

ĐẠI HỌC HUẾ
TRƯỜNG ĐẠI HỌC KHOA HỌC
KHOA CÔNG NGHỆ THÔNG TIN



TIỂU LUẬN

Đề tài:

TẠO HỆ THỐNG BÁO CHÁY VỚI ESP32

Sinh viên thực hiện:

Lê Đức Duy Anh

Khóa:

K45 - Hệ chính quy

Huế, tháng 4 – năm 2025

ĐẠI HỌC HUẾ
TRƯỜNG ĐẠI HỌC KHOA HỌC
KHOA CÔNG NGHỆ THÔNG TIN



TIỂU LUẬN

Đề tài:

TẠO HỆ THỐNG BÁO CHÁY VỚI ESP32

Sinh viên thực hiện:	Lê Đức Duy Anh
Khóa:	K45 - Hệ chính quy
Giảng viên hướng dẫn:	ThS. Võ Việt Dũng

Huế, tháng 4 – năm 2025

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến **Thầy Võ Việt Dũng** đã tận tình hướng dẫn và hỗ trợ em trong quá trình thực hiện bài tiểu luận này. Những kiến thức và kinh nghiệm mà Thầy chia sẻ không chỉ giúp em hoàn thành bài tiểu luận mà còn mở rộng hiểu biết của em về lĩnh vực này. Nhờ sự chỉ bảo tận tình của Thầy, em đã có cơ hội tiếp cận vấn đề một cách có hệ thống hơn, rèn luyện khả năng tư duy và nâng cao kỹ năng trình bày.

Do giới hạn về thời gian và kiến thức, bài tiểu luận chắc chắn không tránh khỏi những thiếu sót. Em mong nhận được những góp ý quý báu từ Thầy để có thể rút kinh nghiệm và cải thiện hơn trong những bài tiểu luận sau. Những nhận xét của Thầy sẽ là động lực để em tiếp tục học hỏi và hoàn thiện bản thân hơn trong tương lai.

Em xin chân thành cảm ơn Thầy!

Huế, tháng 4 năm 2025

Sinh viên thực hiện

Lê Đức Duy Anh

DANH MỤC CÁC TỪ VIẾT TẮT

ABBREVIATIONS	MEANING
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
LED	Light Emitting Diode
TCP	Transmission Control Protocol
API	Application Programming Interface

DANH MỤC CÁC HÌNH ẢNH

Hình 1. *Bảng mạch ESP32*

Hình 2. *Cảm biến khói MQ2*

Hình 3. *Cảm biến độ ẩm, nhiệt độ DHT22*

Hình 4. *Đèn LED trên ESP32*

Hình 5. *Loa Buzzer*

Hình 6. *Kết nối loa Buzzer với ESP32*

Hình 7. *Thành phần giao thức MQTT*

Hình 8. *Cơ chế hoạt động giao thức MQTT*

Hình 9. *Kết nối linh kiện trên Wokwi*

Hình 10. *Nồng độ khói và nhiệt độ ở mức an toàn trên Visual Studio Code*

Hình 11. *Nồng độ khói hoặc nhiệt độ bất thường Visual Studio Code*

Hình 12. *Chạy Mosquitto MQTT Broker*

Hình 13. *Đăng ký nhận dữ liệu cho MQTT Client*

Hình 14. *Hệ thống phát hiện cháy trên MQTT*

Hình 15. *Hệ thống an toàn trên MQTT*

Hình 16. *Hệ thống phát hiện cháy trên Blynk*

Hình 17. *Hệ thống an toàn cháy trên Blynk*

MỤC LỤC

PHẦN MỞ ĐẦU	1
1. Lý do chọn đề tài:	1
2. Mục tiêu nghiên cứu:	1
3. Phương pháp nghiên cứu:	2
CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG BÁO CHÁY	3
1.1. Khái niệm và vai trò của hệ thống báo cháy:	3
1.2. Phương pháp báo cháy truyền thống và hiện đại:	3
1.2.1. Phương pháp báo cháy truyền thống:	3
1.2.2. Phương pháp báo cháy hiện đại:	3
1.3. Ứng dụng IoT trong hệ thống báo cháy:	4
CHƯƠNG 2: TÌM HIỂU CÁC THÀNH PHẦN TRONG HỆ THỐNG	5
2.1. Vi điều khiển ESP32	5
2.1.1. Khái niệm:	5
2.1.2. Cấu hình của ESP32:	5
2.2. Cảm biến khói MQ-2:	6
2.2.1. Khái niệm:	6
2.2.2. Thông số kỹ thuật:	6
2.2.3. Nguyên lý hoạt động:	7
2.3. Cảm biến nhiệt độ DHT22	7
2.3.1. Khái niệm:	7
2.3.2. Thông số kỹ thuật:	8
2.3.3. Nguyên lý hoạt động:	8
2.4. Đèn LED:	9
2.4.1. Khái niệm:	9
2.4.2. Các loại đèn LED trên ESP32:	9
2.4.2.1. LED tích hợp (Onboard LED):	9
2.4.2.2. LED ngoài (External LED):	9
2.5. Loa Buzzer:	10
2.5.1. Khái niệm:	10
2.5.2. Phân loại Buzzer	10
2.5.2.1. Buzzer chủ động (Active Buzzer):	10

2.5.2.2. Buzzer thụ động(Passive Buzzer):	11
2.5.3. Kết nối Buzzer với ESP32:	11
2.6. Giao thức MQTT - Cơ chế truyền dữ liệu và ứng dụng trong IoT	11
2.6.1. Khái niệm:	11
2.6.2. Thành phần:	12
2.6.3. Cơ chế hoạt động:	12
2.6.4. Ứng dụng giao thức MQTT hiện nay:	13
Chương 3: Xây dựng mô hình hệ thống báo cháy	14
3.1. Nguyên lý hoạt động của hệ thống báo cháy:	14
3.2. Thiết lập MQTT để gửi cảnh báo cháy:	14
3.3. Mô phỏng hệ thống:	15
3.3.1. Kết nối bảng mạch, linh kiện trên Wokwi:	15
3.3.2. Code mô phỏng chương trình Hệ Thống Báo Cháy trên Visual Studio Code:	15
3.3.3. Chạy mô phỏng chương trình trên Visual Studio Code:	19
3.3.4. Kiểm tra gửi thông báo đến MQTT:	20
3.5.5. Kiểm tra gửi thông báo đến Blynk:	22
PHẦN KẾT LUẬN	24
TÀI LIỆU THAM KHẢO	

PHẦN MỞ ĐẦU

1. Lý do chọn đề tài:

Cháy nổ là một trong những sự cố nguy hiểm, gây thiệt hại lớn về tài sản và đe dọa tính mạng con người. Nguyên nhân phổ biến có thể xuất phát từ chập điện, rò rỉ khí gas hoặc sự bất cẩn trong sinh hoạt. Để giảm thiểu rủi ro, các hệ thống báo cháy được sử dụng rộng rãi nhằm phát hiện sớm và cảnh báo nguy cơ hỏa hoạn. Tuy nhiên, hầu hết các hệ thống truyền thống chỉ cảnh báo tại chỗ mà không thể gửi thông tin từ xa, dẫn đến việc xử lý đôi khi bị chậm trễ.

Với sự phát triển của **công nghệ IoT (Internet of Things)**, các hệ thống báo cháy thông minh có thể được triển khai với khả năng giám sát từ xa và gửi cảnh báo qua Internet. Trong đó, **vi điều khiển ESP32** là một trong những nền tảng được sử dụng phổ biến nhờ khả năng kết nối Wi-Fi, hiệu suất cao và chi phí thấp. Khi kết hợp ESP32 với **cảm biến khói MQ-2** và **cảm biến nhiệt độ DHT22**, chúng ta có thể xây dựng một hệ thống báo cháy hiệu quả, có thể phát hiện nguy cơ cháy nổ và gửi cảnh báo đến người dùng qua giao thức **MQTT (Message Queuing Telemetry Transport)**.

Việc nghiên cứu và thiết kế hệ thống báo cháy thông minh không chỉ giúp nâng cao khả năng phát hiện sớm nguy cơ hỏa hoạn mà còn góp phần hiện đại hóa công tác phòng cháy chữa cháy, giúp giảm thiểu thiệt hại về tài sản và con người. Chính vì vậy, đề tài **"Tạo hệ thống báo cháy với ESP32 kết hợp cảm biến khói và nhiệt độ, gửi cảnh báo qua MQTT khi phát hiện nguy cơ cháy"** được lựa chọn với mong muốn nghiên cứu, tìm hiểu và đề xuất một giải pháp hiệu quả, ứng dụng thực tế trong lĩnh vực an toàn cháy nổ.

