FACULTY OF ENGINEERING

ARC6173

IoT SYSTEMS & APPLICATIONS

ASSIGNMENT REPORT

T1 2024/2024

Lecturer - Dr. Tan Wooi Haw

Name	ABDELKERIM ALI HASSNA
ID	1211302792
EMAIL	1211302792@studnet.mmu.edu.my

Table of Contents

1. Introduction

Organizational culture represents a pattern of shared basic assumptions that a group learns as it solves problems of external adaptation and internal integration that has worked well enough to be considered valid and, therefore, to be taught to new members as the correct way to perceive, think, and feel in relation to those problems.

Fire and Smoke Detection System: It is just a simple detection of fire or smoke with an alert, then an action by triggering the water pump to suppress the fire. This would speed up the process and reduce the danger through a combined manufacturing of sensors for flame and smoke and the actuator is switched and controlled by the relay. The present project is, in addition to this, added with a real-time monitoring system and an alert system for remote management through the Blynk IoT platform. When also integrated with the appropriate hardware and IoT technologies, it would reduce fire accident risks by prompt response in different areas, including residential, commercial, and industrial areas.

2. Project Objectives

The theme of this project covers an automatic fire-detecting system based on combining flame and smoke sensor signals. The system shall realize the following functionalities:

Continuously monitor and detect fire or smoke.

Raise a voice alarm on the buzzer.

Display the sensor reading with real-time status on an OLED display.

Provide a Relay Module that will trigger a Water pump (fire suppression).

The system will be made more user-friendly and dependable by interfacing it with the Blynk IoT remote-sensing and notification platform.

Ensure safety, reduce human intervention, and help respond to emergencies better is what this project shall work to achieve.

3. Design and Architecture of the System

3.1 Hardware Setup Completed Components:

Fire Sensor Integration: Connect the fire sensor of MHSensorSeries to GPIO35 of ESP32. It senses infrared radiation from flames and sends a signal to the system when the fire is detected.

Smoke Sensor Integration: The GPIO34 of ESP32 has been used to attach the MQ-2 Smoke Sensor, which is capable of gas as well as smoke particle detection in the air to notify the user in case of fire outbreaks.

Indicator Lamps and Alarm Components in Simple:

Green LED: This is attached to GPIO12 to indicate powering up the system and no fire detection.

Red LED: Attached to GPIO14; to be switched on when fire or smoke is detected

3.2 Software Development:
Tasks Done:
Processing of Sensor Data: The input values for the sensors for fire and smoke are analog values.
Below values for these sensors are:
Fire Sensor Threshold: Trigger an alarm when the value < 2500.
Smoke Sensor Threshold: Trigger an alarm when the value is <= 1500.

Alarm System:
When the sensor values < threshold values, the system performs the following actions:
Turn on the Red LED.
Activate the Buzzer.
Display an alert on the OLED.
The alarm stays active for a defined period (10 seconds in this case) and then resets unless the system stabilizes. Otherwise, it stays active.

OLED Display Integration:
Shows the following messages in real-time:
Fire/Smoke not Detected under normal room conditions.
"Fire/Smoke Detected!" in case of any alarm trigger (specific for each).
Shows continuous readings from the sensors for fire and smoke.

Blynk IoT Platform:
First step was to set up the Blynk dashboard to monitor sensor readings remotely.
Over time, this dashboard was improved to give real-time notifications to users upon fire or smoke detection and also to remotely turn on the water pump.

Tasks Done:
Blynk IoT features integration has been finalized.
Sending them alerts once the system notices any smoke or fire.
Dashboard within Blynk for the remote control of the pump of water.
Carried out debugging and completed integration for proper operation.

3.3 System Flowchart:
The ensuing flowchart, offers a visual aid for grasping the initiation, hazard detection, and efficient response of the Fire and Smoke Detection System.

