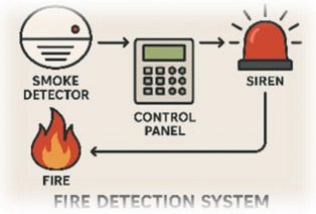
FIRE DETECTION SYSTEM



AIM OF THE PROJECT:

To design and implement a system that can quickly detect the presence of fire or smoke and alert occupants or emergency services to prevent loss of life, minimize property damage, and enable timely evacuation and response.

More specifically, it may aim to:

- **Ensure early detection** of fire using sensors (like smoke, heat, or flame detectors).
- **Trigger alerts** through alarms or notifications.
- **Enable quick response** by linking to emergency services or fire suppression systems
- **Improve safety** in residential, commercial, or industrial environments.

PROBLEM STATEMENT AND SOLUTION:

Problem Statement:

Fire outbreaks pose a serious threat to human life, property, and the environment. In many cases, delayed detection and response lead to uncontrollable damage, injuries, or fatalities. Traditional fire detection systems may be limited by slow response times, lack of real-time monitoring, or inability to notify users remotely. There is a need for a reliable, cost-effective, and efficient system that can detect fire hazards at an early stage and promptly alert people to take action.

☑ Proposed Solution:

The proposed fire detection system uses sensors (such as smoke, temperature, or flame sensors) to detect signs of fire in real-time. When a fire is detected, the system immediately triggers an alarm and optionally sends alerts via buzzer, SMS, or mobile notifications. The system can be integrated with a control panel for monitoring and may be connected to emergency services or smart home networks for automated response.

PROJECT DESIGN SPECIFICATION AND ARCHITECTURE:

The **Project Design Specification and Architecture** of a **Fire Detection System** defines the overall structure, components, and design principles used to create a functional and reliable system. Here's a comprehensive overview you can use or adapt for a report, proposal, or technical documentation.

1. Project Overview

The fire detection system is designed to monitor an area for fire hazards and alert relevant stakeholders through alarms and communication systems. It focuses on early detection to reduce property damage, ensure occupant safety, and initiate emergency response.

2. **Objectives**

- Detect smoke, heat, or flames as early indicators of fire.
- Alert occupants via alarms and notifications.
- Communicate with emergency systems (e.g., fire department).
- Log and monitor events in real-time.

3. **System Components**

a. Sensors

- **Smoke Detectors** (Ionization/Photoelectric)
- **Heat Detectors** (Fixed temperature, Rate-of-Rise)
- **Flame Detectors** (Infrared/Ultraviolet)
- **Gas Sensors** (CO, LPG, etc.)

b. Control Panel (FACP)

- Central processing unit that collects data from sensors and controls alarms and notifications.

c. Alarm Units

- **Audio** (buzzers, sirens)
- **Visual** (strobe lights)
- **Voice Evacuation Systems**

d. Communication Module

- GSM/GPRS/IoT modules for remote alerts (SMS, email, app notifications)
- Integration with Building Management Systems (BMS)

e. Power Supply

- Primary (AC mains)
- Secondary (battery backup or UPS)

4. System Architecture

Here's a typical layered architecture:

A. Sensing Layer

- Collects environmental data (smoke, heat, gas, flame).
- Deploys multiple sensor types for redundancy and accuracy.

B. Processing Layer

- Microcontroller/PLC/Embedded System processes signals.
- Implements threshold-based or AI-based fire detection algorithms.

C. Communication Layer

- Internal communication (wired or wireless)
    - RS485, Modbus, or Zigbee/Bluetooth/Wi-Fi

- External communication (IoT/GSM)
    - Sends alerts to mobile apps, SMS, cloud dashboards.

D. Actuation Layer

- Triggers sirens, sprinklers (if integrated), emergency exits, etc.
- Sends data to fire departments or control centers.

