GOVERNMENT OF TAMILNADU

DIRECTORATE OF TECHNICAL EDUCATION, CHENNAI

NAAN MUDHALVAN SCHEME (TNSDC) SPONSORED

STUDENTS DEVELOPMENT PROGRAMME

ON

IoT AND ITS APPLICATIONS

HOST INSTITUTION

xxxxx

COIMBATORE – 04

TRAINING PARTNER

ENTHU TECHNOLOGY SOLUTIONS INDIA PVT LTD

DATE:

NAME	ROLL NO
NAME 1	1
NAME 2	2
NAME 3	3
NAME 4	4
NAME 5	5

TABLE OF CONTENTS

ABSTRACT

Fire Alarm Circuit is a simple circuit that detects the fire and activates the Siren Sound or Buzzer. Fire Alarm Circuits are very important devices to detect fire in the right time and prevent any damage to people or property.

Fire Alarm Circuits and Smoke Sensors are a part of the security systems which help in detecting or preventing damage. Installing Fire Alarm Systems and Smoke Sensors in commercial buildings like offices, movie theatres, shopping malls and other public places is compulsory.

There are many expensive and sophisticated Fire Alarm Circuit in the form of stand-alone devices, but we have designed five very simple Fire Alarm Circuits using common components like Thermistor, LM358, Germanium Diode, LM341 and NE555.

This is a very simple alarm circuit using Thermistor, LM358 Operational – Amplifier and a Buzzer. The primary purpose of fire alarm system is to provide an early warning of fire so that people can be. evacuated & immediate action can be taken to stop or eliminate of the fire effect as soon as possible. Alarm can be. triggered by using detectors or by manual call point (Remotely).

INTRODUCTION

Overview:

Fire alarm systems are crucial for ensuring safety by providing early warnings in case of a fire. This project focuses on designing and implementing a simple yet effective fire alarm system using the ESP32 microcontroller. The system leverages a flame sensor to detect the presence of fire and promptly activates both audible and visual alarms. Additionally, the system sends real-time alerts to a cloud platform, enabling remote monitoring and quick responses.

Objective:

The primary objective of this project is to create a fire detection system using ESP32 that:

- Detects the presence of fire using a flame sensor.
- Alerts users through a buzzer and LED.
- Sends status updates to a cloud platform for remote monitoring.

Scope:

This project covers the design and implementation of a basic fire alarm system. The system's scalability allows for integration with other sensors or expansion into a more comprehensive home automation system. The cloud integration enables real-time monitoring and logging of fire incidents, which can be extended to trigger other automated responses.

Software &Hardware Requirements:

SOFTWARE:

- Arduino IDE
- Thingzmate

HARDWARE:

- Esp32 Board
- Bread board
- Flame sensor
- Buzzer
- Led
- Jumper Cable

ARDUINO IDE:

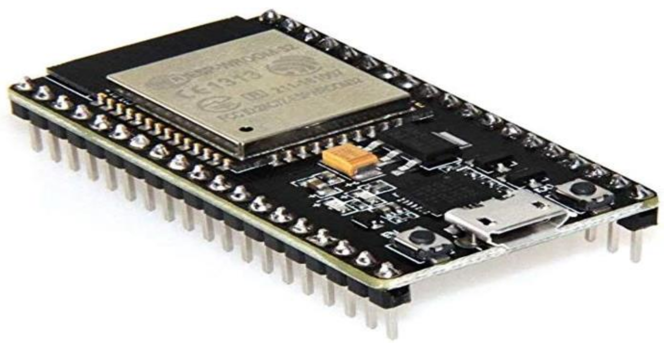
The fire alarm system using Arduino IDE and ESP32 works by continuously monitoring a flame sensor. The sensor detects infrared light emitted by fire and sends a digital signal to the ESP32. If a flame is detected, the ESP32 triggers an alarm by activating a buzzer and lighting an LED, signaling the presence of fire. Simultaneously, the ESP32 sends a notification to a cloud server via WiFi, allowing remote monitoring. The system resets after a delay, continuing to monitor for fire and ensuring real-time alerts and safety.