- **Start:**
  - Power-up initializes the system.
- **Initialize System:**
  - ESP32 microcontroller, sensors (fire and smoke), and outputs (LEDs, buzzer, relay, OLED display) are activated and initialized as components.
  - System running is indicated by the lighting up of Green LED.
- **Read Sensor Values:**
  - Values from Fire Sensor (MH-Sensor-Series) and Smoke Sensor (MQ-2) are read continuously.
  - Check if values exceed the threshold.
- **Check Threshold:**
  - The values of the sensors are compared with some predefined threshold values.
  - Fire Sensor Threshold: If value >= 2500 then Fire is detected.
  - Threshold of Smoke Sensor: Fire is detected if the value exceeds the threshold(<=1500).
  - Fire Detected? (Decision Node):
- **If Fire is detected:**
  - Enable Alarm:
  - Red LED ON for FIRE indication.
  - Buzzer sound for the alarm.
  - And regulate relay module for controlling water pump.
  - Display "Fire Detected" message on OLED.
  - If No Fire is Detected, Check Smoke.
  - Is Smoke Detected? (Decision Node):
- **If True (value > threshold):**
  - Turn the Red LED ON for Showing Smoke Detection.
  - Then, Buzzer will be energized to alarm the System.
  - Water Pump will be activated (by relay module).
  - And the "Smoke Detected" Message will Show on OLED.
  - If No Smoke is detected, The System Stabilizes.
- **Activate Water Pump:**
  - If There is Fire or Smoke, When the Relay Module Will Trigger the Water Pump.
  - The Pump Will Run Until the Alarm State Clears.
- **Display Alert on OLED:**
  - Writes On OLED: "Fire Detected!" or "Smoke Detected!" <if> Trigge<red>

  - Normal conditions - "No Fire/Smoke Detected."
  - Delay - Wait for 10 Seconds:
  - User assessment - 10-second period.
  - During this time, the alarm and pump are active.
  - Reset System: After 10 Seconds:
  - - Red LED and Buzzer off.
  - Green LED blinks - stability.
  - - Water Pump off via relay module.
  - Completion Check:
  - Check if the process is complete.
- **If yes:**
  - System resets and waits for subsequent fire or smoke detection.
- **End:**
- After resetting, the flowchart ends, looping continuously for further sensor input until another alarm condition.

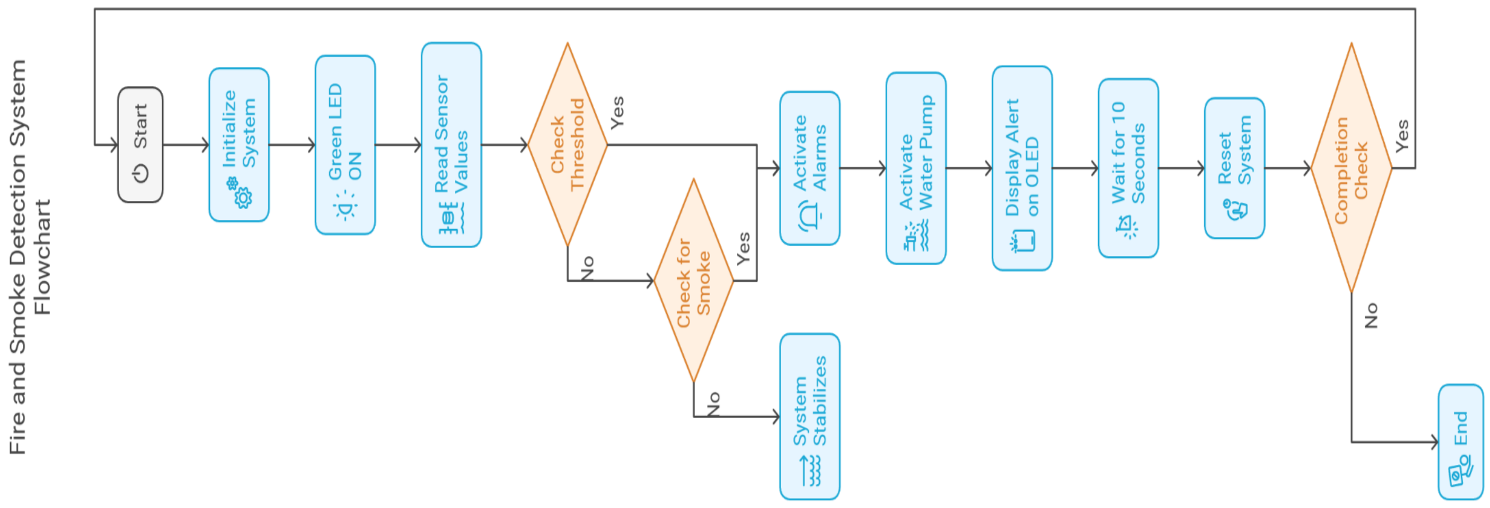
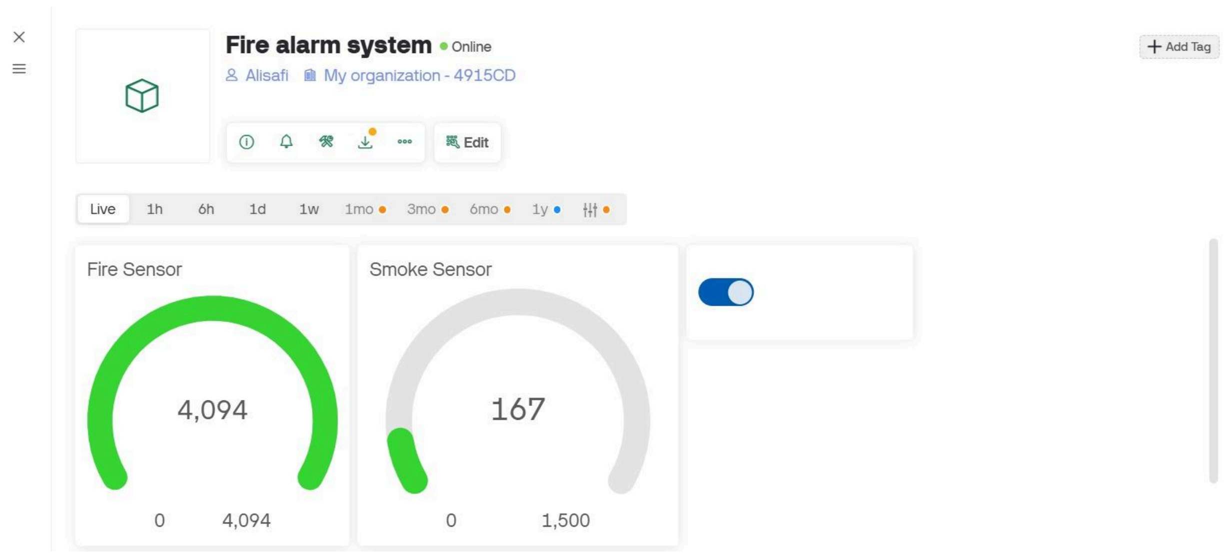


