TRƯỜNG ĐẠI HỌC KHOA HỌC

Bài tiểu luận

Môn: Phát triển ứng dụng IoT

ĐỀ TÀI:

Tạo hệ thống báo cháy với ESP32

Giảng viên hướng dẫn: **Võ Việt Dũng**

Sinh viên thực hiện: **Đinh Xuân Thái**

Khóa: **CNTT K45 – HỆ CHÍNH QUY**

HUẾ, NĂM 2025

Lời nói đầu

Trong bối cảnh hiện nay, với sự phát triển mạnh mẽ của công nghệ đã làm bước đêm để các thiết bị điện tử được len lỏi vào cuộc sống của từng gia đình. Thiết bị điện tử mang lại cho đời sống sinh hoạt của người dân những lợi ích không hề nhỏ chúng giúp ta tiện lợi hơn trong việc nấu ăn, tiết kiệm thời gian làm việc nhà, bảo quản thực phẩm, ... Chúng tiện lợi trong sinh hoạt hàng ngày cũng như rất dễ tiếp cận. Tuy nhiên an toàn cháy nổ của những vật dụng gia đình lại là vấn đề đáng được quan tâm hơn tất cả. Sự thiếu hụt các hệ thống cảnh báo kịp thời và hiệu quả có thể dẫn đến những hậu quả nghiêm trọng, gây thiệt hại lớn về con người lẫn tài sản. Vì vậy, việc nghiên cứu và phát triển các hệ thống cảnh báo cháy nổ thông minh, dễ dàng triển khai và ứng dụng thực tiễn là một yêu cầu cấp thiết hàng đầu được Đảng và Nhà Nước ta vô cùng quan tâm.

Dự án "Tạo hệ thống báo cháy với ESP32" được phát triển nhằm đáp ứng mục tiêu cung cấp giải pháp đơn giản, hiệu quả và kinh tế cho việc giám sát, cảnh báo cháy nổ trong những gia đình, cơ quan, công cộng. Với sự tích hợp của công nghệ hiện đại và vi điều khiển ESP32, hệ thống không chỉ có khả năng phát hiện nguy cơ cháy nổ, mà còn có thể gửi tin nhắn cảnh báo trực tiếp tới người sử dụng qua các thiết bị điện thoại thông minh.

Dự án này không những góp phần nâng cao ý thức của người dân về việc phòng cháy và chữa cháy, mà nó còn thể hiện vai trò của công nghệ trong việc bảo vệ an toàn cho con người và tài sản. Hy vọng, với sự đóng góp từ dự án này, chúng ta có thể tiến gần hơn tới một xã hội an toàn và bền vững hơn.

Danh mục thuật ngữ và các từ viết tắt

Mục lục

Chương 1: Tổng quan về công nghệ và linh kiện sử dụng cho dự án

1. Ngôn ngữ C++

1.1. Giới thiệu

Ngôn ngữ lập trình C++ được sử dụng rộng tãi trong lập trình IoT nhờ hiệu suất cao, khả năng quản lý tài nguyên hiệu quả và tính linh hoạt. Với C++, lập trình viên có thể viết mã điều khiển phần cứng trực tiếp, tối ưu hóa bộ nhớ và tốc độ xử lý trên các thiết bị IoT có tài nguyên hạn chế như ESP32, Arduino hay Raspberry Pi. C++ hỗ trợ lập trình hướng đối tượng và lập trình generic, giúp tổ chức mã nguồn rõ ràng và tái sử dụng cao

1.2. Ứng dụng

- Thiết lập chương trình cho vi mạch.
- Điều khiển và thu thập dữ liệu từ cảm biến
- Xử lý dữ liệu thời gian thực
- Kết nối mạng và giao tiếp với cloud
- Tôi ưu hóa tài nguyên trên thiết bị IoT
- Phát triển firmware cho thiết bị IoT
- Hỗ trợ đa nền tảng và tích hợp thư viện

2. PlatformIO (IDE tích hợp trong vscode)

- Có vai trò là môi trường phát triển (IDE) để viết, biên dịch và nạp mã lên ESP32
- Chi tiết:
  - ❖ PlatformIO là một plugin trong vscode, hỗ trợ phát triển ứng dụng IoT với ESP32
  - ❖ File platformio.ini được dùng để cấu hình dự án
  - ❖ PlatformIO giúp quản lý thư viện, biên dịch mã, và nạp firmware lên ESP32.

