

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH KHOA
ĐÀO TẠO CHẤT LƯỢNG CAO
BỘ MÔN KỸ THUẬT MÁY TÍNH- VIỄN THÔNG

----- oOo -----

ĐỒ ÁN MÔN HỌC 2



Hệ thống giám sát từ xa nhà kho, phân xưởng

GVHD: Trương Ngọc Sơn

Sinh viên thực hiện:

Nhữ Đình Thành - 20119283

Võ Trường Hiếu - 20158153

TP. HỒ CHÍ MINH – 8/2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH KHOA ĐÀO
TẠO CHẤT LƯỢNG CAO
BỘ MÔN KỸ THUẬT MÁY TÍNH- VIỄN THÔNG
----- oOo -----
ĐỒ ÁN MÔN HỌC 2

Hệ thống giám sát từ xa nhà kho, phân xưởng

GVHD: Trương Ngọc Sơn

Sinh viên thực hiện:

Nhữ Đình Thành - 20119283

Võ Trường Hiếu - 2015815

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Ký tên

PGS.TS Trương Ngọc Sơn

LỜI CẢM ƠN

Để hoàn thành đồ án, nhóm sinh viên thực hiện đề tài xin chân thành cảm ơn:

Thầy Trương Ngọc Sơn – Giảng viên ngành Công nghệ kỹ thuật Máy tính đã tận tình giúp đỡ chúng em trong lựa chọn đề tài cũng như trong quá trình thực hiện đề tài. Trong quá trình thực hiện đồ án cũng đã xảy ra nhiều khó khăn, thiếu sót nhưng được sự hỗ trợ và góp ý của Thầy nên nhóm đã hoàn thành được đồ án.

Các bạn sinh viên của tập thể lớp 20119CL4A đã có những giúp đỡ thiết thực, cung cấp tài liệu liên quan, cũng như động viên trong quá trình thực hiện đề tài.

Trong quá trình nghiên cứu, mặc dù nhóm em đã rất cố gắng, song vì trình độ và kiến thức bản thân còn hạn chế nên việc tìm hiểu và mô phỏng đồ án không tránh khỏi những sai sót. Mong Thầy cùng các bạn góp ý, chỉ dẫn để đề tài hoàn thiện hơn và có thể ứng dụng trong thực tế.

Một lần nữa nhóm em xin chân thành cảm ơn

Người thực hiện đề tài

Võ Trường Hiếu

Nhữ Đình Thành

Mục lục

Danh mục bảng	8
Bảng từ viết tắt.....	10
Chương 1: Tổng quan đề tài.....	12
1.1. Đặt vấn đề.....	12
1.2. Mục tiêu và công cụ nghiên cứu	12
1.2.1. Mục tiêu nghiên cứu	12
1.2.2. Công cụ nghiên cứu	13
1.3. Giới hạn đề tài.....	13
1.4. Phương pháp nghiên cứu	14
1.5. Đối tượng và phạm vi nghiên cứu	14
1.6. Tóm tắt	15
Chương 2: Cơ sở lý thuyết.....	16
2.1. Tìm hiểu về ESP32.....	16
2.1.1. Giới thiệu	16
2.1.2. Lập trình cho ESP32	16
2.1.3. Tính năng của ESP32	17
2.1.4. Sơ đồ chân	18
2.2. Cảm biến khí gas MQ-2	20
2.2.1. Chức năng	20
2.2.2. Cấu tạo của MQ-2	20
2.3. Module led thu hồng ngoại/phát hiện lửa	22
2.3.1. Chức năng	22
2.3.2. Cấu tạo của module led thu hồng ngoại/phát hiện lửa	22
2.4. LCD1602.....	23
2.4.1. Chức năng	23

2.4.2. Cấu tạo của LCD1602	23
2.5. Giao thức I2C	25
2.5.1. Khái niệm	25
2.5.2. Cấu tạo.....	25
2.5.3. Khung truyền I2C	26
2.5.4. Quá trình truyền nhận dữ liệu	27
2.5.5. Module I2C	28
2.6. Relay	28
2.6.1. Cấu tạo của Relay	28
2.6.2. Nguyên lý hoạt động.....	28
2.6.3. Sơ đồ nguyên lý của relay	29
2.7. DHT11	29
2.7.1. Chức năng	29
2.7.2. Cấu tạo DHT11	30
2.8. RTOS	30
2.8.1. Khái niệm RTOS	30
2.8.2. Semaphore.....	32
2.9. Ngắt.....	34
2.9.1. Khái niệm	34
2.9.2. Ngắt ngoài trong ESP32.....	35
Chương 3: Thiết kế hệ thống.....	37
3.1. Mô hình hệ thống.....	37
3.2. Thiết kế phần cứng.....	38
3.2.1. Sơ đồ khối.	38
3.2.2. Thiết kế từng khối.....	38

3.2.3. Sơ đồ nguyên lý hệ thống	41
3.3. Thiết kế phần mềm.....	41
3.3.1. Phần mềm lập trình Arduino IDE	41
3.3.2. Tương tác qua Blynk.....	42
3.3.3. Lưu đồ giải thuật hệ thống	43
Chương 4: Kết quả đánh giá	44
4.1. Kết quả mô hình thi công.....	44
4.2. Hoạt động của hệ thống	45
Chương 5: Kết luận và hướng phát triển	49
5.1. Kết luận	49
5.2. Hướng phát triển	49
Tài liệu tham khảo.....	50
Phụ lục	51

Danh mục bảng

Bảng 1: Bảng chân và chức năng của LCD.....	24
Bảng 2: Phân loại các ngắt	35
Bảng 3. Bảng tính toán các thông số điện áp	40

Danh mục hình

Hình 2. 1. Sơ đồ chân ESP32 DEVKIT V1	18
Hình 2. 2. Cảm biến khí gas MQ-2	21
Hình 2. 3. Cảm biến lửa.....	22
Hình 2. 4. Màn hình LCD 16x2.....	23
Hình 2. 5. Cấu tạo mô hình I2C.....	26
Hình 2. 6. Khung truyền I2C	26
Hình 2. 7. Khung tín hiệu SDA và SCL	27
Hình 2. 8. Module I2C.....	28
Hình 2. 9. . Nguyên lý hoạt động của Relay	29
Hình 2. 10. Sơ đồ nguyên lý của Relay	29
Hình 2. 11. Cảm biến nhiệt độ, độ ẩm DHT11	30
Hình 2. 12. Chức năng lập lịch của RTOS	31
Hình 2. 13. Ưu điểm của RTOS	32
Hình 2. 14. Chia sẻ tài nguyên trong Semaphore	33
Hình 2. 15. Ngắt là một sự kiện được ưu tiên xử lý trước	34
Hình 3. 1. Sơ đồ khối của hệ thống	38
Hình 3. 2. Sơ đồ nguyên lý của hệ thống	41
Hình 3. 3. Giao diện tương tác qua Blynk trên điện thoại.....	42
Hình 3. 4. Lưu đồ giải thuật hệ thống.....	43
Hình 4. 1. Thiết kế PCB	44
Hình 4. 2. Phân cứng hệ thống	44
Hình 4. 3. Kết nối tới wifi hệ thống.....	45
Hình 4. 4. Thực hiện nhập mật khẩu	45
Hình 4. 5. LCD hiển thị khi chưa kết nối	46

