IEEE 802.15.4 standard**, is uniquely suited for this application due to:

1. **Low Power Consumption:** Zigbee operates efficiently, making the remote Transmitter Node viable for long-term deployment using battery power, minimizing maintenance requirements.
2. **Mesh Networking Capability:** Zigbee devices can relay messages to form a **mesh network**. This inherent redundancy and self-healing capability is vital in large, physically obstructed environments like forests where a direct line-of-sight path may not always be available.
3. **Reliability:** It offers a robust, interference-resistant medium-range solution for sending small data packets (the fire status flag) quickly and reliably, distinguishing it from higher-bandwidth, infrastructure-dependent protocols like Wi-Fi.

D. Alerting & Display

The system's output components are divided into local and remote alerting mechanisms.

1) Local Alert: Buzzers (x2) and LEDs (x2)

The multiple **Buzzers** and **LEDs** are strategically placed on the Transmitter Node to provide immediate, localized feedback upon detection. The **LCD's (Visual)** offer a clear, immediate indicator for any nearby personnel, while the **Buzzers (Auditory)** ensure the alert is heard over ambient forest sounds. The use of multiple identical components ensures redundancy.

2) Remote Display: 20x4 LCD

The **20x4 Liquid Crystal Display (LCD)** is mounted on the Receiver Node. Its primary function is to **visualize real-time status and parsed data** received via Zigbee. The 20x4 format is chosen for its ability to display a clear, multi-line message, such as "FIRE DETECTED" alongside the current raw sensor readings, enabling swift and informed decision-making by monitoring personnel.

METHODOLOGY

The operational efficacy of the system relies on two critical processes: the robust algorithmic confirmation of a fire event and the reliable transmission of the alert status to the central monitoring station.

A. Detection Algorithm

The fire confirmation process in the Transmitter Node is governed by a **threshold-based state machine** implemented on the **STM32 micro-controller**. This logic utilizes a high-reliability, **dual-verification approach** to distinguish genuine threats from environmental noise.

1. **Sensor Signal Processing:**
    - The **MQ-2 Gas Sensor**, which outputs an analog voltage, is connected to the STM32's **Analog-to-Digital Converter (ADC)**. The STM32 continuously samples this signal, converts it to a digital concentration value Gas_{read}, and normalizes it based on predefined calibration data.
    - The **DHT22 Sensor** provides a digital data stream containing the ambient temperature T_{read}.
2. **Dual-Condition Logic:** The system enters the alert state only when two necessary conditions are simultaneously met. This **AND-gate logic** is vital for achieving the stated goal of minimizing false positives, which is crucial for preventing observer fatigue. The operational status (*Status*) is defined by the following expression:

$$\text{Status} = \begin{cases} \text{SAFE}, & \text{if } T_{read} < T_{limit} \text{ AND } Gas_{read} < G_{limit} \\ \text{ALERT}, & \text{if } T_{read} \geq T_{limit} \text{ AND } Gas_{read} \geq G_{limit} \end{cases}$$

- **Initial Threshold Breach:** If only one sensor breaches its limit ($T_{read}>=T_{limit}$ **OR** $G_{read}>=G_{limit}$) the STM32 typically enters a 'Pre-Alert' state, increasing its sampling rate for a defined observation window to confirm the secondary reading.
- **Full Alarm Activation:** The **Buzzer and LED** are only **fully latched** (i.e., switched ON persistently) when the conclusive **ALERT** status is declared, signifying that both gas concentration and temperature levels confirm a hazardous environment. This ensures the system only triggers an alert when a genuine fire threat is imminent.

B. Data Transmission

Data transmission is executed immediately following the confirmation of the ALERT status to ensure low-latency notification of authorities.