3. ESP32

Hình 1: Mạch ESP32

- Vai trò: ESP32 là vi điều khiển (microcontroller) chính trong dự án, đóng vai trò trung tâm để điều khiển cảm biến, xử lý dữ liệu, và kết nối mạng.
- Chi tiết:
  - ❖ ESP32 là một vi điều khiển mạnh mẽ với WiFi và Bluetooth tích hợp, phù hợp cho các ứng dụng IoT.
  - ❖ Trong dự án, ESP32 được dùng để:
- ❖ Đọc dữ liệu từ cảm biến MQ-2 (khói) và DHT (nhiệt độ) qua các chân GPIO.
- ❖ Điều khiển LED (bật/tắt khi phát hiện nguy cơ cháy).

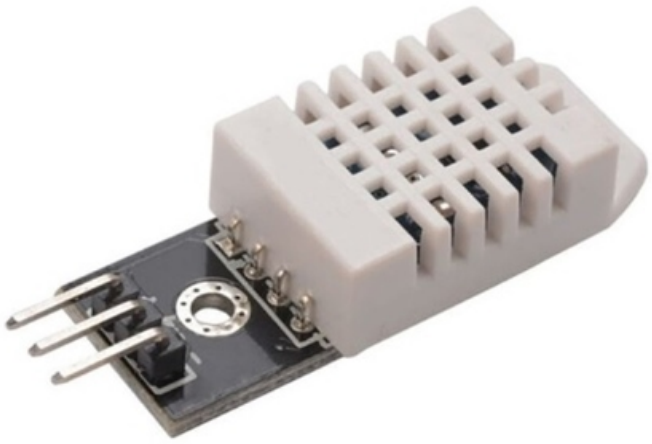
❖ Kết nối WiFi để gửi dữ liệu lên Blynk.

4. Cảm biến khói MQ-2

Hình 2: Cảm biến khói MQ-2

- Vai trò: Phát hiện nồng độ khói trong không khí, giúp nhận biết nguy cơ cháy.
- Chi tiết:
  - ❖ MQ-2 là cảm biến khí, có thể phát hiện các loại khí dễ cháy như LPG, metan, và khói.
  - ❖ Trong dự án, MQ-2 được kết nối với chân analog (GPIO34) của ESP32 để đọc giá trị nồng độ khói

