TRƯỜNG ĐẠI HỌC KHOA HỌC

KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN MÔN HỌC

PHÁT TRIỂN ỨNG DỤNG IOT

<Tạo hệ thống báo cháy với ESP32>

Giáo viên hướng dẫn:

Sinh viên thực hiện:

Lớp: Phát triển ứng dụng IoT

- Nhóm 4

Năm học: 2024 - 2025

Huế, 04/2025

Mục Lục

1.MỞ ĐẦU

Trong thời đại công nghệ 4.0 hiện nay, hỏa hoạn là một trong những thảm họa gây thiệt hại nghiêm trọng về tài sản và con người, đặc biệt trong các khu dân cư, nhà xưởng và tòa nhà cao tầng. Việc phát hiện sớm nguy cơ cháy có ý nghĩa quan trọng trong công tác phòng cháy chữa cháy, giúp giảm thiểu rủi ro và thiệt hại. Nhờ sự phát triển của công nghệ IoT (Internet of Things), các hệ thống báo cháy hiện đại ngày càng trở nên thông minh và hiệu quả hơn, mang lại giải pháp giám sát từ xa và cảnh báo kịp thời. Trong bài tiểu luận này, chúng tôi đề xuất một hệ thống báo cháy sử dụng vi điều khiển ESP32 kết hợp với cảm biến khói và nhiệt độ. Hệ thống có khả năng phát hiện sớm các dấu hiệu của hỏa hoạn thông qua việc đo lường nồng độ khói và nhiệt độ môi trường. Khi các thông số này vượt ngưỡng an toàn, hệ thống sẽ gửi cảnh báo thông qua giao thức MQTT (Message Queuing Telemetry Transport) đến các thiết bị giám sát như điện thoại thông minh hoặc trung tâm điều khiển. Hệ thống này có nhiều ưu điểm như chi phí thấp, dễ triển khai và khả năng kết nối với các nền tảng IoT phổ biến. Bên cạnh đó, việc sử dụng giao thức MQTT giúp tối ưu hóa hiệu suất truyền tải dữ liệu, đảm bảo thông tin được gửi đi nhanh chóng và chính xác. Bài tiểu luận sẽ đi sâu vào các thành phần chính của hệ thống, nguyên lý hoạt động, cách triển khai thực tế và những lợi ích mà hệ thống mang lại. Qua đó, chúng tôi hy vọng rằng hệ thống báo cháy này sẽ là một giải pháp hữu ích, góp phần nâng cao hiệu quả công tác phòng cháy chữa cháy trong thực tế.