THINGZMATE:

In a Thingzmate-integrated fire alarm system, the ESP32 microcontroller continuously monitors a flame sensor. When fire is detected, the sensor sends a signal to the ESP32, which triggers an audible alarm (buzzer) and visual alert (LED). Simultaneously, the ESP32 connects to WiFi and sends a "FIRE!!!" alert to the Thingzmate cloud platform using an HTTP POST request. This real-time data update allows users to monitor the system remotely via the Thingzmate dashboard, ensuring timely responses to fire emergencies and enhancing overall safety.

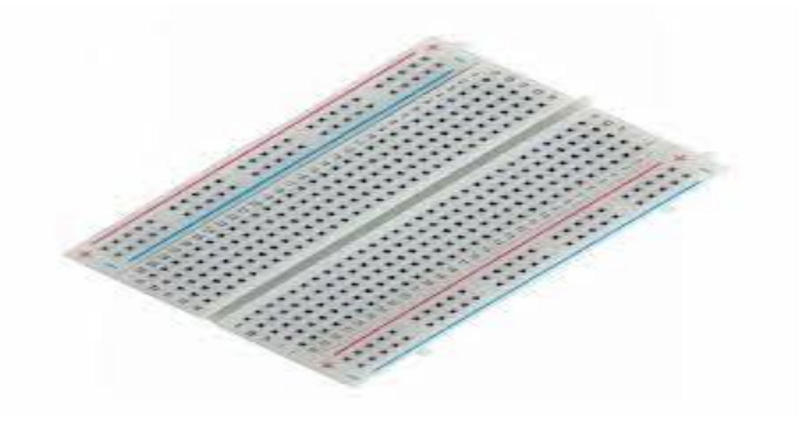
ESP32 BOARD:

The ESP32-based fire alarm system monitors the environment using a flame sensor connected to its digital input pins. When the sensor detects fire, it sends a signal to the ESP32, which then activates a buzzer and an LED to provide audible and visual alarms. Additionally, the ESP32 connects to a WiFi network and sends an alert message, such as "FIRE!!!," to a designated cloud server or IoT platform. This allows for remote monitoring and immediate response to fire emergencies, ensuring safety through real-time alerts and continuous monitoring.



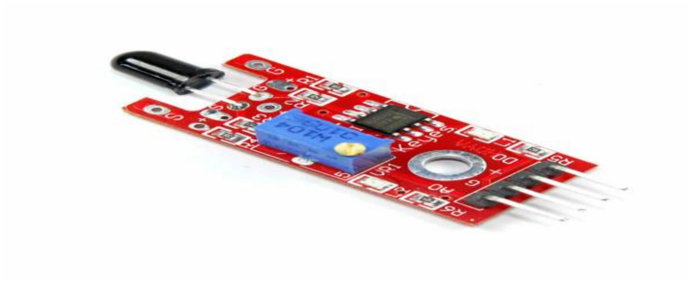
BREAD BOARD:

In a breadboard-based fire alarm system, components like the ESP32 microcontroller, flame sensor, buzzer, and LED are connected on a breadboard for easy prototyping. The flame sensor detects fire by sensing infrared light and sends a signal to the ESP32. When fire is detected, the ESP32 triggers the buzzer for an audible alarm and lights up the LED for visual alert. The breadboard allows for quick assembly and testing of the circuit without soldering, making it ideal for development and experimentation before finalizing the design. The system can also be connected to the cloud for remote monitoring.



FLAME SENSOR:

A flame sensor in a fire alarm system detects the presence of fire by sensing infrared light emitted by flames. The sensor outputs a digital signal when it detects a flame, which is sent to a microcontroller like the ESP32. Upon receiving the signal, the microcontroller activates a buzzer and an LED, providing audible and visual alerts. The flame sensor operates continuously, allowing the system to respond instantly to fire. In advanced setups, the microcontroller can also send alerts to a cloud server for remote monitoring, ensuring real-time notifications and enhancing fire safety.