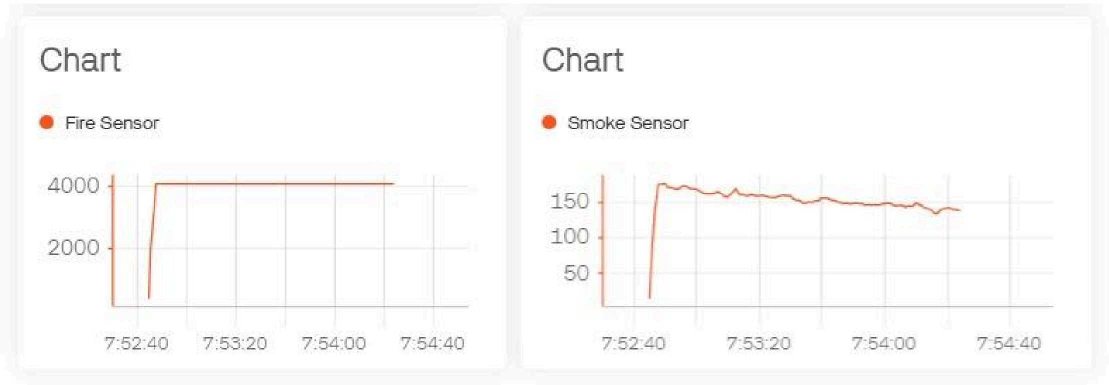
Figure 3.3: System Flowchart

3.4 Blynk IoT Dashboard

The monitoring of real-time measurements of the Fire and Smoke Detection System is shown in the Blynk IoT platform as depicted below.

The snapshot below indicates the existing values from the Fire and Smoke sensors as presented on the Blynk Dashboard. These are values constantly checked by the above system whenever there are readings from the sensors. All normal green indicators are within safe thresholds of sensors.