5. Cảm biến nhiệt độ DHT



Hình 3: Cảm biến nhiệt độ DHT

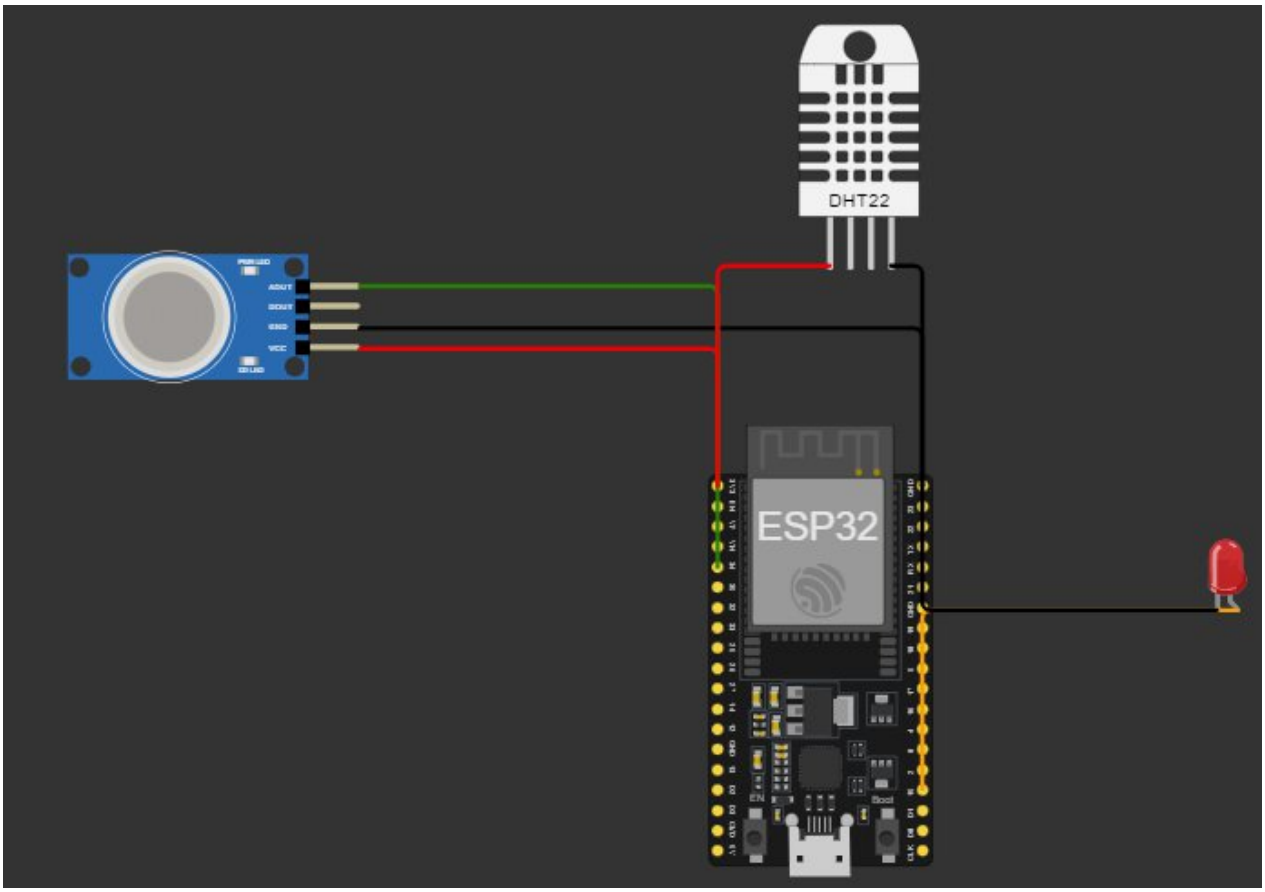
- Vai trò: Đo nhiệt độ để phát hiện nguy cơ cháy (khi nhiệt độ vượt ngưỡng).
- Chi tiết:
  - ❖ DHT là cảm biến nhiệt độ và độ ẩm giá rẻ, phù hợp cho các dự án IoT cơ bản.
  - ❖ Trong dự án, DHT được kết nối với chân digital (GPIO4) của ESP32 để đọc nhiệt độ.

6. Led

- Vai trò: Đóng vai trò cảnh báo trực quan (bật sáng khi phát hiện nguy cơ cháy)
- LED được kết nối với chân GPIO15 của ESP32 và được điều khiển bằng tín hiệu HIGH/LOW.

Chương 2: Nguyên lý hoạt động

1. Sơ đồ mạch hoàn thiện



Hình 4: Sơ đồ hệ thống báo cháy với ESP32

2. Nguyên lý hoạt động

Hệ thống báo cháy sử dụng ESP32 hoạt động dựa trên việc thu thập dữ liệu từ các cảm biến, xử lý dữ liệu để phát hiện nguy cơ cháy, và gửi cảnh báo đến người dùng thông qua ứng dụng Blynk. Dưới đây là các bước chi tiết trong nguyên lý hoạt động:

2.1. Khởi tạo hệ thống

- Mục đích: Chuẩn bị các thành phần phần cứng và phần mềm để hệ thống hoạt động.
- Quy trình:
  - ❖ Khởi động ESP32: Khi cấp nguồn, ESP32 khởi động và chạy chương trình được nạp sẵn (firmware viết bằng C++).
  - ❖ Thiết lập các chân GPIO:

- ❖ Chân GPIO34 (analog) được cấu hình để đọc tín hiệu từ cảm biến khói MQ-2.
- ❖ Chân GPIO4 (digital) được cấu hình để giao tiếp với cảm biến nhiệt độ DHT11.
- ❖ Chân GPIO15 được cấu hình làm đầu ra để điều khiển LED (cảnh báo trực quan).

