ĐẠI HỌC HUẾ

TRƯỜNG ĐẠI HỌC KHOA HỌC

KHOA CÔNG NGHỆ THÔNG TIN

HỆ THỐNG BÁO CHÁY VỚI ESP32

TÊN LỚP HỌC PHẦN: PHÁT TRIỂN ỨNG DỤNG IOT

MÃ HỌC PHẦN: TIN4024.005

GIẢNG VIÊN HƯỚNG DẪN: VÕ VIỆT DŨNG

HUẾ, THÁNG 4 NĂM 2025

Mục lục

Lời mở đầu

Trong bối cảnh xã hội hiện đại, nơi mà sự phát triển của công nghiệp và đô thị hóa diễn ra với tốc độ nhanh chóng, vấn đề an toàn phòng cháy chữa cháy (PCCC) ngày càng trở nên cấp thiết và nhận được sự quan tâm đặc biệt. Hỏa hoạn, một trong những tai nạn nguy hiểm nhất, không chỉ gây ra những thiệt hại to lớn về vật chất mà còn đe dọa trực tiếp đến tính mạng và sức khỏe của con người. Chính vì lẽ đó, việc xây dựng và triển khai các hệ thống báo cháy hiệu quả, có khả năng phát hiện sớm và cảnh báo kịp thời các nguy cơ tiềm ẩn là một yêu cầu không thể thiếu trong mọi công trình, từ nhà ở dân dụng đến các khu công nghiệp và tòa nhà cao tầng.

Sự trỗi dậy mạnh mẽ của Internet of Things (IoT) đã mang đến những giải pháp thông minh và tiện lợi trong nhiều lĩnh vực của đời sống, và PCCC không nằm ngoài xu hướng đó. Các thiết bị IoT, với khả năng thu thập, xử lý và truyền tải dữ liệu một cách linh hoạt, mở ra một kỷ nguyên mới cho các hệ thống báo cháy. Trong số đó, vi điều khiển ESP32 nổi lên như một nền tảng mạnh mẽ và chi phí hợp lý, tích hợp sẵn kết nối Wi-Fi và Bluetooth, tạo điều kiện thuận lợi cho việc xây dựng các hệ thống thông minh, có khả năng kết nối mạng và tương tác từ xa.

Bài tiểu luận này tập trung vào việc nghiên cứu và mô phỏng một hệ thống báo cháy cơ bản, ứng dụng vi điều khiển ESP32 kết hợp với hai loại cảm biến quan trọng: cảm biến khói (MQ2) và cảm biến nhiệt độ (DHT22). Mục tiêu chính là xây dựng một mô hình hoạt động trên nền tảng mô phỏng Wokwi, minh họa khả năng phát hiện đồng thời nguy cơ cháy thông qua sự thay đổi của nồng độ khói và nhiệt độ, đồng thời truyền tải cảnh báo đến người dùng thông qua giao thức MQTT (Message Queuing Telemetry Transport) - một giao thức truyền thông nhẹ nhàng và hiệu quả, đặc biệt phù hợp cho các ứng dụng IoT.

Phạm vi nghiên cứu của bài tiểu luận giới hạn trong việc thiết kế phần cứng dựa trên các linh kiện cụ thể được cung cấp (ESP32, MQ2, DHT22, BUZZER), mô tả logic lập trình firmware cho ESP32 để thu thập dữ liệu cảm biến, xử lý và gửi cảnh báo qua MQTT, cũng như cấu hình và kiểm thử hệ thống trong môi trường mô phỏng Wokwi. Thông qua quá trình này, bài tiểu luận mong muốn làm rõ tiềm năng của việc ứng dụng ESP32 và giao thức MQTT trong việc xây dựng các hệ thống báo cháy thông minh, góp phần nâng cao hiệu quả công tác PCCC trong tương lai.

Chương 1: Cơ sở lý thuyết về hệ thống báo cháy và giao thức MQTT

1.1 Tổng quan về hệ thống báo cháy:

- **Định nghĩa và vai trò của hệ thống báo cháy:** Hệ thống báo cháy là một tập hợp các thiết bị điện tử được thiết kế để tự động phát hiện và cảnh báo khi có dấu hiệu của hỏa hoạn, bao gồm khói, nhiệt độ tăng cao, hoặc sự xuất hiện của ngọn lửa. Vai trò chính của hệ thống báo cháy là cảnh báo sớm cho người trong khu vực nguy hiểm, tạo điều kiện thuận lợi cho việc sơ tán an toàn, đồng thời thông báo cho lực lượng cứu hỏa để có biện pháp can thiệp kịp thời, từ đó giảm thiểu thiệt hại về người và tài sản.
- **Phân loại các hệ thống báo cháy truyền thống và hiện đại:**
    - **Hệ thống báo cháy truyền thống (Conventional Fire Alarm Systems):** Thường sử dụng các đầu báo được kết nối trực tiếp với trung tâm báo cháy theo các zone (vùng). Khi một đầu báo kích hoạt, trung tâm sẽ xác định được zone có sự cố nhưng không xác định được chính xác vị trí đầu báo. Hệ thống này đơn giản, chi phí thấp nhưng khả năng xác định vị trí sự cố không cao và khó mở rộng.
    - **Hệ thống báo cháy địa chỉ (Addressable Fire Alarm Systems):** Mỗi đầu báo và thiết bị trong hệ thống đều có một địa chỉ riêng. Khi có sự cố, trung tâm báo cháy có thể xác định chính xác vị trí của đầu báo bị kích hoạt. Hệ thống này phức tạp hơn, chi phí cao hơn nhưng cung cấp thông tin chi tiết hơn và dễ dàng mở rộng, quản lý.
    - **Hệ thống báo cháy thông minh (Intelligent Fire Alarm Systems):** Đây là các hệ thống hiện đại, tích hợp công nghệ vi xử lý và phần mềm phức tạp. Chúng có khả năng phân tích dữ liệu từ các đầu báo, giảm thiểu báo động giả, tự động điều chỉnh độ nhạy, và có khả năng giao tiếp với các hệ thống khác (ví dụ: hệ thống quản lý tòa nhà). Các hệ thống này thường tích hợp khả năng kết nối mạng, cho phép giám sát và điều khiển từ xa.
- **Các thành phần cơ bản của một hệ thống báo cháy:**

- **Đầu báo (Detectors):** Là các thiết bị cảm biến có khả năng phát hiện các dấu hiệu của cháy. Các loại đầu báo phổ biến bao gồm:
    - Đầu báo khói (Smoke Detectors): Phát hiện sự có mặt của các hạt khói trong không khí (quang điện, ion hóa).
    - Đầu báo nhiệt (Heat Detectors): Phát hiện sự gia tăng nhiệt độ vượt ngưỡng hoặc tốc độ gia tăng nhiệt độ bất thường.
    - Đầu báo lửa (Flame Detectors): Phát hiện bức xạ hồng ngoại hoặc tử ngoại đặc trưng của ngọn lửa.
    - Đầu báo khí (Gas Detectors): Phát hiện sự rò rỉ của các loại khí dễ cháy.
- **Trung tâm báo cháy (Fire Alarm Control Panel - FACP):** Là bộ não của hệ thống, nhận tín hiệu từ các đầu báo, xử lý và đưa ra các tín hiệu cảnh báo, đồng thời giám sát trạng thái hoạt động của toàn bộ hệ thống.
- **Thiết bị cảnh báo (Notification Appliances):** Là các thiết bị phát ra tín hiệu âm thanh (chuông, còi) và/hoặc ánh sáng (đèn chớp) để thông báo cho mọi người về sự cố cháy.
- **Đường dây tín hiệu và nguồn điện:** Đảm bảo việc truyền tín hiệu giữa các thành phần và cung cấp nguồn điện cho toàn bộ hệ thống.