Hình 4. 6. LCD hiển thị ngày và giờ	46
Hình 4. 7. LCD hiển thị giá trị các cảm biến.....	46
Hình 4. 8. Cảnh báo khi có khí gas rò rỉ.....	47
Hình 4. 9. Cảnh báo khi có hỏa hoạn	47
Hình 4. 10. Điều khiển máy bơm chữa cháy	48

Bảng từ viết tắt

STT	Các từ viết tắt	Nghĩa đầy đủ
1	AES	Advanced Encryption Standard
2	AWS IoT	Amazon Web Services Internet of Things
3	ADC	Analog Digital Converter
4	BR/EDR	Basic Rate/Enhanced Data Rate
5	BLE	Bluetooth Low Energy
6	CLK	Clock
7	DAC	Digital Analog Converter
8	DHT	Digital Humidity and Temperature
9	ESP32	Espressif System Platform 32
10	GPIO	General Purpose Input/Output
11	IEEE	Institute of Electrical and Electronics Engineers
12	IDE	Integrated Development Environment
13	I2S	Integrated Interchip Sound
14	I2C	Inter-Integrated Circuit
15	LED	Light Emitting Diode
16	LCD	Liquid Crystal Display
17	LPG	Liquified Petroleum Gas
18	OTP	One-Time Programmable
19	PLL	Phase-Locked Loop
20	PWM	Pulse Width Modulation
21	RNG	Random Number Generator
22	REPL	Read-Eval-Print Loop
23	ROM	Read-Only Memory
24	RTC	Real-Time Clock
25	RTC	Real-Time Clock
26	RTOS	Real-Time Operating System
27	RSA	Rivest-Shamir-Adleman

28	SHA-2	Secure Hash Algorithm 2
29	SPI	Serial Peripheral Interface
30	SDK	Software Development Kit
31	SRAM	Static Random-Access Memory
32	TSMC	Taiwan Semiconductor Manufacturing Company
33	UART	Universal Asynchronous Receiver-Transmitte
34	VCC	Voltage Colector To Colector
35	WFA	Wi-Fi Alliance
36	WPA	Wi-Fi Protected Access
37	WPA2	Wi-Fi Protected Access 2
38	WAPI	WLAN Authentication and Privacy Infrastructure

Chương 1: Tổng quan đề tài

1.1. Đặt vấn đề

Việt Nam là một quốc gia với nền kinh tế đang phát triển. Các nhà máy xí nghiệp của các công ty trong nước và ngoài nước mọc lên trải dài khắp đất nước từ Bắc vào Nam. Ước tính có khoảng gần 30000 doanh nghiệp sản xuất lớn nhỏ. Mỗi nhà máy, xí nghiệp đều có kho lưu trữ hàng hoá, trang thiết bị cũng như máy móc. Tuy nhiên vấn đề về an toàn phòng cháy chữa cháy vẫn chưa được quán triệt. Các vụ hoả hoạn, chập điện vẫn xảy ra thường xuyên gây thiệt hại về người và tài sản. Những sự cố thường xảy ra trong ban đêm khi không có ai túc trực để có thể giám sát, đề phòng. Việt Nam nằm trong vùng khí hậu nhiệt đới và á nhiệt đới có độ ẩm cao cũng là một trong những lý do khiến cho máy móc bị hơi ẩm làm ảnh hưởng đến quá trình vận hành khiến đây cũng là một trong những lý do gây ra chập điện dẫn đến cháy nổ.

Với sự tiến bộ của khoa học kỹ thuật đặc biệt là đối với nền công nghiệp 4.0 thì nhu cầu nâng cao cuộc sống của con người, giảm bớt sức lao động và tiết kiệm được thời gian ngày càng được quan tâm, để đáp ứng được những nhu cầu đó thì nhiều các lĩnh vực cũng phải phát triển theo đặc biệt là các ngành thuộc lĩnh vực khoa học kỹ thuật. Một trong những thành tựu của khoa học kỹ thuật là sự giám sát, điều khiển từ xa.

Sự phát triển nhanh chóng của các trang thiết bị điện, điện tử và kỹ thuật điện đặc biệt là Internet đem lại nhiều ưu điểm cho các sản phẩm phục vụ cho nhu cầu sinh hoạt hàng ngày của con người với tính nhỏ gọn, tính linh hoạt và khả năng lập trình. Từ đó kéo theo sự phát triển của IoT (Internet of Things) ngày càng phát triển cùng với sự đa dụng và thông minh các thiết bị có thể kết nối, tương tác với nhau qua hệ thống Internet. Với mong muốn chọn một đề tài thực tiễn, nhóm chúng em đã quyết định chọn đề tài "Hệ thống giám sát từ xa nhà kho, phân xưởng" nhằm góp phần đóng góp vào việc giảm thiểu sự cố rủi ro gây thiệt hại cho người và tài sản.

1.2. Mục tiêu và công cụ nghiên cứu

1.2.1. Mục tiêu nghiên cứu

Nhóm thực hiện đề tài “Hệ thống giám sát từ xa nhà kho, phân xưởng” nhằm thực hiện các chức năng:

- Thu thập được dữ liệu về các loại khí bị rò rỉ có thể gây ra cháy nổ của môi trường xung quanh.

- Cảnh báo khi có lửa.
- Thu thập dữ liệu nhiệt độ, độ ẩm từ môi trường xung quanh.
- Màn hình LCD hiển thị các thông tin thời gian, nồng độ khí gas, nhiệt độ, độ ẩm
- Giám sát điều khiển các thiết bị khác qua điện thoại.
- Sản phẩm có thể hoạt động một cách ổn định

1.2.2. Công cụ nghiên cứu

- ESP32: Khối điều khiển
- MQ-2: Cảm biến khí gas
- Flame Sensor: Cảm biến lửa
- LCD16x2: Hiển thị nồng độ khí gas và cảnh báo
- DHT11: Cảm biến nhiệt độ, độ ẩm
- I2C: Mạch chuyển đổi giao tiếp cho LCD
- Relay 5VDC

1.3. Giới hạn đề tài

Phân tích các phương pháp phát hiện cháy: Nghiên cứu về các phương pháp phát hiện cháy hiện có như phát hiện khói, phát hiện lửa. Nghiên cứu về ưu điểm và hạn chế của từng phương pháp để phát triển một hệ thống cảnh báo cháy hiệu quả.

Phân tích các khí gây cháy và nguyên nhân rò rỉ: Xem xét các loại khí gây cháy phổ biến như khí tự nhiên, khí hóa lỏng (LPG) và khí hydro. Nghiên cứu các nguyên nhân gây rò rỉ khí như hỏng hóc trong hệ thống cấp khí, quá trình sử dụng không an toàn và sự cố kỹ thuật.

Phân tích nhiệt độ, độ ẩm giới hạn mà các máy móc không bị ảnh hưởng hay hỏng hóc lưu kho không bị hư hỏng.

Thiết kế hệ thống cảnh báo cháy: Phát triển một hệ thống cảnh báo cháy và rò rỉ khí dựa trên kết quả nghiên cứu và phân tích. Nghiên cứu về các yếu tố cần thiết như cảm biến cháy, cảm biến nhiệt độ độ ẩm và cảm biến khí, giao diện người dùng và tích hợp với hệ thống phòng cháy chữa cháy.

Đánh giá hiệu suất hệ thống: Tiến hành thử nghiệm và đánh giá hiệu suất của hệ thống giám sát nhiệt độ, độ ẩm, rò rỉ khí, phát hiện cháy trong các điều kiện thực tế.