2. NỘI DUNG CỦA ĐỀ TÀI

2.1. Các nội dung thực hiện

1. Cấu hình Wifi: STA để kết nối internet.

2. Cấu hình web server để xử lý chức năng: cập nhật WiFi cho ESP32.

3. Cấu hình cảm biến khói: Sử dụng cảm biến MQ-2 để phát hiện khói và khí gas.

4. Cấu hình cảm biến nhiệt độ: Sử dụng cảm biến DHT22 để đo nhiệt độ và độ ẩm.

5. Sử dụng buzzer để kích hoạt còi báo động khi phát hiện nguy cơ cháy.

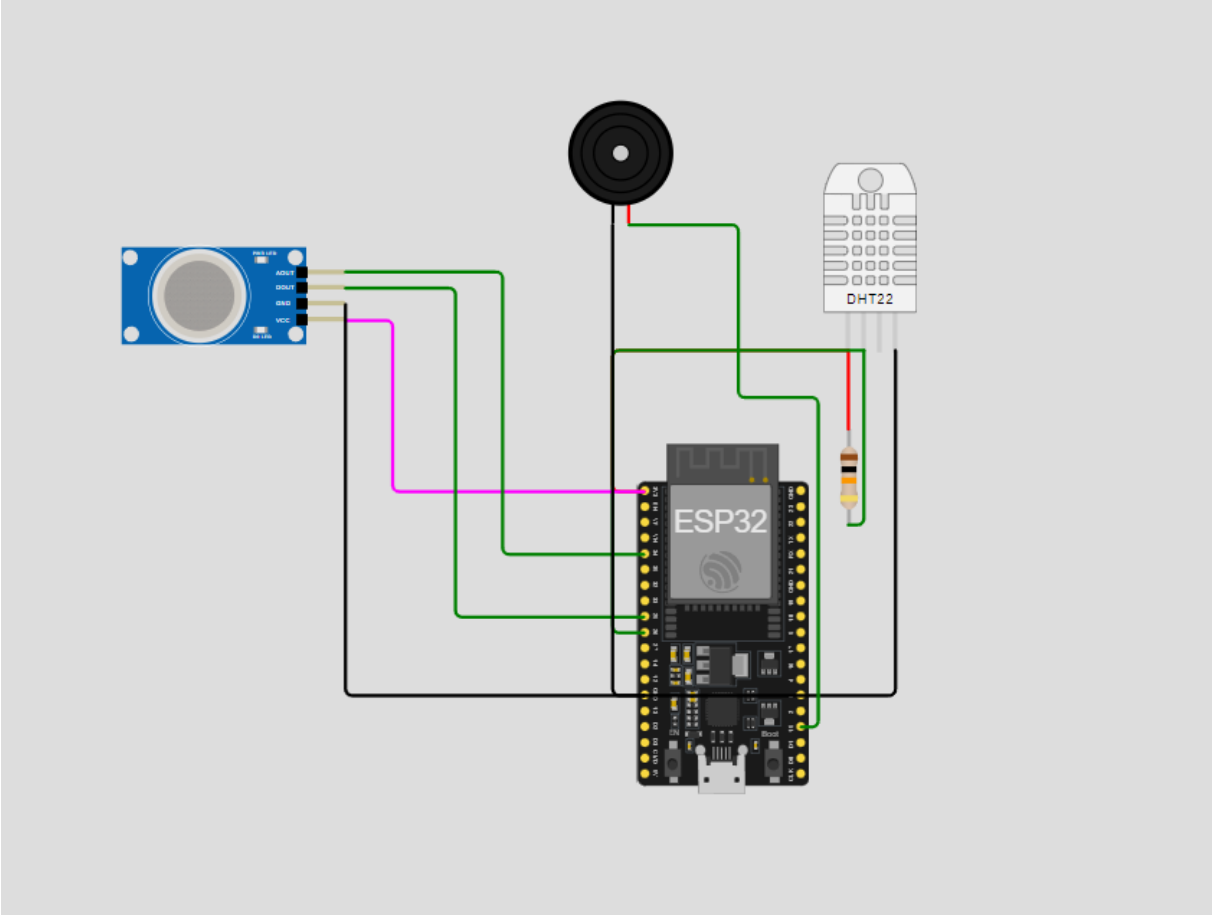
6. Cấu hình để gửi cảnh báo qua MQTT (Message Queuing Telemetry Transport) đến các thiết bị giám sát .

2.2. Thiết kế mạch IoT

2.2.1. Sơ đồ logic của hệ thống

Hình 1 cho thấy sơ đồ logic kết nối các thiết bị ngoại vi vào mạch điều khiển ESP32. Các thiết bị ngoại vi được sử dụng trong hệ thống như sau:

- Cảm biến nhiệt độ: Sử dụng cảm biến DHT22 để đo nhiệt dộ và độ ẩm, kết nối cảm biến với ESP32 qua các chân GPIO.
- buzzer: bật còi báo động khi phát hiện nguy cơ cháy.
- Cảm biến khói: Sử dụng cảm biến MQ-2 để phát hiện khói và khí gas, kết nối cảm biến với ESP32 để nhận tín hiệu cảnh báo.



Hình 1. Sơ đồ logic kết nối các thiết bị ngoại vi vào vi điều khiển ESP32

2.2.2. Chức năng cơ bản của các linh kiện

- Cảm biến DHT22:

- Đo nhiệt độ: DHT22 có khả năng đo nhiệt độ trong khoảng từ -40°C đến 80°C với độ chính xác khoảng ±0.5°C.
- Đo độ ẩm: Cảm biến này đo độ ẩm không khí trong khoảng từ 0% đến 100% với độ chính xác khoảng ±2-5% RH.
- Giao tiếp đơn giản: DHT22 sử dụng giao thức giao tiếp một dây (1-Wire), giúp việc kết nối và lập trình trở nên dễ dàng.
- Thời gian phản hồi: Thời gian phản hồi của DHT22 là khoảng 2 giây, nghĩa là nó có thể cung cấp dữ liệu mới mỗi 2 giây.