2. Mục tiêu nghiên cứu:

Mục tiêu của đề tài **"Tạo hệ thống báo cháy với ESP32 kết hợp cảm biến khói và nhiệt độ, gửi cảnh báo qua MQTT"** là nghiên cứu và đề xuất một giải pháp giám sát, cảnh báo cháy hiệu quả bằng cách ứng dụng công nghệ IoT. Cụ thể, đề tài hướng đến các mục tiêu sau:

- **Tìm hiểu nguyên lý hoạt động** của các cảm biến khói MQ-2 và cảm biến nhiệt độ DHT22 trong việc phát hiện nguy cơ cháy nổ.
- **Tìm hiểu về ứng dụng ESP32** trong việc thu thập dữ liệu từ cảm biến và truyền thông tin qua mạng.
- **Phân tích và đánh giá giao thức MQTT**, tìm hiểu cách thức giao thức này hỗ trợ truyền dữ liệu cảnh báo từ ESP32 đến các thiết bị đầu cuối như điện thoại, máy tính.
- **Xây dựng mô hình mô phỏng hệ thống báo cháy** trên nền tảng **Wokwi**, kiểm nghiệm tính khả thi của hệ thống.
- **Đánh giá ưu, nhược điểm của hệ thống**, so sánh với các giải pháp báo cháy truyền thống và đề xuất hướng phát triển trong tương lai.

3. Phương pháp nghiên cứu:

Để thực hiện đề tài “**Tạo hệ thống báo cháy với ESP32 kết hợp cảm biến khói và nhiệt độ, gửi cảnh báo qua MQTT**”, bài tiểu luận sử dụng các phương pháp nghiên cứu sau:

Phương pháp nghiên cứu tài liệu:

- Thu thập, phân tích tài liệu liên quan đến hệ thống báo cháy, cảm biến khói MQ-2, cảm biến nhiệt độ DHT22, vi điều khiển ESP32 và giao thức MQTT.
- Tìm hiểu nguyên lý hoạt động của từng linh kiện và cách thức giao tiếp giữa chúng.

Phương pháp mô phỏng trên Wokwi:

- Xây dựng mô hình mô phỏng hệ thống báo cháy bằng ESP32 trên nền tảng Wokwi. Kiểm tra hoạt động của các cảm biến và khả năng gửi cảnh báo qua giao thức MQTT.
- Đánh giá hiệu suất hoạt động của hệ thống thông qua các kịch bản giả lập tình huống cháy.

PHẦN NỘI DUNG

CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG BÁO CHÁY

1.1. Khái niệm và vai trò của hệ thống báo cháy:

Hệ thống báo cháy là một tập hợp các thiết bị và cảm biến được thiết kế để phát hiện sự xuất hiện của lửa, khói hoặc nhiệt độ cao bất thường và cung cấp cảnh báo ngay lập tức. Một hệ thống báo cháy thường bao gồm các cảm biến phát hiện khói, nhiệt độ, ngọn lửa, còi báo động, bộ điều khiển trung tâm và hệ thống thông báo từ xa.

Một hệ thống báo cháy có vai trò vô cùng quan trọng trong việc bảo vệ an toàn cho con người và tài sản. Giúp phát hiện sớm các dấu hiệu cháy của hỏa hoạn như khói, nhiệt độ cao bất thường, khí gas rò rỉ. Giúp hạn chế thiệt hại về tài sản và tính mạng bằng cách thông báo sớm cho lực lượng cứu hỏa để xử lý nhanh chóng. Cảnh báo kịp thời đến con người thông qua các thiết bị như còi hú, đèn báo hoặc gửi thông báo từ xa. [1]

1.2. Phương pháp báo cháy truyền thống và hiện đại:

1.2.1. Phương pháp báo cháy truyền thống:

Hệ thống báo cháy truyền thống thường được nhận diện qua cấu trúc đơn giản, chủ yếu dựa vào các thiết bị như đầu báo khói, đầu báo nhiệt, hay nút ấn báo cháy để phát hiện và cảnh báo về tình trạng cháy. Những thiết bị này chỉ báo động tại chỗ thông qua âm thanh hoặc ánh sáng, giúp người trong khu vực kịp thời phát hiện nguy cơ hỏa hoạn.

Tuy nhiên hệ thống này có một số hạn chế như: không thể gửi cảnh báo từ xa, không có khả năng phân tích dữ liệu dẫn đến việc xác định nguy cơ cháy có thể bị chậm trễ hoặc nhầm lẫn, sai sót trong một số trường hợp. [2]

1.2.2. Phương pháp báo cháy hiện đại:

Với sự phát triển của công nghệ, các hệ thống báo cháy thông minh đã ra đời, tích hợp IoT giúp cảnh báo cháy từ xa và giám sát tình trạng cháy nổ hiệu quả hơn:

- **Hệ thống báo cháy không dây:** Các thiết bị cảm biến kết nối với trung tâm điều khiển thông qua Wi-Fi hoặc sóng radio, giúp dễ dàng mở rộng và lắp đặt.
- **Hệ thống báo cháy sử dụng IoT:** Cảm biến kết nối với mạng Internet, gửi cảnh báo qua điện thoại hoặc máy tính, giúp người dùng có thể giám sát từ xa.
- **Ứng dụng trí tuệ nhân tạo (AI) trong báo cháy:** Một số hệ thống hiện đại sử dụng AI để phân tích dữ liệu từ cảm biến và dự đoán nguy cơ cháy.

1.3. Ứng dụng IoT trong hệ thống báo cháy:

Ứng dụng IoT (Internet of Things) trong hệ thống báo cháy mang đến nhiều lợi ích, giúp cải thiện đáng kể hiệu quả & tính chính xác trong việc phát hiện, cảnh báo cháy. Một số ứng dụng cụ thể của IoT trong hệ thống báo cháy phải kể đến như:

Cảm biến IoT

Cảm biến IoT có thể được lắp đặt ở nhiều vị trí khác nhau trong tòa nhà hoặc khu vực cần giám sát. Các loại cảm biến này bao gồm:

- Cảm biến khói: cảnh báo khi phát hiện có sự xuất hiện của khói
- Cảm biến khí gas: phát hiện rò rỉ các loại khí dễ cháy như CO, gas,...
- Cảm biến nhiệt độ: phát hiện nhiệt độ tăng/giảm bất thường

Kết nối & Truyền tải dữ liệu

Các cảm biến IoT sẽ được kết nối với mạng lưới không dây hoặc có dây để truyền tải dữ liệu về trung tâm điều khiển hoặc đám mây. Điều này cho phép:

- Giám sát theo thời gian thực: dữ liệu từ cảm biến được gửi liên tục về trung tâm điều khiển, giúp phát hiện các sự cố kịp thời
- Phân tích dữ liệu: sử dụng các công cụ phân tích dữ liệu để đưa ra các cảnh báo sớm hơn

Hệ thống cảnh báo thông minh

Khi phát hiện nguy cơ cháy nổ, hệ thống IoT sẽ kích hoạt các biện pháp cảnh báo & ứng phó:

- Cảnh báo qua điện thoại di động: gửi thông báo SMS, email hoặc ứng dụng di động đến người quản lý & các cơ quan cứu hỏa

Kích hoạt hệ thống chữa cháy tự động: kích hoạt hệ thống phun nước, hệ thống dập lửa,...[3]

CHƯƠNG 2: TÌM HIỂU CÁC THÀNH PHẦN TRONG HỆ THỐNG

2.1. Vi điều khiển ESP32

2.1.1. Khái niệm:

ESP32 là vi điều khiển giá rẻ, tiêu thụ năng lượng thấp do Espressif Systems phát triển, hỗ trợ WiFi và Bluetooth dual-mode. ESP32 sử dụng bộ vi xử lý Tensilica Xtensa LX6 (lõi đơn hoặc lõi kép) và tích hợp các thành phần như công tắc antenna, RF balun, bộ khuếch đại, bộ lọc và module quản lý năng lượng.

2.1.2. Cấu hình của ESP32:

- **CPU:** Xtensa Dual-Core LX6 (32-bit), tốc độ 160 - 240 MHz
- **RAM:** 520 KB SRAM (bao gồm 8 KB RAM RTC tốc độ cao và 8 KB RAM RTC tốc độ thấp cho chế độ DeepSleep)
- **Kết nối không dây:**
 - Wi-Fi: 802.11 b/g/n/e/I
 - Bluetooth: v4.2 BR/EDR and BLE
- **Giao tiếp ngoại vi:**
 - ADC 12-bit (16 cổng), DAC 8-bit (2 cổng)
 - I²C (2 cổng), UART (3 cổng), SPI (3 cổng), I²S (2 cổng).
 - Hỗ trợ SD card, SDIO/MMC, Ethernet MAC, CAN bus 2.0, IR TX/RX.
 - PWM trên tất cả các chân GPIO.
- **Cảm biến tích hợp:**
 - Cảm biến HALL (từ trường), cảm biến nhiệt độ, cảm biến chạm điện dung (10 đầu vào).
- **Bảo mật:**
 - Hỗ trợ WPA/WPA2, Secure Boot, mã hóa Flash

Hỗ trợ tăng tốc phần cứng cho AES, SHA-2, RSA, ECC, RNG.

- **Điện áp hoạt động:** 2.2V - 3.6V.
- **Nhiệt độ hoạt động:** -40 °C đến +85 °C.
- **Số chân GPIO:** 34.
- **Module phổ biến:** ESP32-WROOM (Doit ESP32 DevKit V1).

