

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**----🙣🕮🙡----**



**BÁO CÁO BÀI TẬP LỚN**

***Đề tài: Hệ thống phát hiện rò rỉ khí ga hoặc khói và cảnh báo cháy.***

**Giảng viên bộ môn:** Đặng Thị Phúc

Nguyễn Ngọc Lễ

**Sinh viên thực hiện :** Nguyễn Hùng Anh -

Hoa Xuân Sang - 21119261

Lê Ngọc Sơn -

**Lớp học phần :** DHKHMT17A

***TP.HCM, ngày 20. tháng 05 năm 2025.***

**LỜI MỞ ĐẦU**

Trong suốt quá trình thực hiện bài tập lớn này, chúng em đã nhận được nhiều sự quan tâm, giúp đỡ và hỗ trợ quý báu từ các thầy cô, bạn bè và gia đình. Chính nhờ những đóng góp ấy, chúng em đã có thể hoàn thành bài tập một cách trọn vẹn, đúng tiến độ và đạt được những kết quả như mong đợi.

Trước tiên, chúng em xin gửi lời cảm ơn chân thành đến Ban Giám hiệu Trường Đại học Công nghiệp Thành phố Hồ Chí Minh và Ban Chủ nhiệm Khoa Công nghệ Thông tin đã tạo điều kiện thuận lợi, môi trường học tập tích cực và hỗ trợ cơ sở vật chất phục vụ cho việc học tập và nghiên cứu của sinh viên.

Chúng em đặc biệt biết ơn Thầy Nguyễn Ngọc Lễ và Cô Đặng Thị Phúc – giảng viên hướng dẫn đã tận tình chỉ bảo, định hướng và theo sát chúng em trong suốt quá trình triển khai đề tài. Thầy không chỉ truyền đạt kiến thức chuyên môn một cách rõ ràng, dễ hiểu mà còn khơi dậy tinh thần tự học, tinh thần trách nhiệm và sự nghiêm túc trong nghiên cứu. Sự hướng dẫn của Thầy Cô là kim chỉ nam quan trọng giúp chúng em vượt qua nhiều khó khăn trong quá trình thực hiện bài tập.

Chúng em cũng xin trân trọng cảm ơn các thầy cô bộ môn khác đã giảng dạy, truyền đạt nền tảng kiến thức vững chắc trong suốt thời gian học tập, giúp chúng em có đủ hành trang để tiếp cận và giải quyết đề tài một cách khoa học và hiệu quả.

Không thể không nhắc đến sự động viên, ủng hộ từ phía gia đình và bạn bè – những người luôn ở bên cạnh, cổ vũ tinh thần và tiếp thêm động lực cho chúng em, đặc biệt trong những giai đoạn áp lực và căng thẳng nhất.

Cuối cùng, chúng em xin chân thành cảm ơn tất cả những ai đã đồng hành, hỗ trợ và góp phần vào quá trình hoàn thành bài tập lớn này. Dù còn nhiều thiếu sót, chúng em luôn trân trọng từng sự giúp đỡ và ghi nhớ những giá trị mà quá trình thực hiện bài tập mang lại.

**Trân trọng cảm ơn!**

**LỜI MỞ ĐẦU**

Trong bối cảnh tốc độ đô thị hóa ngày càng nhanh, nhu cầu đảm bảo an toàn cháy nổ trong các hộ gia đình, khu chung cư, nhà trọ và nhà máy sản xuất ngày càng trở nên cấp thiết. Những vụ cháy lớn gây thiệt hại nghiêm trọng về người và tài sản thường xuất phát từ nguyên nhân chủ quan như rò rỉ khí gas, chập điện hay thiếu hệ thống cảnh báo sớm. Thực tế này đặt ra yêu cầu cấp bách về việc xây dựng các hệ thống giám sát và cảnh báo cháy nổ kịp thời, chính xác, và dễ triển khai trong môi trường dân dụng.

Sự phát triển của Internet of Things (IoT) – mạng lưới các thiết bị có thể kết nối, truyền dữ liệu và tương tác qua internet – đã mở ra những giải pháp hiệu quả cho các bài toán giám sát môi trường trong thời gian thực. Trên cơ sở đó, chúng em đã thực hiện đề tài “Hệ thống phát hiện rò rỉ khí gas hoặc khói và cảnh báo cháy”, nhằm xây dựng một giải pháp đơn giản, giá thành thấp nhưng mang lại hiệu quả cao trong công tác phòng cháy.

Hệ thống bao gồm các cảm biến khí gas và cảm biến khói, được kết nối với vi điều khiển để xử lý tín hiệu và đưa ra cảnh báo tức thời khi phát hiện bất thường. Khi có dấu hiệu rò rỉ khí hoặc phát hiện khói, hệ thống sẽ tự động kích hoạt còi báo động, đèn cảnh báo và có thể gửi tin nhắn hoặc thông báo đến điện thoại người dùng thông qua các module giao tiếp không dây. Hệ thống còn được thiết kế để dễ dàng lắp đặt, sử dụng và có thể mở rộng trong tương lai nếu cần tích hợp thêm cảm biến hoặc tính năng.