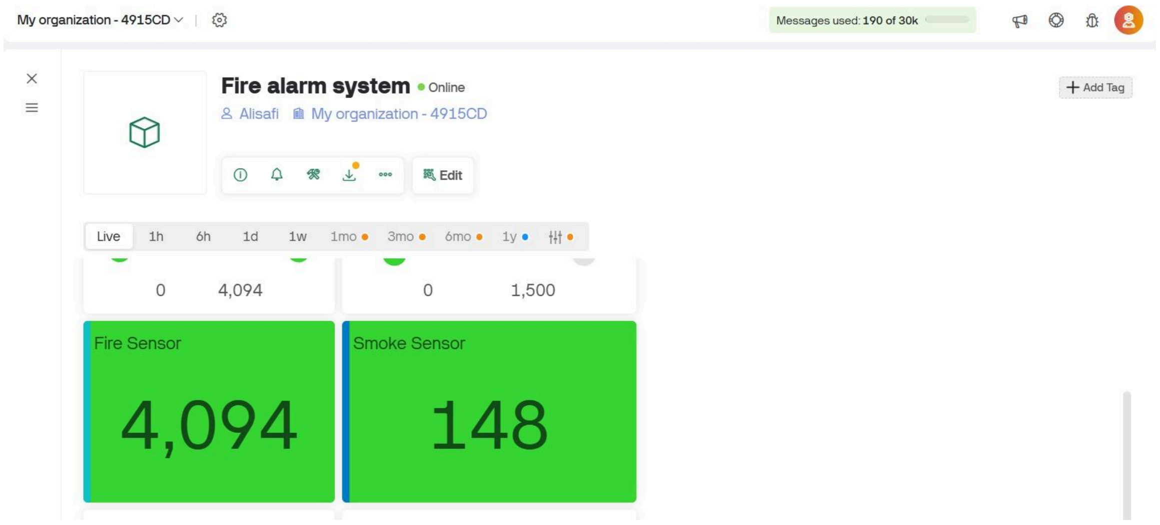


Figure 3.4.1: Fire and Smoke Sensor Status on the Blynk Dashboard

Fire Detection Triggers: Possible fire is detected, the gauge Fire Sensor turns red, crossing the value over the threshold. This can be observed in the Blynk dashboard.

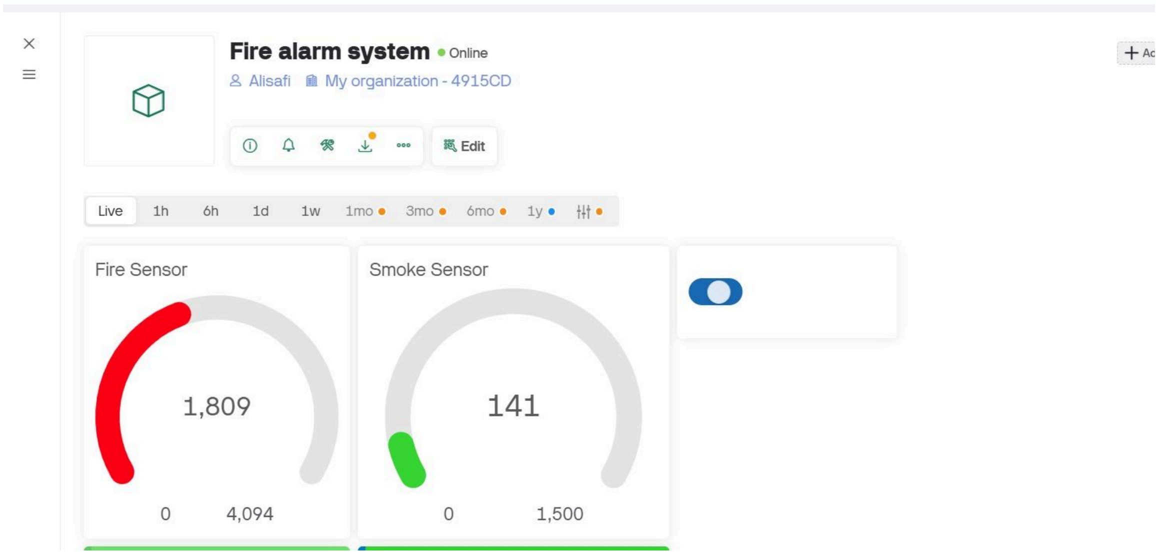


Figure 3,4,2: Fire Detection Triggered on the Blynk Dashboard

Smoke Detection Display: In addition, if smoke is detected, the reading from the Smoke Sensor turns to 1, indicating that smoke is detected. The alarm is triggered by the system and also notifications will be sent to the user by means of the Blynk platform.

