ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÁO CÁO BTL

Đề tài Thiết kế bộ cảnh báo cháy sử dụng cảm biến lửa, cảm biến nhiệt độ, độ ẩm, ESP32

Giảng viên hướng dẫn: Nguyễn Phan Hải Phú

LỚP: L01 NHÓM: 21

Danh sách thành viên

Stt	Họ và tên	MSSV
1	Phan Hoàng Lăng Quân	2014277
2	Nguyễn Phước Thành	2114786
3	Phan Tấn Phát	2114382

TpHCM, ngày 29 tháng 11 năm 2024

Tóm tắt đề tài

Hệ thống báo cháy sử dụng vi điều khiển ESP32 được thiết kế nhằm phát hiện sớm các nguy cơ cháy nổ, có chức năng đo nhiệt độ, độ ẩm, đồng thời kiểm tra khói + lửa và hiển thị thông số lên màn hình LCD. Nếu phát hiện khói mức độ thấp thì mở quạt, cao thì bật còi. Nếu phát hiện lửa thì vừa còi vừa quạt. Có thể bổ sung tính năng gửi cảnh báo ứng dụng trên điện thoại qua wifi.

1. GIỚI THIỆU

1.1 Tổng quan

Hệ thống báo cháy là một trong những giải pháp an toàn quan trọng trong các công trình xây dựng, được thiết kế nhằm phát hiện và cảnh báo sớm về nguy cơ cháy nổ. Chức năng chính của hệ thống là giám sát không gian, phát hiện các yếu tố liên quan đến cháy như khói, nhiệt độ, hoặc khí gas, sau đó phát tín hiệu cảnh báo để con người có thể thực hiện các biện pháp ứng phó kịp thời.

Hệ thống báo cháy hiện đại thường bao gồm ba thành phần chính: thiết bị đầu vào (như đầu báo khói, đầu báo nhiệt, đầu báo lửa), trung tâm điều khiển (bảng điều khiển trung tâm), và thiết bị đầu ra (chuông báo cháy, đèn báo động, hoặc hệ thống thông báo). Mỗi thành phần được kết nối và hoạt động phối hợp chặt chẽ để đảm bảo phát hiện cháy nhanh chóng và chính xác.

Câu hỏi đặt ra:

1. Hệ thống báo cháy bao gồm những thành phần nào và mỗi thành phần đóng vai trò gì trong việc phát hiện và cảnh báo cháy?
2. Làm thế nào để đảm bảo sự phối hợp hiệu quả giữa các thiết bị đầu vào, trung tâm điều khiển, và thiết bị đầu ra trong hệ thống báo cháy?
3. Những hạn chế và thách thức nào đang tồn tại trong việc triển khai và vận hành hệ thống báo cháy?
4. Độ ổn định và giá thành cho hệ thống ?

1.2 Phạm vi và phương pháp nghiên cứu

2. Nội dung 1: Tìm hiểu nguyên lý, các module dùng trong mạch.
3. Nội dung 2: Tìm hiểu cách lập trình vi điều khiển ESP32 bằng thư viện ESP-IDF
4. Nội dụng 3: Tìm hiểu cách lập trình bằng hệ điều hành FreeRTOS
5. Nội dung 3: Chạy mô phỏng sản phẩm trên testboard.

Công việc thực hiện:

1. Tìm hiểu nguyên lý, các giải thuật chương trình.
2. Lập trình ESP32 giao tiếp với các cảm biến, hiển thị dữ liệu qua màn hình LCD bằng thư viện ESP-IDF trên hệ điều hành FreeRTOS
3. Lắp ráp và kết nối chạy thử trên testboard

2. LÝ THUYẾT

2.1 ESP-IDF:

ESP-IDF (Espressif IoT Development Framework) là một framework mã nguồn mở được phát triển bởi Espressif Systems, sử dụng để phát triển phần mềm cho các vi điều khiển trong dòng ESP32 (và các sản phẩm liên quan như ESP32-S2, ESP32-C3, v.v.). ESP-IDF cung cấp một môi trường tích hợp để lập trình và triển khai các ứng dụng IoT, hỗ trợ lập trình bằng C/C++.