Báo cáo này trình bày toàn bộ quá trình từ khâu nghiên cứu, thiết kế, thi công mạch điện, lập trình điều khiển, đến thử nghiệm và đánh giá hiệu quả hệ thống. Mục tiêu của đề tài là hướng đến một giải pháp ứng dụng thực tiễn, phù hợp với điều kiện sử dụng ở Việt Nam, đặc biệt là tại các khu dân cư có nguy cơ cháy nổ cao nhưng thiếu hệ thống giám sát chuyên nghiệp.

Chúng em xin gửi lời cảm ơn chân thành đến quý thầy cô Trường Đại học Công nghiệp TP. Hồ Chí Minh, đặc biệt là Khoa Công nghệ Thông tin, đã tận tình giảng dạy và hỗ trợ chuyên môn cho nhóm trong suốt quá trình thực hiện đề tài. Chúng em cũng xin cảm ơn gia đình và bạn bè đã luôn ủng hộ, tiếp thêm tinh thần để chúng em hoàn thành tốt nhiệm vụ học tập.

Chúng em hy vọng đề tài này không chỉ là một sản phẩm học thuật, mà còn là một bước khởi đầu để nhóm tiếp tục phát triển các giải pháp IoT phục vụ đời sống, hướng đến xây dựng xã hội an toàn và hiện đại hơn.

**Trân trọng cảm ơn!**

1. **Động cơ nghiên cứu**

Trong những năm gần đây, các vụ cháy nổ do rò rỉ khí gas hoặc chập điện xảy ra ngày càng nhiều, gây ra hậu quả nghiêm trọng về người và tài sản. Phần lớn các sự cố này đều xảy ra đột ngột, không được phát hiện kịp thời do thiếu hệ thống giám sát hoặc cảnh báo sớm. Đặc biệt, tại các hộ gia đình, nhà trọ, khu nhà xưởng nhỏ – nơi chưa được trang bị hệ thống phòng cháy chữa cháy chuyên nghiệp – thì nguy cơ thiệt hại càng cao.

Trong khi đó, các công nghệ IoT, cảm biến và vi điều khiển ngày càng trở nên phổ biến, giá thành rẻ, dễ triển khai. Điều này mở ra cơ hội để xây dựng những hệ thống giám sát đơn giản nhưng hiệu quả, có thể tự động phát hiện các nguy cơ cháy nổ và cảnh báo kịp thời cho người dùng, giúp họ có biện pháp xử lý ngay từ sớm.

Từ thực tế trên, nhóm nhận thấy việc nghiên cứu và triển khai một hệ thống phát hiện rò rỉ khí gas hoặc khói và cảnh báo cháy dựa trên nền tảng IoT là cần thiết, có ý nghĩa thực tiễn cao và phù hợp với kiến thức đã học trong chương trình đào tạo. Hệ thống hướng đến mục tiêu: giá thành thấp, dễ sử dụng, dễ mở rộng, hoạt động ổn định, có thể ứng dụng ngay trong các môi trường dân dụng.

Đây cũng là cơ hội để nhóm vận dụng các kiến thức về kỹ thuật cảm biến, lập trình nhúng, truyền thông không dây và thiết kế hệ thống nhúng, đồng thời rèn luyện kỹ năng làm việc nhóm, phân tích yêu cầu và triển khai giải pháp công nghệ trong thực tế.

1. **Mục tiêu**

Mục tiêu của dự án "Hệ thống phát hiện rò rỉ khí ga hoặc khói và cảnh báo cháy bằng IoT" là xây dựng một hệ thống giám sát thông minh, có khả năng phát hiện kịp thời các nguy cơ cháy nổ và rò rỉ khí gas trong môi trường sinh hoạt hoặc làm việc. Cụ thể như sau:

1. Phát triển hệ thống cảm biến IoT đa chức năng có khả năng phát hiện chính xác các hiện tượng rò rỉ khí gas và khói cháy thông qua các cảm biến chuyên dụng (cảm biến khí gas, cảm biến khói, cảm biến nhiệt độ...).
2. Xây dựng hệ thống xử lý dữ liệu và cảnh báo tự động qua mạng Internet, giúp người dùng nhận được thông báo ngay lập tức trên thiết bị di động hoặc máy tính khi phát hiện dấu hiệu nguy hiểm.
3. Tối ưu hóa hệ thống bằng phần mềm điều khiển và giao diện giám sát trực quan, dễ sử dụng, hỗ trợ quản lý từ xa, cho phép theo dõi trạng thái môi trường và lịch sử cảnh báo theo thời gian thực.
4. Đảm bảo hệ thống hoạt động ổn định, chính xác, nhanh chóng trong môi trường thực tế, đồng thời có khả năng mở rộng và tích hợp thêm các chức năng cảnh báo an toàn khác trong tương lai.
5. Nâng cao tính an toàn, giảm thiểu thiệt hại về người và tài sản do cháy nổ hoặc rò rỉ khí gas gây ra, góp phần thúc đẩy ứng dụng công nghệ IoT trong bảo vệ môi trường sống và làm việc.
6. **Cơ sở lý thuyết**
   1. **ESP32**

ESP32 là một vi điều khiển hiệu năng cao được phát triển bởi Espressif Systems, nổi bật với khả năng tích hợp Wi-Fi và Bluetooth trên cùng một chip. Với kiến trúc lõi kép (dual-core), tốc độ xử lý lên đến 240 MHz và bộ nhớ RAM đủ lớn, ESP32 được xem là nền tảng lý tưởng cho các ứng dụng IoT, tự động hóa và các hệ thống nhúng cần xử lý dữ liệu nhanh chóng và truyền thông không dây ổn định. Khả năng kết nối Wi-Fi cho phép ESP32 truyền dữ liệu lên server hoặc các nền tảng đám mây như Firebase, Blynk, hoặc MQTT broker để giám sát từ xa, trong khi kết nối Bluetooth có thể được sử dụng để giao tiếp cự ly ngắn với smartphone hoặc các thiết bị khác.