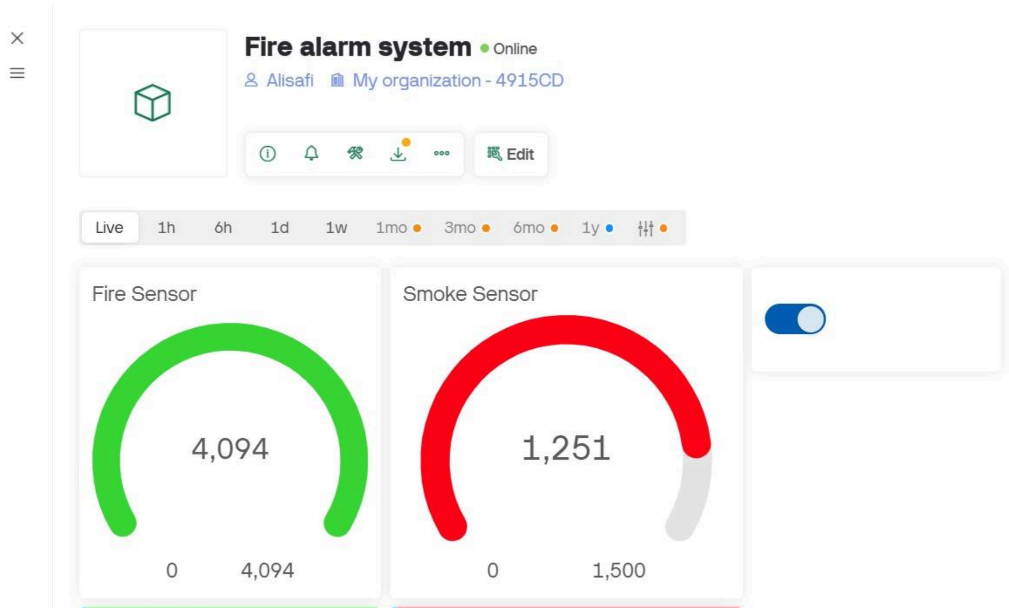


Figure 3.4.3: Smoke Detection Triggered on the Blynk Dashboard

Ending the Blynk Integration Content: The Fire and Smoke Detection System links to the Blynk IoT platform that works as your interface in real-time for watching your sensor data. Through the Blynk dashboard, users can monitor sensor status, receive alerts, and control the system— that is, activate the water pump— all from smartphones and web browsers. This offers a helpful add-on of safety and remote control for users, allowing the system to work better.

Bill of Material :

Component	**Quantity**	**Price per Unit (RM)**	**Justification**
ESP32 Microcontroller	1	35	Detects smoke or gas particles in the air, indicating an impending fire.
MH-Sensor-Series Fire Sensor	1	10	An infrared flame detector detects infrared radiation from the flames.
MQ-2 Smoke Sensor	1	12	Smoke alarms detect smoke in the environment, for early warning of potential fire hazards.
OLED Display (128x64)	1	22	Real-time display of system status, sensor readings, and alerts to users.
Green LED	1	1	This means the system is operating normally and no fire has been detected.
Red LED	1	1	Goes off: If it picks up fire or smoke, signaling an alert.
Buzzer	1	5	Makes a sound, alerting users of a possible danger such as fire or smoke.
Power Supply (9V Battery)	1	12	actuators, and components drawing up to MSC437.
Jumper Wires	1	8	The external circuitry is used to interface different devices (sensors, relays, etc.) with the ESP32 microcontroller.
Breadboard	2	8	A hardware setup is used to test components before doing a final setup.
Relay Model	1	12	Fire Suppression System: The fire suppression system is triggered based on the fire alarm and controls the activation of the water pump to suppress the fire.
Water Pump	1	10	Sprays out water when the system senses fire or smoke to battle fire.

5. Methodology

5.1 Hardware Setup:

Since the microcontroller chosen has plenty of GPIO pins and Wi-Fi support, we opted for using an ESP32 microcontroller for sensor integration with Blynk IoT communication. It was adjusted/fire/smoke sensors in a manner that the system is capable enough to identify developing problems in the environment thus setting off the alarm. Emergency responses are also integrated into via actuators that are connected to specific GPIO pins. (relay, and water pump).

5.2 Software Setup:

We coded our system using the Arduino IDE, which has specific libraries to read sensors, work with the OLED display, and communicate via Blynk. It included the Blynk library for enabling functionalities for the Blynk cloud and mobile app to work for remote control and monitoring to monitor sensor data and receive alerts. When abnormal conditions are detected by the sensors, the relay and RED_LED and buzzer should be triggered in the alarm system.