[4]



Hình 1. Bảng mạch ESP32 (Nguồn: nshopvn.com)

2.2. Cảm biến khói (MQ-2):

2.2.1. Khái niệm:

Mạch Cảm Biến Khí Gas MQ2 là cảm biến khí, dùng để phát hiện các khí có thể gây cháy như LPG, CO, CH₄ (metan), khói, cồn và hydro. Nó được cấu tạo từ chất bán dẫn SnO₂. Chất này có độ nhạy cảm thấp với không khí sạch. Nhưng khi trong môi trường có chất gây cháy, độ dẫn của nó lập tức thay đổi. Chính nhờ đặc điểm này người ta thêm vào mạch đơn giản để biến đổi từ độ nhạy này sang điện áp.

2.2.2. Thông số kỹ thuật:

- Điện áp hoạt động: 5V
- Dòng tiêu thụ: 180 mA
- Công suất: 900 mW

- Chuẩn truyền: Digital & Analog (A0)
- Dãy hoạt động: 300ppm - 1000ppm
- Số chân: 4
- Loại: Module

2.2.3. Nguyên lý hoạt động:

Cảm biến MQ2 hoạt động dựa trên nguyên lý phản ứng hóa học khi tiếp xúc các khí trong môi trường. Nguyên tắc hoạt động của cảm biến MQ2 là khi các khí trong môi trường như khí CO, khí LPG, khói... tiếp xúc với phần tử bên trong cảm biến, làm cho các electron được giải phóng vào SnO_2 cho phép dòng điện chạy qua cảm biến một cách tự do.

Khi được làm nóng, phần tử cảm biến sẽ tạo ra phản ứng hóa học với các khí tiếp xúc và làm thay đổi điện trở của phần tử cảm biến. Cảm biến MQ2 đo lường các biến đổi điện trở này và chuyển đổi chúng thành tín hiệu điện Analog hoặc Digital.

[5]



Hình 2. Cảm biến khói MQ2 (Nguồn: dientu360.com)

2.3. Cảm biến nhiệt độ DHT22

2.3.1. Khái niệm:

Cảm biến nhiệt độ DHT22 là cảm biến đo nhiệt độ và độ ẩm của môi trường, chuyển đổi các giá trị nhiệt độ và độ ẩm sang giá trị số digital, giao tiếp với MCU (vi điều khiển), Arduino, ESP8266, ESP32... qua giao tiếp 1 dây (chân DATA). Từ đó vi điều khiển đọc và so sánh giá trị thực tế và giá trị đặt trước để xuất lệnh cho các cơ cấu chấp hành thực hiện đúng mục đích.

2.3.2. Thông số kỹ thuật:

- Nguồn sử dụng: 3~5 VDC
- Dòng sử dụng: 2.5mA max (khi truyền dữ liệu).
- Đo tốt ở độ ẩm: 0100%RH với sai số 2-5%.
- Đo tốt ở nhiệt độ: -40 °C đến +80 °C sai số $\pm 0.5^{\circ}\text{C}$
- Tần số lấy mẫu tối đa: 0.5Hz (2 giây 1 lần).
- Kích thước: 27mm x 59mm x 13.5mm.

[6]

2.3.3. Nguyên lý hoạt động:

Cảm biến DHT22 (AM2302) hoạt động dựa trên nguyên lý điện dung và nhiệt điện trở để đo độ ẩm và nhiệt độ.

Đo độ ẩm:

DHT22 sử dụng một phần tử cảm biến độ ẩm điện dung, phần tử này gồm hai điện cực đặt trên một lớp polymer hút ẩm, có khả năng hấp thụ hơi nước từ môi trường, khi độ ẩm thay đổi, hằng số điện môi của lớp polymer thay đổi, làm thay đổi điện dung giữa hai điện cực, một bộ mạch tích hợp bên trong sẽ chuyển đổi sự thay đổi này thành tín hiệu số để xuất ra.

Đo nhiệt độ:

DHT22 sử dụng cảm biến nhiệt điện trở (Thermistor) hoặc cảm biến bán dẫn để đo nhiệt độ. Khi nhiệt độ thay đổi, điện trở của cảm biến cũng thay đổi. Mạch xử lý bên trong chuyển đổi giá trị điện trở này thành dữ liệu số để gửi ra ngoài.

Truyền dữ liệu:

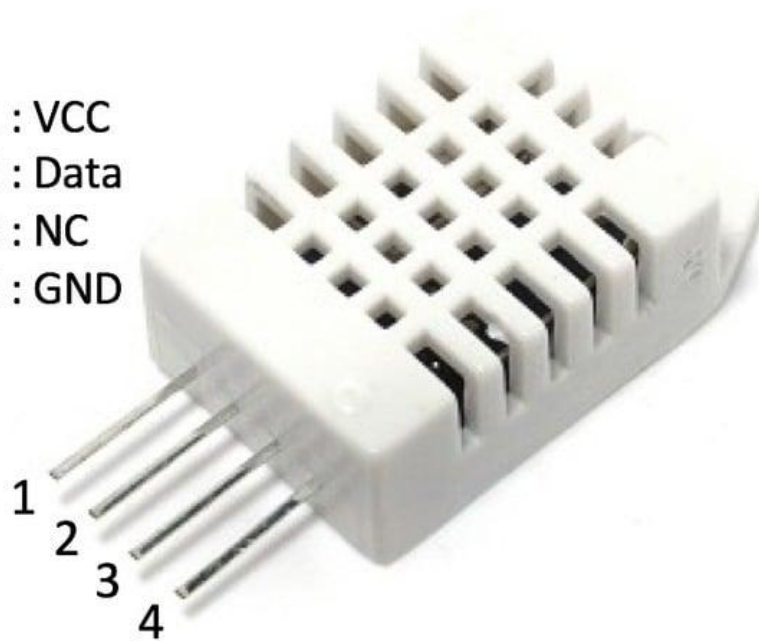
DHT22 truyền dữ liệu qua giao thức 1-Wire với chỉ một dây tín hiệu. Sau khi nhận lệnh từ vi điều khiển, cảm biến gửi một gói tin 40 bit, trong đó:

16 bit đầu: Giá trị độ ẩm

16 bit tiếp theo: Giá trị nhiệt độ

8 bit cuối: Mã kiểm tra (Checksum) để phát hiện lỗi truyền dữ liệu.

1 : VCC
2 : Data
3 : NC
4 : GND



Hình 3. Cảm biến độ ẩm, nhiệt độ DHT22 (hshop.vn)

2.4. Đèn LED:

2.4.1. Khái niệm:

Đèn LED trên ESP32 là một diode phát sáng (Light Emitting Diode) có thể được điều khiển bằng các chân GPIO của ESP32. Đèn LED có thể bật (HIGH / 1) khi có điện áp cấp vào và tắt (LOW / 0) khi không có điện áp cấp.

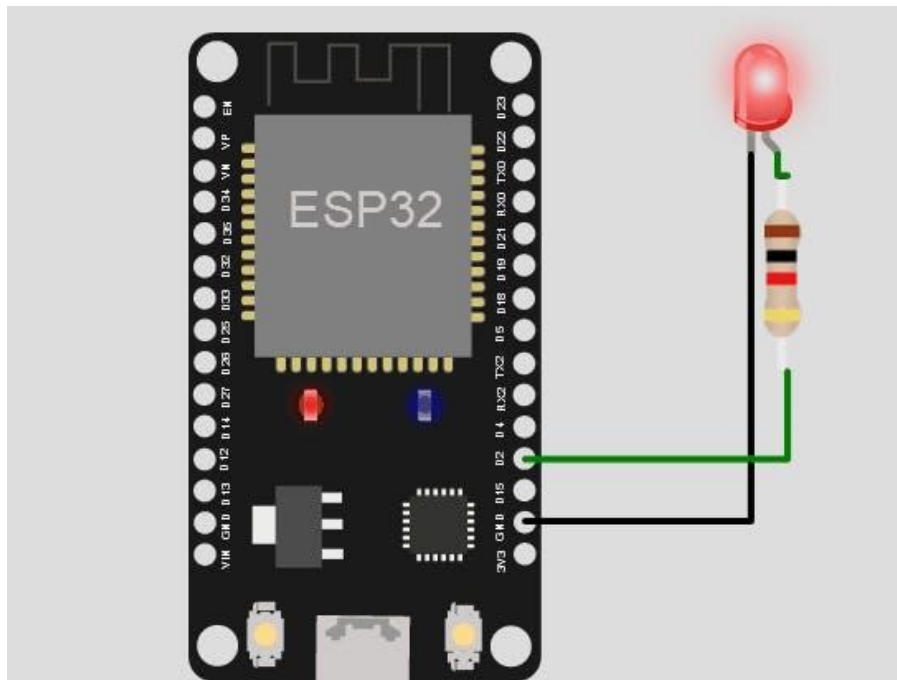
2.4.2. Các loại đèn LED trên ESP32:

2.4.2.1. LED tích hợp (Onboard LED):

- Một số bo mạch ESP 32 có sẵn một đèn LED được nối với GPIO 2.
- Đèn này thường dùng để báo trạng thái hoặc làm thí nghiệm.