Một trong những ưu điểm lớn của ESP32 là sự linh hoạt trong lập trình. Nó có thể được lập trình bằng Arduino IDE, PlatformIO hoặc thậm chí bằng MicroPython, phù hợp với nhiều trình độ phát triển khác nhau. Ngoài ra, vi điều khiển này còn hỗ trợ nhiều giao thức giao tiếp như SPI, I2C, UART, PWM, ADC, và DAC giúp dễ dàng tích hợp với các cảm biến và thiết bị ngoại vi như cảm biến khí gas MQ2, cảm biến nhiệt độ độ ẩm DHT11, LED cảnh báo, máy bơm, v.v. Với mức tiêu thụ điện năng thấp và kích thước nhỏ gọn, ESP32 là lựa chọn phổ biến trong các hệ thống cảnh báo cháy nhờ khả năng thu thập dữ liệu thời gian thực, xử lý tại chỗ và gửi cảnh báo tức thì qua internet hoặc tin nhắn đến người dùng.

* 1. **Python**

Python là một ngôn ngữ lập trình bậc cao, đa mục đích, nổi bật với cú pháp đơn giản, dễ đọc và dễ học, rất phù hợp cho cả người mới bắt đầu lẫn các nhà phát triển chuyên nghiệp. Trong các dự án IoT, Python thường được sử dụng để phát triển các hệ thống giám sát, xử lý và phân tích dữ liệu từ cảm biến, đặc biệt khi kết hợp với các thư viện mạnh mẽ như requests, paho-mqtt, pyserial, hoặc firebase-admin.

Python không chỉ hỗ trợ giao tiếp với phần cứng thông qua cổng serial mà còn cho phép xây dựng các giao diện giám sát từ xa, hệ thống điều khiển qua mạng, hoặc ứng dụng dashboard chạy trên server. Khi sử dụng ESP32 để thu thập dữ liệu từ môi trường (như khí gas, nhiệt độ, khói), Python có thể đóng vai trò trung gian để nhận dữ liệu từ vi điều khiển, lưu trữ và xử lý, đồng thời đưa ra cảnh báo qua email, Telegram bot, hoặc hiển thị thông qua giao diện web. Python cũng thường được sử dụng để tạo ra các kịch bản tự động hóa, chẳng hạn như kích hoạt cảnh báo hoặc điều khiển máy bơm, quạt hút khi cảm biến phát hiện điều kiện bất thường.

Với cộng đồng lớn và tài nguyên phong phú, Python ngày càng khẳng định vai trò quan trọng trong hệ sinh thái IoT, đặc biệt là trong khâu hậu xử lý dữ liệu, điều khiển từ xa, và xây dựng các hệ thống thông minh có khả năng mở rộng linh hoạt.

* 1. **Arduino IDE**

Arduino IDE (Integrated Development Environment) là môi trường phát triển tích hợp được sử dụng phổ biến để lập trình cho các vi điều khiển như Arduino Uno, ESP8266, ESP32 và nhiều dòng board khác. Arduino IDE hỗ trợ lập trình với ngôn ngữ C/C++ đơn giản hóa, kết hợp với hàng loạt thư viện phong phú, giúp người dùng dễ dàng viết mã, biên dịch và nạp chương trình trực tiếp vào bo mạch thông qua cổng USB.

Trong dự án phát hiện và cảnh báo cháy bằng IoT, Arduino IDE đóng vai trò trung tâm trong việc lập trình và điều khiển phần cứng. Cụ thể, nó được sử dụng để lập trình ESP32 thực hiện các chức năng như: đọc dữ liệu từ cảm biến khí gas MQ2, cảm biến nhiệt độ DHT11 hoặc DHT22, kích hoạt LED cảnh báo, điều khiển máy bơm hoặc còi báo động khi phát hiện dấu hiệu cháy hoặc rò rỉ khí gas. Arduino IDE cũng hỗ trợ các thư viện như WiFi.h, HTTPClient.h, FirebaseESP32.h, hoặc DHT.h, giúp việc kết nối mạng, truyền dữ liệu và xử lý cảm biến trở nên dễ dàng và nhanh chóng.

Giao diện trực quan, đơn giản của Arduino IDE cho phép sinh viên và kỹ sư có thể theo dõi log dữ liệu qua Serial Monitor, gỡ lỗi thời gian thực và tinh chỉnh hệ thống một cách hiệu quả. Nhờ tính mở rộng cao, Arduino IDE cũng hỗ trợ việc tích hợp với các dịch vụ đám mây hoặc các nền tảng IoT như Blynk, Firebase, hoặc ThingSpeak, từ đó tạo nên một hệ thống cảnh báo thông minh, tự động và có thể giám sát từ xa qua Internet.