Đánh giá độ nhạy và độ chính xác của hệ thống trong việc phát hiện những bất thường, cũng như thời gian phản ứng và khả năng thông báo kịp thời.

1.4. Phương pháp nghiên cứu

Nghiên cứu lý thuyết: Tiến hành nghiên cứu và thu thập thông tin về các phương pháp phát hiện cháy và rò rỉ khí hiện có, các loại khí gây cháy và nguyên nhân rò rỉ khí. Điều này bao gồm tìm hiểu về nguyên lý hoạt động của các cảm biến cháy, nhiệt độ, độ ẩm và cảm biến khí, các công nghệ phát hiện và các tiêu chuẩn liên quan.

Thiết kế hệ thống: Dựa trên kiến thức thu thập được, thiết kế hệ thống cảnh báo cháy và rò rỉ khí. Đây bao gồm việc xác định loại cảm biến và thiết bị phù hợp, đặc điểm kỹ thuật của hệ thống, và cách kết nối và tích hợp các thành phần với nhau.

Mô phỏng và mô hình hóa: Sử dụng phần mềm mô phỏng và mô hình hóa để đánh giá hiệu suất của hệ thống cảnh báo trong các tình huống cháy, rò rỉ khí khác nhau nhiệt độ, độ ẩm vượt mức cho phép và thời gian hệ thống có đồng bộ với nhau. Điều này giúp xác định thời gian phản ứng, độ nhạy, độ chính xác và khả năng phát hiện của hệ thống.

Thử nghiệm thực tế: Tiến hành các thử nghiệm thực tế trên mô hình hoặc trong môi trường thực để đánh giá hiệu suất thực tế của hệ thống. Điều này có thể bao gồm việc cài đặt hệ thống trong một môi trường được điều chỉnh và tiến hành các cuộc thử nghiệm và phân tích dữ liệu thu được.

Đánh giá và phân tích dữ liệu: Phân tích dữ liệu thu thập được từ các thử nghiệm để đánh giá hiệu suất và sự hiệu quả của hệ thống cảnh báo cháy và rò rỉ khí. Điều này bao gồm xem xét độ nhạy, độ chính xác, thời gian phản ứng và các chỉ số hiệu suất khác. So sánh và đánh giá: So sánh kết quả thu được từ hệ thống cảnh báo cháy và rò rỉ khí với các hệ thống tương tự hiện có để đánh giá ưu điểm và hạn chế của đề tài nghiên cứu.

1.5. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu của đề tài " Hệ thống giám sát từ xa nhà kho, phân xưởng" có thể bao gồm:

Hệ thống cảnh báo cháy: Đối tượng nghiên cứu là các phương pháp, thiết bị và công nghệ được sử dụng để phát hiện cháy trong một môi trường. Điều này có thể bao gồm cảm biến cháy, hệ thống thông báo, hệ thống báo động và giao diện người dùng.

Hệ thống rò rỉ khí: Đối tượng nghiên cứu là các phương pháp, thiết bị và công nghệ được sử dụng để giám sát về rò rỉ khí gây cháy. Điều này có thể bao gồm cảm biến khí, hệ thống thông báo, hệ thống báo động và giao diện người dùng.

Phạm vi nghiên cứu của đề tài bao gồm các khía cạnh sau:

Phân tích và thiết kế: Nghiên cứu về các phương pháp phát hiện cháy và rò rỉ khí, cũng như thiết kế hệ thống giám sát phù hợp. Điều này bao gồm phân tích yêu cầu, lựa chọn và tích hợp các thành phần cần thiết, và đảm bảo tính ứng dụng và hiệu quả của hệ thống.

Thử nghiệm và đánh giá: Tiến hành thử nghiệm và đánh giá hiệu suất của hệ thống giám sát nhiệt độ, độ ẩm, cảnh báo cháy và rò rỉ khí trong các tình huống thực tế. Điều này có thể bao gồm cài đặt hệ thống trong một môi trường được điều chỉnh và tiến hành các cuộc thử nghiệm để đánh giá độ nhạy, độ chính xác, thời gian phản ứng và khả năng phát hiện của hệ thống.

Đánh giá an toàn và tuân thủ quy định: Xem xét yêu cầu an toàn và quy định pháp luật liên quan đến giám sát cháy, rò rỉ khí gas, nhiệt độ, độ ẩm tích hợp thời gian. Điều này bao gồm kiểm tra xem hệ thống đáp ứng các tiêu chuẩn các hướng dẫn về thiết kế, lắp đặt và bảo trì hệ thống cảnh báo.

1.6. Tóm tắt

Báo cáo đề tài gồm 5 chương:

Chương 1: Giới thiệu sơ lược về đề tài (lý do chọn đề tài, mục tiêu đề ra, giới hạn của đề tài, phương pháp và phạm vi nghiên cứu).

Chương 2: Cơ sở lý thuyết (giới thiệu về lý thuyết liên quan tới các vấn đề cần giải quyết trong đề tài).

Chương 3: Thiết kế hệ thống (giới thiệu về ý tưởng thiết kế, thiết kế phần cứng và phần mềm).

Chương 4: Kết quả (các kết quả đạt được của đề tài).

Chương 5: Kết luận và hướng phát triển (kết luận tổng kết lại các vấn đề đã giải quyết được của đề tài và hướng phát triển xa hơn cho đề tài).

Chương 2: Cơ sở lý thuyết

2.1. Tìm hiểu về ESP32

2.1.1. Giới thiệu

ESP32 là một series các vi điều khiển trên một vi mạch giá rẻ, năng lượng thấp có hỗ trợ WiFi và dual-mode Bluetooth (tạm dịch: Bluetooth chế độ kép). Dòng ESP32 sử dụng bộ vi xử lý Tensilica Xtensa LX6 ở cả hai biến thể lõi kép và lõi đơn, và bao gồm các công tắc antenna tích hợp, RF balun, bộ khuếch đại công suất, bộ khuếch đại thu nhiễu thấp, bộ lọc và module quản lý năng lượng. ESP32 được chế tạo và phát triển bởi Espressif Systems, một công ty Trung Quốc có trụ sở tại Thượng Hải, và được sản xuất bởi TSMC bằng cách sử dụng công nghệ 40nm [1] [2]. ESP32 là sản phẩm kế thừa từ vi điều khiển ESP8266.

2.1.2. Lập trình cho ESP32

Espressif hỗ trợ Espressif IoT Development Framework (IDF) (viết tắt là ESP-IDF), framework chính thức cho ESP32.

Ngoài ra, các ngôn ngữ lập trình, framework, platform và môi trường lập trình khác được sử dụng để lập trình ESP32 bao gồm:

Arduino IDE với ESP32 Arduino Core

MicroPython: Phiên bản triển khai gọn của Python 3 cho vi điều khiển

Espressif Mesh Development Framework: Framework hỗ trợ phát triển mạng Bluetooth mesh trên ESP32.

Espruino: JavaScript SDK và firmware mô phỏng giống với Node.js

Lua RTOS dành cho ESP32

Moddable SDK: bao gồm ngôn ngữ JavaScript và hỗ trợ thư viện cho ESP32

Mongoose OS: hệ điều hành dành cho các sản phẩm kết nối trên vi điều khiển, có thể lập trình bằng JavaScript hoặc C. Mongoose OS là một nền tảng được đề xuất bởi Espressif Systems [3], AWS IoT [4], và Google Cloud IoT [5].

mruby cho ESP32

.NET nanoFramework: Lập trình bằng .NET C #, triển khai và gỡ lỗi bằng Visual Studio.

