**PROJECT OF INTERNET OF THINGS**

**Class: SE1816**

**Group: 03**

**Topic name: Fire Alarm**

**Objective of project:**

The goal of this project is to design and implement an automatic fire alarm system that utilizes Internet of Things (IoT) technology to ensure safety in fire-prone areas. The system incorporates the **ESP32** microcontroller, which serves as the central unit for processing and controlling the sensors and alerts. It integrates three key sensors:

* **DHT11 Sensor**: Monitors environmental conditions such as temperature and humidity. These parameters are crucial in detecting potential fire risks.
* **MQ2 Sensor**: Detects the presence of gases and smoke in the air, which are strong indicators of fire hazards.

The system continuously monitors the environment, and once abnormal levels of temperature, humidity, or gas/smoke are detected, it triggers immediate alerts to warn of danger. These warnings are issued through **LED lights** (to signal the presence of danger) and a **buzzer** (to provide an audible alarm).

Additionally, the system displays real-time data on an **LCD screen**, offering an easily accessible way to view the current temperature, humidity, and gas levels in the environment. This helps users make informed decisions to act swiftly in case of a fire risk.

Overall, the objective is to enhance safety by providing real-time monitoring and alerts in environments that are at risk of fire, thus helping to prevent damage and protect lives and property. The system can also be integrated with a cloud platform like **Blynk** for remote monitoring, allowing users to keep track of the environment even from a distance.