- Cảm biến MQ-2:

- Phát hiện khí gas: MQ-2 có khả năng phát hiện nhiều loại khí khác nhau, bao gồm khí gas tự nhiên (CH4), khí propane, khí butane, khói, và các khí độc hại khác.
- Đo nồng độ khí: Cảm biến này có thể đo nồng độ khí trong khoảng từ 300 đến 10,000 ppm (parts per million) tùy thuộc vào loại khí.
- Giao tiếp analog và digital: MQ-2 có thể cung cấp tín hiệu đầu ra analog (để đo nồng độ khí) và tín hiệu đầu ra digital (để phát hiện ngưỡng khí).
- Thời gian khởi động: Cảm biến MQ-2 cần thời gian khởi động khoảng 20 giây để ổn định trước khi có thể cung cấp dữ liệu chính xác.

- Chức năng buzzer:

  Phát âm thanh cảnh báo khi phát hiện nguy cơ cháy.

- Kết nối mạng và gửi cảnh báo qua MQTT(Message Queuing Telemetry Transport) đến các thiết bị giám sát :

  Kết nối mạng và gửi cảnh báo qua MQTT(Message Queuing Telemetry Transport) đến các thiết bị giám sát là một phần quan trọng để ESP32 có thể gửi cảnh báo với các thiết bị giám sát , ví dụ như việc nhận cảnh báo nhiệt độ tăng đột ngột cao từ cảm biến DHT22 để gửi cảnh báo qua MQTT(Message Queuing Telemetry Transport) đến các thiết bị giám sát như điện thoại di động,máy tính,...

- Quản lý kết nối và cấu hình hệ thống:

  Quản lý kết nối và cấu hình hệ thống là chức năng đảm bảo rằng ESP32 có thể lưu trữ và sử dụng thông tin cấu hình (như SSID, mật khẩu wifi) trong suốt quá trình hoạt động.