* 1. **Cảm biến đo nhiệt độ, cảm biến khói và phát hiện cháy**

Trong hệ thống IoT phát hiện và cảnh báo cháy, việc sử dụng kết hợp các cảm biến như cảm biến đo nhiệt độ và cảm biến khói là yếu tố then chốt để đảm bảo độ chính xác và độ tin cậy trong việc phát hiện nguy cơ cháy nổ. Cảm biến đo nhiệt độ, điển hình là DHT11 hoặc DHT22, có khả năng ghi nhận sự thay đổi nhiệt độ môi trường xung quanh với độ chính xác tương đối cao, từ đó giúp phát hiện những tình huống bất thường khi nhiệt độ tăng đột ngột – một trong những dấu hiệu quan trọng của hiện tượng cháy.

Song song với đó, cảm biến khói như MQ2 hoặc MQ135 được sử dụng để phát hiện sự hiện diện của khói và khí độc hại (CO, CH4, LPG...), vốn thường phát sinh trong các đám cháy âm ỉ hoặc các tình huống rò rỉ khí. Khi khói hoặc khí vượt quá một mức ngưỡng nhất định, cảm biến sẽ gửi tín hiệu đến vi điều khiển (như ESP32) để xử lý và đưa ra cảnh báo.

Việc phối hợp hai loại cảm biến này trong một hệ thống không chỉ tăng cường khả năng phát hiện cháy sớm mà còn giúp loại bỏ các cảnh báo sai (false alarm) khi chỉ dựa vào một yếu tố. Khi hệ thống phát hiện đồng thời nhiệt độ cao và nồng độ khói vượt ngưỡng, nó có thể tự động kích hoạt các cơ chế phản ứng như: bật còi báo động, nhấp nháy đèn LED, vận hành máy bơm dập lửa hoặc gửi cảnh báo từ xa đến điện thoại người dùng thông qua kết nối WiFi. Giải pháp này mang lại hiệu quả thiết thực, đặc biệt trong các khu vực dân cư, nhà kho, phòng máy chủ, hoặc các khu vực dễ cháy nổ.

* 1. **Giao tiếp không dây và giao thức truyền dữ liệu**

Trong hệ thống IoT phát hiện và cảnh báo cháy, giao tiếp không dây là thành phần then chốt để kết nối giữa các cảm biến, vi điều khiển và thiết bị điều khiển như đèn báo, còi hoặc máy bơm nước. Dự án sử dụng WiFi nội bộ được tích hợp sẵn trong ESP32 để thiết lập kết nối giữa các thiết bị trong cùng một mạng LAN, mà không cần truy cập Internet hoặc máy chủ web từ xa.

Thay vì truyền dữ liệu qua các nền tảng web, hệ thống lựa chọn giao tiếp truyền dữ liệu cục bộ thông qua giao thức MQTT (Message Queuing Telemetry Transport) hoặc Serial/UART (truyền nối tiếp). MQTT hoạt động theo cơ chế publish-subscribe, cho phép ESP32 gửi cảnh báo (ví dụ: nhiệt độ vượt ngưỡng, phát hiện khói) đến các thiết bị khác trong mạng như một nút trung tâm điều khiển còi báo động hoặc kích hoạt máy bơm. Với thiết kế này, dữ liệu được truyền đi nhanh chóng, độ trễ thấp và không phụ thuộc vào kết nối internet, tăng tính ổn định và khả năng phản ứng thời gian thực của hệ thống.

Ngoài MQTT, một số phần của hệ thống cũng sử dụng giao tiếp nối tiếp (UART) để trao đổi dữ liệu giữa các thiết bị phần cứng như ESP32 với các cảm biến nhiệt độ, cảm biến khói, hoặc với mạch điều khiển relay bật/tắt máy bơm và LED cảnh báo. Đây là phương thức truyền tin phổ biến trong các hệ thống nhúng vì độ tin cậy cao và dễ triển khai.

Việc sử dụng giao tiếp không dây nội bộ thay vì các nền tảng cloud giúp hệ thống hoạt động ổn định ngay cả khi mất kết nối Internet, đồng thời vẫn đảm bảo chức năng cảnh báo và kích hoạt thiết bị một cách hiệu quả trong các tình huống khẩn cấp.

* 1. **Thiết bị điều khiển – LED, Máy bơm**

Trong hệ thống IoT phát hiện và cảnh báo cháy, các thiết bị điều khiển như LED và máy bơm đóng vai trò quan trọng trong việc báo hiệu và xử lý sự cố. LED được sử dụng như một tín hiệu trực quan để cảnh báo khi hệ thống phát hiện nhiệt độ cao hoặc khói, giúp người dùng dễ dàng nhận biết tình trạng nguy hiểm ngay lập tức. Máy bơm được tích hợp để tự động kích hoạt khi có dấu hiệu cháy, nhằm mục đích dập tắt hoặc làm giảm nguy cơ lây lan của ngọn lửa. Việc điều khiển các thiết bị này thường được thực hiện thông qua các module relay kết nối với vi điều khiển ESP32, cho phép bật/tắt thiết bị điện một cách chính xác và an toàn. ESP32 gửi tín hiệu điều khiển dựa trên dữ liệu thu thập từ cảm biến nhiệt độ và cảm biến khói, đảm bảo hệ thống phản ứng nhanh chóng và hiệu quả trong các tình huống khẩn cấp, góp phần nâng cao tính an toàn và tự động hóa cho môi trường giám sát.