Framework này được thiết kế để tối ưu hóa hiệu suất của các chip ESP32, cung cấp nhiều API và thư viện để giúp các nhà phát triển tạo ra các ứng dụng IoT phức tạp như Wi-Fi, Bluetooth, kết nối đám mây, điều khiển thiết bị, cảm biến và nhiều ứng dụng khác.

ESP-IDF hoạt động trên nhiều hệ điều hành như Windows, macOS, và Linux, và hỗ trợ nhiều IDE phổ biến như VS Code, Eclipse, hoặc sử dụng dòng lệnh với công cụ CMake và Ninja.

Ưu điểm:

- ESP-IDF cung cấp một tập hợp API chi tiết và hoàn chỉnh, cho phép khai thác tối đa khả năng của vi điều khiển ESP32.
- Tích hợp sẵn các giao thức như Wi-Fi, Bluetooth, MQTT, HTTPS, v.v

- ESP-IDF là mã nguồn mở, nên cộng đồng phát triển lớn và thường xuyên có các bản cập nhật.
- Dễ dàng tuỳ chỉnh và mở rộng các tính năng theo nhu cầu.
- Espressif cung cấp tài liệu chính thức chi tiết, cùng với nhiều ví dụ mẫu giúp nhà phát triển học và triển khai nhanh chóng.
- Hỗ trợ các công cụ phát triển hiện đại như CMake, Ninja, và tích hợp với IDE như VS Code.
- Có thể sử dụng cùng với ESP-IDF Plugin để cải thiện trải nghiệm phát triển.
- Được tối ưu hóa cho phần cứng ESP32, ESP-IDF giúp khai thác tối đa tài nguyên như CPU, RAM, và bộ nhớ flash.
- ESP-IDF sử dụng FreeRTOS làm hệ điều hành thời gian thực, giúp quản lý đa tác vụ hiệu quả, rất hữu ích cho các ứng dụng IoT phức tạp.

Nhược điểm:

- ESP-IDF có thể khó tiếp cận đối với người mới bắt đầu vì yêu cầu hiểu biết sâu về lập trình nhúng và cấu trúc hệ thống.
- Sử dụng dòng lệnh để thiết lập và biên dịch ứng dụng có thể gây khó khăn cho người chưa quen.
- Cài đặt ESP-IDF yêu cầu nhiều bước như cấu hình Python, thiết lập môi trường PATH, và cài đặt các công cụ phát triển khác, dễ gây nhầm lẫn.
- Dù ESP32 mạnh mẽ, nhưng tài nguyên như RAM và flash vẫn có giới hạn, đòi hỏi nhà phát triển tối ưu hóa mã nguồn.
- Debug trên ESP-IDF, đặc biệt là với các tính năng phức tạp như Wi-Fi hoặc Bluetooth, có thể mất nhiều thời gian để xác định lỗi.

- Vì ESP-IDF tích hợp FreeRTOS, các nhà phát triển cần có kiến thức cơ bản về hệ điều hành thời gian thực, điều này có thể là một rào cản đối với người mới.

2.2 FreeRTOS:

FreeRTOS là một hệ điều hành thời gian thực (Real-Time Operating System - RTOS) mã nguồn mở dành cho các vi điều khiển và các bộ vi xử lý nhúng. Được phát triển bởi Richard Barry vào năm 2003, FreeRTOS hiện nay được duy trì bởi Amazon Web Services (AWS). Nó được thiết kế để hoạt động hiệu quả trên các hệ thống nhúng có tài nguyên hạn chế, như vi điều khiển ARM Cortex-M, AVR, ESP32, v.v.

FreeRTOS cung cấp môi trường quản lý đa nhiệm, trong đó các tác vụ (tasks) được lập lịch và thực hiện dựa trên độ ưu tiên hoặc thời gian. Đây là một nền tảng phổ biến trong các ứng dụng IoT, công nghiệp, và điều khiển nhúng nhờ tính đơn giản, ổn định, và khả năng mở rộng.