- Kết nối WiFi:
  - ❖ ESP32 sử dụng module WiFi tích hợp để kết nối với mạng không dây, dựa trên thông tin SSID và mật khẩu được khai báo.
  - ❖ Nếu kết nối thành công, ESP32 sẽ in thông báo "Connected to WiFi" qua Serial Monitor.
- Kết nối với Blynk:
  - ❖ ESP32 sử dụng thư viện BlynkSimpleEsp32 để kết nối với Blynk Cloud, thông qua Auth Token, SSID, và mật khẩu.
  - ❖ Sau khi kết nối thành công, ESP32 có thể gửi/nhận dữ liệu từ ứng dụng Blynk.
- Thiết lập timer: Một timer (BlynkTimer) được thiết lập để gọi hàm sendSensorData mỗi giây, đảm bảo dữ liệu cảm biến được gửi liên tục lên Blynk.

2.2. Thu thập dữ liệu từ cảm biến

- Mục đích: Đọc dữ liệu từ cảm biến khói (MQ-2) và nhiệt độ (DHT11) để phát hiện nguy cơ cháy.
- Quy trình:
  - Cảm biến MQ-2 (khói):
    - ❖ MQ-2 được kết nối với chân GPIO34 (analog) của ESP32.
    - ❖ ESP32 đọc giá trị analog từ MQ-2 (giá trị từ 0 đến 4095, tương ứng với nồng độ khói
    - ❖ Giá trị này được lưu vào biến **smokeValue**.
  - DHT11 được kết nối với chân GPIO4 (digital) của ESP32.
    - ❖ ESP32 sử dụng thư viện DHT.h để đọc giá trị nhiệt độ từ DHT11.
    - ❖ Nếu không đọc được dữ liệu (do lỗi cảm biến), hệ thống sẽ in thông báo lỗi qua Serial Monitor.
    - ❖ ứng với nồng độ khói).
    - ❖ Giá trị nhiệt độ được lưu vào biến **temperature**.

2.3. Xử lý dữ liệu và phát hiện nguy cơ

- Mục đích: Phân tích dữ liệu từ cảm biến để xác định có nguy cơ cháy hay không.
- Quy trình:
  - So sánh với ngưỡng:

Hệ thống kiểm tra giá trị smokeValue và temperature so với các ngưỡng được định nghĩa:

- ❖ Ngưỡng khói: 1000 (giá trị analog).
- ❖ Ngưỡng nhiệt độ: 50°C.

- Nếu smokeValue > 1000 hoặc temperature > 50, hệ thống xác định có nguy cơ cháy.
- Kích hoạt cảnh báo trực quan:
  - ❖ Nếu phát hiện nguy cơ cháy, ESP32 gửi tín hiệu HIGH đến chân GPIO15 để bật LED.
  - ❖ Nếu không có nguy cơ cháy, LED được tắt (tín hiệu LOW).
- In thông tin debug: Hệ thống in thông tin về giá trị khói và nhiệt độ qua Serial Monitor để hỗ trợ debug.