5. **Functional Requirements**

- Real-time monitoring
- Event logging
- Fault detection (sensor failure, wiring fault)
- Battery monitoring
- User interface (touchscreen panel or mobile app)
- Zone-based control and monitoring

6. **Non-Functional Requirements**

- **Reliability:** Fault-tolerant, redundancy in sensors
- **Scalability:** Can add more sensors/zones
- **Maintainability:** Easy replacement and diagnostics
- **Security:** Secure communication, anti-tamper features
- **Compliance:** Adheres to local fire safety standards (e.g., NFPA, IS, EN)

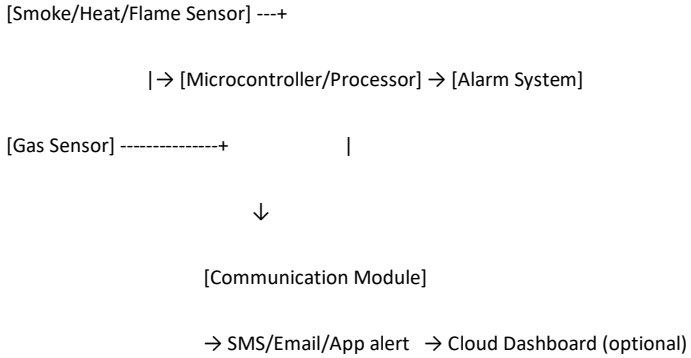
7. **Deployment Model**

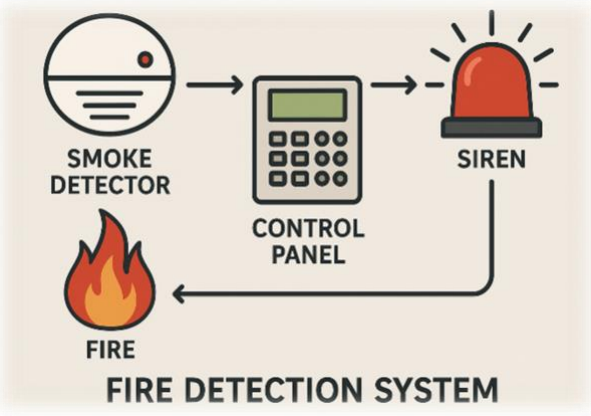
- **Standalone System:** For small buildings, houses
- **Networked System:** For multi-story buildings or campuses
- **Cloud-based Monitoring:** For remote access, large-scale systems

8. **User Interface**

- LCD or touch panel for control and status viewing
- Mobile/web application for alerts and remote control
- Admin roles for configuration, logs, and maintenance

9. **Diagram (Simplified)**





10. Future Enhancements

- AI-based fire prediction
- Integration with drones for fire source tracking
- Automated sprinkler activation with zone targeting
- Voice control and smart assistant integration.

PROGRAMMING SOLUTION:

Here's a simple Python version that captures the logic:

```
class SmokeDetector:
    def detect_smoke(self):
        # Simulating smoke detection
        print("Smoke detected!")
        return True
class ControlPanel:
    def process_signal(self, smoke_detected):
        if smoke_detected:
            print("Control Panel: Signal received. Activating siren...")
            return True
        else:
            print("Control Panel: No smoke detected.")
            return False
class Siren:
    def activate(self):
        print("Siren: WEEEOOO WEEEOOO! (Alert!)")
class FireDetectionSystem:
    def __init__(self):
       self.smoke_detector = SmokeDetector()
        self.control_panel = ControlPanel()
        self.siren = Siren()
```

```
    def run_system(self):
        smoke_detected = self.smoke_detector.detect_smoke()
        if self.control_panel.process_signal(smoke_detected):
            self.siren.activate()
# Run the system
if __name__ == "__main__":
    fire_detection_system = FireDetectionSystem()
    fire_detection_system.run_system()
```

EXPLANATION:

- **Smoke Detector** detects smoke.
- **Control Panel** processes the signal.
- **Siren** gets activated if smoke is detected.