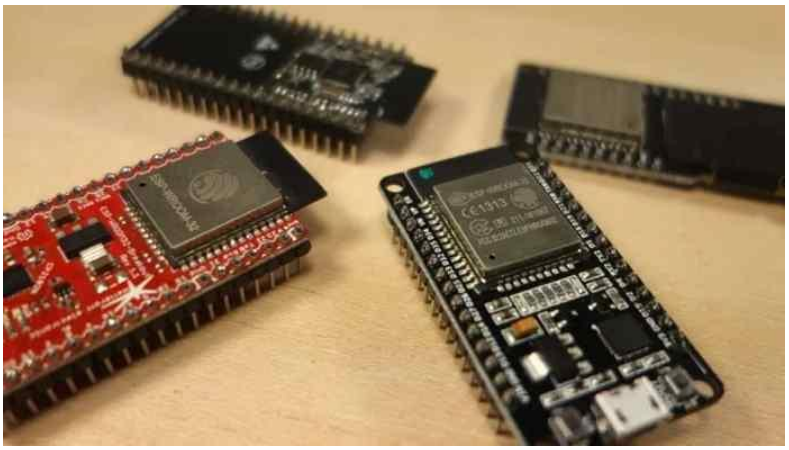
2.3 Phần mềm biên dịch và nạp code cho ESP 32 – Espressif IDE:

Espressif IDE là một môi trường phát triển tích hợp (Integrated Development Environment - IDE) chính thức do Espressif Systems phát triển. IDE này được thiết kế dành riêng cho việc lập trình và phát triển các ứng dụng nhúng trên các vi điều khiển của Espressif như ESP32, ESP32-S2, ESP32-C3, và các dòng sản phẩm khác.

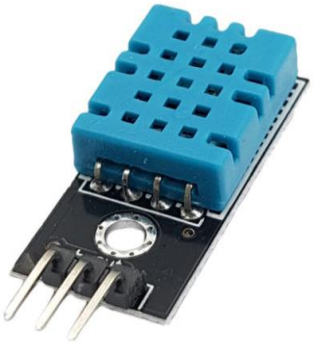
Espressif IDE dựa trên Eclipse IDE, một nền tảng mã nguồn mở phổ biến, nhưng được tùy chỉnh để tích hợp sâu với ESP-IDF (Espressif IoT Development Framework), giúp lập trình viên dễ dàng xây dựng, biên dịch, và nạp chương trình cho các chip ESP32.

2.4 Các loại module, cảm biến và vi điều khiển trong hệ thống:

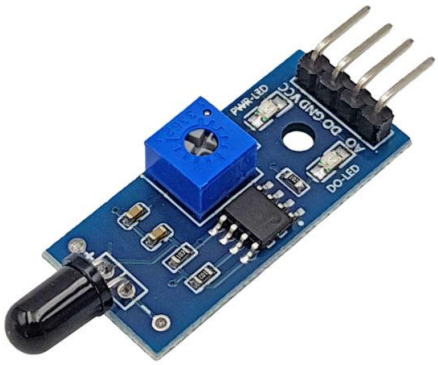
ESP32:



Cảm biến độ ẩm, nhiệt độ DHT11:

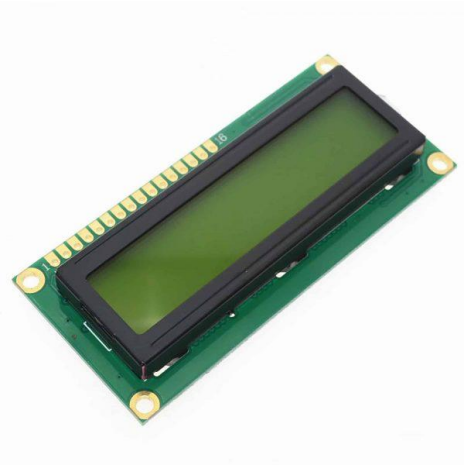


Cảm biến lửa:



Cảm biến khói MQ2:

Màn hình LCD 1602:



Còi buzzer:

3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

3.1 Yêu cầu thiết kế (Đặc tả hệ thống):

3.1.1 Tên và mục đích hệ thống

Tên: Hệ thống báo cháy

Mục đích hệ thống: Phát hiện dấu hiệu cháy và ngăn nguy cơ cháy.