Testing the Relay and Water Pump: It should be verified whether or not the relay could initiate the water pump as a response to detecting either fire or smoke; hence, proving the functionality of the fire suppression mechanism.

Calibration of Fire and Smoke Detectors: Calibrate fire and smoke detectors to check if they can sense hazardous conditions to continue sensing until environmental conditions overpower them.

Challenges and Solutions: This part of the project highlights the challenges faced, in particular detecting fire and smoke, and the measures taken to address them up to where the proposed system can be assessed as able to provide access from a remote area.

Challenges:

Sensor Calibration: Tuning reading thresholds of the fire and smoke sensors to be more responsive, while keeping them stable against fluctuations of the environment, to minimize false positives and negatives.

Relay and Water Pump Activation: Make sure that the relay does what it is supposed to do activate the water pump when the alarm goes off under varying electrical conditions.

Solutions:

These involved manual tests and repeated tweaks of threshold values to ensure that the sensors were load-tested for calibration as well as the relay module for smooth working.

Conclusion

This completes the development of the Fire and Smoke Detection System wherein all the major features like sensor monitoring, alarm, OLED display update, and Blynk IoT functions have been completed. This would provide an easy, efficient, and automated solution towards fire prevention that can immediately raise an alarm as well as take required action to save home, office, and industry. In future, I wish to add more features but Blynk does not provide for free so I will have to check for other IoT clouds as well and make changes to the sensor places plus more tests to ensure that the system is stable.

Future Enhancements

Adjusting additional sensor emplacement and calibration for better detection in diversified environments (such as temperature sensor, gas sensor, etc.).

System scalability: What it means to extend systems to scale up for bigger installations and different environments.