* 1. **Cảnh báo và giám sát**

Trong hệ thống IoT phát hiện rò rỉ khí gas và khói cùng cảnh báo cháy, chức năng cảnh báo và giám sát đóng vai trò then chốt nhằm đảm bảo an toàn tối đa cho môi trường sử dụng. Khi các cảm biến phát hiện nhiệt độ vượt ngưỡng an toàn hoặc phát hiện khói, tín hiệu sẽ được truyền về vi điều khiển ESP32 để xử lý ngay lập tức. Hệ thống sẽ kích hoạt các thiết bị cảnh báo như còi hú, đèn LED nhấp nháy nhằm thu hút sự chú ý của người xung quanh và báo động nguy cơ cháy hoặc rò rỉ khí gas. Đồng thời, dữ liệu thu thập được từ các cảm biến cũng sẽ được lưu trữ hoặc truyền tải qua giao tiếp không dây nội bộ để giám sát liên tục và theo dõi từ xa, giúp người quản lý có thể kiểm tra trạng thái hệ thống một cách nhanh chóng và chính xác. Việc giám sát liên tục cho phép phát hiện sớm các dấu hiệu bất thường, từ đó giảm thiểu tối đa thiệt hại về người và tài sản. Hệ thống cảnh báo và giám sát không chỉ đảm bảo phản ứng kịp thời mà còn hỗ trợ trong việc bảo trì, nâng cấp và cải tiến hệ thống an toàn một cách hiệu quả.

1. **Mã chương trình**
   1. **Mã báo cháy sử dụng cảm biến lửa và cảm biến ánh sáng**

#define FLAME\_ANALOG 34 // KY-026 A0

#define FLAME\_DIGITAL 13 // KY-026 D0

#define LDR\_PIN 14 // LDR Analog Output

void setup() {

Serial.begin(115200);

pinMode(FLAME\_DIGITAL, INPUT);

}

void loop() {

int flameAnalog = analogRead(FLAME\_ANALOG); // Read flame intensity

int flameDigital = digitalRead(FLAME\_DIGITAL); // Detect flame presence

int lightLevel = analogRead(LDR\_PIN); // Read ambient light

Serial.print("Flame Analog: ");

Serial.print(flameAnalog);

Serial.print(" | Flame Digital: ");

Serial.print(flameDigital);

Serial.print(" | Light Level: ");

Serial.println(lightLevel);

// Fire Detection Logic (Adjust thresholds based on testing)

if (flameDigital == 1 && lightLevel < 500) {

Serial.println("FIRE DETECTED!");

} else {

Serial.println("No fire detected.");

}

delay(500);

}

**Giải thích mã**

Mã được xây dựng trên nền tảng Arduino, sử dụng cảm biến phát hiện lửa KY-026 kết hợp với cảm biến ánh sáng LDR (Light Dependent Resistor). Mục tiêu của mã là xác định sự hiện diện của ngọn lửa trong điều kiện môi trường cụ thể, từ đó đưa ra cảnh báo khi có nguy cơ xảy ra cháy.

**1. Khai báo các chân kết nối**

#define FLAME\_ANALOG 34

#define FLAME\_DIGITAL 13

#define LDR\_PIN 14

Ba chân tín hiệu được khai báo để kết nối với các cảm biến:

* FLAME\_ANALOG: Nhận tín hiệu analog từ cảm biến lửa, cho biết cường độ ngọn lửa. Ngược lại, nếu không đủ điều kiện, hệ thống thông báo "No fire detected.".
* FLAME\_DIGITAL: Nhận tín hiệu digital từ cảm biến lửa, xác định có hay không có ngọn lửa.
* LDR\_PIN: Nhận tín hiệu analog từ cảm biến LDR, phản ánh mức độ sáng của môi trường xung quanh.

**2. Thiết lập ban đầu (setup())**

Serial.begin(115200);

pinMode(FLAME\_DIGITAL, INPUT);

Trong hàm setup(), Serial Monitor được khởi tạo ở tốc độ truyền 115200 bps để phục vụ việc hiển thị dữ liệu giám sát. Đồng thời, chân digital kết nối cảm biến lửa được thiết lập ở chế độ đầu vào (INPUT).

**3. Vòng lặp chính (loop())**

int flameAnalog = analogRead(FLAME\_ANALOG);

int flameDigital = digitalRead(FLAME\_DIGITAL);

int lightLevel = analogRead(LDR\_PIN);

Trong mỗi chu kỳ lặp, chương trình đọc giá trị từ ba cảm biến:

* flameAnalog: Giá trị analog từ cảm biến lửa, phản ánh cường độ của ngọn lửa. Giá trị càng nhỏ thì ngọn lửa càng mạnh.
* flameDigital: Giá trị digital xác định sự hiện diện của ngọn lửa (1 là có lửa, 0 là không có).
* lightLevel: Giá trị đo ánh sáng môi trường từ cảm biến LDR. Giá trị thấp tương ứng với môi trường tối.