3.1.2 Ngõ vào và ngõ ra của hệ thống:

- Ngõ vào:

  Cảm biến khói, cảm biến nhiệt độ độ ẩm, cảm biến cháy

- Ngõ ra:

  Màn hình lcd, quạt hút khói

3.2 Phân tích thiết kế

3.2.1 Đánh giá và lựa chọn linh kiện

a) Tại sao sử dụng ESP32:

ESP32 thông dụng, dễ lập trình, số chân kết nối nhiều hơn so với các vi điều khiển khác như ESP8266. ESP32 có hai lõi, phù hợp cho chạy nhiều tác vụ, phù hợp cho việc năng cấp sau này.

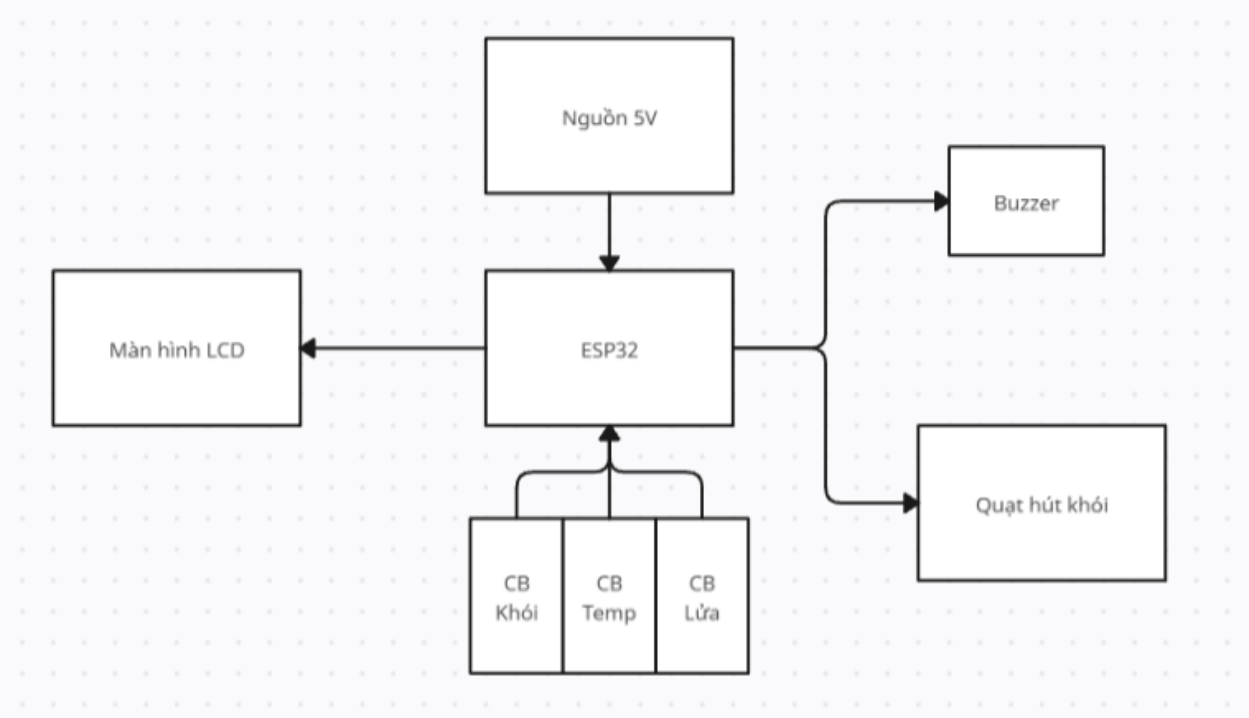
Có khả năng kết nối Wifi & Bluetooh, ưu việt hơn các loại chip hiện hành, phù hợp với mức chi phí.

Tốc độ xử lý: Với chu kỳ hoạt động 160/240MHz thì ESP32 xử lý rất nhanh.

b) Đối với các linh kiện khác:

Có giá thành rẻ, thông dụng và chạy ổn định. Ngoài ra, các linh kiện trên dễ dàng tìm kiếm trên các trang thương mại điện tử.