Appendices

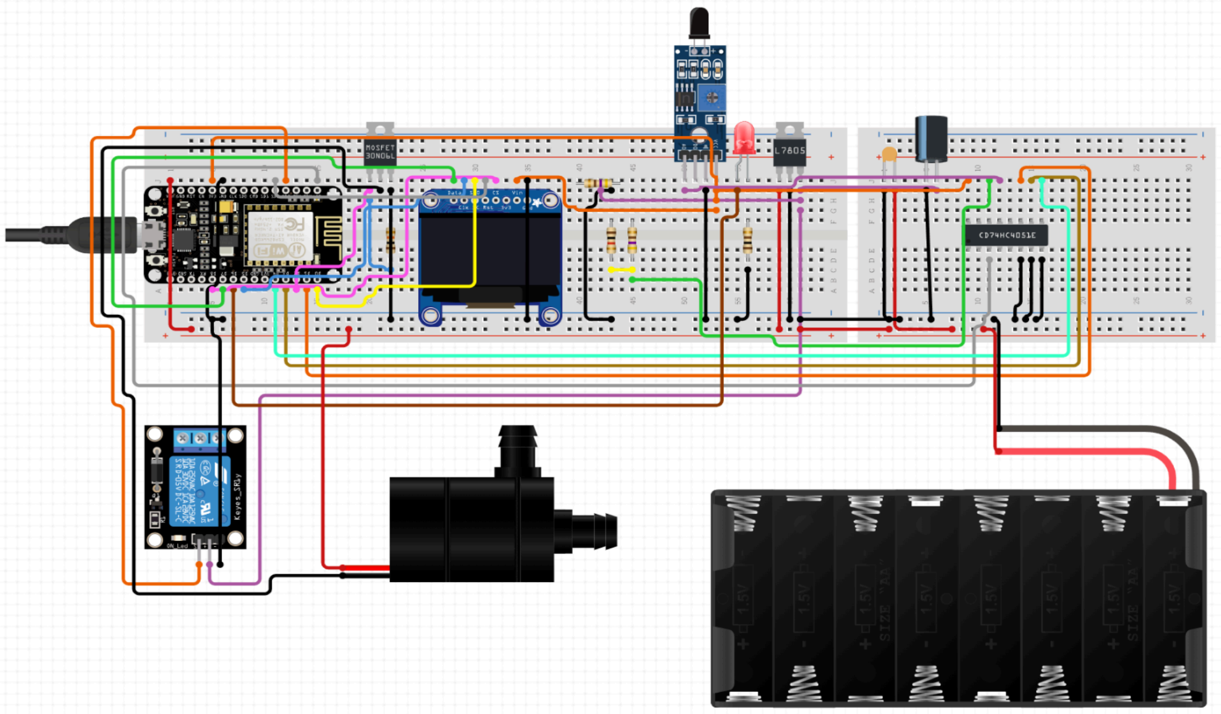
A. Source Code: Full code listing for the fire and smoke detection system, including sensor readings, display integration, and Blynk IoT communication.

```
#define BLYNK_TEMPLATE_ID "TMPL4MfJj7V5h"
#define BLYNK_TEMPLATE_NAME "ABDELERIM"
#define BLYNK_AUTH_TOKEN "5OwHNwwMoIsdxVJGShpXeltUMCqG2mTz"

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <BlynkSimpleEsp32.h>

// Pin Definitions
#define RED_LED_PIN 14     // Red LED
#define GREEN_LED_PIN 12   // Green LED
#define BUZZER_PIN 13      // Buzzer
#define FIRE_SENSOR_PIN 35 // Fire sensor (MH-Sensor-Series)
#define SMOKE_SENSOR_PIN 34 // Smoke sensor (MQ-2)
#define RELAY_PIN 27        // Relay Pin for controlling the water pump

// Wi-Fi credentials
const char* ssid = "ABRGhamry";
const char* pass = "ghsam9999";

// OLED Display Settings
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RESET);

// Fire and Smoke sensor thresholds
int fireThreshold = 2500;  // Fire sensor threshold
int smokeThreshold = 1500;  // Smoke sensor threshold

// Parameters for averaging
```

```
#define NUM_READINGS 10
int fireReadings[NUM_READINGS];
int smokeReadings[NUM_READINGS];
int fireIndex = 0;
int smokeIndex = 0;
int fireTotal = 0;
int smokeTotal = 0;
int fireAverage = 0;
int smokeAverage = 0;

// Timer for alarm duration
unsigned long lastCheckTime = 0;  // Time of the last check
unsigned long alarmStartTime = 0;  // Start time of the alarm
bool alarmActive = false;  // Flag to track if alarm is active

void setup() {
  Serial.begin(115200);  // Start serial communication

  // Initialize OLED display (using I2C address 0x3C)
  Serial.println("Initializing OLED display...");
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("OLED allocation failed"));
    while (1);  // Stay here if OLED initialization fails
  }

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.print("System Initialized");
  display.display();
  delay(2000);  // Delay to show initialization message

  pinMode(RED_LED_PIN, OUTPUT);
  pinMode(GREEN_LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(RELAY_PIN, OUTPUT);  // Set the relay pin as output

  // Start with the first Green LED ON
  digitalWrite(GREEN_LED_PIN, HIGH);
  digitalWrite(RED_LED_PIN, LOW);
  digitalWrite(BUZZER_PIN, HIGH);
  digitalWrite(RELAY_PIN, HIGH);  //  water pump is OFF by default

  // Initialize all readings to 0
  for (int i = 0; i < NUM_READINGS; i++) {
    fireReadings[i] = 0;
    smokeReadings[i] = 0;
  }

  // Initialize Wi-Fi and Blynk connection
  Serial.println("Connecting to Wi-Fi...");
  WiFi.begin(ssid, pass);  // Connect to Wi-Fi

  int attempts = 0;  // Connection attempts counter
  while (WiFi.status() != WL_CONNECTED && attempts < 10) {  // Timeout
after 10 attempts
    delay(1000);
    Serial.print(".");
    attempts++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("Connected to Wi-Fi");
  } else {
    Serial.println("Failed to connect to Wi-Fi");
    return;  // If Wi-Fi connection fails, stop setup here
  }

  // Attempt Blynk connection
  Serial.println("Connecting to Blynk...");
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);  // Connect to Blynk server

  attempts = 0;  // Reset connection attempts counter
  while (!Blynk.connected() && attempts < 10) {  // Timeout after 10
attempts
    delay(1000);
    Serial.print(".");
    attempts++;
  }

  if (Blynk.connected()) {
    Serial.println("Connected to Blynk");
  } else {
    Serial.println("Failed to connect to Blynk");
    return;  // If Blynk connection fails, stop setup here
  }
}

void loop() {
  Blynk.run();  // Run Blynk background tasks

  unsigned long currentMillis = millis();

  // Always update the sensor values continuously
  fireTotal = fireTotal - fireReadings[fireIndex];