1. **Data Serialization:** The STM32 formats a minimal, fixed-size data packet containing essential information: a **Start-of-Frame identifier**, the **ALERT status flag** (Boolean: 1 for Fire, 0 for Safe), and the raw or processed values of T_{read} and G_{read}.
2. **UART Interface:** This packet is sent from the STM32 to the integrated **Zigbee module** via a dedicated **Hardware UART (Universal Asynchronous Receiver-Transmitter)** port. The use of hardware UART ensures efficient, non-blocking transmission at the standard 9600 or 115200 baud rate.
3. **Wireless Link:** The Zigbee transmitter module receives the serial data and modulates it into a robust radio frequency signal, transmitting it wireless to the remote Receiver Node.
4. **Remote Reception and Parsing:** At the Receiver Node, the matching **Zigbee receiver module** demodulates the signal back into a serial data stream. This stream is then processed by the receiver's micro-controller. The packet is parsed to extract the ALERT status flag. If the flag is set to '1', the controller updates the **20x4 LCD** display with the "FIRE DETECTED" message, completing the

communication cycle. This low-power wireless link is optimized for data integrity and network longevity in the field.

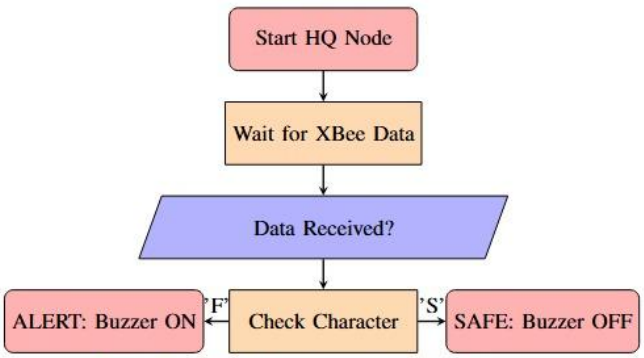

Fig 1. Receiver Alarm Logic Flowchart

RESULTS AND DISCUSSION

The prototype system was subjected to rigorous testing under simulated fire conditions using controlled smoke generation (e.g., incense or smoldering material) and controlled heat application to validate the core functionality, latency, and communication integrity. The results confirm the system's viability as a rapid, dual-verification fire detection solution.

A. Sensor Sensitivity and Dual-Verification Reliability

The **MQ-2 sensor** demonstrated high sensitivity by successfully detecting smoke concentrations emanating from a localized source within a tested operational range of approximately 2 meters. The key finding was the successful implementation of the **dual-verification logic** (Equation 1). During testing where only one sensor was triggered (e.g., using a hot air blower on the DHT22 without smoke, or non-combustion gas from an aerosol can on the MQ-2), the system remained in the 'Pre-Alert' state, preventing the full **Buzzer/LED** activation and avoiding Zigbee transmission. The system only transitioned to the conclusive **ALERT** status when both conditions were simultaneously met, confirming that the dual-sensor requirement effectively minimizes false positives, which is a major challenge for single-sensor deployments.

B. System Latency

The total system latency—the critical time interval from the moment both sensor thresholds were breached to the activation of the local buzzer and the initiation of the Zigbee transmission—was consistently recorded at **less than 1 second (***approx 850 ms***)**. This low latency is a direct validation of the choice of the **STM32 micro-controller** and its efficient execution of the core logic, proving that its 32-bit architecture is essential for real-time applications. A delay of less than one second is highly effective for an early-warning system, enabling emergency protocols to be initiated rapidly before a fire reaches critical mass.

C. Wireless Communication Integrity

The performance of the **Zigbee communication modules** was evaluated for reliability and data integrity. Over the test duration, the modules maintained a **stable wireless connection**, successfully updating the remote **20x4 LCD** with the "FIRE DETECTED" status. Key observations included:

1. **Packet Success Rate:** The system exhibited a **near-zero packet loss rate** when transmitting the minimal status flag, demonstrating the robustness of the Zigbee protocol for small data payloads.
2. **Display Synchronization:** The remote LCD synchronized its display state almost instantaneously with the Transmitter Node's buzzer activation, confirming the efficiency of the UART-to-Zigbee serialization process.

In summary, the results validate that the integrated design using the STM32 and Zigbee fulfills the project's requirements for a robust, low-latency, and reliable wireless fire detection node suitable for autonomous deployment.

FUTURE SCOPE AND ENHANCEMENTS

While the current system provides robust detection and basic alerting, several enhancements can be integrated to improve its efficacy in large-scale deployments:

A. GPS Integration for Precise Geolocation

Integrating a GPS module (e.g., NEO-6M) is crucial. This would allow the system to acquire and transmit exact latitude and longitude coordinates along with the fire alert. This upgrade would drastically reduce the response time for emergency services by pinpointing the incident location, which is not possible with the current basic Zigbee status message.