3.2.2 Sơ đồ khối hệ thống :



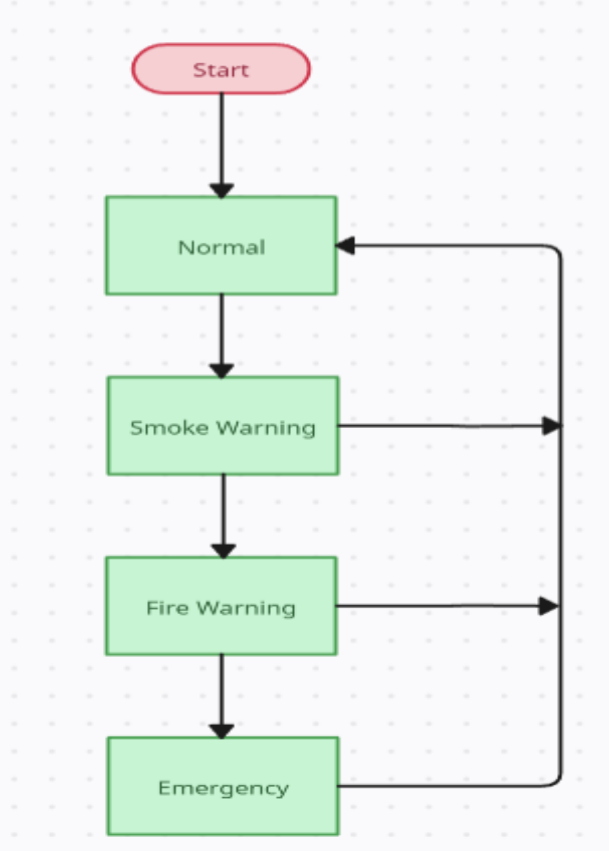
1. Nguồn cung cấp:
    - Cung cấp điện áp 5V cho toàn bộ hệ thống.
2. ESP32 (Bộ xử lý trung tâm):
    - Thu thập dữ liệu từ các cảm biến: cảm biến khói, cảm biến nhiệt độ/độ ẩm (DHT11), cảm biến cháy.
    - Xử lý dữ liệu và thực hiện các quyết định:
        - Kích hoạt quạt hút khói.
        - Hiển thị thông tin trên màn hình LCD.
3. Ngõ vào:
    - Cảm biến khói: Phát hiện nồng độ khói trong không khí, gửi tín hiệu về ESP32.
    - Cảm biến nhiệt độ và độ ẩm (DHT11): Cung cấp dữ liệu về nhiệt độ và độ ẩm để phân tích điều kiện nguy cơ cháy.

- Cảm biến lửa: Phát hiện ánh sáng từ lửa hoặc ngọn lửa để kích hoạt cảnh báo tức thì.

4. Ngõ ra:

- Màn hình LCD: Hiển thị trạng thái hệ thống, thông tin nhiệt độ, độ ẩm, và cảnh báo nguy cơ.
- Quạt hút khói: Tự động bật khi phát hiện khói, giúp giảm nồng độ khói trong không gian.
- Buzzer: Phát tín hiệu âm thanh để cảnh báo khi phát hiện nguy cơ cháy hoặc khói.

3.2.3 Sơ đồ máy trạng thái:



Các trạng thái chính:

Trạng thái khởi tạo (Start):

Hệ thống khởi động và kiểm tra các thiết bị (cảm biến, màn hình LCD, quạt hút khói).

Trạng thái bình thường (Normal):

Không phát hiện nguy cơ cháy hoặc khói. Hiển thị thông tin nhiệt độ và độ ẩm trên LCD.

Trạng thái cảnh báo khói (Smoke Warning):

Phát hiện nồng độ khói vượt ngưỡng. Kích hoạt quạt hút khói và hiển thị thông báo trên màn hình LCD.