2.4. Gửi dữ liệu và thông báo qua Blynk

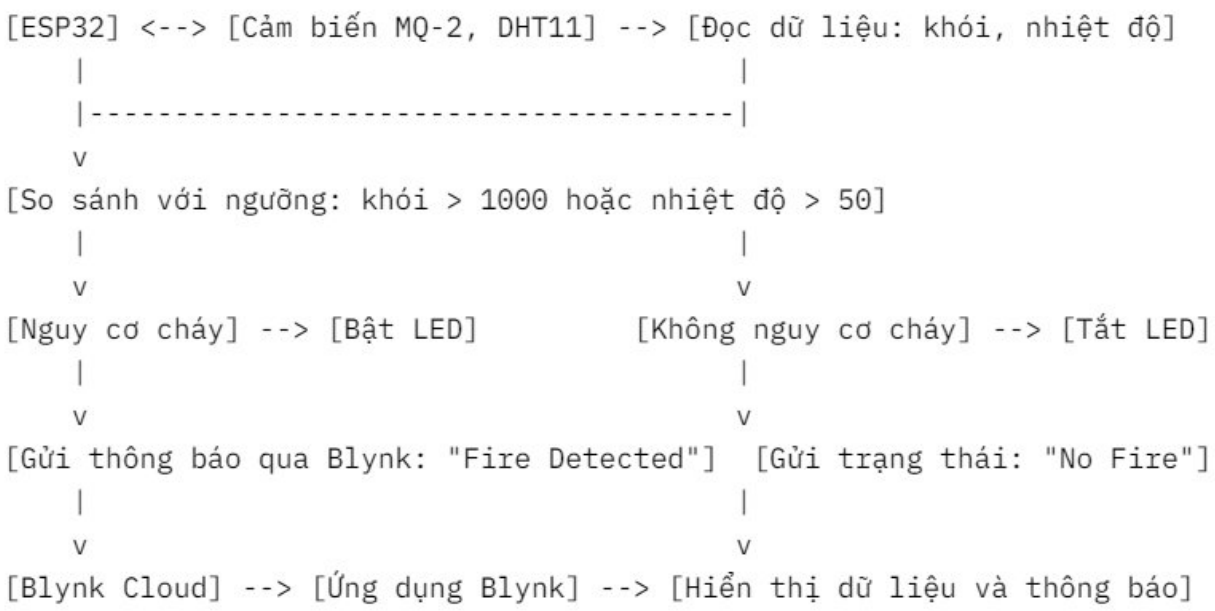
- Mục đích: Gửi dữ liệu cảm biến và thông báo cháy đến người dùng qua ứng dụng Blynk.
- Quy trình:
  - Gửi dữ liệu cảm biến:
    - ❖ Giá trị khói (smokeValue) được gửi lên Virtual Pin V0.
    - ❖ Giá trị nhiệt độ (temperature) được gửi lên Virtual Pin V1.
    - ❖ Các giá trị này được hiển thị trên giao diện Blynk (ứng dụng di động).
  - Gửi thông báo cháy:
    - ❖ Nếu phát hiện nguy cơ cháy, hệ thống gửi trạng thái "Fire Detected" lên Virtual Pin V2.
    - ❖ Nếu không có nguy cơ cháy, trạng thái "No Fire" được gửi lên V2.
    - ❖ Trên ứng dụng Blynk, sự kiện FireAlert được thiết lập để gửi thông báo đến điện thoại khi V2 = "Fire Detected".
  - Giao tiếp với Blynk Cloud:
    - ❖ ESP32 gửi dữ liệu qua giao thức HTTP/HTTPS đến Blynk Cloud.
    - ❖ Blynk Cloud xử lý dữ liệu và chuyển tiếp đến ứng dụng Blynk trên điện thoại người dùng.

2.5. Lặp lại quy trình

- Mục đích: Đảm bảo hệ thống hoạt động liên tục để giám sát thời gian thực.

3. Sơ đồ nguyên lý hoạt động

Dưới đây là sơ đồ minh họa quy trình hoạt động của hệ thống:



Hình 5: sơ đồ nguyên lý hoạt động của hệ thống

Chương 3: Đánh giá và kết luận

1. Đánh giá hiệu năng của mạch

Hệ thống báo cháy sử dụng ESP32 đã được triển khai và thử nghiệm trong môi trường thực tế để đánh giá hiệu năng của các thành phần chính, bao gồm cảm biến nhiệt độ (DHT), cảm biến khói (MQ-2), và khả năng hoạt động tổng thể của mạch. Dưới đây là phân tích chi tiết về hiệu năng của từng thành phần và các vấn đề có thể gặp phải.