**4. Hiển thị dữ liệu và xử lý logic cảnh báo**

Serial.print("Flame Analog: ");

Serial.print(flameAnalog);

Serial.print(" | Flame Digital: ");

Serial.print(flameDigital);

Serial.print(" | Light Level: ");

Serial.println(lightLevel);

Các giá trị đọc được từ cảm biến được in ra màn hình Serial Monitor để người dùng theo dõi và đánh giá tình trạng hoạt động của hệ thống.

if (flameDigital == 1 && lightLevel < 500) {

Serial.println("FIRE DETECTED!");

} else {

Serial.println("No fire detected.");

}

Tiếp theo, chương trình thực hiện kiểm tra điều kiện phát hiện cháy. Cụ thể:

* Nếu cảm biến lửa báo có ngọn lửa (flameDigital == 1) và độ sáng môi trường thấp (lightLevel < 500), hệ thống đưa ra cảnh báo "FIRE DETECTED!".
* Ngược lại, nếu không đủ điều kiện, hệ thống thông báo "No fire detected.".

Ngưỡng giá trị ánh sáng (<500) được sử dụng nhằm giảm thiểu cảnh báo giả trong điều kiện ánh sáng mạnh (như ánh đèn điện hoặc ánh sáng ban ngày).

delay(500);

Cuối vòng lặp, chương trình tạm dừng trong 500 mili-giây trước khi tiếp tục đọc dữ liệu mới. Điều này giúp ổn định tín hiệu và tránh ghi nhận liên tục gây nhiễu.

* 1. **Mã điều khiển còi báo động (Buzzer)**

const int buzzerPin = 12;

void setup(){

pinMode(buzzerPin, OUTPUT);

}

void loop(){

tone(buzzerPin, 1000);

delay(1000);

noTone(buzzerPin);

delay(1000);

}

**Giải thích mã**

Trong hệ thống phát hiện cháy, còi báo động (buzzer) là một thành phần quan trọng để cảnh báo người dùng khi có sự cố. Đoạn mã minh họa cách sử dụng buzzer loại chủ động để tạo ra âm thanh nhịp nhàng.

**1. Khai báo chân kết nối**

const int buzzerPin = 12;

* buzzerPin được khai báo là chân số 12 trên Arduino, nơi kết nối với chân tín hiệu của buzzer.

**2. Thiết lập ban đầu (setup())**

void setup() {

pinMode(buzzerPin, OUTPUT);

}

* Chân buzzerPin được cấu hình là OUTPUT để gửi tín hiệu điều khiển còi.

**3. Vòng lặp chính (loop())**

tone(buzzerPin, 1000);

delay(1000);

noTone(buzzerPin);

delay(1000);

* tone(buzzerPin, 1000): Phát ra âm thanh với tần số 1000 Hz (1 kHz) từ buzzer.
* delay(1000): Duy trì âm thanh trong 1000 mili-giây (1 giây).
* noTone(buzzerPin): Ngắt âm thanh đang phát.
* delay(1000): Tạm dừng 1 giây trước khi phát lại.

Vòng lặp này khiến còi phát ra tiếng "bíp" ngắt quãng với chu kỳ 1 giây kêu, 1 giây tắt – tạo hiệu ứng cảnh báo đơn giản nhưng hiệu quả.

* 1. **Lập trình đo nhiệt độ và độ ẩm sử dụng cảm biến DHT22**

#include <DHT.h>

#define DHTPIN 4 // GPIO4 for data

#define DHTTYPE DHT22 // DHT22 sensor

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(115200);

dht.begin();

}

void loop() {

float temperature = dht.readTemperature();

float humidity = dht.readHumidity();

if (isnan(temperature) || isnan(humidity)) {

Serial.println("Failed to read from DHT sensor!");

} else {

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.print(" °C, Humidity: ");

Serial.print(humidity);

Serial.println(" %");

}

delay(2000); // Wait 2 seconds between readings

}

**Giải thích mã**

Giải thích chương trình đo nhiệt độ và độ ẩm sử dụng cảm biến DHT22

Chương trình này sử dụng cảm biến DHT22 để đo nhiệt độ (°C) và độ ẩm tương đối (%) của môi trường xung quanh. Dữ liệu sau đó được in ra Serial Monitor để theo dõi và giám sát điều kiện môi trường trong thời gian thực.

**1. Khai báo thư viện và cấu hình cảm biến**

#include <DHT.h>

#define DHTPIN 4

#define DHTTYPE DHT22

#include <DHT.h>

* Thư viện dùng để tương tác với các cảm biến dòng DHT (DHT11, DHT22...).
* DHTPIN: GPIO số 4 được sử dụng để nhận tín hiệu dữ liệu từ cảm biến.
* DHTTYPE: Định nghĩa loại cảm biến là DHT22 (có độ chính xác cao hơn DHT11).

DHT dht(DHTPIN, DHTTYPE);

* Khởi tạo một đối tượng cảm biến DHT22 với các tham số chân kết nối và loại cảm biến.

**2. Hàm thiết lập (setup())**

void setup() {

Serial.begin(115200);

dht.begin();

}

* Serial.begin(115200): Khởi tạo kết nối Serial để in dữ liệu ra máy tính.
* dht.begin(): Khởi động cảm biến DHT22 để chuẩn bị đo dữ liệu.