Trạng thái cảnh báo cháy (Fire Warning):

Phát hiện cháy (nhiệt độ cao hoặc cảm biến lửa). Kích hoạt Buzzer, hiển thị cảnh báo trên LCD

Trạng thái nguy hiểm (Emergency):

Phát hiện cả khói và lửa, chuyển sang trạng thái khẩn cấp. Kích hoạt toàn bộ hệ thống báo động, quạt hút khói hoạt động ở công suất tối đa.

Mô tả quá trình chuyển đổi trạng thái:

Từ Init → Normal:

Hệ thống hoàn thành khởi tạo và không phát hiện nguy cơ.

Normal → Smoke Warning:

Phát hiện khói từ cảm biến khói (giá trị đo vượt ngưỡng).

Normal → Fire Warning:

Phát hiện cháy từ cảm biến nhiệt độ hoặc cảm biến cháy.

Smoke Warning → Normal:

Mức khói giảm dưới ngưỡng an toàn.

Fire Warning → Normal:

Không còn tín hiệu từ cảm biến cháy, nhiệt độ trở lại bình thường.

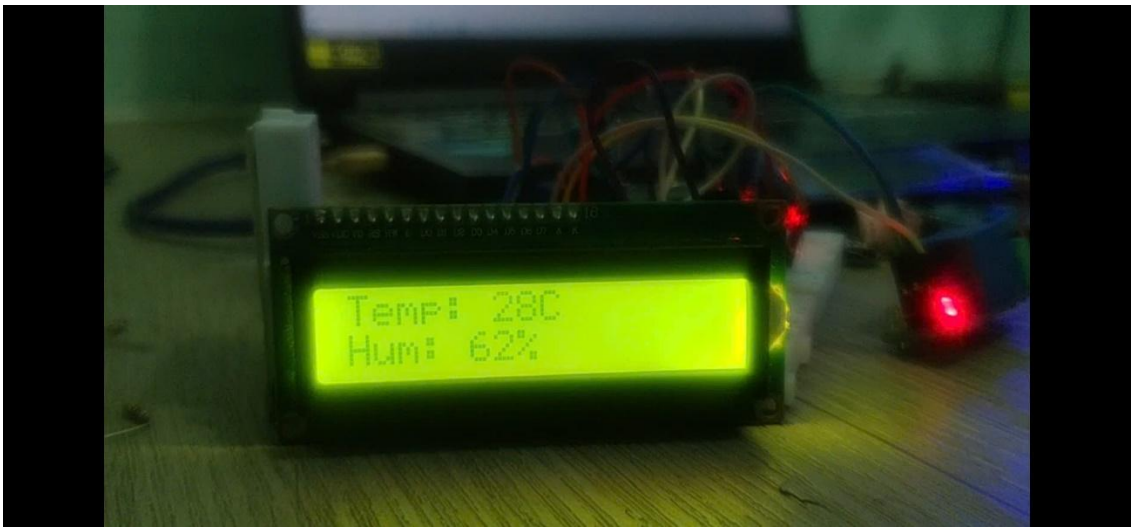
Smoke Warning / Fire Warning → Emergency:

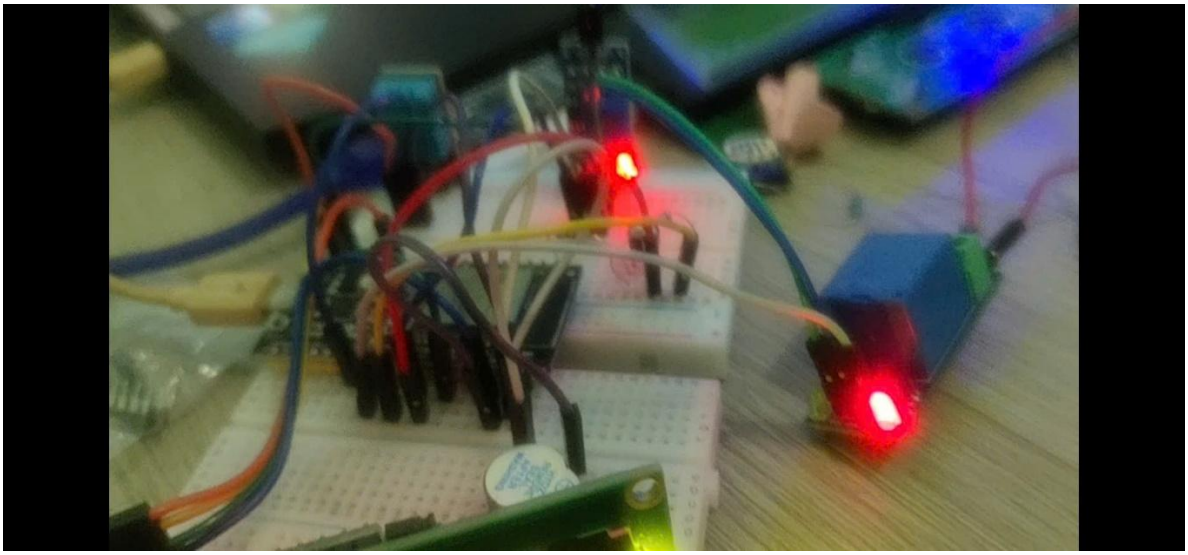
Phát hiện đồng thời cả khói và lửa.