2.4.2.2. LED ngoài (External LED):

- Đèn LED rời có thể được kết nối với bất kỳ chân GPIO nào của ESP32.
- Khi dùng LED ngoài, cần thêm điện trở (thường 220Ω - $1k\Omega$) để bảo vệ LED.



Hình 4. Đèn LED trên ESP32 (Nguồn: hackster.io)

2.5. Loa Buzzer:

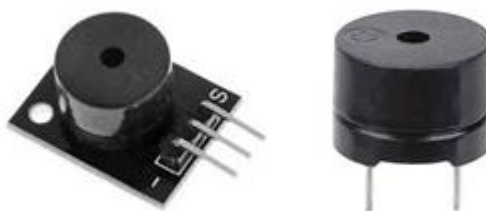
2.5.1. Khái niệm:

Buzzer là một loại loa nhỏ thường được sử dụng trong các ứng dụng nhúng để phát ra âm thanh cảnh báo hoặc tín hiệu âm thanh đơn giản.

2.5.2. Phân loại Buzzer

2.5.2.1. Buzzer chủ động (Active Buzzer):

- Tích hợp mạch dao động bên trong, chỉ cần cấp nguồn là có thể phát âm thanh.
- Hoạt động đơn giản, chỉ cần dùng lệnh `digitalWrite(pin, HIGH)` để bật và `digitalWrite(pin, LOW)` để tắt.



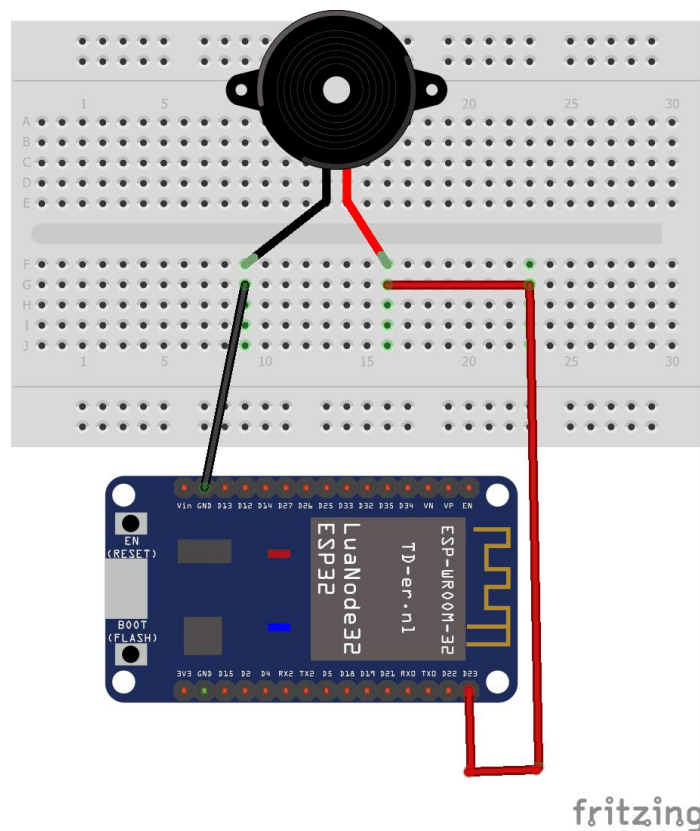
Hình 5. Loa Buzzer (Nguồn: robotique.tech)

2.5.2.2. Buzzer thụ động(Passive Buzzer):

- Không có mạch dao động bên trong, cần điều khiển bằng tín hiệu PWM để tạo âm thanh.
- Có thể tạo ra nhiều tần số khác nhau để phát các âm thanh khác nhau.
- Dùng lệnh tone(pin, frequency) để phát âm với tần số mong muốn.

2.5.3. Kết nối Buzzer với ESP32:

- Chân dương (+) của Buzzer: Kết nối với 1 chân GPIO trên ESP32.
- Chân âm (-) của Buzzer: Kết nối với GND trên ESP32.



Hình 6. Kết nối loa Buzzer với ESP32 (Nguồn: robotique.tech)

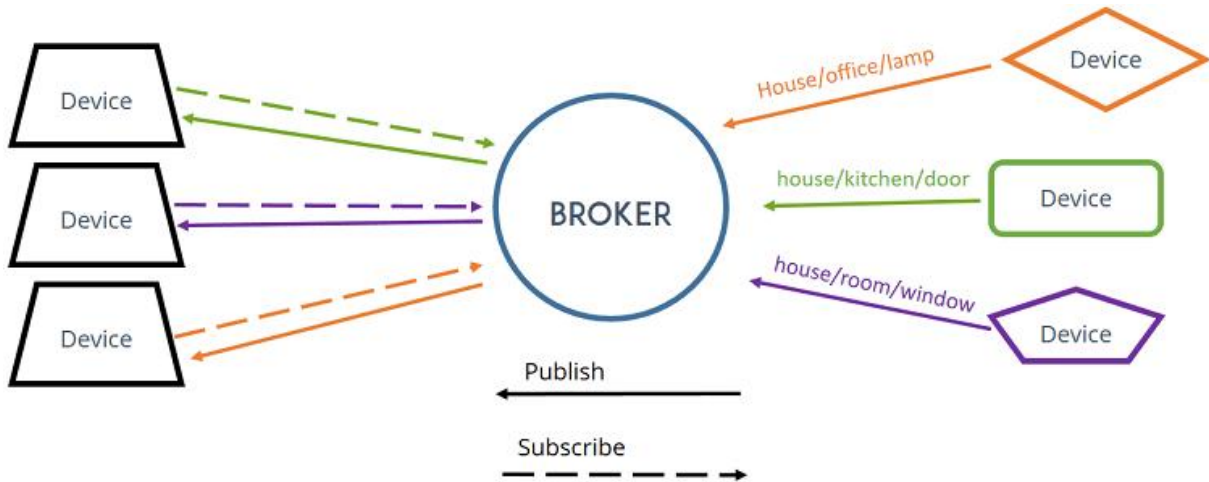
2.6. Giao thức MQTT - Cơ chế truyền dữ liệu và ứng dụng trong IoT

2.6.1. Khái niệm:

MQTT là giao thức truyền thông điệp (message) theo mô hình publish/subscribe (cung cấp / thuê bao), được sử dụng cho các thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Nó dựa trên một Broker

(tạm dịch là “Máy chủ môi giới”) “nhẹ” (khá ít xử lý) và được thiết kế có tính mở (tức là không đặc trưng cho ứng dụng cụ thể nào), đơn giản và dễ cài đặt.

2.6.2. Thành phần:



Hình 7. Thành phần giao thức MQTT (Nguồn: smartindustry.vn)

Thành phần chính của MQTT là Client (Publisher/Subscriber), Server (Broker), Sessions (tạm dịch là Phiên làm việc), Subscriptions và Topics.

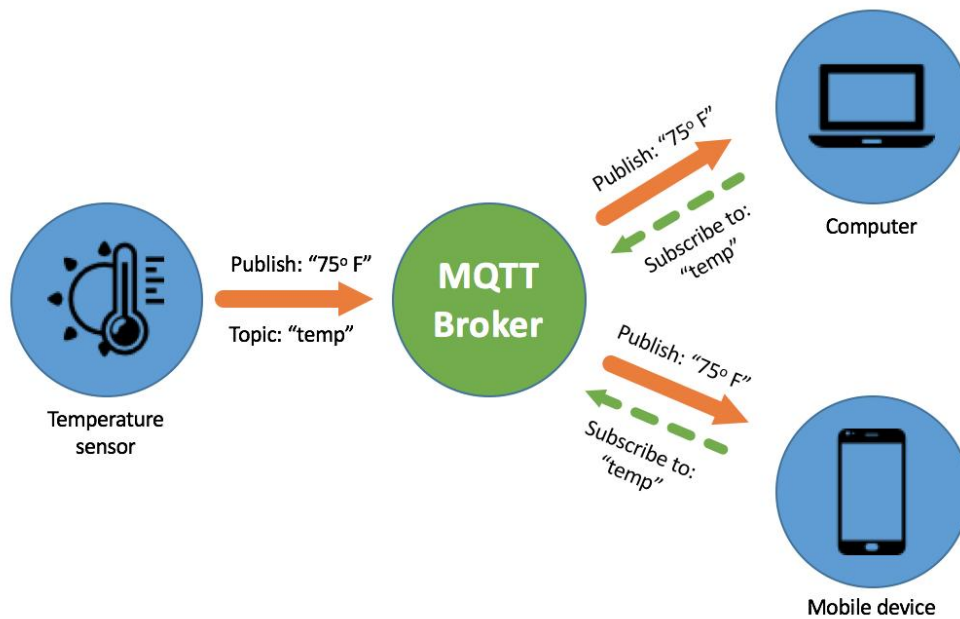
MQTT Client (Publisher/Subscriber): Clients sẽ subscribe một hoặc nhiều topics để gửi và nhận thông điệp từ những topic tương ứng.