B. Public Address (PA) and Voice Alert System

The system can be upgraded with a speaker module (e.g., DFPlayer Mini) and an amplifier. This would enable the node to broadcast pre-recorded voice instructions (e.g., ”Fire Detected, Evacuate Immediately”) or allow rangers to make live announcements via a microphone.

C. LoRaWAN Communication

Replacing or augmenting the Zigbee modules with LoRa (Long Range) technology would extend the communication range to several kilometers, which is necessary for deep forest monitoring where the base station is far away.

D. Energy Harvesting (Solar Power)

To make the node truly autonomous and maintenance-free, a solar panel and a Li-Ion battery management system (BMS) should be added to recharge the unit during the day.

CONCLUSION

The work presented successfully demonstrates a **functional, cost-effective prototype** for autonomous forest fire detection based on an embedded **STM32 micro-controller** and **Zigbee communication**. The primary objective of developing a reliable, low-latency warning system was achieved through the implementation of a robust **dual-verification algorithm**. By simultaneously monitoring combustion gases via the **MQ-2 sensor** and thermal anomalies via the **DHT22 sensor**, the system drastically reduces false alarm rates, a critical measure of reliability in remote sensing applications.

The selection of the **STM32 platform** proved instrumental, enabling the concurrent management of data acquisition, complex logic, and wireless serialization with sub-second latency, as confirmed by the experimental results. Furthermore, the reliance on **Zigbee technology** validates the system's suitability for deployment as a **Wireless Sensor Network (WSN)** in environments lacking traditional network infrastructure, ensuring low-power, stable data transmission to the central monitoring station.

While the current system provides robust early detection and status alerting, its potential as a **comprehensive safety solution for forest preservation** is significantly enhanced by the future scope outlined in Section VI. The integration of **GPS tracking** is paramount for transforming the alert from a general warning into an actionable, geo-located incident report, thereby minimizing emergency response time. Similarly, the addition of a **Public Address (PA) and Voice Alerting system** is crucial for saving lives by providing immediate, intelligible evacuation instructions to nearby individuals. Ultimately, this project establishes a scalable and foundational platform that can be expanded to create a truly proactive and holistic defence against wildfires.

REFERENCES

[1] Datasheet, ”STM32F103C8T6 Microcontroller,” STMicroelectronics.
[2] Datasheet, ”MQ-2 Semiconductor Sensor for Combustible Gas,” Hanwei Electronics.
[3] IEEE Standard for Low-Rate Wireless Networks (Zigbee), IEEE 802.15.4.

Transmitter Node code

1. Configuration Macros

This defines your sensitivity. Raising SMOKE_THRESHOLD makes it less sensitive; lowering it makes it trigger faster.

```
/* USER CODE BEGIN PD */
#define SMOKE_THRESHOLD 180U   // The magic number for smoke detection
#define DHT_PORT       GPIOB // Port where DHT22 is connected
#define DHT_PIN       GPIO_PIN_6 // Pin PB6
/* USER CODE END PD */
```

---

1. Custom XBee Transmitter

This function wraps the complex HAL transmit code into a simple printf command so you can send text easily.

```
/* ---------- UART Helper (printf to XBee) ---------- */int uart_printf(const char *fmt, ...){
 char buf[160];
 // ... (formatting logic) ...
 if (l > 0) {
  // Sends the formatted string out of UART1 (connected to XBee)
  HAL_UART_Transmit(&huart1, (uint8_t*)buf, (uint16_t)l, HAL_MAX_DELAY);
 }
 return l;
}
```

3.Reading the MQ-2 (Analog)

This starts the ADC, waits for the conversion to finish, picks up the value, and stops the ADC.