NodeMCU: Firmware dựa trên Lua

Hệ sinh thái PlatformIO và IDE

Pymakr IDE: IDE được thiết kế để sử dụng với các thiết bị Pycom; xử lý các nâng cấp firmware và bao gồm bảng điều khiển MicroPython REPL

Nền tảng lập trình nhúng Simba

IDb E wea trên khối hệ sinh thái Whitecat

Zerynth: Python cho IoT và vi điều khiển, bao gồm cả ESP32

AtomVM: Erlang/Elixir Abstract machine (BEAM) cho ESP32

2.1.3. Tính năng của ESP32

Các tính năng của ESP32 bao gồm: [6]

Bộ xử lý:

CPU: Bộ vi xử lý Xtensa lõi kép (hoặc lõi đơn) 32-bit LX6, hoạt động ở tần số 240 MHz (160 MHz cho ESP32-S0WD và ESP32-U4WDH) và hoạt động ở tối đa 600 MIPS (200 MIPS với ESP32-S0WD/ESP32-U4WDH)

Bộ đồng xử lý (co-processor) công suất cực thấp (Ultra low power, viết tắt: ULP) hỗ trợ việc đọc ADC và các ngoại vi khi bộ xử lý chính (main processor) vào chế độ deep sleep.

Hệ thống xung nhịp: CPU Clock, RTC Clock và Audio PLL Clock

Bộ nhớ nội:

448 KB bộ nhớ ROM cho việc booting và các tính năng lõi

520 KB bộ nhớ SRAM trên chip cho dữ liệu và tập lệnh

Kết nối không dây:

Wi-Fi: 802.11 b/g/n

Bluetooth: v4.2 BR/EDR và BLE (chia sẻ sóng vô tuyến với Wi-Fi)

Bảo mật:

Hỗ trợ tất cả các tính năng bảo mật chuẩn IEEE 802.11, bao gồm WFA, WPA/WPA2 và WAPI.

Secure boot (tạm dịch: khởi động an toàn)

Mã hóa flash

1024-bit OTP, lên đến 768-bit cho khách hàng

Tăng tốc mã hóa phần cứng: AES, SHA-2, RSA, elliptic curve cryptography (ECC, tạm dịch: mật mã đường cong ellip), trình tạo số ngẫu nhiên (random number generator, viết tắt: RNG)

Quản lý năng lượng:

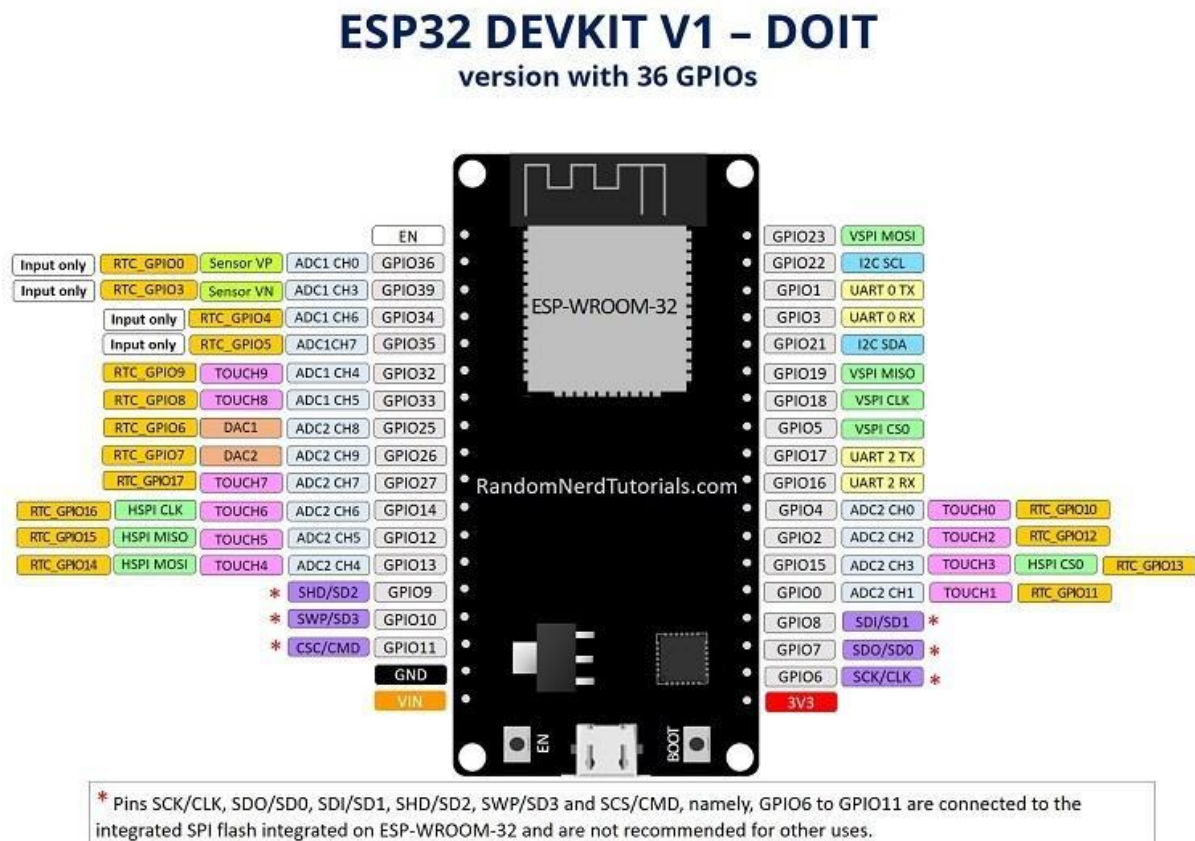
Hỗ trợ 5 chế độ hoạt động với mức tiêu thụ năng lượng khác nhau: Active, Modem-sleep, Light-sleep, Deep-sleep và Hibernation

Bộ ổn áp nội với điện áp rơi thấp (internal low-dropout regulator)

Miền nguồn riêng (individual power domain) cho RTC

Trở lại hoạt động từ ngắt GPIO, timer, đo ADC, ngắt với cảm ứng điện dung

2.1.4. Sơ đồ chân



Hình 2. 1. Sơ đồ chân ESP32 DEVKIT V1

Sơ đồ chân của ESP32 DEVKIT V1 gồm: [7]

- 34 GPIO có thể lập trình
- 18 kênh ADC 12 bit
- 2 kênh DAC 8-bit
- 16 kênh PWM
- 3 giao diện UART
- 3 giao diện SPI
- 2 Giao diện I2C

- 2 Giao diện I2S
- 10 GPIO cảm ứng điện dung
- 16 GPIO RTC

GPIO

Thiết bị ngoại vi được sử dụng phổ biến nhất là GPIO. ESP32 có 34 chân GPIO với mỗi chân thực hiện nhiều hơn một chức năng (chỉ một chân hoạt động). Có thể cấu hình chân dưới dạng GPIO hoặc ADC hoặc UART trong chương trình.

Các chân ADC và DAC được xác định trước và phải sử dụng các chân do nhà sản xuất chỉ định. Nhưng các chức năng khác như PWM, SPI, UART, I2C, ... có thể được gán cho bất kỳ chân GPIO nào thông qua chương trình.

RTC GPIO

ESP32 có 16 GPIO RTC, là một phần của hệ thống con RTC Low-Power. Các chân này có thể được sử dụng để đánh thức ESP32 khỏi chế độ ngủ sâu làm nguồn đánh thức bên ngoài.