**3. Vòng lặp chính (loop())**

float temperature = dht.readTemperature();

float humidity = dht.readHumidity();

* Đọc nhiệt độ và độ ẩm từ cảm biến:
  + readTemperature(): Trả về nhiệt độ hiện tại theo đơn vị °C.
  + readHumidity(): Trả về độ ẩm tương đối theo phần trăm (% RH).

if (isnan(temperature) || isnan(humidity)) {

Serial.println("Failed to read from DHT sensor!");

}

* Kiểm tra nếu cảm biến không trả về giá trị hợp lệ (trường hợp lỗi hoặc mất kết nối), in thông báo lỗi.

else {

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.print(" °C, Humidity: ");

Serial.print(humidity);

Serial.println(" %");

}

* Nếu dữ liệu hợp lệ, in ra thông tin nhiệt độ và độ ẩm theo định dạng dễ đọc.

delay(2000);

* Chờ 2 giây trước khi thực hiện phép đo tiếp theo. Khoảng cách thời gian này là phù hợp để giảm tải cho cảm biến và giữ dữ liệu ổn định.
  1. **Lập trình phát hiện khí gas sử dụng cảm biến MQ-5**

#define MQ5\_ANALOG 35 // A0 Pin on MQ-5

#define MQ5\_DIGITAL 17 // D0 Pin on MQ-5

void setup() {

Serial.begin(115200);

pinMode(MQ5\_DIGITAL, INPUT);

}

void loop() {

int gasAnalog = analogRead(MQ5\_ANALOG);

// Serial.print("Gas Level: ");

// Serial.print(gasAnalog);

if (gasAnalog > 700) {

Serial.println("HIGH GAS DETECTED!");

} else {

Serial.println("Gas levels are normal.");

}

delay(500);

}

**Giải thích mã**

Mã này sử dụng cảm biến khí gas MQ-5 để phát hiện nồng độ khí gas trong không khí. MQ-5 là loại cảm biến nhạy với các loại khí như LPG, butan, methane, alcohol, hydrogen, v.v. Hệ thống giúp đưa ra cảnh báo khi nồng độ khí vượt mức an toàn, phục vụ các ứng dụng như phòng cháy chữa cháy hoặc giám sát rò rỉ khí đốt.

**1. Khai báo chân kết nối**

#define MQ5\_ANALOG 35

#define MQ5\_DIGITAL 17

* MQ5\_ANALOG: Chân A0 của cảm biến, đầu ra analog phản ánh nồng độ khí.
* MQ5\_DIGITAL: Chân D0 của cảm biến, xuất ra tín hiệu mức cao/thấp dựa trên ngưỡng được điều chỉnh bằng biến trở trên module MQ-5.

**2. Hàm thiết lập (setup())**

Serial.begin(115200);

pinMode(MQ5\_DIGITAL, INPUT);

* Khởi tạo giao tiếp Serial với tốc độ 115200 bps để hiển thị dữ liệu lên màn hình Serial Monitor.
* Chân digital được thiết lập là INPUT để có thể đọc tín hiệu mức logic nếu cần (dù hiện tại chưa dùng đến).

**3. Vòng lặp chính (loop())**

int gasAnalog = analogRead(MQ5\_ANALOG);

* Đọc giá trị analog từ cảm biến. Giá trị càng lớn cho thấy nồng độ khí càng cao. Dải giá trị nằm trong khoảng từ 0 đến 4095 (với ESP32).

if (gasAnalog > 700) {

Serial.println("HIGH GAS DETECTED!");

} else {

Serial.println("Gas levels are normal.");

}

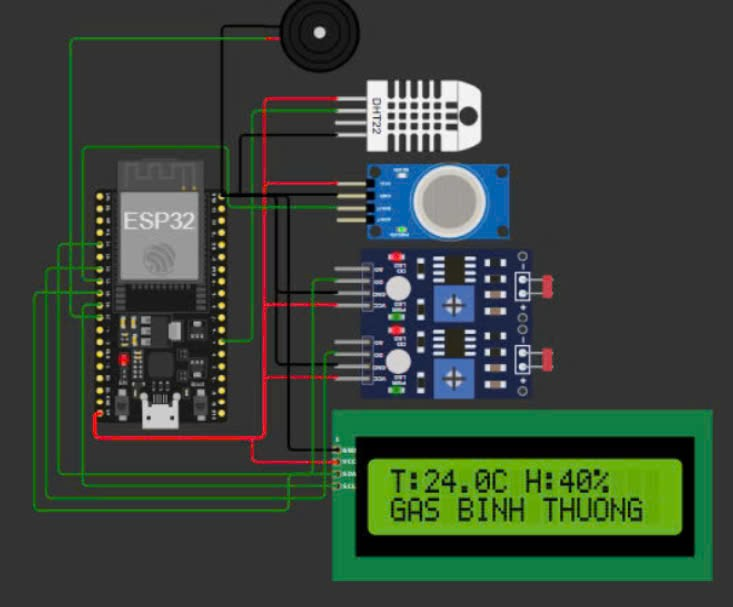
* Nếu giá trị vượt ngưỡng 700, hệ thống nhận diện có nồng độ khí cao và in cảnh báo "HIGH GAS DETECTED!".
* Nếu nhỏ hơn hoặc bằng 700, coi như mức khí trong giới hạn an toàn.

delay(500);

* Dừng 500 mili-giây trước lần đọc tiếp theo để ổn định tín hiệu.

Ghi chú về hiệu chuẩn: Ngưỡng 700 là giá trị tương đối và cần được hiệu chỉnh qua thực nghiệm trong môi trường thực tế để tránh cảnh báo sai hoặc bỏ sót.