FLOW EXPLANATION:

Here's a simple explanation of the **flow** of a **fire detection system** based on the diagram you shared:

Fire Detection System Flow:

1. **Fire Happens**
   A fire starts, producing **smoke** in the environment.
2. **Smoke Detector Activates**
   The **smoke detector** senses the presence of smoke particles in the air.
   ➔ It **sends a signal** to the **control panel**.
3. **Control Panel Processes the Signal**
   The **control panel** receives the signal from the smoke detector.
   ➔ It **decides** that there is a potential fire situation.
   ➔ It **triggers** the **siren**.
4. **Siren Alarms**
   The **siren** sounds loudly and flashes lights.
   ➔ This **warns** people nearby about the fire so they can **evacuate** or **take action**.
5. **Result: Fire Response**
   People are alerted quickly, and emergency actions like **calling firefighters** or **using extinguishers** can happen.

Summary in One Line:

Fire ➔ Smoke ➔ Smoke Detector ➔ Control Panel ➔ Siren ➔ Human Response

WIRING DIAGRAM:

THERE ARE THREE TYPES OF WIRING DIAGRAMS THEY ARE,

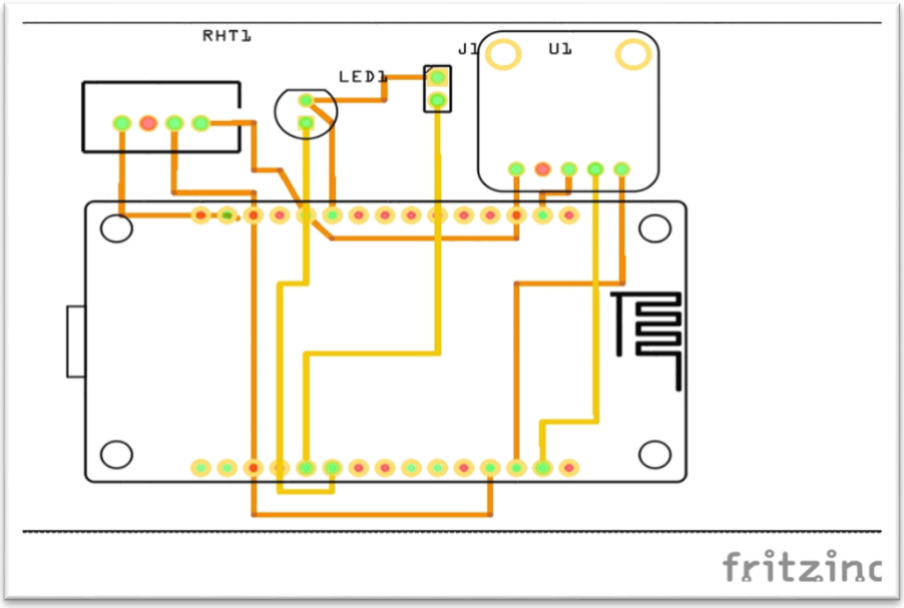
1 . PCB DIAGRAM

2 . SCHEMATIC DIAGRAM

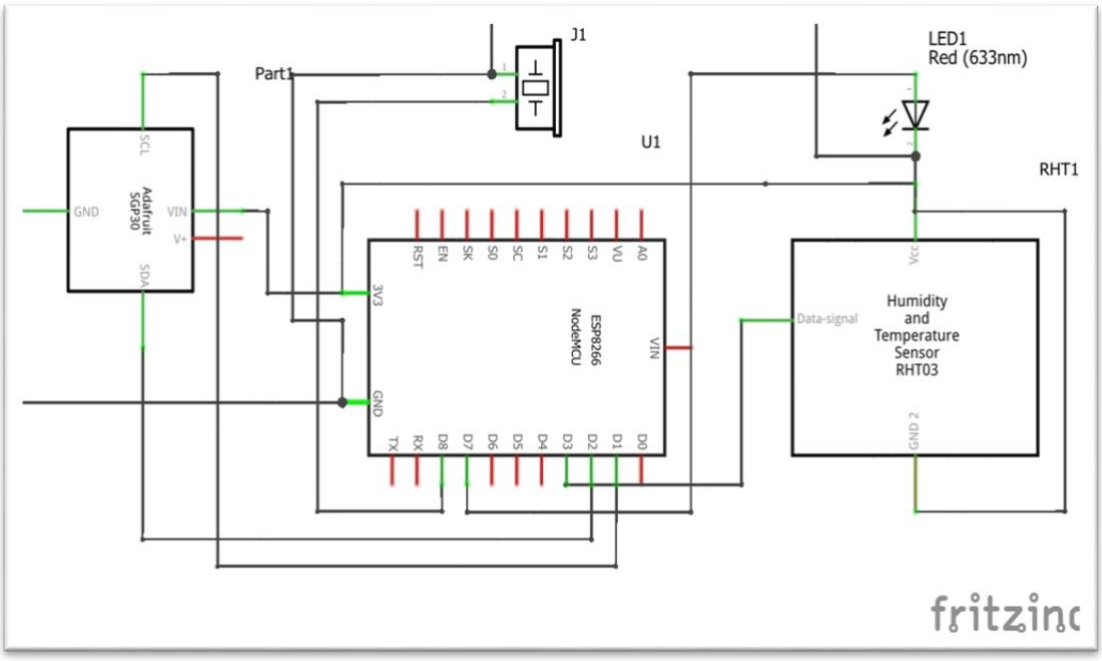
3 . BREAD BOARD DIAGRAM

LET'S SEE THE DIAGRAMS

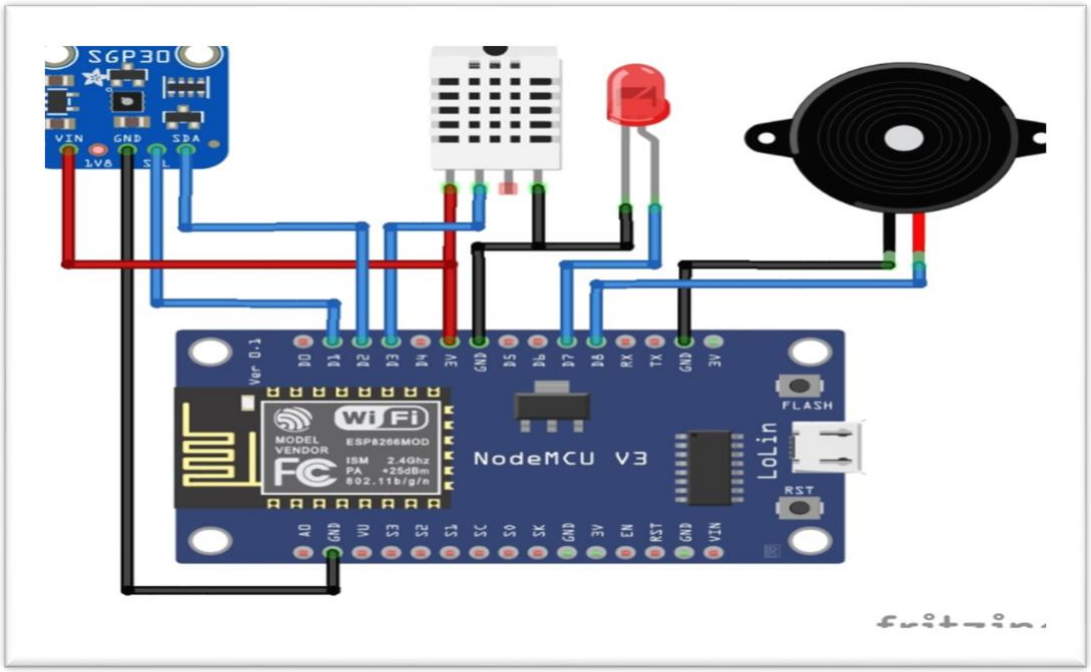
1.PCB DIAGRAM:



2.SCHEMATIC DIAGRAM:



3. BREAD BOARD DIAGRAM:



COMPONENTS WORKING PRINCIPLES/PRINCPLES:

In a **Fire Detection System**, the **main components** and their **working principles** or **functionality** are:

1. Fire Detectors

(Detect early signs of fire)

- **Smoke Detectors**
  - **Working Principle**: Detect particles in the air.
    - **Ionization Type**: Measures change in electrical current caused by smoke particles.
    - **Photoelectric Type**: Uses a light beam; smoke scatters the light and triggers an alarm.
- **Heat Detectors**
  - **Working Principle**: Detects rise in temperature.
    - **Fixed Temperature**: Alarms when temperature exceeds a set limit.
    - **Rate-of-Rise**: Alarms when temperature rises quickly.
- **Flame Detectors**
  - **Working Principle**: Detects light emitted by flames (ultraviolet or infrared).