**Member**

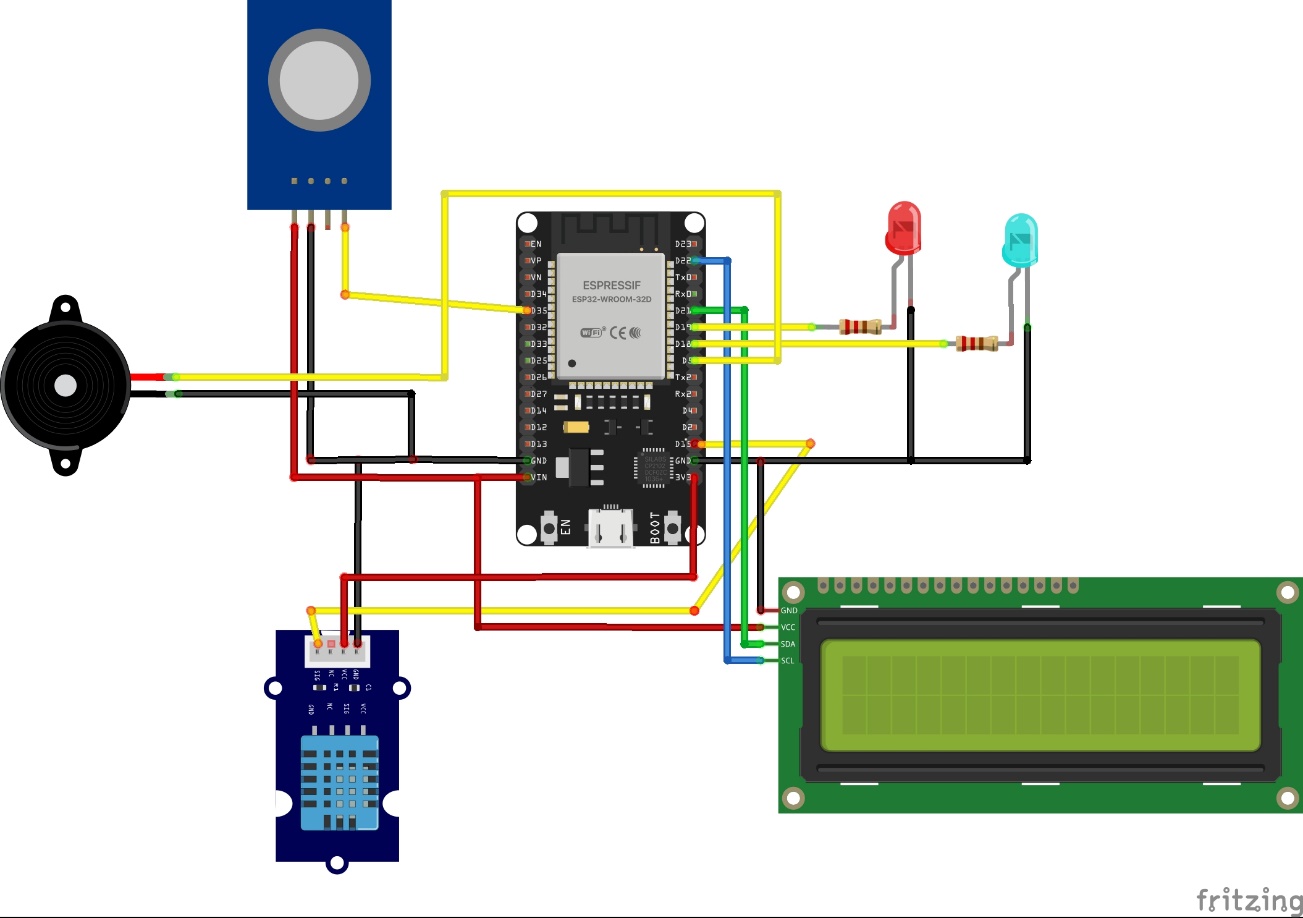
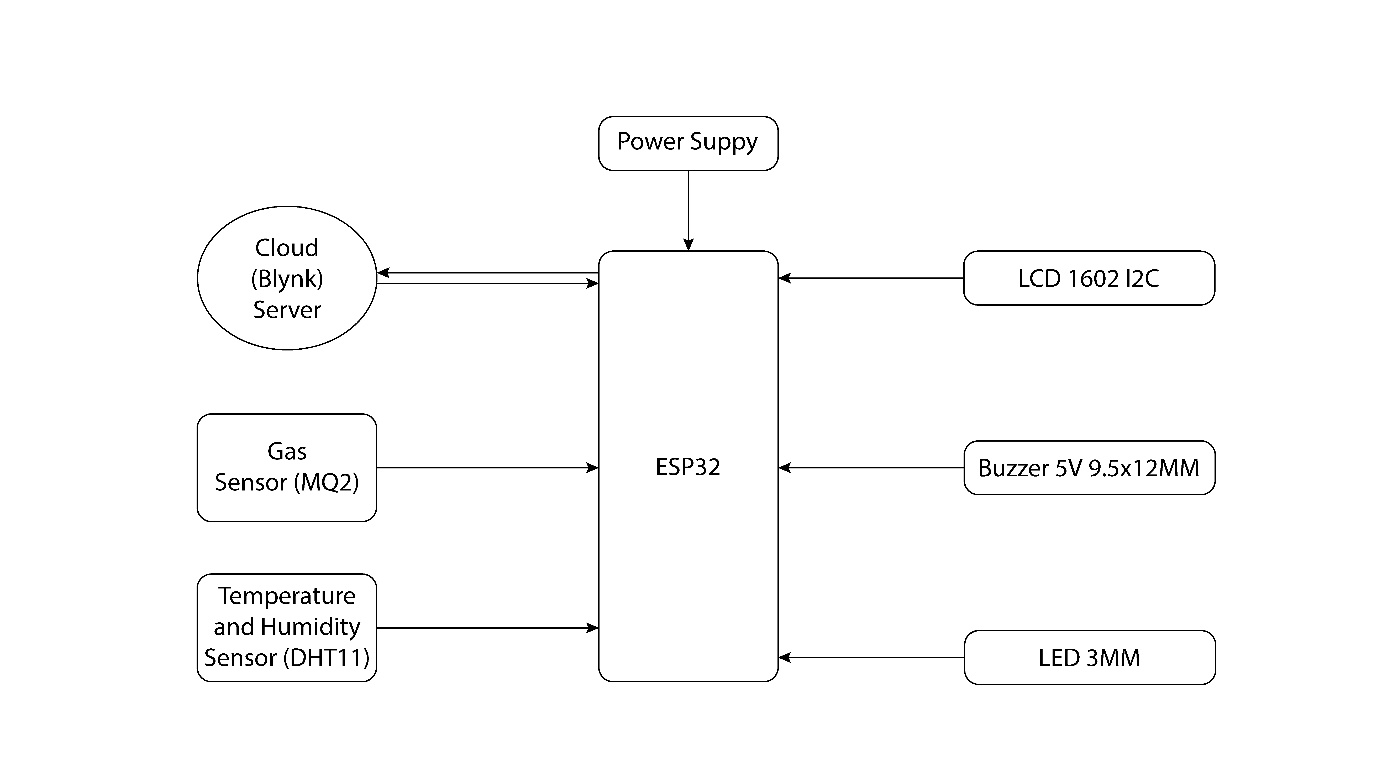
|  |  |  |  |
| --- | --- | --- | --- |
| No. | Roll number | Fullname | Contribution percentage\* |
| 1  2  3  4  5 | CE180905  CE180074  CE181389  CE181566  CE180621 | Lê Anh Tuấn  Phạm Tấn Hưng  Phạm Thanh Điền  Nguyễn Thị Như Huỳnh  Lương Thị Diễm | 30%  15%  15%  20%  20% |