1. **Sản phẩm**



**Mô tả về cài đặt hệ thống mô phỏng**

1. Cảm biến DHT22

* + Chân VCC của DHT22 được nối với chân 5V của ESP32 để cấp nguồn.
  + Chân GND của DHT22 được nối với chân GND của ESP32.
  + Chân dữ liệu (SDA) của DHT22 được nối với chân GPIO 4 của ESP32 để truyền dữ liệu nhiệt độ và độ ẩm.

2. Cảm biến khí gas (loại có ngõ ra kỹ thuật số - DOUT)

* + Chân VCC của cảm biến khí gas được nối với chân 5V của ESP32.
  + Chân GND của cảm biến khí gas được nối với chân GND của ESP32.
  + Chân DOUT (ngõ ra kỹ thuật số) của cảm biến khí gas được nối với chân GPIO 33 của ESP32 để phát hiện khí gas.

3. Màn hình LCD 1602 giao tiếp I2C

* + Chân VCC của LCD được nối với chân 5V của ESP32.
  + Chân GND của LCD được nối với GND của ESP32.
  + Chân SDA của LCD được nối với chân GPIO 25 của ESP32.
  + Chân SCL của LCD được nối với chân GPIO 26 của ESP32.

4. Hai cảm biến ánh sáng (LDR, loại có ngõ ra DO)

LDR1:

* + Chân VCC của LDR1 được nối với 5V của ESP32.
  + Chân GND của LDR1 được nối với GND của ESP32.
  + Chân DO của LDR1 được nối với chân GPIO 34 của ESP32 để đọc trạng thái sáng/tối.

LDR2:

* + Chân VCC của LDR2 được nối với 5V của ESP32.
  + Chân GND của LDR2 được nối với GND của ESP32.
  + Chân DO của LDR2 được nối với chân GPIO 32 của ESP32.

5. Loa buzzer (chủ động, không cần PWM)

* + Chân dương (+) của buzzer được nối với chân GPIO 27 của ESP32 để điều khiển âm thanh.
  + Chân âm (-) của buzzer được nối với chân GND của ESP32.

6. Kết nối Serial Monitor (UART)

* + Chân TX của ESP32 được nối với RX của Serial Monitor.
  + Chân RX của ESP32 được nối với TX của Serial Monitor.

1. **Kết luận và hướng phát triển**

**1. Kết luận**

Đề tài đã triển khai thành công một hệ thống giám sát và cảnh báo cháy – rò rỉ khí gas với các chức năng chính bao gồm:

* Phát hiện lửa bằng cảm biến KY-026, kết hợp đọc tín hiệu analog và digital để tăng độ chính xác.
* Đo cường độ ánh sáng môi trường thông qua cảm biến LDR, hỗ trợ lọc nhiễu và đánh giá môi trường bất thường.
* Phát hiện khí gas với cảm biến MQ-5, hoạt động hiệu quả trong việc giám sát các loại khí dễ cháy như LPG, butan, methane.
* Giám sát nhiệt độ và độ ẩm thông qua cảm biến DHT22, góp phần cảnh báo sớm các điều kiện dễ gây cháy.
* Cảnh báo âm thanh thông qua còi buzzer, giúp tăng khả năng phát hiện tại chỗ.
* Giao tiếp Serial để hiển thị và ghi nhận dữ liệu trong thời gian thực.

Kết quả thực nghiệm cho thấy hệ thống hoạt động ổn định, có khả năng mở rộng và dễ dàng tích hợp thêm các tính năng nâng cao.

**2. Bảng đánh giá hiệu quả cảm biến**

| Tên cảm biến | Chức năng | Ngưỡng thử nghiệm | Độ tin cậy | Phản hồi |
| --- | --- | --- | --- | --- |
| KY-026 | Phát hiện lửa | Ánh sáng < 500, D0 = 1 | Tốt | 0.5–1 giây |
| LDR | Đo ánh sáng môi trường | Giá trị < 500 | Tốt | Liên tục, ổn định |
| MQ-5 | Phát hiện khí gas dễ cháy | Analog > 700 | Tốt | 0.5–2 giây |
| DHT22 | Đo nhiệt độ, độ ẩm | Sai số ±0.5°C | Rất tốt | Cập nhật mỗi 2 giây |
| Buzzer | Cảnh báo âm thanh | Khi phát hiện cháy/gas | Tốt | Rất nhanh (âm thanh ngay) |

**3. Hướng phát triển**

Để nâng cao tính ứng dụng và mở rộng hệ thống trong tương lai, có thể triển khai các hướng sau:

* Tích hợp Wi-Fi / GSM module để gửi cảnh báo qua điện thoại (SMS, email, app).
* Xây dựng giao diện người dùng qua Web hoặc ứng dụng mobile để theo dõi thời gian thực.
* Kết nối Cloud (Firebase, Thingspeak...) để lưu trữ dữ liệu phục vụ phân tích hành vi cháy/rò rỉ theo thời gian.
* Tối ưu phần cứng bằng việc thiết kế mạch in PCB, dùng nguồn dự phòng hoặc năng lượng mặt trời để hệ thống hoạt động liên tục.
* Ứng dụng AI để phân tích dữ liệu cảm biến, lọc báo động giả và dự đoán nguy cơ cháy nổ từ điều kiện môi trường.
* Kết hợp camera hoặc mô hình thị giác máy tính (Computer Vision) để xác thực có người/vật bị ảnh hưởng, hỗ trợ ứng cứu thông minh hơn.