  smokeTotal = smokeTotal - smokeReadings[smokeIndex];

  // Read the fire and smoke sensor values
  fireReadings[fireIndex] = analogRead(FIRE_SENSOR_PIN);
  smokeReadings[smokeIndex] = analogRead(SMOKE_SENSOR_PIN);

  // Add the new readings to the total
  fireTotal = fireTotal + fireReadings[fireIndex];
  smokeTotal = smokeTotal + smokeReadings[smokeIndex];

  // Move to the next index
  fireIndex = fireIndex + 1;
  smokeIndex = smokeIndex + 1;

  // If we've reached the end of the array, wrap around to the beginning
  if (fireIndex >= NUM_READINGS) {
    fireIndex = 0;
  }
  if (smokeIndex >= NUM_READINGS) {
    smokeIndex = 0;
  }
  // Calculate the average value
  fireAverage = fireTotal / NUM_READINGS;
  smokeAverage = smokeTotal / NUM_READINGS;

  // Print fire and smoke sensor averages to Serial Monitor
  Serial.print("Fire Sensor Average Value: ");
  Serial.println(fireAverage);
  Serial.print("Smoke Sensor Average Value: ");
  Serial.println(smokeAverage);

  // Display on OLED
  display.clearDisplay();
  display.setCursor(0, 0);

  bool fireDetected = false;
  bool smokeDetected = false;

  // Check for fire detection
  if (fireAverage >= fireThreshold) {
    fireDetected = true;
    digitalWrite(RED_LED_PIN, HIGH);    // Red LED ON
    digitalWrite(GREEN_LED_PIN, LOW);   // Green LED OFF
    digitalWrite(BUZZER_PIN, LOW);     // Buzzer ON
    digitalWrite(RELAY_PIN, LOW);       // Activate the relay (turn on
the water pump)
    display.print("Fire Detected! Water Pump Activated.");
    alarmStartTime = currentMillis;  // Reset the alarm start time
    alarmActive = true;
  }

  // Check for smoke detection
  if (smokeAverage >= smokeThreshold) {
    smokeDetected = true;
    digitalWrite(RED_LED_PIN, HIGH);    // Red LED ON
    digitalWrite(GREEN_LED_PIN, LOW);   // Green LED OFF
    digitalWrite(BUZZER_PIN, LOW);     // Buzzer ON
    digitalWrite(RELAY_PIN, LOW);       // Activate the relay (turn on
the water pump)
```

```
    display.print("Smoke Detected! Water Pump Activated.");
    alarmStartTime = currentMillis;  // Reset the alarm start time
    alarmActive = true;
  }

  // If neither fire nor smoke is detected
  if (!fireDetected && !smokeDetected) {
    digitalWrite(RED_LED_PIN, LOW);      // Red LED OFF
    digitalWrite(BUZZER_PIN, HIGH);       // Buzzer OFF
    digitalWrite(GREEN_LED_PIN, HIGH);  // Green LED ON
    digitalWrite(RELAY_PIN, HIGH);      // Deactivate the relay (turn
off the water pump)
    display.print("No Fire/Smoke Detected. Water Pump OFF.");
  }

  // Reset the alarm after 10 seconds if active
  if (alarmActive && (currentMillis - alarmStartTime >= 10000)) {
    alarmActive = false;  // Reset alarm state
    digitalWrite(RED_LED_PIN, LOW);       // Red LED OFF
    digitalWrite(BUZZER_PIN, HIGH);        // Buzzer OFF
    digitalWrite(GREEN_LED_PIN, HIGH);   // Green LED ON
    digitalWrite(RELAY_PIN, HIGH);       //  water pump is OFF
    Serial.println("Alarm ended.");
  }

  // Send sensor values to Blynk
  Blynk.virtualWrite(V0, fireAverage);  // Fire sensor value on V0
  Blynk.virtualWrite(V1, smokeAverage); // Smoke sensor value on V1

  // Send relay control status (Water pump) to Blynk
  if (digitalRead(RELAY_PIN) == LOW) {
    Blynk.virtualWrite(V2, 1);  // Water pump ON
  } else {
    Blynk.virtualWrite(V2, 0);  // Water pump OFF
  }

  display.setCursor(0, 20);
  display.print("Fire Value: ");
  display.println(fireAverage);
  display.setCursor(0, 40);
  display.print("Smoke Value: ");
  display.println(smokeAverage);

  display.display();  // Update OLED display
  delay(100);  // Short delay to allow for faster display updates
}
```

B. Circuit Diagram: A detailed circuit diagram showing how all hardware components are connected to the ESP32.


MOSFET 30N06L
L7805
CD74HC4051E
Keyes_SRly