MQTT Server (Broker): Broker nhận những thông tin subscribe (Subscriptions) từ client, nhận thông điệp, chuyển những thông điệp đến các Subscriber tương ứng dựa trên Subscriptions từ client.

Topic: là kênh giao tiếp giữa Publishers và Subscribers. Ví dụ: /cambien/khoi dùng để chứa dữ liệu về cảm biến khói

Subscription: Không giống như session, subscription về mặt logic là kết nối từ client đến topic. Khi đã subscribe một topic, Client có thể nhận/gửi thông điệp (message) với topic đó.

2.6.3. Cơ chế hoạt động:



Hình 8. Cơ chế hoạt động giao thức MQTT (Nguồn: atscada.com)

MQTT hoạt động theo cơ chế client/server, nơi mà mỗi cảm biến là một khách hàng (client) và kết nối đến một máy chủ, có thể hiểu như một Máy chủ môi giới (broker), thông qua giao thức TCP (Transmission Control Protocol). Broker chịu trách nhiệm điều phối tất cả các thông điệp giữa phía gửi đến đúng phía nhận.

MQTT là giao thức định hướng bản tin. Mỗi bản tin là một đoạn rời rạc của tín hiệu và broker không thể nhìn thấy. Mỗi bản tin được publish một địa chỉ, có thể hiểu như một kênh (Topic). Client đăng ký vào một vài kênh để nhận/gửi dữ liệu, gọi là subscribe. Client có thể subscribe vào nhiều kênh. Mỗi client sẽ nhận được dữ liệu khi bất kỳ trạm nào khác gửi dữ liệu vào kênh đã đăng ký. Khi một client gửi một bản tin đến một kênh nào đó gọi là publish.

[7]

2.6.4. Ứng dụng giao thức MQTT hiện nay:

Facebook Messenger: Trò chuyện trực tuyến chính là ứng dụng được sử dụng. Facebook đã sử dụng các khía cạnh của MQTT trong Facebook Messenger.

Amazon Web Services đã công bố Amazon IoT dựa trên MQTT vào năm 2015.

Các tổ chức không gian địa lý Sensor Things API. Họ đã công bố đặc điểm kỹ thuật tiêu chuẩn có một phần mở rộng MQTT. Tiêu chuẩn lúc này như một giao thức

thông báo bổ sung ràng buộc. Nó đã được chứng minh trong một thí điểm IoT của Bộ An ninh Nội địa Hoa Kỳ.

Adafruit đưa ra một MQTT miễn phí. Đó chính là Cloud Service cho thí nghiệm IoT. Chúng còn được biết đến với tên gọi Adafruit IO trong năm 2015.

Microsoft Azure IoT Hub sử dụng MQTT làm giao thức chính cho các tin nhắn từ xa.

XIM, Inc. đã ra mắt ứng dụng khách MQTT có tên MQTT Buddy vào năm 2017. Đây là ứng dụng MQTT dành cho Android và iOS. Người dùng được sử dụng ngôn ngữ có sẵn bằng tiếng Anh, Nga và Trung Quốc.

Node-RED hỗ trợ các nút MQTT kể từ phiên bản 0.14. Nhiệm vụ phát minh để định dạng cấu hình đúng các kết nối TLS.

[8]

Chương 3: Xây dựng mô hình hệ thống báo cháy

3.1. Nguyên lý hoạt động của hệ thống báo cháy:

Chương trình này sử dụng ESP32 để đọc dữ liệu từ cảm biến khói (MQ-2) và cảm biến nhiệt độ & độ ẩm (DHT22). Nếu nồng độ khói vượt quá mức nguy hiểm hoặc nhiệt độ quá cao, hệ thống sẽ kích hoạt cảnh báo bằng cách bật đèn LED, phát âm thanh cảnh báo từ buzzer và gửi dữ liệu lên nền tảng Blynk để hiển thị tình trạng thực tế trên điện thoại. Đồng thời, ESP32 cũng kết nối với một máy chủ MQTT để gửi cảnh báo cháy đến hệ thống giám sát từ xa.

3.2. Thiết lập MQTT để gửi cảnh báo cháy:

Hệ thống sử dụng giao thức MQTT để truyền tải thông tin cảnh báo theo thời gian thực. MQTT broker được thiết lập trên Mosquitto với các thông tin sau:

- Broker: test.mosquitto.org
- Port: 1833
- Topic: esp32/fire_alarm

ESP32 sẽ thực hiện publish cảnh báo khi nồng độ khói đo được từ cảm biến MQ-2 hoặc nhiệt độ đo từ cảm biến DHT22 vượt quá ngưỡng an toàn. Khi xảy ra tình huống nguy hiểm, hệ thống sẽ gửi tin nhắn cảnh báo qua MQTT với nội dung:

“CANH BAO CHAY!!! Nong do Khoi, Nhiet Do vuot qua nguong an toan”

Nếu điều kiện trở lại bình thường sẽ hiển thị thông báo:

“Nong do Khoi, Nhiet Do an toan”

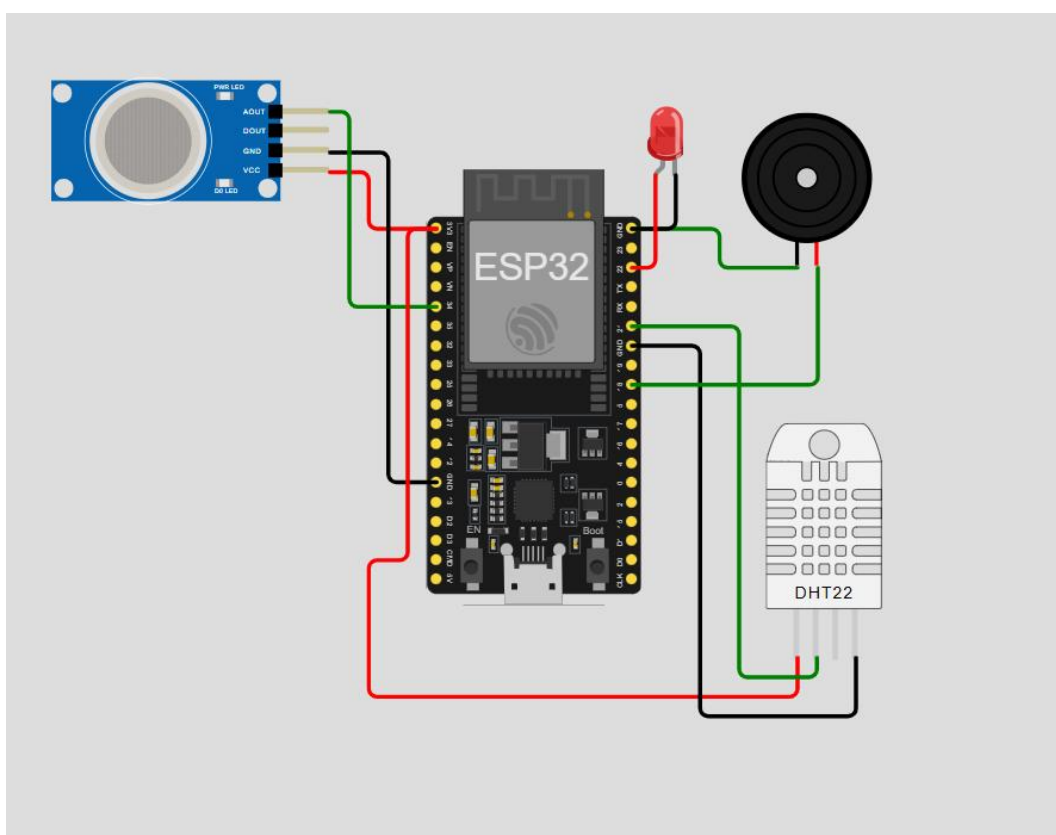
Các thông báo này có thể được kiểm tra thông qua việc sử dụng lệnh `mosquitto_sub` trên máy tính để theo dõi trạng thái của hệ thống. Việc thiết lập MQTT giúp hệ thống có thể dễ dàng tích hợp với các nền tảng IoT khác, đồng thời hỗ trợ theo dõi và xử lý sự cố kịp thời.