BUZZER:

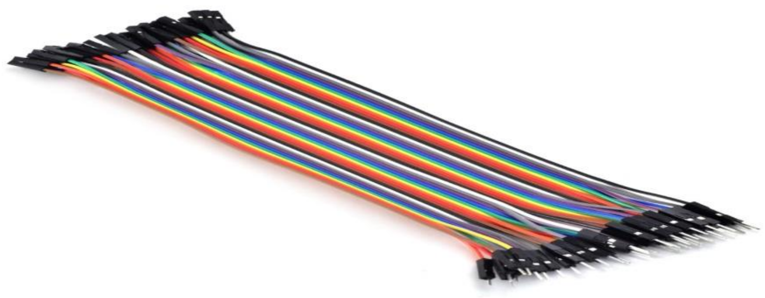
In a fire alarm system, a buzzer serves as an audible alert device. When a flame sensor detects fire, it sends a signal to a microcontroller like the ESP32. The microcontroller then activates the buzzer, causing it to emit a loud, continuous sound to alert people in the vicinity of the danger. The buzzer's purpose is to provide an immediate and unmistakable warning, prompting evacuation or other safety measures. The system may also include visual alerts (LEDs) and cloud notifications, but the buzzer is critical for ensuring that the alarm is heard, even without visual cues.

LED:

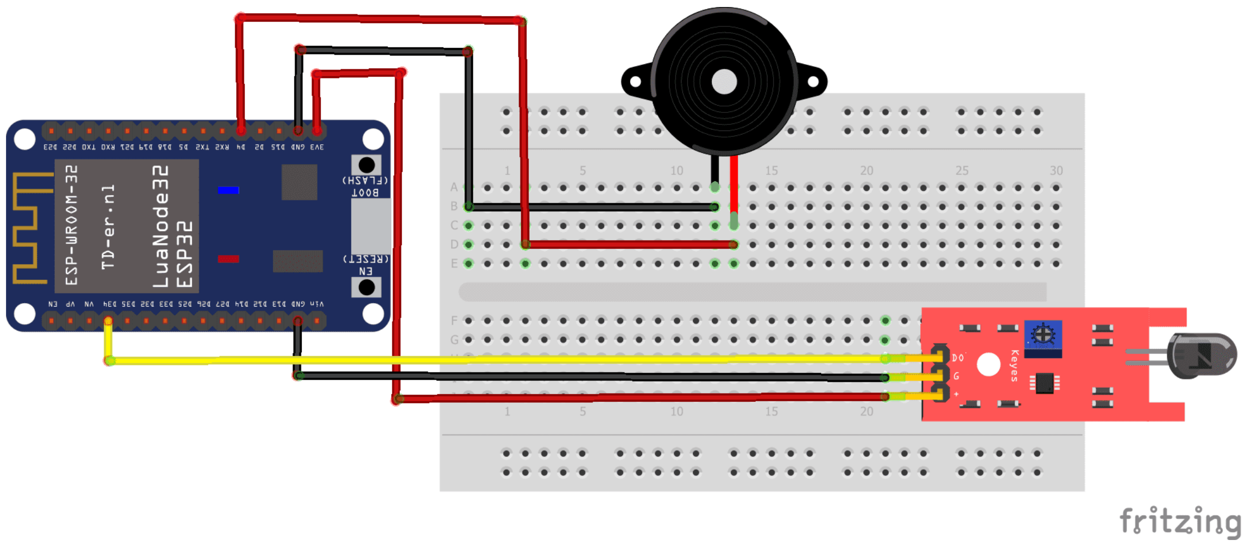
In a fire alarm system, an LED serves as a visual alert indicator. When a flame sensor detects fire, it sends a signal to a microcontroller like the ESP32. The microcontroller then powers the LED, causing it to light up, often in red, to visually indicate the presence of fire. This visual alert complements other warning systems, such as buzzers, by providing a clear, visible sign of danger. The LED remains illuminated as long as the fire is detected, ensuring that even in noisy environments or for individuals with hearing impairments, the alarm is noticeable and prompts immediate action.

JUMPER CABLE:

In a fire alarm system, jumper cables are essential for connecting various components, such as the flame sensor, ESP32 microcontroller, LED, and buzzer, on a breadboard or prototyping setup. These flexible wires allow for quick and temporary connections without soldering. When the flame sensor detects fire, it sends a signal through the jumper cables to the ESP32. The ESP32 then activates the LED and buzzer through other jumper cables, triggering visual and audible alarms. The reliable connections provided by jumper cables ensure proper communication between components, making the system functional and effective in detecting and responding to fire.