2. Fire Alarm Control Panel (FACP)

(Central brain of the system)

- **Working Principle**:
  - Receives signals from detectors.
  - Processes and verifies alarms.
  - Activates outputs like sirens, notifications, and fire suppression if needed.

3. Notification Devices

(Alert occupants)

- **Types**: Sirens, bells, flashing strobes, voice announcements.
- **Working Principle**:
  - When triggered by the control panel, they create sound/visual alerts to warn people.

4. **Manual Pull Stations**

(Human-triggered alarms)

- **Working Principle**:
  - A person pulls a lever to manually send a fire alarm signal to the control panel.

5. **Power Supplies**

(Ensure system reliability)

- **Working Principle**:
  - Provide primary (main) and backup (battery) power so the system stays active even during power outages.

6. **Fire Suppression Systems** (sometimes integrated)

(Put out the fire)

- **Examples**: Sprinkler systems, gas suppression.
- **Working Principle**:
  - Activates automatically or manually when fire is detected to extinguish or control fire spread.

BILL OF MATERIALS PREPARATION(BOM):

Here's the **BOM** for the project in your picture:

No.	Component	Specification/Model	Quantity	Notes
1	**Microcontroller**	NodeMCU V3 (ESP8266 WiFi Module)	1	Main controller
2	**Gas Sensor**	SGP30 Air Quality Sensor	1	Detects VOCs and CO_2
3	**Temperature & Humidity Sensor**	DHT22 (or similar DHT series)	1	Monitors temp and humidity
4	**LED**	5mm Red LED	1	Visual fire/gas alarm
5	**Buzzer**	Active Buzzer	1	Audible alarm
6	**Resistors**	220Ω Resistor (for LED)	1	Protects the LED
7	**Jumper Wires**	Male-to-Male	~10	For wiring connections
8	**Breadboard (optional)**	Small breadboard	1	For neat prototyping

Extra Notes:

- **SGP30** connects via **I2C** (SCL and SDA lines).
- **DHT22** connects via a digital pin.
- **LED** and **Buzzer** are outputs controlled by NodeMCU.
- Make sure to use **appropriate pull-up resistors** if necessary for I2C or DHT sensors (some modules already have them built-in).

PROJECT OUTCOME:

Alright! Based on the wiring diagram you sent (NodeMCU + SGP30 Gas Sensor + DHT22 Sensor + LED + Buzzer), the **project outcome** in a real-life **video demonstration** would typically look like this:

🔥 **Fire/Smoke/Heat Detection System — Expected Outcome (Video Scenario):**

1. **Normal Condition**:
   - The **LED** is OFF.
   - The **Buzzer** is silent.
   - The system is reading air quality (gas levels) and temperature/humidity regularly.
2. **When Fire/Smoke/High Gas Detected**:
   - The **SGP30 sensor** detects high VOC or CO_2 levels.
   - The **DHT22 sensor** detects an abnormal **high temperature** (above a set threshold, e.g., 50°C).
   - The **LED** turns ON (bright red warning light).
   - The **Buzzer** starts beeping (or continuous alarm sound).
   - (Optional: If programmed, the NodeMCU could send a notification via Wi-Fi to a phone or web dashboard.)
3. **After Fire/Smoke is Gone**:
   - Sensors detect normal air and temperature levels.
   - The **LED** turns OFF.
   - The **Buzzer** stops.

⚡ **In a simple video**, you would show:

- Powering up the system.
- Blowing smoke, or heating near the sensor.
- The LED lighting up and buzzer sounding when thresholds are crossed.
- Returning to normal when the smoke/heat is removed.

THANK YOU