- **Nguyên lý hoạt động chung của hệ thống báo cháy:** Khi có dấu hiệu của cháy, một hoặc nhiều đầu báo sẽ được kích hoạt và gửi tín hiệu về trung tâm báo cháy. Trung tâm sẽ xử lý tín hiệu này, xác định vị trí (zone hoặc địa chỉ cụ thể) của sự cố, và kích hoạt các thiết bị cảnh báo để thông báo cho người обитающий. Đồng thời, trung tâm cũng có thể gửi tín hiệu đến các hệ thống liên quan khác (ví dụ: hệ thống chữa cháy tự động, hệ thống quản lý tòa nhà) hoặc đến lực lượng cứu hỏa (thông qua các module truyền thông).

1.2 Giới thiệu về ESP32:

- **Tổng quan về vi điều khiển ESP32:** ESP32 là một dòng chip hệ thống trên một vi mạch (System-on-a-Chip - SoC) được phát triển bởi Espressif Systems.

ESP32 DevKitC V4 là một board phát triển phổ biến dựa trên chip ESP32-WROOM-32. Kiến trúc của ESP32 bao gồm một hoặc hai lõi xử lý LX6 Tensilica mạnh mẽ, hoạt động ở tốc độ xung nhịp lên đến 240 MHz, cung cấp hiệu năng xử lý tốt cho nhiều ứng dụng khác nhau. ESP32 tích hợp nhiều ngoại vi và tính năng, bao gồm bộ nhớ Flash, SRAM, các giao tiếp như GPIO, ADC, DAC, SPI, I2C, UART, và đặc biệt là khả năng kết nối Wi-Fi (802.11 b/g/n) và Bluetooth (Classic và Low Energy) tích hợp. Ưu điểm nổi bật của ESP32 là hiệu năng cao, chi phí thấp, khả năng kết nối mạng mạnh mẽ và tiêu thụ năng lượng thấp, làm cho nó trở thành một lựa chọn lý tưởng cho các ứng dụng IoT.

- **Các module và phiên bản phổ biến của ESP32:** Ngoài ESP32-WROOM-32, còn có nhiều module và phiên bản khác của ESP32 với các đặc điểm và kích thước khác nhau, phục vụ cho các yêu cầu ứng dụng cụ thể (ví dụ: ESP32-WROVER với bộ nhớ PSRAM ngoài, ESP32-SOLO-1 với một lõi xử lý).
- **Khả năng kết nối Wi-Fi và Bluetooth tích hợp:** ESP32 tích hợp chuẩn Wi-Fi 802.11 b/g/n, cho phép nó dễ dàng kết nối với mạng không dây và giao tiếp với các thiết bị hoặc dịch vụ trên internet. Bluetooth tích hợp (cả Bluetooth Classic và Bluetooth Low Energy - BLE) mở ra khả năng giao tiếp với các thiết bị di động, cảm biến Bluetooth và các ứng dụng Bluetooth khác, tạo ra sự linh hoạt trong việc xây dựng các hệ thống IoT.
- **Ứng dụng của ESP32 trong lĩnh vực IoT:** ESP32 được ứng dụng rộng rãi trong nhiều lĩnh vực của IoT nhờ vào hiệu năng, khả năng kết nối và chi phí hợp lý. Một số ứng dụng tiêu biểu bao gồm:
    - Hệ thống nhà thông minh (Smart Home): Điều khiển đèn, ổ cắm, khóa cửa, cảm biến.
    - Thiết bị đeo thông minh (Wearable Devices): Đồng hồ thông minh, vòng đeo tay sức khỏe.
    - Cảm biến và giám sát môi trường: Đo nhiệt độ, độ ẩm, chất lượng không khí.

- Hệ thống điều khiển công nghiệp: Tự động hóa quy trình, giám sát thiết bị.
- Các ứng dụng liên quan đến mạng không dây và Bluetooth.

1.3 Giới thiệu về giao thức MQTT:

- **Định nghĩa và lịch sử phát triển của MQTT:** MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông theo kiểu publish/subscribe (xuất bản/đăng ký) cực kỳ đơn giản và nhẹ nhàng, được thiết kế đặc biệt cho các thiết bị có tài nguyên hạn chế và mạng có độ trễ cao hoặc không ổn định, điển hình là trong môi trường IoT. MQTT được phát triển bởi Andy Stanford-Clark của IBM và Arlen Nipper của Arcom (nay là Eurotech) vào năm 1999 cho một dự án giám sát đường ống dẫn dầu. Sau đó, nó trở thành một giao thức mở và ngày càng phổ biến trong cộng đồng IoT.
- **Kiến trúc Publisher-Subscriber trong MQTT:** MQTT hoạt động dựa trên mô hình publish/subscribe, trong đó các thiết bị (publisher) gửi thông điệp đến một trung gian (broker) theo các chủ đề (topic) cụ thể. Các thiết bị khác (subscriber) có thể đăng ký (subscribe) vào các topic mà chúng quan tâm để nhận các thông điệp được xuất bản trên các topic đó. Publisher và subscriber không cần biết trực tiếp về nhau, mà chỉ tương tác thông qua broker.
- **Các thành phần chính của MQTT:**
  - **Broker:** Là máy chủ trung tâm chịu trách nhiệm nhận tất cả các thông điệp từ publisher và chuyển chúng đến các subscriber phù hợp dựa trên các topic mà subscriber đã đăng ký. Broker đóng vai trò trung gian và quản lý toàn bộ luồng thông tin trong hệ thống MQTT.
  - **Publisher:** Là các thiết bị hoặc ứng dụng gửi thông điệp đến broker. Trong ngữ cảnh của bài tiểu luận, ESP32 sẽ đóng vai trò là publisher khi gửi thông tin cảnh báo cháy. Mỗi thông điệp được publisher gửi đi sẽ được gán một topic.
  - **Subscriber:** Là các thiết bị hoặc ứng dụng đăng ký nhận thông điệp từ broker trên một hoặc nhiều topic cụ thể. Trong hệ thống báo cháy thông