1.1. Cảm biến nhiệt độ

- Cảm biến nhiệt độ DHT11 được sử dụng để đo nhiệt độ môi trường, với mục tiêu phát hiện nguy cơ cháy khi nhiệt độ vượt ngưỡng 50°C. Qua quá trình thử nghiệm, cảm biến DHT11 cho thấy một số ưu điểm và hạn chế như sau:
- Ưu điểm:
  - Cảm biến DHT có chi phí thấp và dễ tích hợp với ESP32 thông qua thư viện DHT.h.
  - Trong điều kiện nhiệt độ bình thường (20°C - 40°C), cảm biến cung cấp kết quả đo khá chính xác, với sai số khoảng ±2°C, phù hợp cho các ứng dụng IoT cơ bản.
  - Thời gian phản hồi của cảm biến (khoảng 2 giây) đủ nhanh để giám sát nhiệt độ trong thời gian thực, đáp ứng yêu cầu của hệ thống.
- Hạn chế:
  - Cảm biến DHT có dải đo nhiệt độ hạn chế (0°C - 50°C), không phù hợp trong các tình huống cháy thực tế, nơi nhiệt độ có thể vượt quá 50°C. Khi nhiệt độ vượt ngưỡng, cảm biến trả về giá trị không chính xác hoặc lỗi (NaN).
  - Độ bền của cảm biến không cao, dễ bị ảnh hưởng bởi môi trường có độ ẩm cao hoặc nhiệt độ khắc nghiệt, dẫn đến nguy cơ hỏng hóc trong thời gian dài.
  - Trong một số trường hợp, cảm biến không đọc được dữ liệu do kết nối không ổn định hoặc lỗi giao tiếp với ESP32, gây gián đoạn quá trình giám sát.

1.2. Cảm biến khói

- Cảm biến khói MQ-2 được sử dụng để phát hiện nồng độ khói trong không khí, với ngưỡng phát hiện cháy được đặt ở mức 1000 (giá trị analog). Hiệu năng của cảm biến MQ-2 được đánh giá như sau:
- Ưu điểm:
  - Cảm biến MQ-2 có khả năng phát hiện nhiều loại khí dễ cháy (như LPG, metan, và khói), phù hợp để phát hiện nguy cơ cháy trong môi trường gia đình hoặc văn phòng.

- Độ nhạy của cảm biến khá cao, có thể phát hiện khói ở mức độ thấp (giá trị analog từ 300 trở lên), giúp hệ thống cảnh báo sớm trước khi đám cháy lan rộng.
- Tích hợp dễ dàng với ESP32 qua chân analog (GPIO34), với thời gian phản hồi nhanh (dưới 1 giây), đáp ứng yêu cầu giám sát thời gian thực.

- Hạn chế:
  - Cảm biến MQ-2 dễ bị ảnh hưởng bởi các yếu tố môi trường, như hơi nước, bụi, hoặc các loại khí khác (như khí CO từ bếp gas), dẫn đến nguy cơ báo động giả (false positive). Trong thử nghiệm, hệ thống đôi khi kích hoạt cảnh báo khi có hơi nước từ nồi nước sôi gần cảm biến.
  - Cảm biến cần thời gian khởi động (pre-heating) khoảng 1-2 phút để đạt độ ổn định, gây chậm trễ khi hệ thống vừa khởi động.
  - Giá trị analog từ cảm biến có thể dao động nhẹ (±50) do nhiễu điện từ hoặc chất lượng kết nối, đòi hỏi phải hiệu chỉnh ngưỡng phát hiện cẩn thận để tránh báo động sai.