3.3. Mô phỏng hệ thống:

3.3.1. Kết nối bảng mạch, linh kiện trên Wokwi:

Dưới đây là sơ đồ kết nối giữa các linh kiện:

- ESP32: Vi điều khiển trung tâm, giao tiếp với WiFi
- MQ-22: Kết nối với chân ADC (GPIO34)
- DHT22: Kết nối với chân Digital (GPIO21)
- LED báo động: Kết nối với chân Digital (GPIO2)
- Loa Buzzer: Kết nối với chân Digital (GPIO18)



Hình 9. Kết nối linh kiện trên Wokwi

3.3.2. Code mô phỏng chương trình Hệ Thống Báo Cháy trên Visual Studio Code:

```

1 #define BLYNK_TEMPLATE_ID "TMPL644gL5d-g"
2 #define BLYNK_TEMPLATE_NAME "MQTT BLYNK"
3 #define BLYNK_AUTH_TOKEN "gZeUNnNrPRLCosk_shI__oR1UtpHpuuz"
4
5 #include <WiFi.h>
6 #include <PubSubClient.h>
7 #include <DHT.h>
8 #include <BlynkSimpleEsp32.h>
9
10 #define MQ2_PIN 34
11 #define LED_PIN 22
12 #define BUZZER_PIN 18
13 #define DHT_PIN 21
14 #define DHT_TYPE DHT22
15
16 unsigned long lastAlertTime = 0;
17 const unsigned long alertInterval = 2000;
18 const char* ssid = "Wokwi-GUEST";
19 const char* password = "";
20
21 const char* mqtt_server = "test.mosquitto.org";
22 const int mqtt_port = 1883;
23 const char* mqtt_topic = "esp32/fire_alarm";
24
25 WiFiClient espClient;
26 PubSubClient mqttClient(espClient);
27 DHT dht(DHT_PIN, DHT_TYPE);
28
29 void reconnect() {
30     while (!mqttClient.connected()) {
31         Serial.print("Đang kết nối MQTT...");
32         if (mqttClient.connect("ESP32_Client")) {

```

```

33     Serial.println("Thành công!");
34 } else {
35     Serial.print("Thất bại, mã lỗi: ");
36     Serial.print(mqttClient.state());
37     Serial.println(" Thử lại sau 5 giây...");
38     delay(5000);
39 }
40 }
41 }
42
43 void setup() {
44     Serial.begin(115200);
45     WiFi.begin(ssid, password);
46     dht.begin();
47
48     pinMode(LED_PIN, OUTPUT);
49     pinMode(BUZZER_PIN, OUTPUT);
50     digitalWrite(LED_PIN, LOW);
51     digitalWrite(BUZZER_PIN, LOW);
52
53     while (WiFi.status() != WL_CONNECTED) {
54         delay(500);
55         Serial.print(".");
56     }
57     Serial.println("\nWiFi đã kết nối!");
58
59     mqttClient.setServer(mqtt_server, mqtt_port);
60     Blynk.begin(BLYNK_AUTH_TOKEN, ssid, password);
61
62     delay(2000);
63 }
64

```



```

65 void loop() {
66   if (!mqttClient.connected()) {
67     reconnect();
68   }
69   mqttClient.loop();
70   Blynk.run();
71
72   int sensorValue = analogRead(MQ2_PIN);
73   float ppm = (sensorValue / 4095.0) * 5000;
74   float temperature = dht.readTemperature();
75   float humidity = dht.readHumidity();
76
77   Serial.print("ADC Raw: ");
78   Serial.print(sensorValue);
79   Serial.print(" | Nồng độ khí: ");
80   Serial.print(ppm);
81   Serial.println(" ppm");
82   Serial.print("Nhiệt độ: ");
83   Serial.print(temperature);
84   Serial.println("°C");
85   Serial.print("Độ ẩm: ");
86   Serial.print(humidity);
87   Serial.println("%");
88
89   Blynk.virtualWrite(V0, ppm);
90   Blynk.virtualWrite(V2, temperature);
91   Blynk.virtualWrite(V3, humidity);
92
93   if (ppm > 2000 || temperature > 50) {
94     Serial.println("CẢNH BÁO CHÁY!!! Đèn đỏ bật + Buzzer kêu!");
95     digitalWrite(LED_PIN, HIGH);
96     tone(BUZZER_PIN, 1000);

```

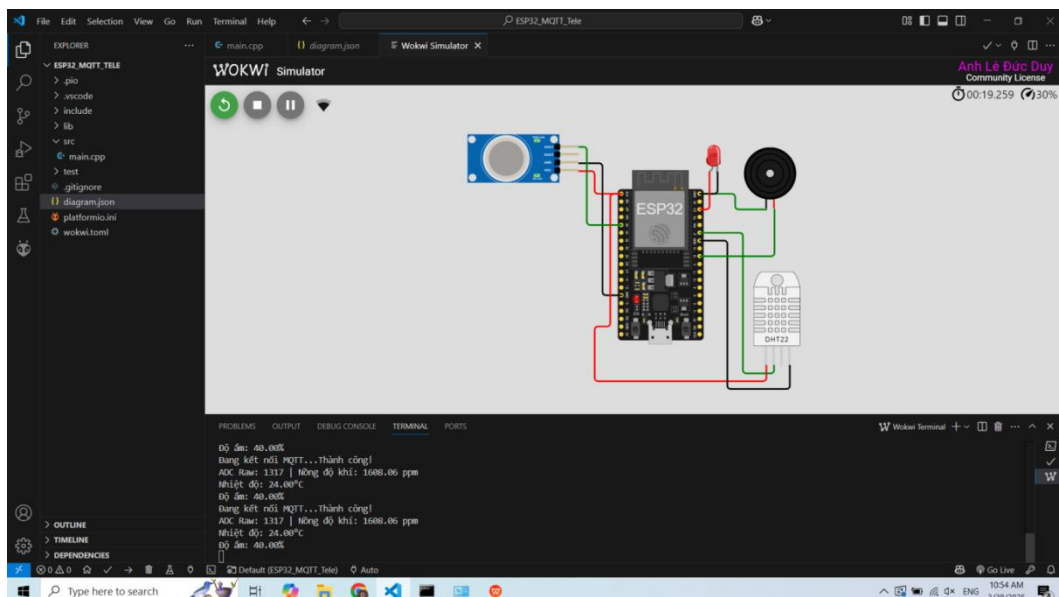
```

97     Blynk.virtualWrite(V1, 1);
98
99     if (millis() - lastAlertTime > alertInterval) {
100         Serial.println("Gửi tin nhắn MQTT: CẢNH BÁO CHÁY!");
101         mqttClient.publish(mqtt_topic, "CANH BAO CHAY! Nong do Khi, Nhiet
102 Do vuot qua nguong an toan");
103         lastAlertTime = millis();
104     }
105 } else {
106     digitalWrite(LED_PIN, LOW);
107     noTone(BUZZER_PIN);
108     Blynk.virtualWrite(V1, 0);
109     mqttClient.publish(mqtt_topic, "Nong do Khi, Nhiet Do an toan");
110 }
111
112 delay(2000);
113 }

```

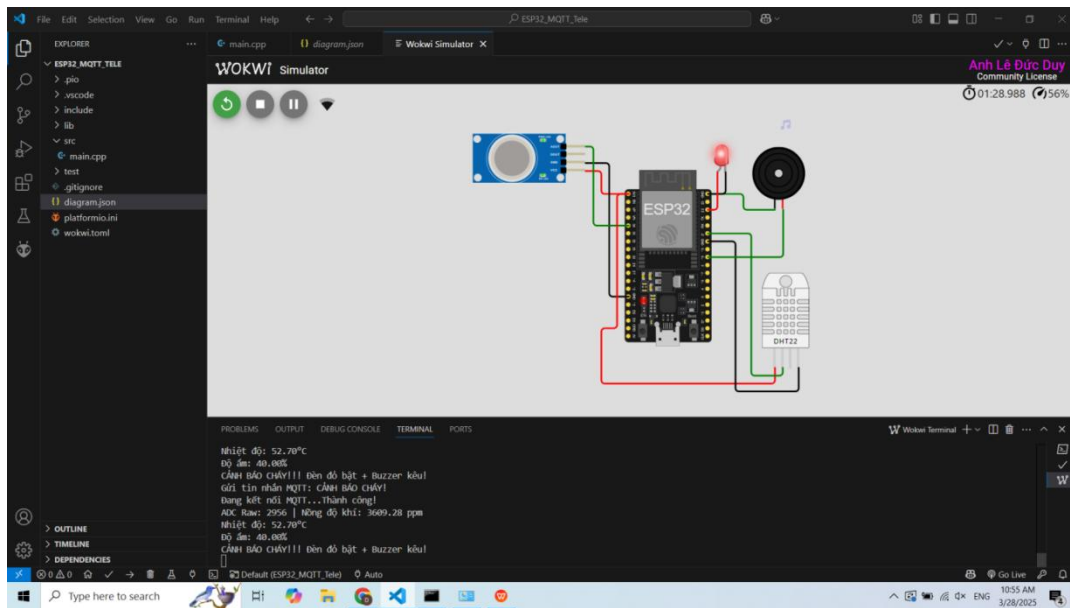
3.3.3. Chạy mô phỏng chương trình trên Visual Studio Code:

Khi nồng độ khói và nhiệt độ ở mức an toàn (Nồng độ khói dưới 2000, nhiệt độ dưới 50 °C):



Hình 10. Nồng độ khói và nhiệt độ ở mức an toàn trên Visual Studio Code

Khi nồng độ khói vượt ngưỡng 2000, nhiệt độ lớn hơn 50°C:



Hình 11. Nồng độ khói hoặc nhiệt độ bất thường Visual Studio Code

3.3.4. Kiểm tra gửi thông báo đến MQTT:

Chạy lệnh `mosquitto -c mosquitto.conf -v` trên cmd. Lệnh `mosquitto -c mosquitto.conf -v` được sử dụng để chạy **Mosquitto MQTT Broker** với tệp cấu hình `mosquitto.conf` và chế độ hiển thị chi tiết (`-v - verbose`).

```
Command Prompt - mosquitto -c mosquitto.conf -v
Microsoft Windows [Version 10.0.19045.5608]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Duy Anh>E:

E:>cd mosquitto

E:\mosquitto>mosquitto -c mosquitto.conf -v
1743129075: mosquitto version 2.0.21 starting
1743129075: Config loaded from mosquitto.conf.
1743129075: Opening ipv6 listen socket on port 1883.
1743129075: Opening ipv4 listen socket on port 1883.
1743129075: mosquitto version 2.0.21 running
```

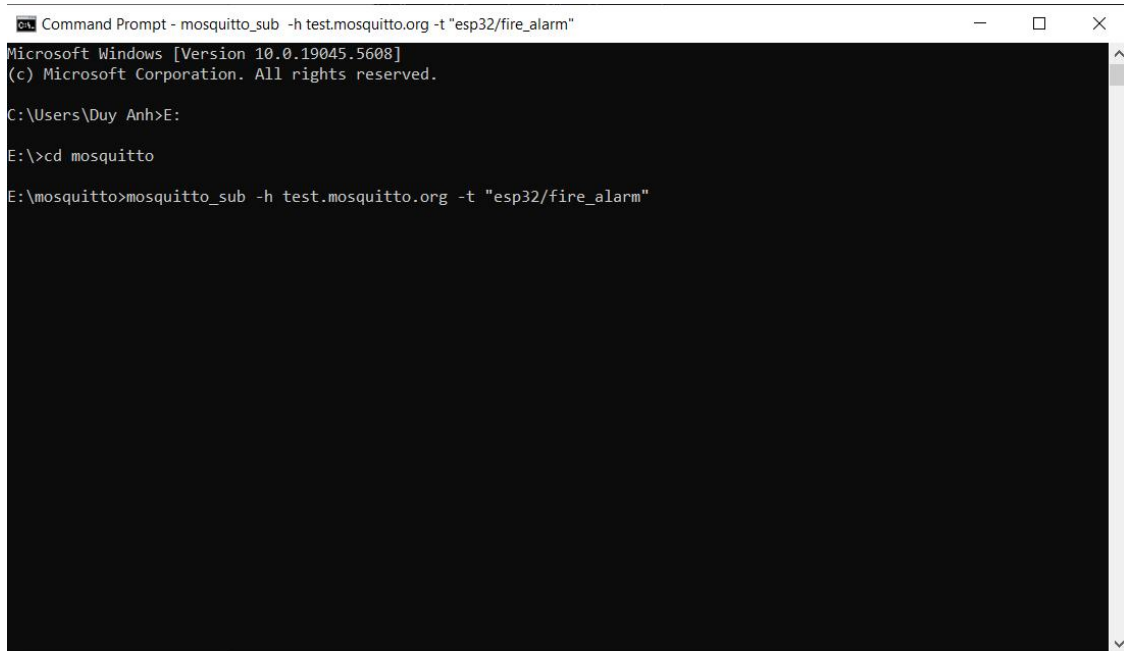
Hình 12. Chạy Mosquitto MQTT Broker

Chạy lệnh `mosquitto_sub -h test.mosquitto.org -t "esp32/fire_alarm"` trên cmd.

`mosquitto_sub`: MQTT client dùng để đăng ký nhận dữ liệu (subscribe)

`-h test.mosquitto.org`: Kết nối đến MQTT broker tại địa chỉ `test.mosquitto.org`

`-t "esp32/fire_alarm"`: Đăng ký nhận tin nhắn từ topic “esp32/fire_alarm”



```
Command Prompt - mosquitto_sub -h test.mosquitto.org -t "esp32/fire_alarm"
Microsoft Windows [Version 10.0.19045.5608]
(c) Microsoft Corporation. All rights reserved.

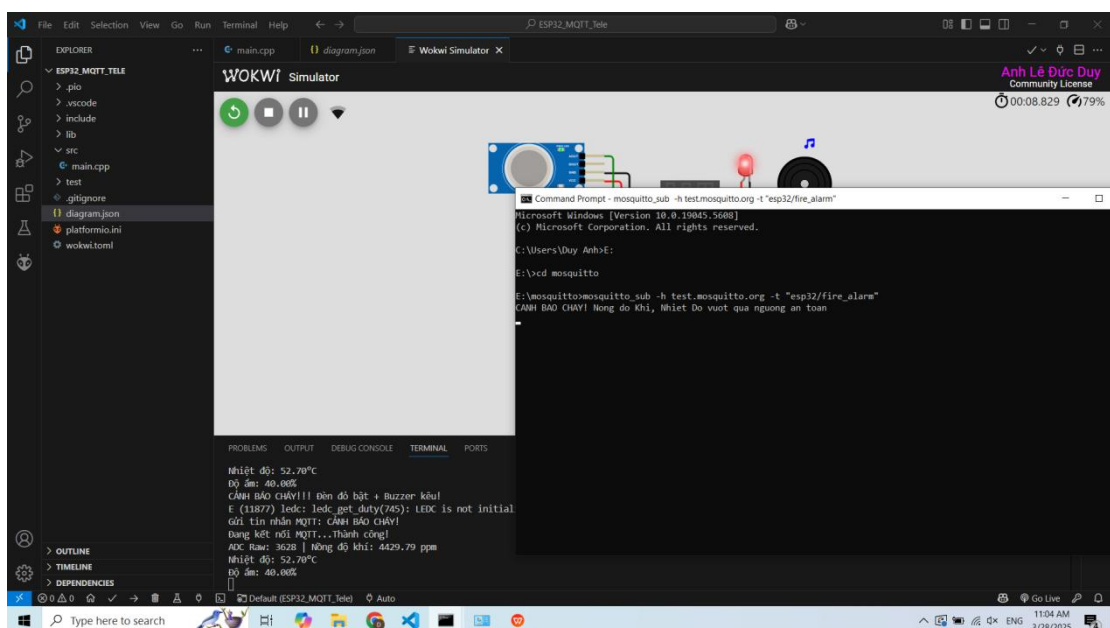
C:\Users\Duy Anh>E:

E:\>cd mosquitto

E:\mosquitto>mosquitto_sub -h test.mosquitto.org -t "esp32/fire_alarm"
```

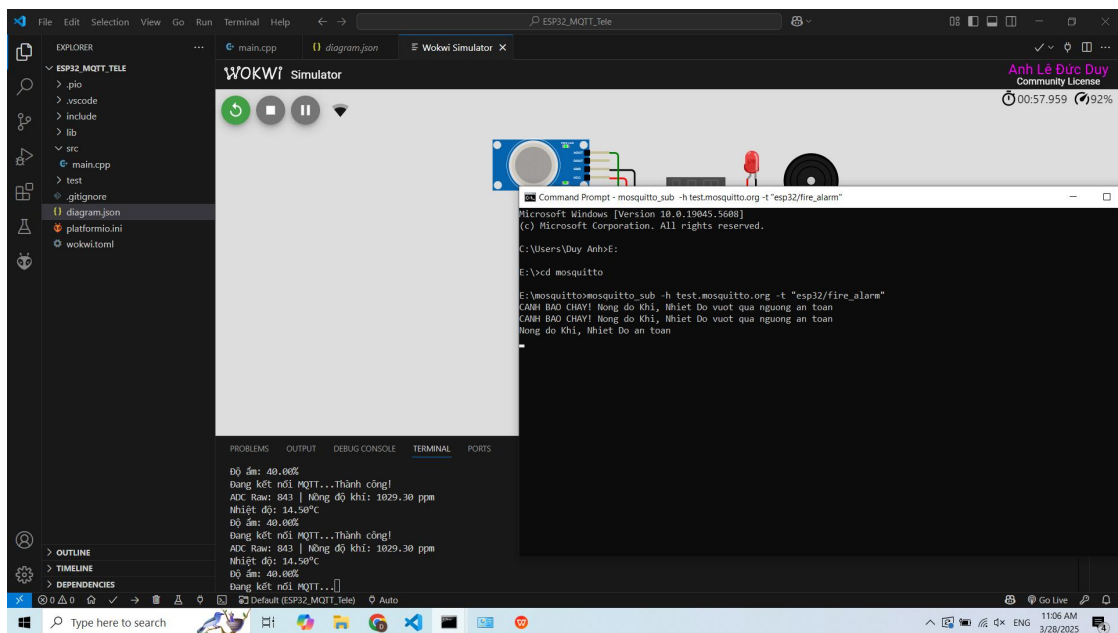
Hình 13. Đăng ký nhận dữ liệu cho MQTT Client

Khi chạy chương trình, nếu nồng độ khói lớn hơn 2000 hoặc nhiệt độ lớn hơn 50 °C thì MQTT client sẽ nhận thông báo “CANH BAO CHAY!!! Nong do Khi, Nhiệt Do vượt ngưỡng an toàn” và chương trình sẽ kiểm tra và gửi tin nhắn sau mỗi 10 phút.