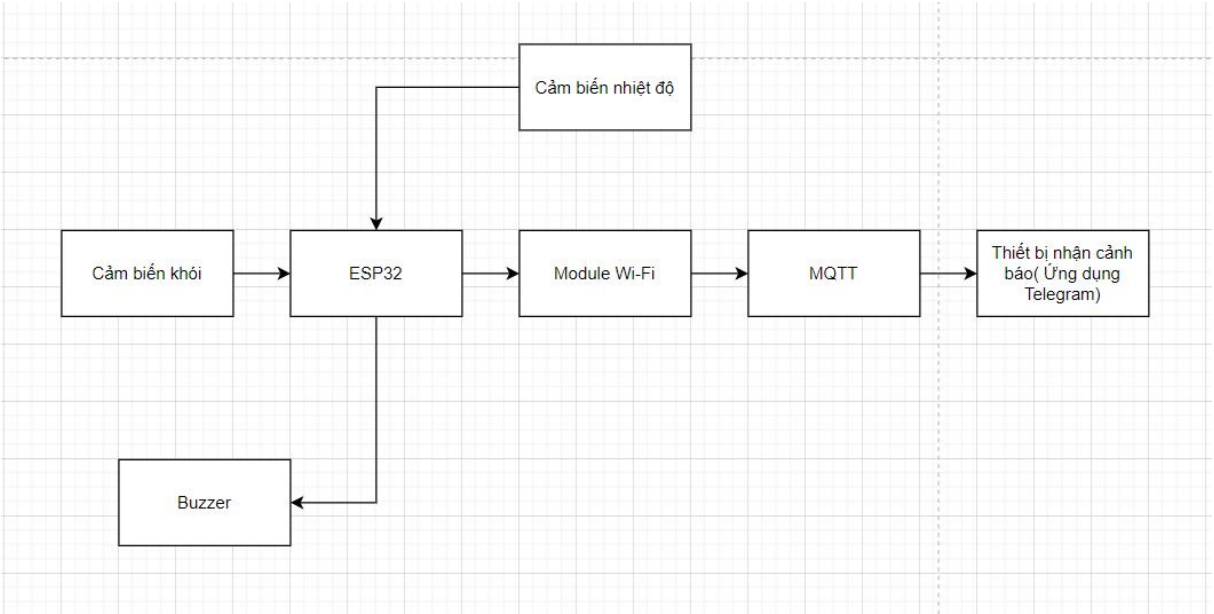
minh này, một ứng dụng trên điện thoại hoặc một hệ thống giám sát trung tâm có thể là subscriber để nhận cảnh báo từ ESP32.

- **Ưu điểm của MQTT trong các ứng dụng IoT:**
    - **Nhẹ (Lightweight):** MQTT có overhead thấp, kích thước gói tin nhỏ, tiêu thụ ít băng thông và tài nguyên, rất phù hợp cho các thiết bị có tài nguyên hạn chế và mạng có băng thông thấp.
    - **Tiết kiệm băng thông (Bandwidth Efficient):** Do thiết kế đơn giản và kích thước gói tin nhỏ, MQTT giúp giảm thiểu lượng dữ liệu truyền tải trên mạng.
    - **Tin cậy (Reliable):** MQTT hỗ trợ các mức chất lượng dịch vụ (Quality of Service - QoS) khác nhau để đảm bảo độ tin cậy của việc truyền tải thông điệp, ngay cả trong môi trường mạng không ổn định.
    - **Khả năng mở rộng (Scalable):** Kiến trúc publish/subscribe cho phép hệ thống MQTT dễ dàng mở rộng với số lượng lớn thiết bị và người dùng.
    - **Hoạt động tốt trên mạng không ổn định (Works well on unreliable networks):** MQTT được thiết kế để hoạt động hiệu quả ngay cả trên các kết nối mạng có độ trễ cao hoặc thường xuyên bị gián đoạn.
- **QoS (Quality of Service) trong MQTT:** MQTT cung cấp ba mức chất lượng dịch vụ (QoS) để đảm bảo độ tin cậy của việc truyền tải thông điệp:
    - **QoS 0 (At most once):** Thông điệp được gửi đi một lần mà không có bất kỳ cơ chế xác nhận nào. Có thể xảy ra mất mát thông điệp.
    - **QoS 1 (At least once):** Thông điệp được gửi đi ít nhất một lần và sẽ được gửi lại cho đến khi publisher nhận được xác nhận (PUBACK) từ broker. Đảm bảo thông điệp đến được subscriber nhưng có thể có trường hợp trùng lặp.
    - **QoS 2 (Exactly once):** Đảm bảo thông điệp được gửi và nhận chính xác một lần duy nhất thông qua một quy trình trao đổi bốn bước phức tạp hơn. Đây là mức QoS cao nhất nhưng cũng tốn tài nguyên nhất.

Chương 2: Thiết kế phần cứng và lựa chọn linh kiện

2.1 Sơ đồ khối tổng quan của hệ thống:

- Hệ thống báo cháy thông minh sử dụng ESP32, cảm biến khói MQ2 và cảm biến nhiệt độ DHT22 có thể được biểu diễn bằng sơ đồ khối sau:



Mô tả:

- **Cảm biến khói MQ2:** Phát hiện nồng độ khói và các loại khí dễ cháy trong môi trường, gửi tín hiệu analog đến ESP32.
- **Cảm biến nhiệt độ DHT22:** Đo nhiệt độ và độ ẩm của môi trường, gửi dữ liệu số đến ESP32.
- **ESP32 (Vi điều khiển và xử lý):** Nhận dữ liệu từ cảm biến MQ2 (qua chân analog) và DHT22 (qua giao tiếp số). ESP32 thực hiện việc xử lý dữ liệu, so sánh với các ngưỡng cài đặt. Khi phát hiện nguy cơ cháy (khói vượt ngưỡng hoặc nhiệt độ tăng cao), ESP32 sẽ kích hoạt cảnh báo cục bộ (BUZZER) và gửi thông điệp cảnh báo qua module Wi-Fi tích hợp.
- **Module Wi-Fi (Tích hợp):** ESP32 sử dụng module Wi-Fi tích hợp để kết nối với mạng không dây và giao tiếp với MQTT Broker.

- **MQTT Broker(Telegram):** Là máy chủ trung gian nhận thông điệp cảnh báo từ ESP32 (publisher) và chuyển tiếp đến các thiết bị hoặc ứng dụng đã đăng ký (subscriber).
- **Thiết bị nhận cảnh báo (Ứng dụng di động):** Ứng dụng đã đăng ký theo dõi topic cảnh báo từ hệ thống sẽ nhận được thông báo khi có nguy cơ cháy.
- **Buzzer (Cảnh báo cục bộ):** Được điều khiển trực tiếp bởi ESP32, phát chuông khi phát hiện nguy cơ cháy để cảnh báo cho những người ở gần.

2.2 Lựa chọn cảm biến:

2.2.1 Cảm biến khói MQ2 (chip2):

- Giới thiệu về cảm biến khói MQ2: MQ2 là một cảm biến khói phổ biến, có khả năng phát hiện nhiều loại khí như LPG, i-butan, propan, metan, rượu, hydro và khói. Nó hoạt động dựa trên nguyên lý thay đổi điện trở của một lớp vật liệu nhạy cảm khi tiếp xúc với các loại khí này.
- Nguyên lý hoạt động và đặc điểm kỹ thuật của cảm biến MQ2: Cảm biến MQ2 có một phần tử đốt nóng bên trong để duy trì nhiệt độ hoạt động. Điện trở của lớp cảm biến thay đổi tỷ lệ với nồng độ khí trong môi trường. Cảm biến cung cấp đầu ra analog (AO) mà giá trị điện áp này sẽ thay đổi theo nồng độ khí. Ngoài ra, một số module MQ2 còn có đầu ra số (DO) với một ngưỡng có thể điều chỉnh. Tuy nhiên, với mục tiêu thu thập dữ liệu chi tiết hơn, việc sử dụng đầu ra analog sẽ phù hợp hơn.
- **Cách kết nối chân AO, VCC, GND của MQ2 với ESP32 (dựa trên dữ liệu connections):**
  - Chân **AO** (Analog Output) của MQ2 (ID: chip2) được kết nối với chân **GPIO35** của ESP32 để đọc giá trị analog (dây màu xanh lá cây).
  - Chân **VCC** của MQ2 (ID: chip2) được kết nối với chân **3V3** của ESP32 để cấp nguồn (dây màu đỏ).
  - Chân **GND** của MQ2 (ID: chip2) được kết nối với chân **GND.1** của ESP32 để đảm bảo mạch điện chung (dây màu đen).