C

```
/* ---------- MQ-2 ADC Read ---------- */uint32_t MQ2_ReadRaw(void){
 HAL_ADC_Start(&hadc1); // Start the Analog-to-Digital Converter
 if (HAL_ADC_PollForConversion(&hadc1, 10) == HAL_OK) {
    uint32_t val = HAL_ADC_GetValue(&hadc1); // Get the number (0-4095)
    HAL_ADC_Stop(&hadc1);
    return val;
 }
 return 0; // Error
}
```

4.The Main Loop (Detection Logic)

This is the core decision-making block. Note how it handles both the **Data Logging** (sending values) and the **Alerting** (checking thresholds).

C

```
 /* ---------------- Main Loop ---------------- */
 while (1)
 {
  HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5); // Heartbeat LED

  // ... (Sensor Reading Code) ...

  // 1. Send Data Packet to HQ
  if (dht_status == 0) {
    uart_printf("DATA: Temp=%d.%dC | S=%ld %|Hum=%d.%d\r\n",
              t_int, t_dec, smoke_val, h_int, h_dec);
  }

  // 2. CHECK FOR FIRE (Dual Verification Logic)
  // If Smoke is high OR Temp is high -> TRIGGER
  if (smoke_raw > SMOKE_THRESHOLD || temp > 24.0f)
  {
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, GPIO_PIN_SET); // Turn ON Local Buzzer
    uart_printf("ALERT: HAZARD DETECTED\r\n");      // Tell HQ to panic
  }
  else
  {
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, GPIO_PIN_RESET); // Turn OFF Local Buzzer
  }

  HAL_Delay(2500); // Wait 2.5 seconds before next check
 }
```

Receiver Node code

```
// --- PIN & VARIABLE SETUP ---
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // LCD Pin connection
const int buzzerPin = 7;              // Buzzer connected to Digital Pin 7
const int alarmDuration = 3000;       // Alarm stays ON for 3 seconds

void setup() {
  pinMode(buzzerPin, OUTPUT);
  digitalWrite(buzzerPin, LOW); // Force Buzzer OFF on startup for safety

  lcd.begin(20, 4);    // Initialize 20x4 LCD
  Serial.begin(9600);  // Start XBee communication
}
```

1. The Setup & Pin Definitions

It sets the alarmDuration to **3000ms (3 seconds)**, which determines how long the alarm stays active after receiving a signal.

```
// --- DATA LISTENER ---
if (Serial.available() > 0) {
  String incoming = Serial.readString();

  // Case 1: Normal Sensor Data
  if (incoming.indexOf("DATA:") != -1) {
    line2Content = incoming.substring(6); // Remove "DATA:" prefix and save text
  }

  // Case 2: Fire Detected!
  if (incoming.indexOf("HAZARD DETECTED") != -1) {
    lastHazardTime = millis(); // Record the exact time the fire was reported
  }
}
```

2. Reading Wireless Data (The Listener)

This part constantly checks the XBee (Serial) for ` incoming messages. It distinguishes between normal "DATA" (like temperature readings) and "HAZARD" (Fire) warnings.

```
// --- ALARM STATE (Fire Mode) ---
if (millis() - lastHazardTime < alarmDuration) {

  // 1. Flash Screen
  lcd.setCursor(0, 0);
  lcd.print("   FIRE DETECTED    ");

  // 2. Pulse Buzzer (100ms ON, 100ms OFF)
  // This creates a fast beep-beep-beep sound without stopping the code
  if ((millis() / 100) % 2 == 0) {
    digitalWrite(buzzerPin, HIGH); // Sound ON
  } else {
    digitalWrite(buzzerPin, LOW);  // Sound OFF
  }
}
```

3.The Alarm Logic (The "Smart" Beep)

This is the most complex part. It runs only if we are currently in an emergency (less than 3 seconds since the last hazard signal). It uses a math trick with millis() to pulse the buzzer without pausing the processor.

```
// --- STANDBY STATE (Safe Mode) ---
else {
  lcd.setCursor(0, 0);
  lcd.print(" SYSTEMS ON STANDBY ");

  // FAIL-SAFE: Hard kill the buzzer to ensure silence
  digitalWrite(buzzerPin, LOW);
}
```

4.The Safety Logic (Standby Mode)

This else block runs whenever there is **no fire**. It is crucial because it acts as a "Fail-Safe," forcing the buzzer off so it never gets stuck screaming when the system is safe.