Hình 14. Hệ thống phát hiện cháy trên MQTT

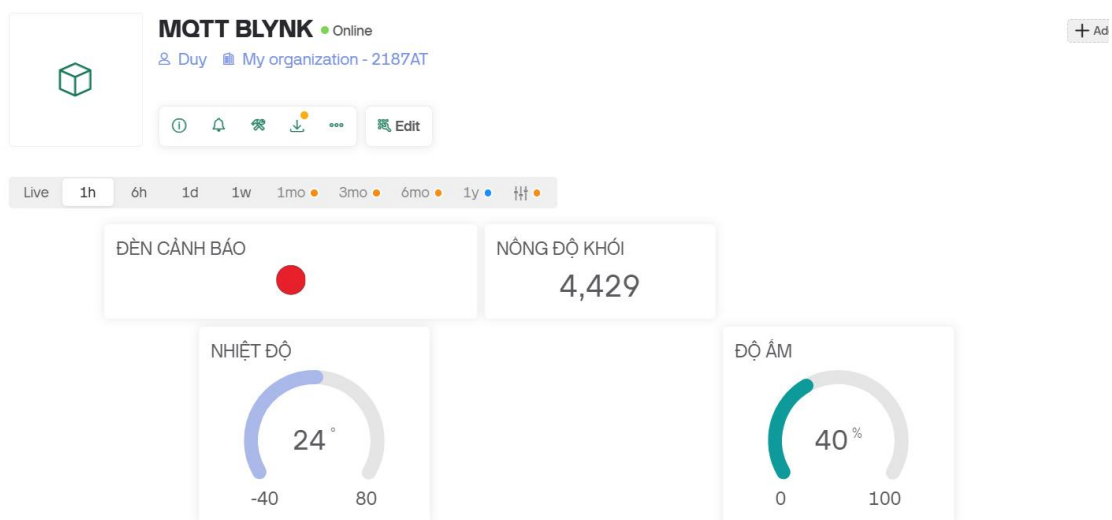
Ngược lại, nếu nồng độ khói nhỏ hơn 2000 và nhiệt độ nhỏ hơn 50 °C thì hiển thị thông báo “Nong do Khi, Nhiet Do an toan”



Hình 15. Hệ thống an toàn trên MQTT

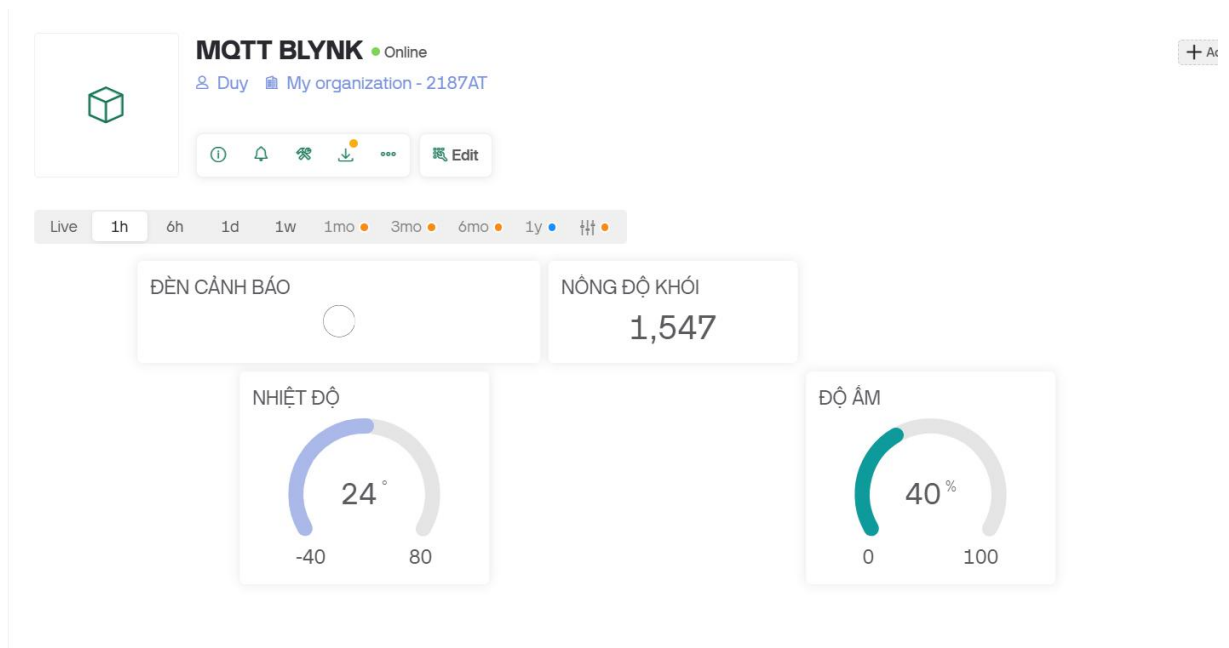
3.5.5. Kiểm tra gửi thông báo đến Blynk:

Khi hệ thống báo cháy phát hiện nồng độ khói hoặc nhiệt độ bất thường:



Hình 16. Hệ thống phát hiện cháy trên Blynk

Khi hệ thống báo cháy trở lại bình thường:



Hình 17. Hệ thống an toàn cháy trên Blynk

PHẦN KẾT LUẬN

Trong bài tiểu luận này, tôi đã nghiên cứu và triển khai thành công hệ thống báo cháy sử dụng ESP32, kết hợp cảm biến khói MQ-2 và cảm biến nhiệt độ, độ ẩm DHT22. Hệ thống hoạt động bằng cách giám sát liên tục môi trường, phát hiện sự gia tăng bất thường của nhiệt độ hoặc mức khói vượt ngưỡng cho phép, từ đó kích hoạt cảnh báo cục bộ bằng đèn LED và còi buzzer, đồng thời gửi cảnh báo từ xa qua giao thức MQTT.

Việc sử dụng ESP32 làm bộ điều khiển trung tâm mang lại nhiều lợi ích, bao gồm khả năng kết nối Wi-Fi, tiêu thụ điện năng thấp và hỗ trợ nhiều giao thức truyền thông. MQTT đã được lựa chọn để đảm bảo truyền tải dữ liệu nhanh chóng, hiệu quả, giúp người dùng có thể nhận cảnh báo cháy kịp thời trên các nền tảng như điện thoại hoặc máy tính.

Qua quá trình thực hiện, tôi nhận thấy hệ thống có độ chính xác cao trong việc phát hiện nguy cơ cháy, đồng thời có thể mở rộng bằng cách tích hợp thêm các cảm biến khác như cảm biến khí CO, camera giám sát hoặc hệ thống chữa cháy tự động. Ngoài ra, việc tối ưu thuật toán đo lường và xử lý tín hiệu cũng có thể giúp giảm thiểu các cảnh báo sai, tăng độ tin cậy của hệ thống.

Tóm lại, hệ thống báo cháy dựa trên ESP32 không chỉ là một ứng dụng thực tế hữu ích mà còn có tiềm năng phát triển trong các mô hình nhà thông minh, công trình dân dụng và công nghiệp. Trong tương lai, tôi sẽ tiếp tục nghiên cứu để cải tiến hệ thống, nâng cao hiệu suất và tích hợp thêm nhiều tính năng nhằm phục vụ tốt hơn nhu cầu thực tế.

TÀI LIỆU THAM KHẢO

[1] Hệ thống báo cháy: Vai trò, duy trì và nâng cấp để bảo vệ an toàn.

<https://pcccپnn.com/danh-muc/he-thong-bao-chay/>

[2] So Sánh Hệ Thống Báo Cháy Thông Minh vs Truyền Thống.

<https://pccc.vn/so-sanh-bao-chay-thong-minh-truyen-thong/>

[3] IoT Là Gì? Ứng Dụng IoT Trong Hệ Thống Báo Cháy Như Thế Nào?

<https://myrobot.asia/ung-dung-iot-trong-he-thong-bao-chay/>

[4] Lập Trình ESP32 từ A đến Z - Khuê Nguyễn Creator

<https://khuenguyencreator.com/lap-trinh-esp32-tu-a-toi-z/>

[5] CẢM BIẾN MQ2, CẢM BIẾN NỒNG ĐỘ KHÍ GA.

<https://dientunhattung.com/san-pham/cam-bien-mq2-cam-bien-nong-do-khi-ga/>

[6] DHT22 CẢM BIẾN ĐO NHIỆT ĐỘ & ĐỘ ẨM CHÍNH XÁC CAO.

<https://dientuduchuy.com/products/dht22-cam-bien-do-nhiet-do-do-am-do-chinh-xac-cao-kem-day-ket-noi-loai-3-pin>

[7] MQTT là gì? Vai trò của MQTT trong IoT.

<https://viblo.asia/p/mqtt-la-gi-vai-tro-cua-mqtt-trong-iot-V3m5WL3bKO7>

[8] Giao thức MQTT trong IoT là gì ? Những ứng dụng của MQTT như thế nào

<https://smartindustry.vn/technology/internet-of-things/giao-thuc-mqtt-la-gi-nhung-ung-dung-cua-mqtt-nhu-the-nao/>