2.2.2 Cảm biến nhiệt độ DHT22 (dht1):

- Giới thiệu về cảm biến nhiệt độ DHT22: DHT22 là một cảm biến nhiệt độ và độ ẩm kỹ thuật số, cung cấp độ chính xác cao hơn so với DHT11. Nó giao tiếp với vi điều khiển thông qua một giao thức truyền dữ liệu nối tiếp một dây.
- Nguyên lý hoạt động và đặc điểm kỹ thuật của cảm biến DHT22: DHT22 sử dụng một phần tử cảm biến điện dung cho độ ẩm và một nhiệt điện trở cho nhiệt độ. Một chip bên trong cảm biến xử lý các tín hiệu và chuyển đổi chúng thành dữ liệu số. DHT22 có dải đo nhiệt độ từ -40°C đến 125°C với độ chính xác ±0.5°C và dải đo độ ẩm từ 0% đến 100% RH với độ chính xác ±2-5% RH.
- **Cách kết nối chân SDA, VCC, GND của DHT22 với ESP32 (dựa trên dữ liệu connections):**
  - Chân **SDA** (Data) của DHT22 (ID: dht1) được kết nối với chân **GPIO4** của ESP32 để truyền dữ liệu số (dây màu xanh lá cây).
  - Chân **VCC** của DHT22 (ID: dht1) được kết nối với chân **3V3** của ESP32 để cấp nguồn (dây màu đỏ).
  - Chân **GND** của DHT22 (ID: dht1) được kết nối với chân **GND.2** của ESP32 để đảm bảo mạch điện chung (dây màu đen).

2.3 ESP32 và module Wi-Fi:

- Lựa chọn module ESP32 DevKitC V4: Board ESP32 DevKitC V4 (ID: esp) sử dụng module ESP32-WROOM-32, một module mạnh mẽ và phổ biến, tích hợp Wi-Fi 802.11 b/g/n (2.4 GHz) và Bluetooth v4.2 BR/EDR & BLE. ESP32-WROOM-32 có bộ vi xử lý lõi kép, bộ nhớ Flash và SRAM đủ lớn cho các ứng dụng IoT, và cung cấp nhiều chân GPIO để kết nối với các cảm biến và thiết bị ngoại vi.
- Giải thích về khả năng kết nối Wi-Fi tích hợp của ESP32: ESP32 tích hợp một chip Wi-Fi cho phép nó hoạt động như một trạm (station) để kết nối với mạng Wi-Fi hiện có, hoặc như một điểm truy cập (access point) để các thiết bị khác có thể kết nối đến nó. Trong hệ thống báo cháy này, ESP32 sẽ hoạt động ở chế

độ trạm để kết nối với mạng Wi-Fi cục bộ và giao tiếp với MQTT Broker (Blynk) trên internet.

2.4 Các linh kiện phụ trợ:

- Đề cập đến việc sử dụng Buzzer để cảnh báo cục bộ: Một còi Buzzer (ID: bz1) được sử dụng để cung cấp cảnh báo âm thanh tại chỗ khi hệ thống phát hiện nguy cơ cháy.
- **Cách kết nối chân 2 và 1 của Buzzer với ESP32 (dựa trên dữ liệu connections):**
    - Chân **2** của Buzzer (ID: bz1) được kết nối với chân **GPIO15** của ESP32 (dây màu xanh lá cây).
    - Chân **1** của Buzzer (ID: bz1) được kết nối với chân **GND.1** của ESP32 để hoàn thành mạch điện (dây màu đen). **Lưu ý:** Cần có điện trở hạn dòng mắc nối tiếp với Buzzer trong thực tế.

2.5 Bảng kết nối phần cứng

Linh kiện	**Chân linh kiện**	**ESP32 Chân**	**Mục đích**	**Màu dây (trong JSON)**
DHT22	SDA	GPIO4	Truyền dữ liệu nhiệt độ và độ ẩm	Xanh lá
DHT22	VCC	3V3	Cấp nguồn cho DHT22	Đỏ
DHT22	GND	GND.2	Mass (Ground) cho DHT22	Đen
MQ2	AO	GPIO35	Tín hiệu analog nồng độ khí/khói	Xanh lá
MQ2	VCC	3V3	Cấp nguồn cho MQ2	Đỏ

MQ2	GND	GND.1	Mass (Ground) cho MQ2	Đen
Buzzer	2	GPIO15	Chân điều khiển Buzzer (tín hiệu ra)	Xanh lá
Buzzer	1	GND.1	Mass (Ground) cho Buzzer	Đen
ESP32 DevKitC V4	TX	$serialMonitor:RX	Gửi dữ liệu đến Serial Monitor	
ESP32 DevKitC V4	RX	$serialMonitor:TX	Nhận dữ liệu từ Serial Monitor	

2.6 Cảm biến không xác định (chip3):

- Cấu hình JSON bao gồm một thành phần chip3 với các kết nối tương tự như MQ2, được kết nối với chân GPIO34 của ESP32. Tuy nhiên, loại cảm biến này không được xác định rõ trong danh sách parts. Nếu đây là một cảm biến analog khác (ví dụ: một cảm biến khí khác), nó có thể được đọc và xử lý tương tự như MQ2.
    - Chân **AO** của chip3 được kết nối với chân **GPIO34** của ESP32 (dây màu xanh lá cây).
    - Chân **VCC** của chip3 được kết nối với chân **3V3** của ESP32 (dây màu đỏ).
    - Chân **GND** của chip3 được kết nối với chân **GND.1** của ESP32 (dây màu đen).