CIRCUIT DIAGRAM:



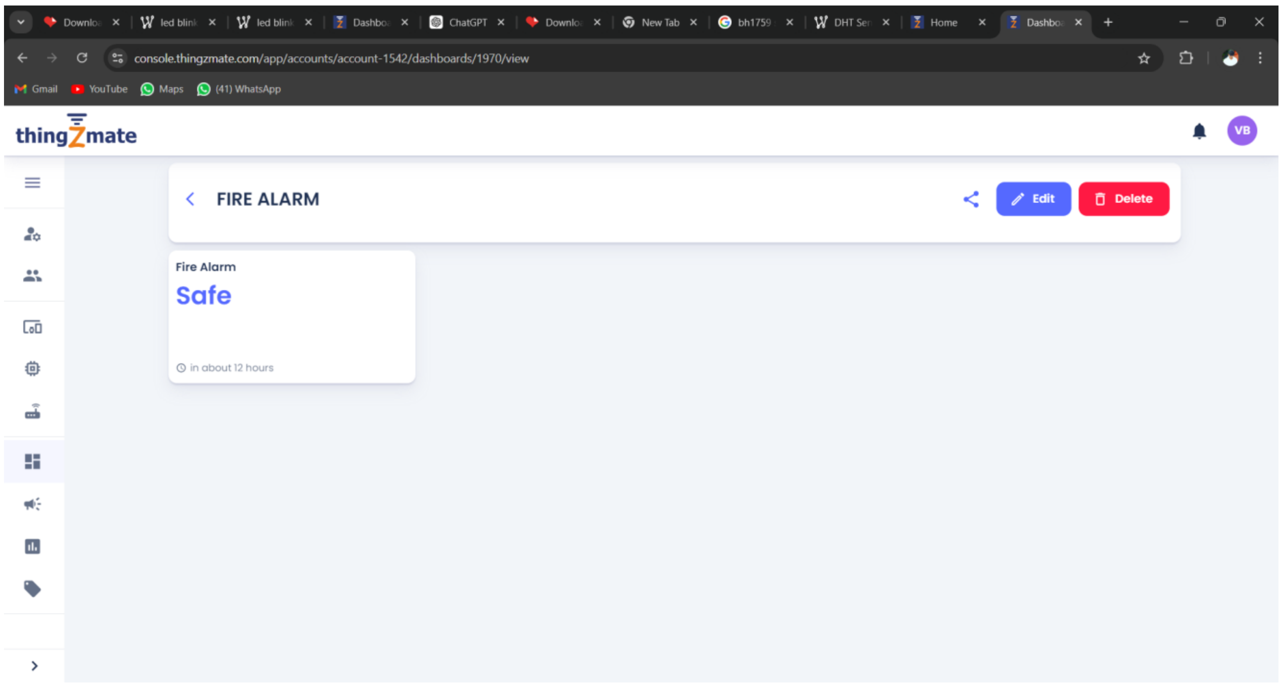
Coding:

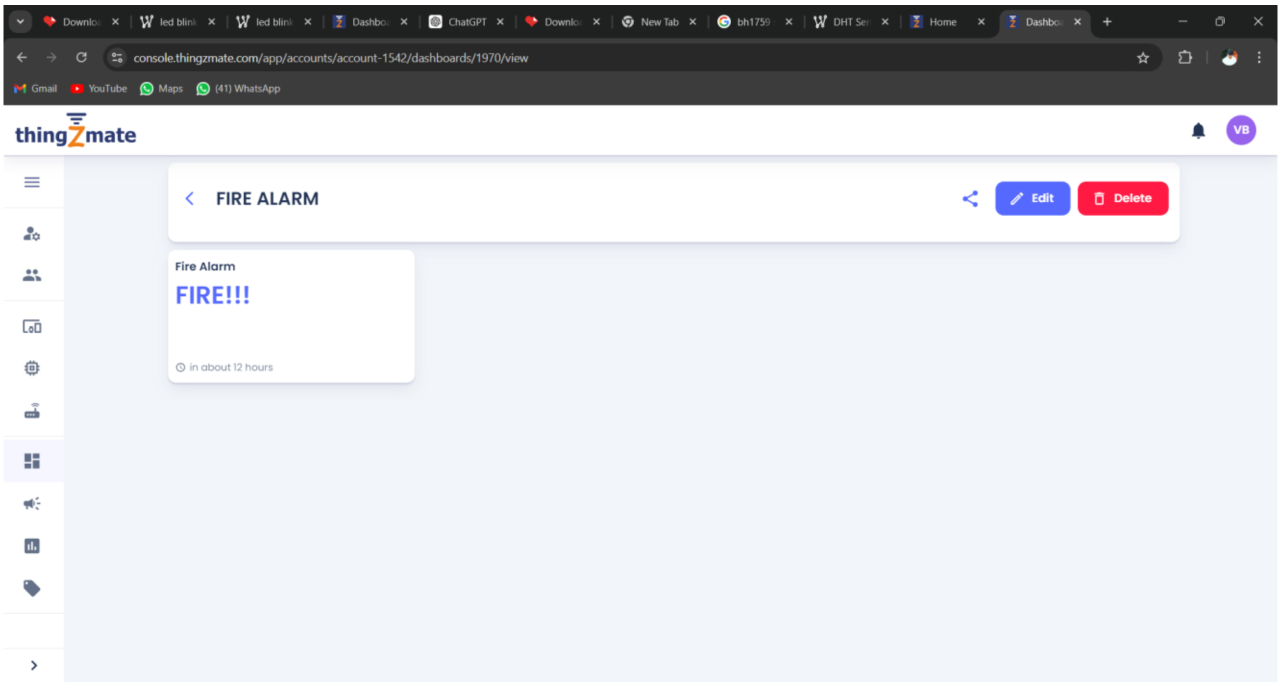
```
#include <WiFi.h>
#include <HTTPClient.h>

#define WIFI_SSID "Vignesh"
#define WIFI_PASSWORD "Password"

#define flame 26
#define led 27
#define buzzer 25
```

```
const char *serverUrl = "https://console.thingzmate.com/api/v1/device-
types/esp11/devices/esp3211/uplink"; // Replace with your server endpoint
String AuthorizationToken = "Bearer 4b847551b0644057f8f7e6480050b5b6";

void setup() {
  pinMode(flame, INPUT);
  pinMode(led, OUTPUT);
  pinMode(buzzer, OUTPUT);
  Serial.begin(115200);




// Connect to WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("Connected to WiFi");
}

void loop() {
  int value = digitalRead(flame);
  String status;

  if (value == 1) {
    status = "FIRE!!!";
    Serial.println(status);
    digitalWrite(led, HIGH);
    digitalWrite(buzzer, HIGH);
  } else {
    status = "Safe";
    Serial.println(status);
    digitalWrite(led, LOW);
    digitalWrite(buzzer, LOW);
  }

  // Send status to the cloud
  sendToCloud(status);

  delay(1000);

}
```