ADC

ESP32 có hai module chuyển đổi analog sang kỹ thuật số SAR 12 bit với 8 kênh và 10 kênh mỗi module. Vì vậy, khối ADC1 và ADC2 kết hợp với nhau có 18 kênh ADC 12-bit.

Với độ phân giải 12-bit, các giá trị kỹ thuật số đầu ra sẽ nằm trong khoảng 0 - 4093.

DAC

Bộ vi điều khiển ESP32 có hai kênh chuyển đổi kỹ thuật số sang analog 8 bit độc lập để chuyển đổi các giá trị kỹ thuật số sang tín hiệu điện áp analog. DAC có mạng điện trở bên trong và sử dụng nguồn điện làm điện áp tham chiếu đầu vào.

Hai chân GPIO sau được liên kết với các chức năng của DAC.

DAC1 - GPIO25

DAC2 - GPIO26

GPIO cảm ứng điện dung

SoC ESP32 có 10 GPIO cảm ứng điện dung, có thể phát hiện các biến thể về điện dung trên chân cảm do chạm hoặc tiếp cận chân GPIO bằng ngón tay hoặc bút stylus.

Các GPIO cảm ứng này có thể được sử dụng để triển khai các miếng cảm ứng điện dung mà không cần bất kỳ phần cứng bổ sung nào.

SPI

Chip Wi-Fi ESP32 có ba khối SPI (SPI, HSPI và VSPI) ở cả chế độ master và chế độ slave. SPI được sử dụng để giao tiếp với bộ nhớ Flash. Vì vậy, có hai giao diện SPI.

I2C

Có hai giao diện I2C trong ESP32 với sự linh hoạt hoàn toàn trong việc gán các chân, tức là, người dùng có thể gán các chân SCL và SDA cho cả hai giao diện I2C trong chương trình.

Nếu sử dụng Arduino IDE, thì các chân I2C mặc định là:

SDA - GPIO21

SCL - GPIO22

PWM

Bộ điều khiển PWM trong ESP32 có 16 kênh dạng sóng PWM độc lập với tần số và chu kỳ nhiệm vụ có thể định cấu hình. Dạng sóng PWM có thể được sử dụng để điều khiển động cơ và LED. Có thể cấu hình tần số tín hiệu PWM, kênh, chân GPIO và cả chu kỳ nhiệm vụ.

2.2. Cảm biến khí gas MQ-2

2.2.1. Chức năng

Cảm biến khí gas MQ-2 sử dụng phần tử SnO₂ có độ dẫn điện thấp hơn trong không khí sạch, khi khí dễ cháy tồn tại, cảm biến có độ dẫn điện cao hơn, nồng độ chất dễ cháy càng cao thì độ dẫn điện của SnO₂ sẽ càng cao và được tương ứng chuyển đổi thành mức tín hiệu điện.

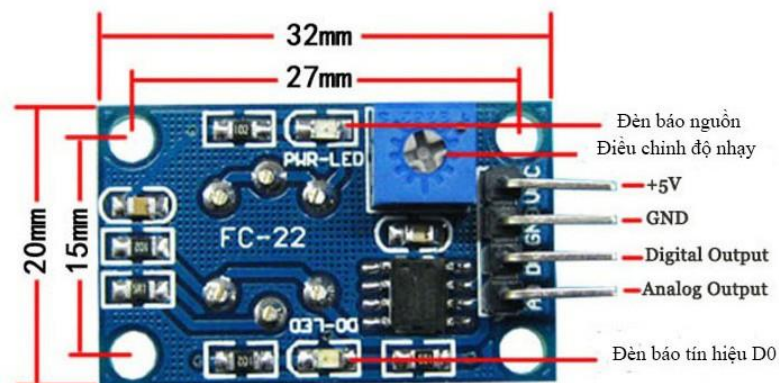
Cảm biến khí gas MQ-2 là cảm biến khí có độ nhạy cao với LPG, Propane và Hydrogen, mê-tan (CH₄) và hơi dễ bắt lửa khác, với chi phí thấp và phù hợp cho các ứng dụng khác nhau. [8]

2.2.2. Cấu tạo của MQ-2

Thông số kỹ thuật : [9]

- Điện áp hoạt động: 3.3V-5V
- Kích thước PCB: 3cm * 1.6cm

- Led đỏ báo nguồn vào, Led xanh báo gas
- IC so sánh : LM393
- VCC: 3.3V-5V
- GND: 0V
- DO: Đầu ra tín hiệu số (0 và 1)
- AO: Đầu ra Analog (Tín hiệu tương tự)
- Cấu tạo từ chất bán dẫn SnO_2
- Phạm vi phát hiện: 300 ~ 10000ppm
- Độ nhạy: R trong không khí / khí đặc trưng của $R_{in} \geq 5$
- Thời gian đáp ứng: ≤ 10 giây
- Thời gian khôi phục: ≤ 30 giây
- Trở kháng nhiệt: $31\Omega \pm 3\Omega$
- Dòng điện: $\leq 180\text{mA}$
- Điện áp làm nóng: $5.0\text{V} \pm 0.2\text{V}$
- Công suất làm nóng: $\leq 900\text{mW}$
- Điện áp đo: $\leq 24\text{VDC}$
- Điều kiện làm việc:
- Nhiệt độ: $-20^\circ\text{C} \sim +55^\circ\text{C}$
- Độ ẩm: $\leq 95\% \text{ RH}$
- Oxy môi trường: 21%



Hình 2. 2. Cảm biến khí gas MQ-2

2.4. LCD1602

2.4.1. Chức năng

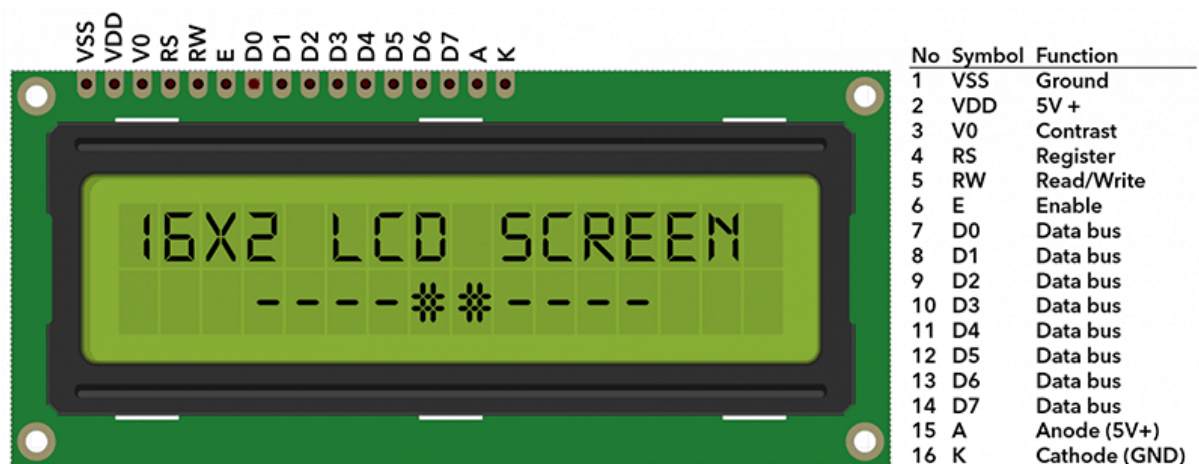
Màn hình LCD 1602 xanh lá sử dụng driver HD44780, có khả năng hiển thị 2 dòng với mỗi dòng 16 ký tự, màn hình có độ bền cao, rất phổ biến, nhiều code mẫu và dễ dàng sử dụng hơn nếu đi kèm mạch chuyển tiếp I2C. [11]