Chương 3: Thiết kế và lập trình firmware cho ESP32 (Mô tả logic)

3.1 Mô tả hoạt động hệ thống

1. **Bắt đầu**: Hệ thống khởi động và bắt đầu quá trình hoạt động.
2. **Kết nối Wi-Fi thành công:** ESP32 sẽ cố gắng thiết lập kết nối với mạng Wi-Fi đã được cấu hình trước đó. Quá trình này cần thành công để có thể thực hiện các bước tiếp theo liên quan đến giao tiếp MQTT. Nếu kết nối không thành công, ESP32 sẽ tiếp tục thử lại cho đến khi kết nối được thiết lập.
3. **Kết nối MQTT Broker (Telegram) thành công**: Sau khi kết nối Wi-Fi thành công, ESP32 sẽ tiến hành thiết lập kết nối với MQTT Broker (Telegram) đã được chỉ định (địa chỉ và cổng). Kết nối này là cần thiết để ESP32 có thể gửi các thông điệp cảnh báo khi phát hiện nguy cơ cháy. Tương tự như kết nối Wi-Fi, nếu kết nối MQTT không thành công, ESP32 sẽ tiếp tục thử lại.
4. **Đọc giá trị cảm biến MQ2**: ESP32 sẽ đọc dữ liệu từ cảm biến khói MQ2. Cảm biến này cung cấp một giá trị analog tương ứng với nồng độ khói và các loại khí khác. Giá trị này cần được đọc để phân tích.
5. **Đọc giá trị cảm biến DHT22**: Tiếp theo, ESP32 sẽ đọc dữ liệu từ cảm biến nhiệt độ DHT22. Cảm biến này cung cấp giá trị nhiệt độ môi trường hiện tại.
6. **Giá trị khói > 6%:** Giá trị đọc được từ cảm biến MQ2 sẽ được chuyển đổi hoặc so sánh với một ngưỡng tương ứng với nồng độ khói vượt quá 6%. Nếu giá trị này lớn hơn ngưỡng, hệ thống xác định có nguy cơ cháy do khói.
7. **Phát chuông Buzzer:** Nếu nguy cơ cháy do khói được xác định (giá trị khói vượt ngưỡng), Buzzzer cảnh báo cục bộ sẽ được kích hoạt (bật sáng) để thông báo cho những người ở gần.
8. **Gửi cảnh báo MQTT (Khói):** Một thông điệp sẽ được tạo và gửi qua giao thức MQTT đến MQTT Broker (Telegram), thông báo rằng nguy cơ cháy đã được phát hiện do nồng độ khói cao. Thông điệp này có thể bao gồm thông tin chi tiết về thời điểm phát hiện và giá trị khói đo được.

9. **Giá trị nhiệt độ > 50°C:** Nếu không phát hiện nguy cơ cháy do khói, hệ thống sẽ kiểm tra xem giá trị nhiệt độ đọc được từ cảm biến DHT22 có vượt quá ngưỡng 50°C hay không. Nếu nhiệt độ vượt quá ngưỡng này, hệ thống xác định có nguy cơ cháy do nhiệt độ cao.

10. **Không phát chuông Buzzer**: Nếu cả giá trị khói và nhiệt độ đều không vượt quá ngưỡng nguy hiểm, Buzzer cảnh báo sẽ được tắt (nếu trước đó nó đã được bật). Điều này cho thấy môi trường hiện tại không có dấu hiệu của nguy cơ cháy.

11. **Gửi cảnh báo MQTT (Nhiệt độ) - tùy chọn**: Tùy thuộc vào yêu cầu của hệ thống, một thông điệp MQTT có thể được gửi để thông báo về nhiệt độ cao, ngay cả khi không có khói.

12. **Delay:** Sau khi thực hiện các bước kiểm tra và cảnh báo (nếu có), hệ thống sẽ tạm dừng trong một khoảng thời gian ngắn trước khi quay lại bước đọc giá trị cảm biến. Vòng lặp này đảm bảo hệ thống liên tục theo dõi môi trường để phát hiện các thay đổi.

3.2 Cài đặt môi trường phát triển cho ESP32 trên Wokwi:

Wokwi cung cấp một môi trường phát triển tích hợp trực tuyến. Để bắt đầu, người dùng chỉ cần truy cập trang web Wokwi và tạo một dự án ESP32 mới. Nền tảng này cung cấp một trình soạn thảo code trực quan, nơi có thể viết và chỉnh sửa code cho ESP32 mà không cần cài đặt bất kỳ phần mềm bổ sung nào trên máy tính. Wokwi đã bao gồm các thư viện cần thiết cho ESP32, giúp người dùng tập trung vào logic của ứng dụng.

3.3 Lập trình đọc dữ liệu từ cảm biến (Mô tả logic):

- **Đọc giá trị từ cảm biến khói MQ2 (kết nối với GPIO35):** ESP32 sẽ sử dụng chức năng analogRead() để đọc giá trị điện áp analog từ chân GPIO35, nơi chân AO của cảm biến MQ2 được kết nối. Giá trị analog này sẽ tương ứng với nồng độ khói hiện tại. Để xác định xem có nguy cơ cháy hay không, giá trị analog đọc được cần được chuyển đổi sang một thang đo phần trăm hoặc so sánh với một giá trị ngưỡng đã được xác định trước tương ứng với 6% khói. Quá trình

chuyển đổi này có thể dựa trên datasheet của cảm biến hoặc thông qua hiệu chuẩn thực tế.

- **Đọc giá trị từ cảm biến nhiệt độ DHT22 (kết nối với GPIO4):** ESP32 sẽ sử dụng một thư viện (ví dụ: Adafruit DHT) để giao tiếp với cảm biến DHT22 thông qua chân GPIO4, nơi chân SDA của DHT22 được kết nối. Thư viện này sẽ xử lý các chi tiết giao tiếp phức tạp và cung cấp giá trị nhiệt độ hiện tại dưới dạng số (thường là độ Celsius).
- **Đọc giá trị từ cảm biến không xác định (nếu có, kết nối với GPIO34):** Nếu cảm biến chip3 là một cảm biến analog, ESP32 sẽ đọc giá trị từ chân GPIO34, nơi chân AO của cảm biến này được kết nối, tương tự như cách đọc giá trị từ MQ2.
- **Xử lý dữ liệu cảm biến:** Dữ liệu đọc được từ cảm biến có thể được xử lý để loại bỏ nhiễu (ví dụ: sử dụng bộ lọc). Giá trị từ cảm biến MQ2 và cảm biến không xác định (nếu có) cần được diễn giải để xác định nồng độ tương đối. Giá trị nhiệt độ từ DHT22 sẽ được sử dụng trực tiếp.
- **Định nghĩa ngưỡng phát hiện cháy:** Hệ thống sẽ sử dụng hai ngưỡng (hoặc nhiều hơn nếu có thêm cảm biến) để xác định nguy cơ cháy:
    - **Ngưỡng khói:** Một giá trị tương ứng với nồng độ khói > 6% (dựa trên giá trị đọc từ GPIO35).
    - **Ngưỡng nhiệt độ:** 50°C (dựa trên giá trị đọc từ GPIO4).
    - **Ngưỡng cho cảm biến không xác định (nếu có):** Một ngưỡng phù hợp với loại cảm biến được kết nối với GPIO34**.**

3.4 Lập trình kết nối Wi-Fi (Mô tả logic):

ESP32 sẽ được lập trình để tự động kết nối với mạng Wi-Fi đã được cấu hình (tên mạng - SSID và mật khẩu). Quá trình này bao gồm việc quét các mạng khả dụng, tìm kiếm SSID đã cấu hình và thực hiện quá trình xác thực. ESP32 sẽ liên tục kiểm tra trạng thái kết nối và cố gắng kết nối lại nếu kết nối bị mất.