1.3. Cách khắc phục

- Để cải thiện hiệu năng của hệ thống và khắc phục các hạn chế của cảm biến nhiệt độ và cảm biến khói, một số giải pháp đã được đề xuất và áp dụng như sau:
- Đối với cảm biến nhiệt độ (DHT11):
  - Thay thế cảm biến: Sử dụng cảm biến nhiệt độ chuyên dụng như DS18B20, có dải đo rộng hơn (-55°C đến 125°C) và độ chính xác cao hơn (±0.5°C), phù hợp hơn cho các tình huống cháy thực tế.
  - Cải thiện kết nối: Sử dụng dây dẫn chất lượng cao và thêm điện trở pull-up (4.7kΩ) giữa chân dữ liệu của DHT11 và nguồn 3.3V để tăng độ ổn định của tín hiệu.
  - Xử lý lỗi phần mềm: Thêm cơ chế kiểm tra và thử lại khi đọc dữ liệu thất bại, đồng thời lưu giá trị nhiệt độ gần nhất để tránh gián đoạn giám sát.
- Đối với cảm biến khói (MQ-2):
  - Hiệu chỉnh ngưỡng: Tăng ngưỡng phát hiện khói (ví dụ: từ 1000 lên 1200) để giảm nguy cơ báo động giả do nhiễu từ hơi nước hoặc khí khác. Đồng thời, kết hợp với cảm biến nhiệt độ để xác nhận nguy cơ cháy (chỉ kích hoạt cảnh báo khi cả khói và nhiệt độ đều vượt ngưỡng).
  - Lọc nhiễu: Thêm bộ lọc phần mềm (ví dụ: lấy trung bình giá trị khói trong 5 lần đọc liên tiếp) để giảm dao động do nhiễu.
  - Bảo vệ cảm biến: Lắp đặt cảm biến trong vỏ bảo vệ để giảm ảnh hưởng từ hơi nước và bụi, đồng thời đặt cảm biến ở vị trí thông thoáng, tránh gần nguồn nhiệt hoặc khu vực nấu ăn.

- Đối với hệ thống tổng thể:
  - Dự phòng kết nối: Thêm cơ chế lưu trữ dữ liệu cục bộ (trên EEPROM của ESP32) khi mất kết nối WiFi, và gửi lại dữ liệu khi kết nối được khôi phục.
  - Tăng độ tin cậy: Sử dụng nguồn cấp điện dự phòng (pin hoặc UPS) để đảm bảo hệ thống hoạt động liên tục trong trường hợp mất điện.

2. Kết luận

Dự án hệ thống báo cháy sử dụng ESP32 đã đạt được các mục tiêu đề ra, bao gồm phát hiện nguy cơ cháy, cảnh báo trực quan tại chỗ, và gửi thông báo từ xa qua ứng dụng Blynk. Hệ thống hoạt động ổn định trong môi trường thử nghiệm, với khả năng giám sát thời gian thực và phản hồi nhanh khi phát hiện khói hoặc nhiệt độ vượt ngưỡng. Cảm biến MQ-2 và DHT11, dù có một số hạn chế về độ chính xác và độ bền, vẫn đáp ứng được yêu cầu cơ bản của một hệ thống báo cháy cấp độ nghiên cứu.

Tuy nhiên, hệ thống vẫn tồn tại một số nhược điểm, như nguy cơ báo động giả từ cảm biến khói và hạn chế về dải đo của cảm biến nhiệt độ. Các giải pháp khắc phục đã được đề xuất, bao gồm thay thế cảm biến, hiệu chỉnh ngưỡng, và cải thiện phần mềm, nhằm tăng độ tin cậy và hiệu quả của hệ thống. Dự án này không chỉ chứng minh tính khả thi của việc ứng dụng ESP32 và IoT trong hệ thống báo cháy, mà còn mở ra hướng phát triển cho các ứng dụng giám sát an toàn khác trong tương lai.

3. Ứng dụng

Hệ thống báo cháy sử dụng ESP32 có tiềm năng ứng dụng rộng rãi trong nhiều lĩnh vực, đặc biệt trong các môi trường yêu cầu giám sát và cảnh báo cháy nhanh chóng. Dưới đây là một số ứng dụng cụ thể:

- Giám sát an toàn trong gia đình:
- Ứng dụng trong văn phòng và tòa nhà nhỏ
- Hỗ trợ trong các khu công nghiệp nhỏ
- Giáo dục và nghiên cứu
- Mở rộng cho các hệ thống an toàn khác