2.4.2. Cấu tạo của LCD1602

Thông số kỹ thuật: [11]

- Điện áp hoạt động là 5V.
- Kích thước: 80 x 36 x 12.5mm
- Chữ trắng, nền xanh dương
- Khoảng cách giữa hai chân kết nối là 0.1 inch tiện dụng khi kết nối với Breadboard.
- Tên các chân được ghi ở mặt sau của màn hình LCD hỗ trợ việc kết nối, đi dây điện.
- Có đèn led nền, có thể dùng biến trở hoặc PWM điều chỉnh độ sáng để sử dụng ít điện năng hơn.
- Có thể được điều khiển với 6 dây tín hiệu

Sơ đồ chân Màn hình LCD 1602 Xanh Lá:



Hình 2. 4. Màn hình LCD 16x2

Bảng 1: Bảng chân và chức năng của LCD

Chân	Ký hiệu	Mô tả	Giá trị
1	VSS	GND	0V
2	VCC		5V
3	V0	Độ tương phản	
4	RS	Lựa chọn thanh ghi	RS=0 (mức thấp) chọn thanh ghi lệnh RS=1 (mức cao) chọn thanh ghi dữ liệu
5	R/W	Chọn thanh ghi đọc/viết dữ liệu	R/W=0 thanh ghi viết R/W=1 thanh ghi đọc
6	E	Enable	
7	DB0	Chân truyền dữ liệu	8 bit: DB0DB7
8	DB1		
9	DB2		
10	DB3		
11	DB4		
12	DB5		
13	DB6		
14	DB7		
15	A	Cực dương led nền	0V đến 5V

16	K	Cực âm led nền	0V
----	---	----------------	----

2.5. Giao thức I2C

2.5.1. Khái niệm

I2C (Inter – Integrated Circuit) là 1 giao thức giao tiếp nối tiếp đồng bộ được phát triển bởi Philips Semiconductors, sử dụng để truyền nhận dữ liệu giữa các IC với nhau chỉ sử dụng hai đường truyền tín hiệu.

Các bit dữ liệu sẽ được truyền từng bit một theo các khoảng thời gian đều đặn được thiết lập bởi 1 tín hiệu đồng hồ.

Bus I2C thường được sử dụng để giao tiếp ngoại vi cho rất nhiều loại IC khác nhau như các loại vi điều khiển, cảm biến, EEPROM,

2.5.2. Cấu tạo

I2C sử dụng 2 đường truyền tín hiệu:

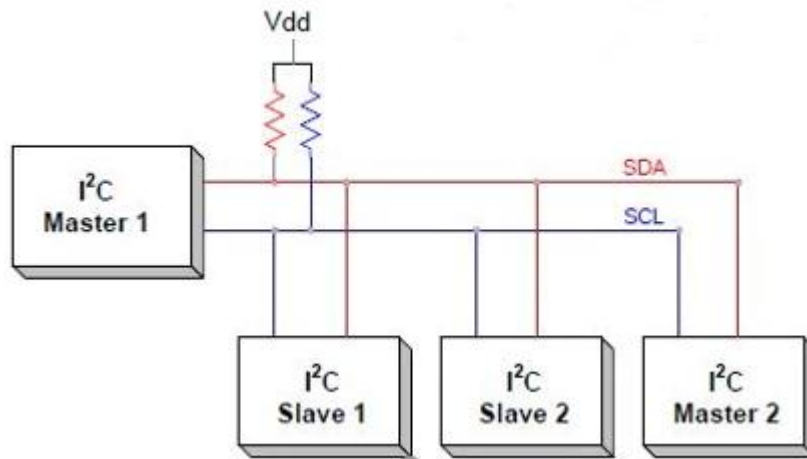
- SCL - Serial Clock Line : Tạo xung nhịp đồng hồ do Master phát đi
- SDA - Serial Data Line : Đường truyền nhận dữ liệu.

Giao tiếp I2C bao gồm quá trình truyền nhận dữ liệu giữa các thiết bị chủ tớ, hay Master - Slave.

Thiết bị Master là 1 vi điều khiển, nó có nhiệm vụ điều khiển đường tín hiệu SCL và gửi nhận dữ liệu hay lệnh thông qua đường SDA đến các thiết bị khác.

Các thiết bị nhận các dữ liệu lệnh và tín hiệu từ thiết bị Master được gọi là các thiết bị Slave. Các thiết bị Slave thường là các IC, hoặc thậm chí là vi điều khiển.

Master và Slave được kết nối với nhau như hình trên. Hai đường bus SCL và SDA đều hoạt động ở chế độ Open Drain, nghĩa là bất cứ thiết bị nào kết nối với mạng I2C này cũng chỉ có thể kéo 2 đường bus này xuống mức thấp (LOW), nhưng lại không thể kéo được lên mức cao. Vì để tránh trường hợp bus vừa bị 1 thiết bị kéo lên mức cao vừa bị 1 thiết bị khác kéo xuống mức thấp gây hiện tượng ngắn mạch. Do đó cần có 1 điện trở (từ 1 – 4,7 kΩ) để giữ mặc định ở mức cao.



Hình 2. 5. Cấu tạo mô hình I2C

2.5.3. Khung truyền I2C

Bắt đầu	7 bit địa chỉ	Bit Read/Write	Bit ACK/NACK	8 bit dữ liệu	Bit ACK/NACK	Kết thúc
---------	---------------	----------------	--------------	---------------	--------------	----------

Hình 2. 6. Khung truyền I2C

Khởi bit địa chỉ :

Thông thường quá trình truyền nhận sẽ diễn ra với rất nhiều thiết bị, IC với nhau. Do đó để phân biệt các thiết bị này, chúng sẽ được gán 1 địa chỉ vật lý 7 bit cố định.

Bit Read/Write:

Bit này dùng để xác định quá trình là truyền hay nhận dữ liệu từ thiết bị Master. Nếu Master gửi dữ liệu đi thì ứng với bit này bằng '0', và ngược lại, nhận dữ liệu khi bit này bằng '1'.

Bit ACK/NACK:

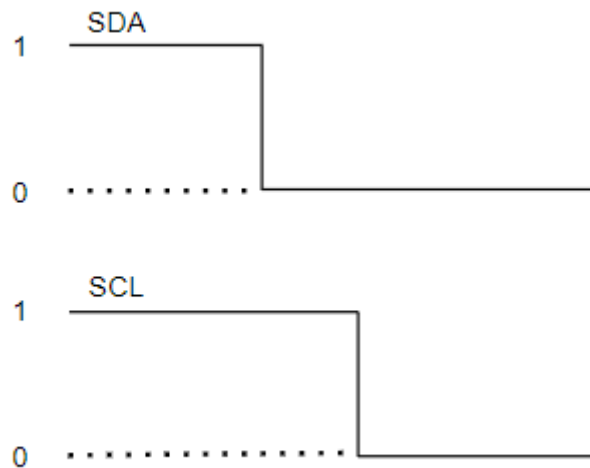
Viết tắt của Acknowledged / Not Acknowledged. Dùng để so sánh bit địa chỉ vật lý của thiết bị so với địa chỉ được gửi tới. Nếu trùng thì Slave sẽ được đặt bằng '0' và ngược lại, nếu không thì mặc định bằng '1'.