3.5 Lập trình giao tiếp MQTT (Mô tả logic):

Sau khi kết nối Wi-Fi thành công, ESP32 sẽ thiết lập kết nối với một MQTT Broker (Telegram). ESP32 sẽ hoạt động như một "publisher", gửi các thông điệp cảnh báo đến một topic cụ thể. Khi phát hiện nguy cơ cháy (khói > 6% hoặc nhiệt độ > 50°C), ESP32 sẽ tạo một thông điệp MQTT chứa thông tin về loại cảnh báo (khói hoặc nhiệt độ) và giá trị đo được, sau đó gửi thông điệp này đếnTelegram.

3.6 Tích hợp các module và xử lý logic:

Chương trình trên ESP32 sẽ hoạt động trong một vòng lặp liên tục. Trong mỗi lần lặp, nó sẽ thực hiện các bước sau: đọc dữ liệu từ cả hai cảm biến, so sánh các giá trị với ngưỡng đã định nghĩa (6% cho khói và 50°C cho nhiệt độ). Nếu một trong hai ngưỡng bị vượt quá, ESP32 sẽ bật **Buzzer (kết nối với GPIO15)** để cảnh báo cục bộ và gửi một thông điệp cảnh báo qua MQTT đến Telegram. Nếu tất cả các giá trị đều dưới ngưỡng, Buzzer sẽ tắt và không có cảnh báo MQTT nào được gửi (trừ khi có yêu cầu gửi thông báo trạng thái định kỳ).

Chương 4: Mô phỏng hệ thống trên Wokwi

4.1 Giới thiệu về nền tảng Wokwi:

Wokwi là một trình giả lập trực tuyến cho phép mô phỏng các dự án điện tử, bao gồm ESP32, cảm biến và giao tiếp mạng MQTT.

4.2 Thiết kế mạch trên Wokwi:

Mạch được thiết kế trên Wokwi với các linh kiện ESP32 DevKitC V4 (esp), Gas Sensor (MQ2) (chip2), DHT22 (dht1), và Buzzer (bz1) kết nối như sau:

chip2:AO → esp:35

chip2:VCC → esp:3V3

chip2:GND → esp:GND.1

dht1:SDA → esp:4

dht1:VCC → esp:3V3

dht1:GND → esp:GND.2

bz1:2 → esp:15

bz1:1 → esp:GND.1

4.3 Viết code trên Wokwi:

Code Arduino được viết để đọc dữ liệu từ cảm biến MQ2 (chân AO kết nối với GPIO35) và DHT22 (chân SDA kết nối với GPIO4), kết nối Wi-Fi, giao tiếp MQTT với Broker Telegram, và điều khiển Buzzer (chân 2 kết nối với GPIO15) dựa trên ngưỡng phát hiện cháy (khói > 6% và nhiệt độ > 50°C). Nếu có cảm biến không xác định (chip3) kết nối với GPIO34, code cũng sẽ đọc và xử lý dữ liệu từ cảm biến này.

4.4 Cấu hình MQTT Broker:

Sử dụng MQTT Broker trực tuyến (Telegram) được cấu hình trong code ESP32.

4.5 Mô phỏng :

Trường hợp 1: Nhiệt độ và khói chưa vượt quá ngưỡng

- **Thiết lập mô phỏng:**
    - Giá trị concentration của mq2 được điều chỉnh để tương ứng với nồng độ khói dưới 6%.
    - Giá trị temperature của dht1 được đặt dưới 50°C.

Trường hợp 2: Khói vượt quá ngưỡng, nhiệt độ chưa vượt quá ngưỡng

- **Thiết lập mô phỏng:**
    - Giá trị concentration của mq2 được điều chỉnh để tương ứng với nồng độ khói trên 6%.
    - Giá trị temperature của dht1 được đặt dưới 50°C.

Trường hợp 3: Nhiệt độ vượt quá ngưỡng, khói chưa vượt quá ngưỡng

- **Thiết lập mô phỏng:**
    - Giá trị concentration của mq2 được điều chỉnh để tương ứng với nồng độ khói dưới 6%.

- Giá trị temperature của dht1 được đặt trên 50°C.

Trường hợp 4: Cả khói và nhiệt độ đều vượt quá ngưỡng

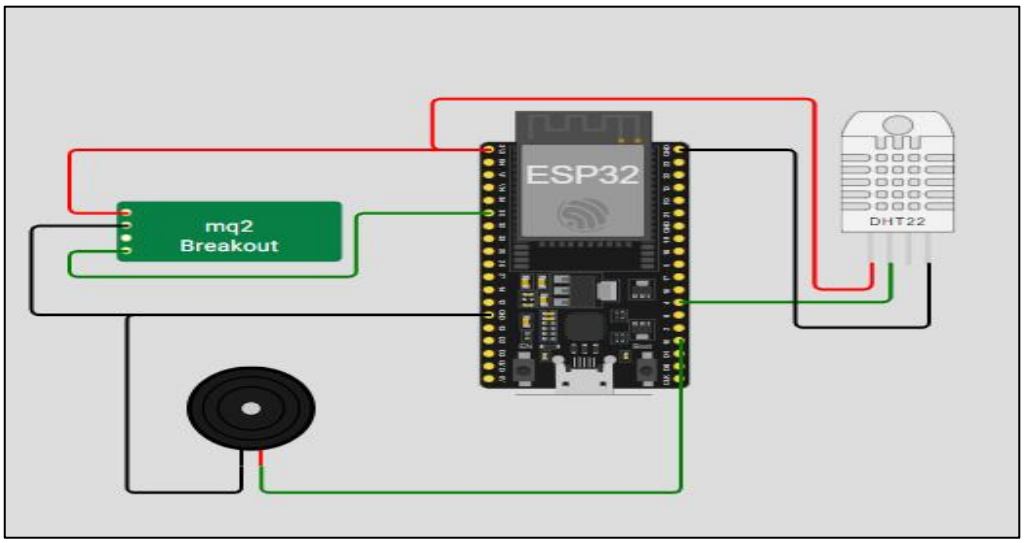
- **Thiết lập mô phỏng:**
  - Giá trị concentration của mq2 được điều chỉnh để tương ứng với nồng độ khói trên 6%.
  - Giá trị temperature của dht1 được đặt trên 50°C.