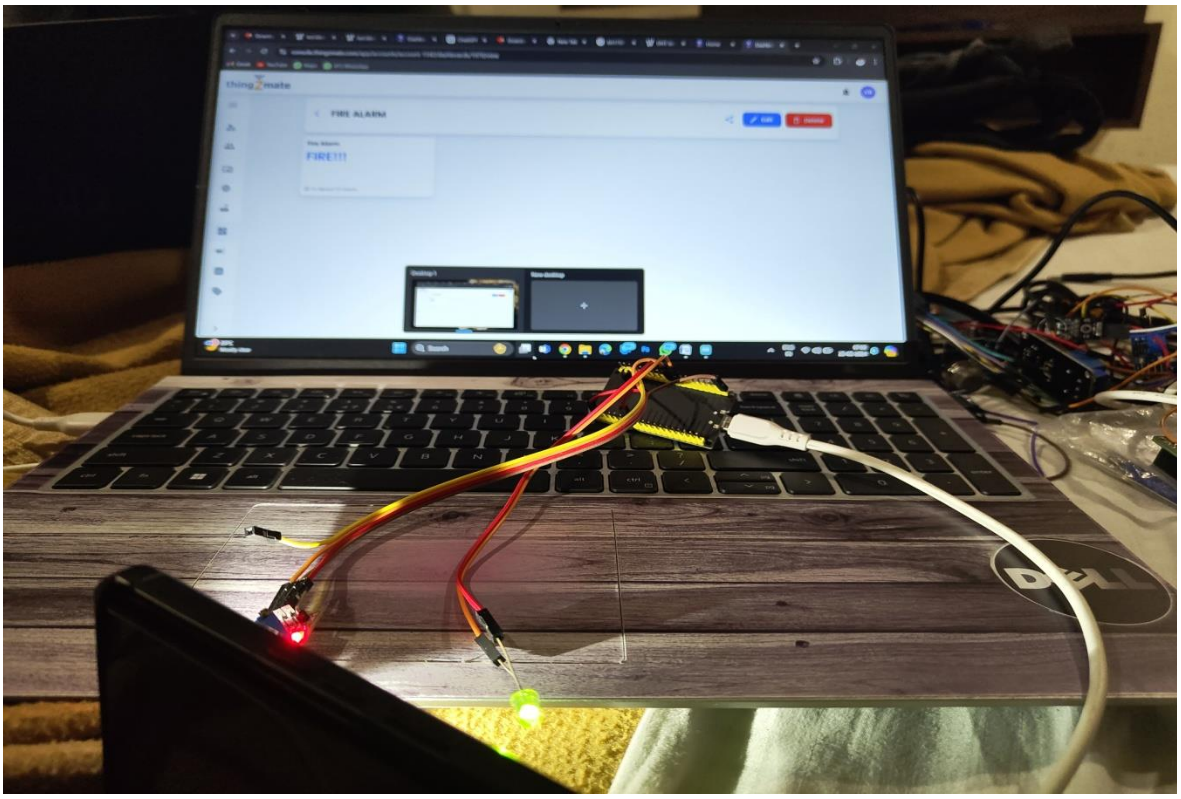
```
void sendToCloud(String status) {
  HTTPClient http;
  http.begin(serverUrl);
  http.addHeader("Content-Type", "application/json");
  http.addHeader("Authorization", AuthorizationToken);

  // Create JSON payload with the status text
  String payload = "{\"status\":\"" + status + "\"}";

  // Send POST request
  int httpResponseCode = http.POST(payload);

if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.println("HTTP Response code: " + String(httpResponseCode));
    Serial.println(response);
  } else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }

  http.end(); // Free resources
}
```

CLOUD OUTPUT:


console.thingzmate.com/app/accounts/account-1542/dashboards/1970/view
thingzmate
FIRE ALARM
Edit
Delete
Fire Alarm
Safe
in about 12 hours


console.thingzmate.com/app/accounts/account-1542/dashboards/1970/view
thingZmate
FIRE ALARM
Edit
Delete
Fire Alarm
FIRE!!!
in about 12 hours

OUTPUT RESULTS:



Testing and Results:

Testing Procedure:

- **Step 1:** Assemble the circuit on a breadboard and upload the code to the ESP32.
- **Step 2:** Power the system and verify that the ESP32 connects to WiFi.
- **Step 3:** Simulate a fire by exposing the flame sensor to a lighter or match flame.
- **Step 4:** Observe the LED and buzzer activation and check the cloud platform for the received status message.
- **Step 5:** Test the system under different conditions (e.g., varying distances and angles of the flame).

Results:

The system successfully detected the presence of fire, activated the buzzer and LED, and sent the appropriate status to the cloud server. The test confirmed that the system is reliable for fire detection and remote monitoring.

Challenges Faced:

- **WiFi Connectivity:** Ensuring stable WiFi connectivity was crucial for consistent cloud communication.
- **Sensor Sensitivity:** Adjusting the sensitivity of the flame sensor was necessary to avoid false alarms.

Conclusion:

Summary:

The project achieved its goal of developing a functional fire alarm system using ESP32. The system effectively detects fire, alerts users through a buzzer and LED, and sends real-time data to a cloud server. The design is simple, cost-effective, and can be scaled or integrated into larger IoT systems.

Future Work:

- **Enhancements:** Integrating additional sensors (e.g., smoke or temperature sensors) to improve accuracy.
- **Notifications:** Adding SMS or email alerts for immediate user