The sum of this part is 100%

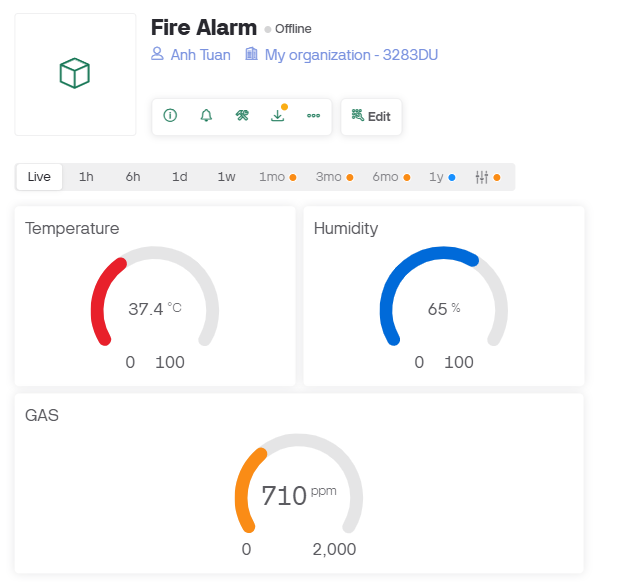
**Hardware Required:**

* Kit Wifi ESP-32 ESP-WROOM-32S
* Voltage: ESP32 works well with 5V supply (via USB port or external source).
* Power connection: You can supply power to ESP32 via USB port (5V and GND) or use a power converter from 5V to 3.3V (ESP32 working application).
* MQ2 Gas Sensor Module
* Operating voltage: 5V (usually 5V or 3.3V is used for modules, but the output signal needs to be adjusted).
* Gas detection range: 10-1000 ppm for CO, 300-10000 ppm for methane, alcohol, and smoke.
* Compatible sensors: The sensor can detect various gases and output analog or digital signals.
* Stabilization time: About 24 hours after starting, the MQ2 sensor needs time to stabilize.
* DHT11 temperature - humidity sensor
* Operating voltage: 3.3V - 5V (suitable for many microcontrollers such as ESP32 and Arduino).
* Measurement temperature range: 0°C to 50°C with accuracy ±2°C.
* Measurement humidity range: 20% to 90% with accuracy ±5%.
* Communication: DHT11 uses 1-wire communication (only one pin is needed to communicate with the microcontroller).
* Response time: About 1 second per data reading.
* 3MM Red & Blue SS LED x2
* Size: 3mm, compact and easy to integrate into electronic circuits.
* Color: LED has two main colors:
* Red: Usually used to indicate dangerous or warning status.
* Blue: Can be used to indicate normal or safe status.
* Operating voltage:
  + Red: About 2.0V to 2.2V.
  + Blue: About 3.0V to 3.2V.
* Current: About 20mA (LED consumption current, usually limited by resistor to protect LED from damage).
* Type: High quality LED with stable operation and long life.
* LCD1602 IC2 Display 5V
* Size: 16 x 2 (16 characters x 2 lines).
* Operating voltage: 5V (can use 5V power from ESP32 or external power source).
* I2C communication: Using I2C protocol, helps reduce the number of connection pins, only need 4 wires: VCC, GND, SDA and SCL.
* Display: Display letters and characters on LCD screen with black background and white light letters.
* Low power consumption: LCD1602 screen is very energy efficient, suitable for projects requiring high energy efficiency.
* Buzzer 9.5x12MM 5V
* Size: 9.5mm x 12mm, compact size, easy to integrate into electronic circuits.
* Operating voltage: 5V (usually using power from ESP32 or external 5V power source).
* Current consumption: About 30mA - 50mA when operating.
* Audio frequency: About 2kHz - 4kHz (depending on the type of buzzer, usually produces loud and easy-to-hear sound).
* Type: Active buzzer, capable of emitting sound as soon as power is supplied.
* Wire and Connectors
* Number of pins: 40 pins, each pin can be used to transmit signals or power devices.
* Length: 20 cm, just enough to connect modules, sensors or displays without using too much space.
* Wire type: Soft, flexible wire, resistant to temperature and working environment in electronic projects.
* Connector: The wire has DuPont connectors on both ends (female and male), making it easy to connect to GPIO pins of microcontrollers or modules.