Chương 5: Kết quả

5.1 Trình bày kết quả mô phỏng:

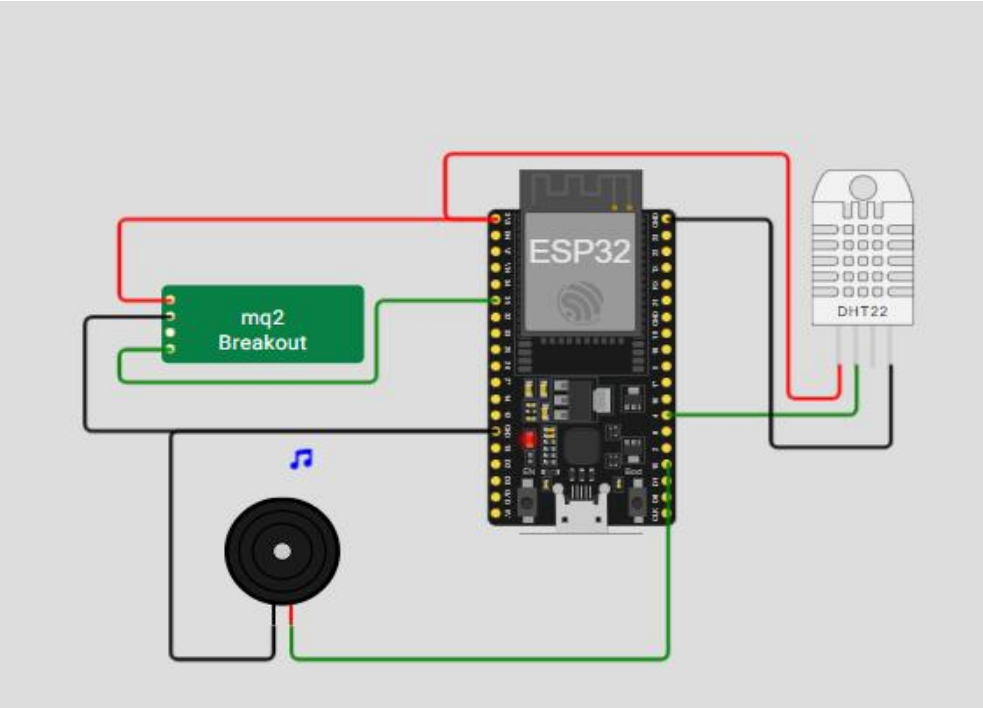
Sau quá trình mô phỏng hệ thống báo cháy thông minh trên nền tảng Wokwi với các trường hợp kiểm thử khác nhau, các kết quả sau đã được quan sát:

- **Trường hợp 1: Nhiệt độ và khói chưa vượt quá ngưỡng:**



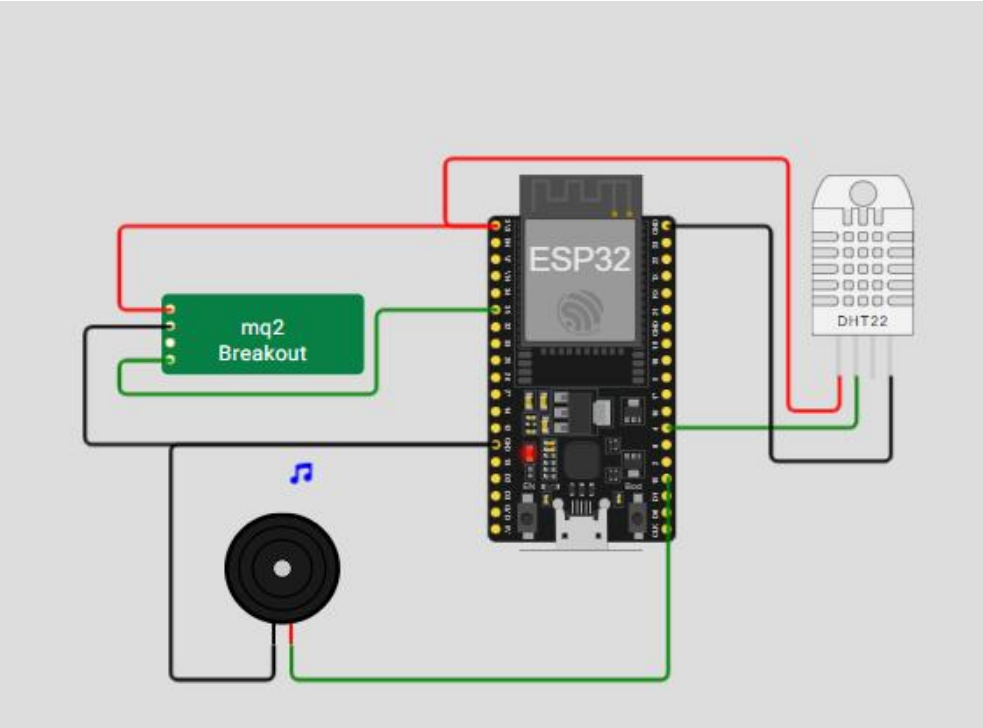
Khi này cả nhiệt độ và khói đều không vượt quá ngưỡng cảnh báo cháy, hệ thống chưa gửi thông tin đến Telegram

- **Trường hợp 2: Khói vượt quá ngưỡng, nhiệt độ chưa vượt quá ngưỡng:**



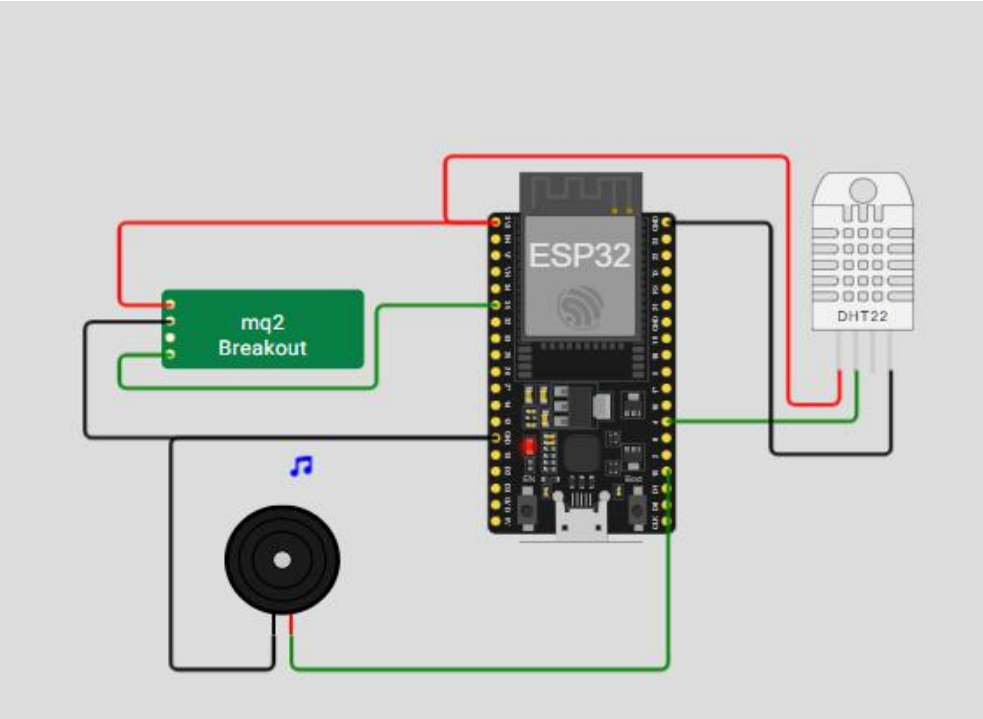
Khi này khói đã vượt ngưỡng 6%, hệ thống liền gửi cảnh báo "Khói vượt ngưỡng cảnh báo và có nguy cơ cháy cao." và gửi chính xác nồng độ khói đo được, đồng thời chuông Buzzer cũng reo lên.

- **Trường hợp 3: Nhiệt độ vượt quá ngưỡng, khói chưa vượt quá ngưỡng:**



Khi này nhiệt độ đã vượt ngưỡng 50°C., hệ thống liền gửi cảnh báo "Nhiệt độ vượt ngưỡng cảnh báo có nguy cơ cháy cao." và gửi chính xác nhiệt độ đo được, đồng thời chuông Buzzer cũng reo lên.