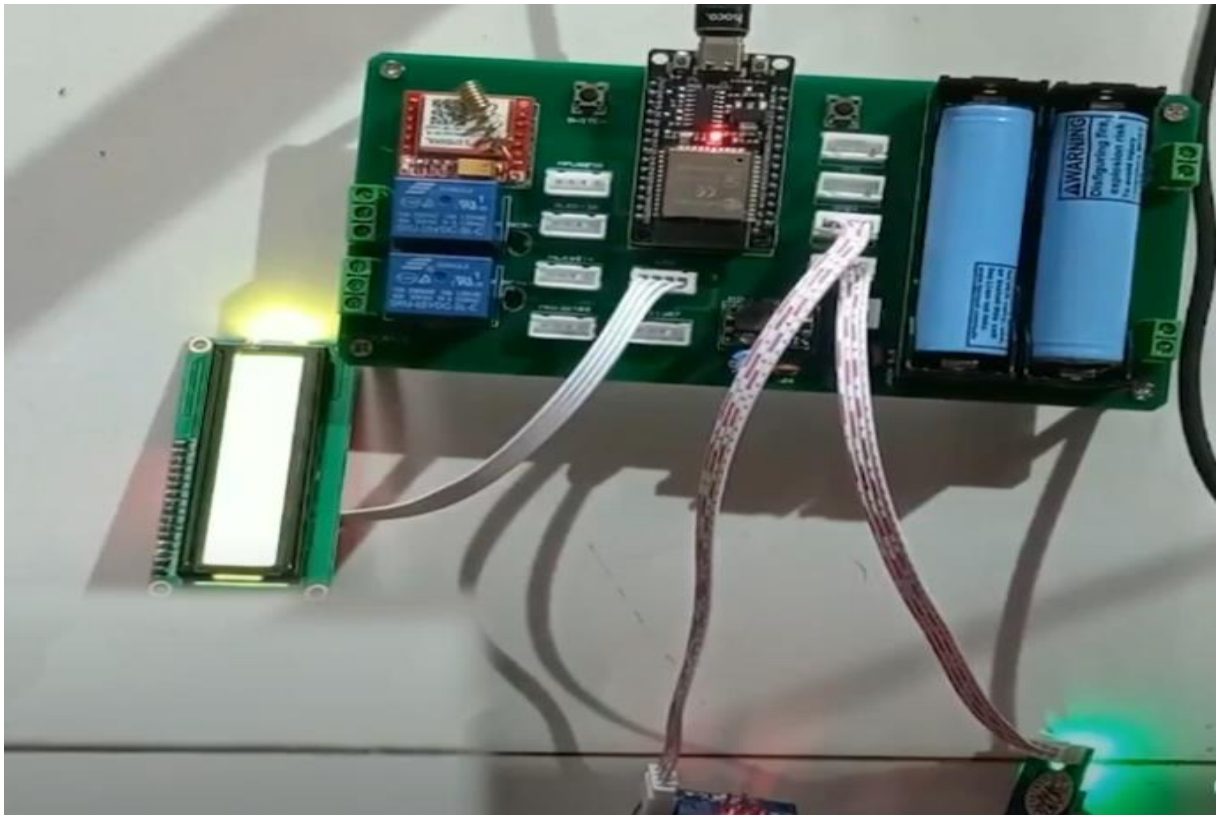
2.2.3. Nguyên lý hoạt động

Hệ thống báo cháy với ESP32 hoạt động bằng cách liên tục giám sát mức độ khói và nhiệt độ trong môi trường thông qua cảm biến MQ-2 và DHT22. Cảm biến MQ-2 đo nồng độ khí gas và khói, trong khi DHT22 theo dõi nhiệt độ. ESP32 thu thập dữ

liệu từ hai cảm biến này và so sánh với ngưỡng cảnh báo đã thiết lập. Nếu mức khói vượt quá giới hạn hoặc nhiệt độ cao bất thường (ví dụ: trên 60°C), hệ thống sẽ kích hoạt còi báo động(buzzer**)** để cảnh báo ngay tại chỗ. Đồng thời, ESP32 gửi thông báo qua giao thức MQTT đến một máy chủ hoặc ứng dụng giám sát trên điện thoại, giúp người dùng nhận cảnh báo từ xa nhờ khả năng kết nối WiFi giúp phản ứng nhanh hơn trong trường hợp có hỏa hoạn.

2.2.4. Mạch vật lý

Thiết bị vật lý được dùng trong hệ thống như hình 3



Hình 3. Mạch vật lý của hệ thống

3. KẾT QUẢ THỰC HIỆN

Hệ thống đã được thử nghiệm trong môi trường thực tế, Kết quả thử nghiệm cho thấy hệ thống hoạt động ổn định trong các điều kiện thực tế. Cụ thể:

- Về kết nối Wifi: Thời gian khởi động nhanh chóng, thiết bị kết nối nhanh với mạng wifi và duy trì kết nối ổn định trong quá trình hoạt động.
- Về cảm biến DHT22: Khi được tích hợp vào hệ thống, DHT22 hoạt động bằng cách gửi tín hiệu đến vi điều khiển (như ESP32 hoặc Arduino) để truyền tải thông tin về nhiệt độ và độ ẩm. Kết quả đo được hiển thị trên màn hình hoặc gửi qua mạng Internet để người dùng có thể theo dõi từ xa. Độ chính xác của DHT22, với sai số khoảng ±0.5°C cho nhiệt độ và ±2-5% RH cho độ ẩm, cho phép người dùng có được thông tin chi tiết và đáng tin cậy về điều kiện môi trường.
- Về cảm biến MQ-2: Cảm biến MQ-2 đã cho thấy kết quả thực hiện ấn tượng trong việc phát hiện và đo lường nồng độ các loại khí gas và khói, góp phần quan trọng trong việc đảm bảo an toàn cho con người và môi trường. Với khả năng phát hiện nhiều loại khí như khí gas tự nhiên (CH4), khí propane, khí butane, và khói, cảm biến MQ-2 hoạt động bằng cách cảm nhận sự thay đổi nồng độ khí trong không khí. Khi nồng độ khí vượt quá ngưỡng an toàn, cảm biến sẽ gửi tín hiệu đến vi điều khiển (như ESP32 hoặc Arduino), cho phép hệ thống thực hiện các hành động cần thiết.
- Về vi mạch ESP32: Khi phát hiện nguy cơ cháy, ESP32 sẽ gửi thông báo cảnh báo đến máy chủ MQTT qua kết nối Wi-Fi. Điều này cho phép hệ thống truyền tải thông tin một cách nhanh chóng và hiệu quả. Mạch sẽ gửi thông báo đến các thiết bị giám sát như smartphone hoặc máy tính, giúp người dùng nhận biết kịp thời về tình trạng an toàn trong không gian sống hoặc làm việc của họ.
- Mạch có thể được lập trình để ghi nhận và lưu trữ dữ liệu về nồng độ khí và thời gian phát hiện, giúp người dùng theo dõi và phân tích tình trạng an toàn theo thời gian.