Khởi bit dữ liệu:

Gồm 8 bit và được thiết lập bởi thiết bị gửi truyền đến thiết bị nhận. Sau khi các bit này được gửi đi, lập tức 1 bit ACK/NACK được gửi ngay theo sau để xác nhận rằng thiết bị nhận đã nhận được dữ liệu thành công hay chưa. Nếu nhận thành công thì bit ACK/NACK được set bằng '0' và ngược lại.

2.5.4. Quá trình truyền nhận dữ liệu

Bắt đầu: Thiết bị Master sẽ gửi đi 1 xung Start bằng cách kéo lần lượt các đường SDA, SCL từ mức 1 xuống 0.



Hình 2. 7. Khung tín hiệu SDA và SCL

Tiếp theo đó, Master gửi đi 7 bit địa chỉ tới Slave muốn giao tiếp cùng với bit Read/Write.

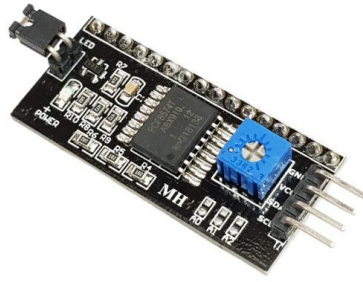
Slave sẽ so sánh địa chỉ vật lý với địa chỉ vừa được gửi tới. Nếu trùng khớp, Slave sẽ xác nhận bằng cách kéo đường SDA xuống 0 và set bit ACK/NACK bằng '0'. Nếu không trùng khớp thì SDA và bit ACK/NACK đều mặc định bằng '1'.

Thiết bị Master sẽ gửi hoặc nhận khung bit dữ liệu. Nếu Master gửi đến Slave thì bit Read/Write ở mức 0. Ngược lại nếu nhận thì bit này ở mức 1.

Nếu như khung dữ liệu đã được truyền đi thành công, bit ACK/NACK được set thành mức 0 để báo hiệu cho Master tiếp tục.

Sau khi tất cả dữ liệu đã được gửi đến Slave thành công, Master sẽ phát 1 tín hiệu Stop để báo cho các Slave biết quá trình truyền đã kết thúc bằng cách chuyển lần lượt SCL, SDA từ mức 0 lên mức 1.

2.5.5. Module I2C



Hình 2. 8. Module I2C

Thông số mạch chuyển đổi giao tiếp I2C

- Kích thước: 41.5mm(l)x19mm(w)x15.3mm(h)
- Trọng lượng: 5g
- Điện áp hoạt động: 2.5v-6v
- Jump chốt: cung cấp đèn cho lcd hoặc ngắt
- Biến trở xoay độ tương phản cho lcd

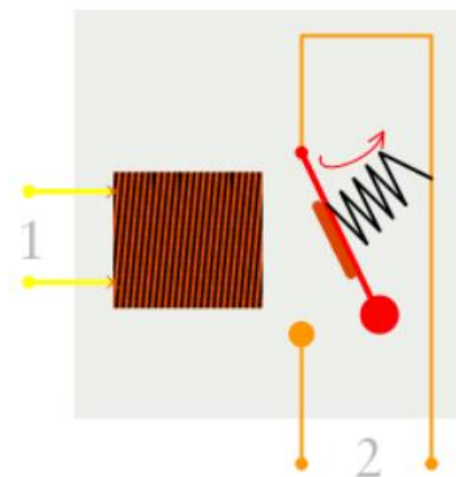
2.6. Relay

2.6.1. Cấu tạo của Relay

Về cấu tạo của relay bao gồm một cuộn dây kim loại làm bằng đồng hoặc nhôm được quấn quanh một lõi sắt từ. Bộ phận này có phần tĩnh gọi là ách từ (Yoke). Còn phần động được gọi là phần cứng (Armature). Phần cứng của relay sẽ được kết nối với một tiếp điểm động. Cuộn dây có tác dụng hút thanh tiếp điểm lại để từ đó tạo thành trạng thái NO và NC. Nhiệm vụ của mạch tiếp điểm (mạch lực) là đóng cắt các thiết bị tải với dòng điện nhỏ và được cách ly bởi một cuộn hút. [12]

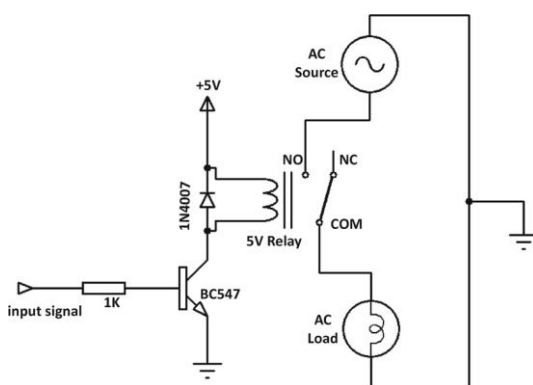
2.6.2. Nguyên lý hoạt động

Khi dòng điện chạy qua mạch thứ nhất (1), nó sẽ kích hoạt nam châm điện. Từ đó tạo ra từ trường để thu hút một tiếp điểm (màu đỏ). Sau đó sẽ kích hoạt mạch thứ hai (2). Khi tắt nguồn, một lò xo được lắp trước vào tiếp điểm sẽ có nhiệm vụ là kéo tiếp điểm trở lại vị trí ban đầu, tắt mạch thứ hai lại một lần nữa.



Hình 2. 9. . Nguyên lý hoạt động của Relay

2.6.3. Sơ đồ nguyên lý của relay



Hình 2. 10. Sơ đồ nguyên lý của Relay

Khi tín hiệu đưa vào là mức 0 (Tức =0V) thì transistor không dẫn do không có dòng $I_{BE} \gg$ Role không làm việc.

Khi tín hiệu đưa vào là mức 1 (Tức =5V) thì sẽ qua điện trở hạn dòng, làm cho transistor dẫn thông lúc này ta có dòng I_{ce} là dòng điện chạy qua cuộn dây \gg transistor \gg Mát, Role đóng tiếp điểm thường mở (ĐK thiết bị nào đó).

Diode trong mạch có tác dụng chống lại dòng điện cảm ứng do cuộn dây sinh ra làm hỏng transistor.

2.7. DHT11

2.7.1. Chức năng

DHT11 Cảm Biến Nhiệt Độ Độ Ẩm kỹ thuật số chất lượng tốt chi phí thấp. Nó sử dụng một cảm biến độ ẩm điện dung và một cảm biến nhiệt để đo không khí xung quanh, và tạo ra một tín hiệu số trên pin dữ liệu, đơn giản để sử dụng, nhưng đòi hỏi thời gian cần thiết để lấy dữ liệu. [13]

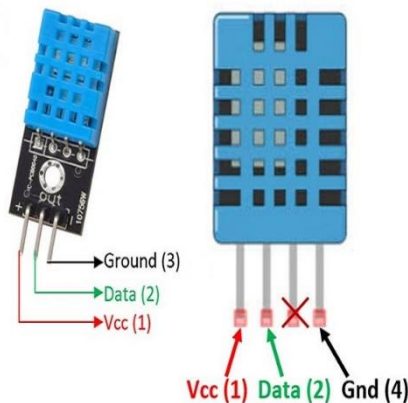
2.7.2. Cấu tạo DHT11

Cảm biến DHT11 bao gồm một phần tử cảm biến độ ẩm điện dung và một điện trở nhiệt để cảm nhận nhiệt độ. Tụ điện cảm biến độ ẩm có hai điện cực với chất nền giữ ẩm làm chất điện môi giữa chúng. Thay đổi giá trị điện dung xảy ra với sự thay đổi của các mức độ ẩm. IC đo, xử lý các giá trị điện trở đã thay đổi này và chuyển chúng thành dạng kỹ thuật số.