- **Trường hợp 4: Cả khói và nhiệt độ đều vượt quá ngưỡng:**



Khi này nhiệt độ và khói đã vượt ngưỡng: Nhiệt độ > 50°C, nồng độ khói > 6% hệ thống liền gửi cảnh báo "Nhiệt độ và khói vượt ngưỡng cảnh báo có nguy cơ cháy rất cao." và gửi chính xác nhiệt độ, nồng độ khói đồng thời chuông Buzzer cũng reo lên.

5.2 Phân tích kết quả thu được:

Kết quả mô phỏng cho thấy hệ thống báo cháy thông minh cơ bản được thiết kế đã hoạt động theo đúng logic dự kiến trong môi trường Wokwi. Hệ thống có khả năng:

- **Giám sát đồng thời** nồng độ khói và nhiệt độ môi trường thông qua cảm biến MQ2 và DHT22.
- **Phát hiện** các nguy cơ cháy tiềm ẩn khi giá trị khói vượt quá ngưỡng 6% hoặc nhiệt độ vượt quá 50°C.

- **Cảnh báo cục bộ** bằng cách kích hoạt đèn Buzzer khi phát hiện nguy cơ cháy từ bất kỳ cảm biến nào.
- **Gửi cảnh báo từ xa** thông qua giao thức MQTT đến một MQTT Broker (Telegram), cho phép các thiết bị hoặc ứng dụng đã đăng ký nhận thông tin cảnh báo kịp thời.

Việc hệ thống phản ứng chính xác trong cả bốn trường hợp kiểm thử cho thấy tính khả thi của việc sử dụng ESP32, cảm biến MQ2 và DHT22 kết hợp với giao thức MQTT để xây dựng một hệ thống báo cháy thông minh cơ bản.

5.3 Thảo luận về những ưu điểm và hạn chế của hệ thống mô phỏng:

Ưu điểm:

Khả năng phát hiện đa yếu tố: Hệ thống có khả năng phát hiện nguy cơ cháy dựa trên cả khói và nhiệt độ, tăng độ tin cậy và giảm nguy cơ báo động giả do một yếu tố duy nhất.

Cảnh báo cục bộ và từ xa: Việc tích hợp cả cảnh báo bằng đèn LED tại chỗ và gửi thông báo qua MQTT đảm bảo thông tin về nguy cơ cháy được truyền đạt đến người dùng ở cả gần và xa.

Sử dụng nền tảng IoT phổ biến: ESP32 là một vi điều khiển mạnh mẽ, chi phí thấp và được cộng đồng hỗ trợ rộng rãi, phù hợp cho các ứng dụng IoT.

Giao thức MQTT hiệu quả: MQTT là một giao thức nhẹ, tiết kiệm băng thông và phù hợp cho việc truyền tải thông tin cảnh báo trong môi trường mạng không ổn định.

Mô phỏng hiệu quả trên Wokwi: Nền tảng Wokwi cung cấp một môi trường thuận tiện và trực quan để thiết kế, lập trình và kiểm thử hệ thống mà không cần phần cứng thực tế.

Hạn chế:

Mô hình cảm biến đơn giản: Mô hình cảm biến MQ2 trên Wokwi có thể không hoàn toàn phản ánh độ nhạy và đặc tính phức tạp của cảm biến thực tế đối với các loại khói và khí khác nhau. Việc hiệu chuẩn cảm biến thực tế sẽ phức tạp hơn.

Ngưỡng cảnh báo cố định: Ngưỡng cảnh báo cho khói và nhiệt độ được đặt cố định trong code. Trong thực tế, có thể cần cơ chế điều chỉnh ngưỡng linh hoạt hơn.

Thiếu các tính năng nâng cao: Hệ thống mô phỏng này là một phiên bản cơ bản, thiếu các tính năng nâng cao như phân biệt loại khói, khả năng tự kiểm tra lỗi, hoặc tích hợp với các hệ thống PCCC khác.

Giả lập MQTT: Việc mô phỏng giao tiếp MQTT trên Wokwi có thể đơn giản hóa một số khía cạnh so với việc tương tác với một Broker thực tế trên internet.

Kết luận

Bài tiểu luận này đã trình bày quá trình nghiên cứu, thiết kế và mô phỏng một hệ thống báo cháy thông minh cơ bản, ứng dụng vi điều khiển ESP32 kết hợp với cảm biến khói MQ2 và cảm biến nhiệt độ DHT22. Mục tiêu chính của việc xây dựng mô hình trên nền tảng Wokwi là minh họa khả năng phát hiện đồng thời nguy cơ cháy dựa trên sự thay đổi của nồng độ khói và nhiệt độ, đồng thời truyền tải cảnh báo đến người dùng thông qua giao thức MQTT và Telegram.

Quá trình mô phỏng đã thành công trong việc thể hiện logic hoạt động của hệ thống. Trong các trường hợp kiểm thử khác nhau, hệ thống đã phản ứng chính xác bằng cách kích hoạt cảnh báo cục bộ (Buzzer) và gửi thông báo đến ứng dụng Telegram khi phát hiện nồng độ khói hoặc nhiệt độ vượt quá ngưỡng an toàn đã định. Điều này cho thấy tiềm năng của việc kết hợp ESP32, các cảm biến phổ biến và giao thức MQTT trong việc xây dựng các hệ thống báo cháy thông minh, mang lại khả năng cảnh báo sớm và từ xa.

Mặc dù hệ thống mô phỏng này chỉ là một phiên bản cơ bản và còn tồn tại một số hạn chế như mô hình cảm biến đơn giản, ngưỡng cảnh báo cố định và thiếu các tính năng nâng cao, nó vẫn cung cấp một nền tảng vững chắc để hiểu rõ nguyên lý hoạt động và khả năng ứng dụng của công nghệ IoT trong lĩnh vực phòng cháy chữa cháy.

Trong tương lai, hệ thống này có thể được phát triển thêm bằng cách tích hợp các loại cảm biến khác, cải thiện thuật toán xử lý dữ liệu để giảm thiểu báo động giả, bổ sung khả năng cấu hình ngưỡng từ xa, và kết nối với các hệ thống PCCC chuyên nghiệp hơn. Việc triển khai hệ thống trên phần cứng thực tế cũng sẽ đặt ra những thách thức và yêu cầu về hiệu chuẩn cảm biến, đảm bảo độ ổn định của kết nối mạng và nguồn điện, cũng như tính an toàn và độ tin cậy của toàn bộ hệ thống trong môi trường thực tế.

Tuy nhiên, với sự phát triển không ngừng của công nghệ IoT, các hệ thống báo cháy thông minh dựa trên các nền tảng như ESP32 và giao thức MQTT hứa hẹn sẽ đóng một vai trò ngày càng quan trọng trong việc nâng cao hiệu quả công tác phòng cháy chữa cháy, bảo vệ tính mạng và tài sản của cộng đồng.

Tài liệu tham khảo

Tiếng Việt:

[1] IoTZone.vn. (n.d.). *Hướng dẫn DHT11, DHT22 ESP32 trên Arduino IDE*. Truy cập từ https://www.iotzone.vn/esp32/huong-dan-dht11-dht22-esp32-tren-arduino-ide/

[2] Wokwi. (n.d.). *Project: Fire Alarm System with ESP32, MQ2, and DHT22*. Truy cập từ https://wokwi.com/projects/404279161915648001