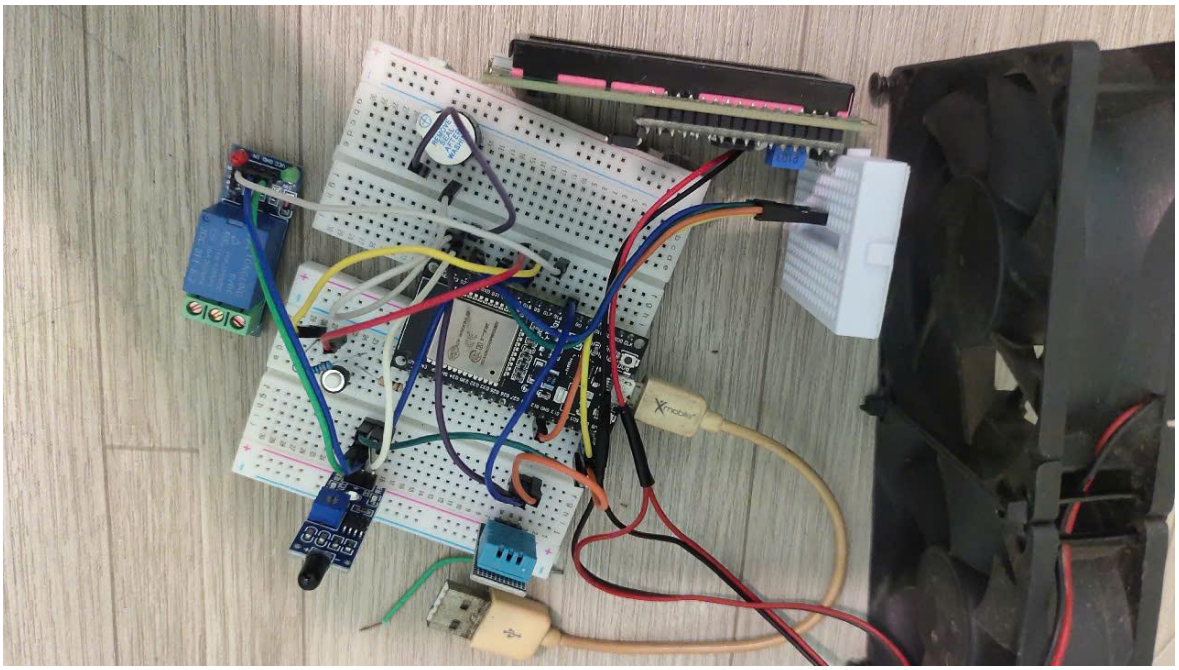
Emergency → Normal:

Hệ thống khôi phục sau khi khói và lửa được kiểm soát.

4. KẾT QUẢ THỰC HIỆN







(*Kết quả tính đến ngày 29/11/2024)

Do chưa có cảm biến MQ2 nên thay thế bằng 1 nút bấm đồng thời code đang tạm sửa thành đọc digital từ “đại diện MQ2” (nút bấm) nên chỉ kiểm tra có khói hoặc không, chưa đọc mức độ.