Hình 4. Giao diện trang Web khi chưa kết nối wifi

Giao diện của trang web như Hình 4, hiển thị đầy đủ các thông tin về việc kết nối wifi. Với Hình 5 thì đây là giao diện của trang web khi đã kết nối wifi thành công.

Hình 5. Giao diện trang Web khi kết nối wifi thành công

Hình 6. Gửi cảnh báo qua MQTT (Message Queuing Telematry Transport) đến thiết bị giám sát bằng tin nhắn.

Hình 7. Gửi cảnh báo qua MQTT (Message Queuing Telematry Transport) đến thiết bị giám sát bằng cuộc gọi.

4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

- Các công việc đã thực hiện được:

+ Mạch đã sử dụng cảm biến DHT22 đã thực hiện nhiều công việc quan trọng trong việc đo lường và giám sát điều kiện môi trường như đo nhiệt độ trong khoảng từ -40°C đến 80°C với độ chính xác cao (±0.5°C),đo độ ẩm không khí trong khoảng từ 0% đến 100% với độ chính xác khoảng ±2-5% RH rồi gửi dữ liệu về nhiệt độ và độ ẩm đến máy chủ hoặc ứng dụng di động. Điều này cho phép người dùng theo dõi điều kiện môi trường từ xa.

+ Mạch đã sử dụng cảm biến MQ-2 để phát hiện sự hiện diện của các loại khí gas như khí gas tự nhiên (CH4), khí propane, khí butane, và khói. Khi nồng độ khí vượt quá ngưỡng an toàn, cảm biến sẽ gửi tín hiệu đến vi điều khiển.

+ Mạch có thể được kết nối với các thiết bị cảnh báo khác như còi báo động hoặc đèn nhấp nháy, tự động kích hoạt khi phát hiện nguy cơ cháy, nhằm tăng cường cảnh báo cho người dùng.

+ Mạch đã sử dụng ESP32 để nhận tín hiệu từ cảm biến MQ-2. ESP32 thực hiện việc xử lý tín hiệu, xác định xem có nguy cơ cháy hay không dựa trên dữ liệu từ cảm biến.

+ Khi phát hiện nguy cơ cháy, ESP32 đã gửi thông báo cảnh báo đến máy chủ MQTT qua kết nối Wi-Fi. Điều này cho phép hệ thống truyền tải thông tin một cách nhanh chóng và hiệu quả.

+ Mạch đã gửi thông báo đến các thiết bị giám sát như smartphone hoặc máy tính, giúp người dùng nhận biết kịp thời về tình trạng an toàn trong không gian sống hoặc làm việc của họ.

+ Mạch có thể được lập trình để ghi nhận và lưu trữ dữ liệu về nồng độ khí và thời gian phát hiện, giúp người dùng theo dõi và phân tích tình trạng an toàn theo thời gian.

- Các công việc chưa thực hiện được:

+ Phát hiện ánh sáng từ đám cháy.

+ Chữa cháy tự động bằng relay hoặc bơm nước.

- Hướng phát triển:

+ Triển khai thương mại hóa.

+ Tích hợp thêm cảm biến lửa để phát hiện ánh sáng từ đám cháy.

+ Điều khiển hệ thống chữa cháy tự động bằng relay hoặc bơm nước.

CÁC PHỤ LỤC

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

// **Cấu hình cảm biến**
#define DHTPIN 4          // Chân DATA của DHT22 nối với GPIO4 của ESP32
#define DHTTYPE DHT22     // Loại cảm biến DHT22
#define MQ2PIN 34         // Chân Analog của cảm biến MQ-2 nối với GPIO34
#define BUZZER_PIN 25     // Còi báo động nối với GPIO25

DHT dht(DHTPIN, DHTTYPE);

// **Thông tin WiFi**
const char* ssid = "YOUR_WIFI_SSID";
const char* password = "YOUR_WIFI_PASSWORD";

// **Thông tin MQTT**
const char* mqtt_server = "broker.hivemq.com";
const char* mqtt_topic = "fire/alarm";  // Chủ đề gửi cảnh báo
const char* mqtt_status_topic = "fire/status";  // Chủ đề gửi trạng thái
const char* device_id = "ESP32_FireSystem";  // ID thiết bị trên MQTT

WiFiClient espClient;
PubSubClient client(espClient);