**Circuit:**



**Blynk web server**

****

**Code:**

|  |
| --- |
| #define BLYNK\_TEMPLATE\_ID "TMPL6zt0MVFYt" // Define Template ID on Blynk  #define BLYNK\_TEMPLATE\_NAME "Fire Alarm" // Name of the project on Blynk  #define BLYNK\_AUTH\_TOKEN "Inx-92dSRFRUHBUQQxkkx3u1VAMw8qgz" // Blynk Auth Token  #include <WiFi.h> // Wi-Fi library to connect ESP32 to the network  #include <WiFiClient.h> // Wi-Fi client library  #include <BlynkSimpleEsp32.h> // Blynk library for ESP32  #include <Wire.h> // I2C library  #include <LiquidCrystal\_I2C.h> // LCD I2C library  #include <DHT.h> // DHT11 sensor library  #include <NTPClient.h> // NTP library to get real time  #include <WiFiUDP.h> // UDP for NTPClient  // Define LCD 1602 I2C (I2C address might be 0x27 or 0x3F)  LiquidCrystal\_I2C lcd(0x27, 16, 2);  // Khai báo DHT11  #define DHTPIN 15  #define DHTTYPE DHT11 // Type of DHT sensor  DHT dht(DHTPIN, DHTTYPE); // Create DHT sensor object with GPIO15 pin  // MQ2 sensor (connected to ADC pin on ESP32)  const int gasSensor = 35; // Chân ADC ESP32  // Define LEDs and Buzzer pins  const int ledRed = 19;  const int ledGreen = 18;  const int buzzer = 5;  // Wi-Fi credentials  char auth[] = BLYNK\_AUTH\_TOKEN;  char ssid[] = "Anh Tuan";  char pass[] = "12344233";  // NTP setup  const char\* ntpServer = "pool.ntp.org" // NTP server  const long gmtOffset\_sec = 7 \* 3600; // Timezone offset for GMT+7  const int daylightOffset\_sec = 0; // No daylight saving time for GMT+7  WiFiUDP udp; // UDP for NTPClient  NTPClient timeClient(udp, ntpServer, gmtOffset\_sec, daylightOffset\_sec); // Initialize NTP client  void setup() {  // Start Serial Monitor for debugging  Serial.begin(115200);  dht.begin(); // Initialize DHT11 sensor  // Start Blynk connection  Blynk.begin(auth, ssid, pass);    // Configure sensors and devices  pinMode(gasSensor, INPUT); // MQ2 gas sensor as input  pinMode(ledRed, OUTPUT); // Red LED as output  pinMode(ledGreen, OUTPUT); // Green LED as output  pinMode(buzzer, OUTPUT); // Buzzer as output  // Initialize LCD  lcd.init(); // Initialize LCD  lcd.backlight(); // Turn on LCD backlight  lcd.setCursor(0, 0); // Set cursor to row 1, column 0  lcd.print("Fire Alarm Ready!"); // Display message "Fire Alarm Ready!"  delay(2000); // Wait for 2 seconds to read the message  // Initialize NTP client  timeClient.begin();  timeClient.update(); // Get the initial time  }  void loop() {  Blynk.run(); // Run Blynk to process the app data  // Read data from DHT11 sensor (temperature and humidity)  float temp = dht.readTemperature();  int humidity = (int)dht.readHumidity();  // Read value from MQ2 sensor (gas level)  int gasValue = analogRead(gasSensor);  // Get current time (hour, minute, second) from NTP server  timeClient.update(); // Update time from NTP server  // Get current time as string (HH:MM:SS)  String currentTime = timeClient.getFormattedTime();  // Check if DHT11 sensor data is valid  if (isnan(temp) || isnan(humidity)) {  Serial.println("ERROR: Unable to read data from DHT11!");  lcd.clear();  lcd.setCursor(0, 0);  lcd.print("DHT11 ERROR!");  delay(2000);  return;  }  // Send data to Blynk App  Blynk.virtualWrite(V0, temp);  Blynk.virtualWrite(V1, humidity);  Blynk.virtualWrite(V3, gasValue);  // Print data to Serial Monitor  Serial.print("\nTime: ");  Serial.println(currentTime); // Print current time  Serial.print("Temperature: "); Serial.print(temp); Serial.print("°C");  Serial.print("\nHumidity: "); Serial.print(humidity); Serial.print("%");  Serial.print("\nGas: "); Serial.print(gasValue); Serial.print("ppm\n");    // Display data on LCD  lcd.clear();  lcd.setCursor(0, 0);  lcd.print("T:"); lcd.print(temp); lcd.print("C ");  lcd.print(" H:"); lcd.print(humidity); lcd.print("%");  lcd.setCursor(0, 1);  lcd.print("Gas: "); lcd.print(gasValue); lcd.print("ppm");  // Fire alarm conditions  if (gasValue > 900 || temp > 33) {  digitalWrite(ledRed, HIGH);  digitalWrite(ledGreen, LOW);  digitalWrite(buzzer, HIGH);  lcd.clear();  lcd.setCursor(0, 0);  lcd.print("WARNING!");  lcd.setCursor(0, 1);  lcd.print("CHECK MY AREA!");  Serial.println("\nWARNING: Gas or high temperature detected!");  } else {  digitalWrite(ledRed, LOW);  digitalWrite(ledGreen, HIGH);  digitalWrite(buzzer, LOW);  }  delay(2000); // DHT11 requires at least a 2-second delay to read data accurately  } |

**Link video demo:**

**https://drive.google.com/file/d/1HimW-7FQ-NXW52KPwjxDm7mC5nhLEguH/view?usp=sharing**