// **Biến kiểm soát trạng thái hệ thống**
bool fire_detected = false;  // Trạng thái cảnh báo cháy
```

```
unsigned long lastMessageTime = 0; // Lưu thời gian gửi MQTT gần nhất
const int messageInterval = 10000; // Gửi dữ liệu MQTT mỗi 10 giây

void setup() {
  Serial.begin(115200);
  dht.begin();
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);

  // **Kết nối WiFi**
  Serial.print("📎 Đang kết nối WiFi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\n✅ WiFi đã kết nối!");

  // **Kết nối MQTT**
  client.setServer(mqtt_server, 1883);
  reconnectMQTT();

  // **Gửi trạng thái khởi động**
  client.publish(mqtt_status_topic, "ESP32 hệ thống cảnh báo cháy đã khởi
động!");
}

void loop() {
  if (!client.connected()) {
    reconnectMQTT();
```

```
}
client.loop();

// **Đọc dữ liệu từ cảm biến**
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
int smoke_level = analogRead(MQ2PIN);

// **Kiểm tra lỗi khi đọc cảm biến**
if (isnan(temperature) || isnan(humidity)) {
  Serial.println("⚠ Lỗi đọc cảm biến DHT22!");
  return;
}

// **Hiển thị dữ liệu lên Serial Monitor**
Serial.print("🌡 Nhiệt độ: ");
Serial.print(temperature);
Serial.print("°C | 💧 Độ ẩm: ");
Serial.print(humidity);
Serial.print("% | 💨 Khói: ");
Serial.println(smoke_level);

// **Kiểm tra nguy cơ cháy**
if (temperature > 60 || humidity < 20 || smoke_level > 300) {
  if (!fire_detected) {
    Serial.println("🔥🔥 CẢNH BÁO CHÁY! 🔥🔥");

    // **Bật còi báo động**
    digitalWrite(BUZZER_PIN, HIGH);
```

```
      // **Gửi cảnh báo lên MQTT**
      String message = "🔥 CẢNH BÁO CHÁY! 🌡 Nhiệt độ: " + String(temperature) +
                       "°C | 💧 Độ ẩm: " + String(humidity) +
                       "% | 💨 Khói: " + String(smoke_level);
      client.publish(mqtt_topic, message.c_str());

      fire_detected = true; // Đánh dấu trạng thái cháy
    }
  } else {
    // **Tắt cảnh báo nếu không còn nguy cơ cháy**
    if (fire_detected) {
      Serial.println("✅ Nguy cơ cháy đã giảm, hệ thống trở lại bình thường.");
      digitalWrite(BUZZER_PIN, LOW);
      client.publish(mqtt_topic, "✅ Hệ thống bình thường, không có nguy cơ cháy.");
      fire_detected = false;
    }
  }

  // **Gửi dữ liệu lên MQTT định kỳ**
  if (millis() - lastMessageTime > messageInterval) {
    String status_message = "📊 Cập nhật: 🌡 " + String(temperature) + "°C | 💧 " +
                            String(humidity) + "% | 💨 " + String(smoke_level);
    client.publish(mqtt_status_topic, status_message.c_str());
    lastMessageTime = millis();
```

```
  }

  delay(5000);  // Đọc dữ liệu mỗi 5 giây
}

// **Hàm kết nối lại MQTT nếu bị mất kết nối**
void reconnectMQTT() {
  while (!client.connected()) {
    Serial.print("🔄 Đang kết nối MQTT...");
    if (client.connect(device_id)) {
      Serial.println("✅ Kết nối MQTT thành công!");
      client.publish(mqtt_status_topic, "ESP32 đã kết nối lại MQTT.");
    } else {
      Serial.print("❌ Lỗi, mã: ");
      Serial.print(client.state());
      Serial.println(" -> Thử lại sau 5s");
      delay(5000);
    }
  }
}
```

PHỤC LỤC C. Tài liệu tham khảo

[1] Arduino và giao tiếp SPI http://arduino.vn/bai-viet/1081-arduino-va-giao-tiep-spi?fbclid=IwZXh0bgNhZW0CMTEAAR1xn9z0jn0C14YrmcmjRoiNwToRn8HVfcFO0bRjKilOJaxrQLSX6cDfIus_aem_Qczo5MMSCK9uz3aND03awQ

[2] ChatGPT – OpenAI

[3] BLACKBOX_AI