Để đo nhiệt độ, cảm biến này sử dụng một nhiệt điện trở có hệ số nhiệt độ âm, làm giảm giá trị điện trở của nó khi nhiệt độ tăng. Để có được giá trị điện trở lớn hơn ngay cả đối với sự thay đổi nhỏ nhất của nhiệt độ, cảm biến này thường được làm bằng gốm bán dẫn hoặc polymer.

Thông số kỹ thuật:

- Điện áp cấp và I/O: 3~5 VDC
- Dòng điện tối đa: 2.5mA (trong khi xử lý dữ liệu)
- Độ ẩm đọc tốt: 20-80% với độ chính xác 5%
- Nhiệt độ đọc tốt: $0\sim 50^{\circ}\text{C} \pm 2^{\circ}\text{C}$
- Tốc độ lấy mẫu nhiều hơn 1 Hz (một lần mỗi giây)
- Kích thước cơ thể 15.5mm x 12mm x 5.5mm
- 4 chân với khoảng cách 2.54mm



Hình 2. 11. Cảm biến nhiệt độ, độ ẩm DHT11

2.8. RTOS

2.8.1. Khái niệm RTOS

RTOS hoạt động dựa trên hai cơ chế là hướng sự kiện (event-driven) hoặc chia sẻ thời gian (time-sharing). Cơ chế hướng sự kiện sẽ giải quyết và điều phối các tác vụ

(task) thông qua mức độ ưu tiên của chúng, còn cơ chế chia sẻ thời gian sẽ chuyển đổi các tác vụ dựa trên phản ứng ngắt của xung nhịp. Phần lớn các hệ điều hành RTOS đều sử dụng giải thuật pre-emptive scheduling (tạm dịch là lập lịch trước).

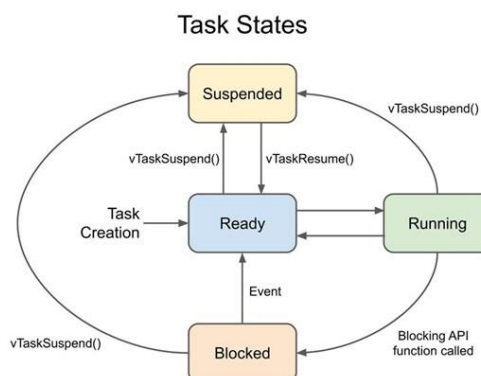
Hệ điều hành RTOS thường được chia thành ba loại chính là:

- Hard RTOS (Tạm dịch: Hệ điều hành thời gian thực cứng): Các hệ điều hành này sẽ luôn đảm bảo các tác vụ được hoàn thành trong một khoảng thời gian cố định, xác định cụ thể, không có sự sai sót.
- Soft RTOS (Tạm dịch: Hệ điều hành thời gian thực mềm) : Đây là các hệ điều hành với thời gian thực hiện tác vụ có thể nhận được một số nới lỏng trong phạm vi cho phép, chỉ cần hoàn thành đúng thời gian quy định.
- Firm RTOS (Tạm dịch: Hệ điều hành thời gian thực bền vững) : Hệ điều hành cũng có các giới hạn thời gian được xác định cụ thể với mức độ chính xác cao nhất, Firm RTOS cũng đảm bảo các tác vụ luôn được thực hiện thành công ngay cả trong trường hợp quá thời hạn quy định.

Các chức năng cơ bản của RTOS

Scheduler (Bộ lập lịch), trong Scheduler mỗi tác vụ sẽ có 3 trạng thái mặc định là:

- Ready to run: Trạng thái chuẩn bị của tác vụ khi đã có đủ các tài nguyên để khởi chạy.
- Running: Trạng thái tác vụ đang thực thi.
- Blocked: Đây là trạng thái mà các tác vụ không đủ tài nguyên để xử lý sẽ được về trạng thái khóa.



Hình 2. 12. Chức năng lập lịch của RTOS

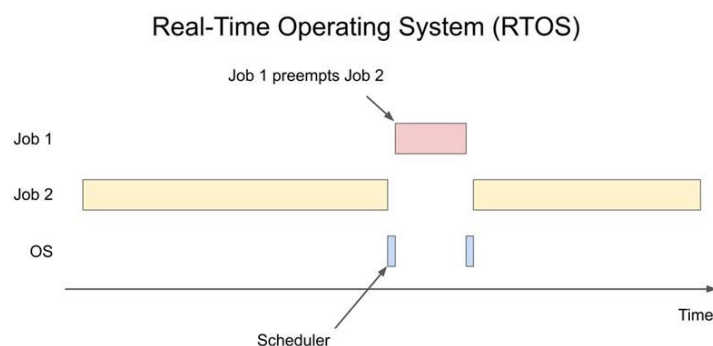
RTOS Services (Dịch vụ thời gian thực) với các dịch vụ:

- Dịch vụ xử lý ngắt (Interrupt handling services).
- Dịch vụ thời gian (Time services).
- Dịch vụ quản lý thiết bị (Device management services).
- Dịch vụ quản lý bộ nhớ (Memory management services).
- Dịch vụ quản lý kết nối (IO services).

Messaging (Các thông điệp). Các thông điệp này sẽ dùng để trao đổi thông tin giữa các tác vụ với nhau, bao gồm:

- Semaphores: Đồng bộ hóa quyền truy cập các tài nguyên dùng chung.
- Event flags: Đồng bộ hóa hoạt động cần có sự phối hợp của nhiều tác vụ.
- Mailboxes, Pipes, Message queues: Quản lý các thông điệp đã được gửi đi và đến giữa các task.

Ưu điểm của RTOS



Hình 2. 13. Ưu điểm của RTOS

Bên cạnh đó hệ điều hành thời gian thực còn có một số ưu điểm khác như:

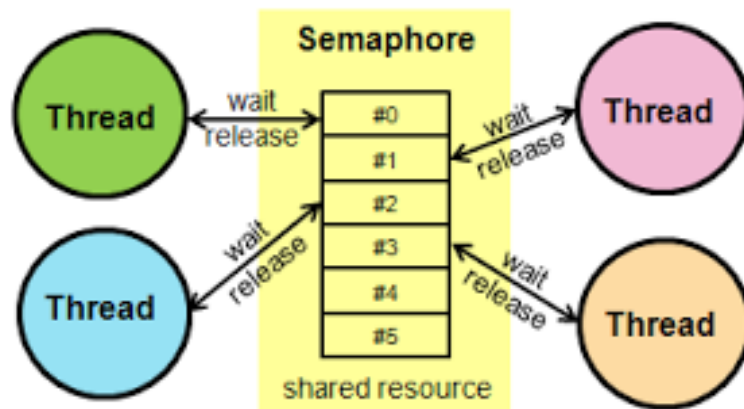
- Độ ổn định và tin cậy cao, nên có thể hoạt động trong thời gian dài mà không cần quá nhiều sự can thiệp của con người.
- Hiệu suất tốt hơn cùng mức tiêu thụ bộ nhớ thấp nên không gây tiêu tốn quá nhiều tài nguyên hoặc RAM.
- RTOS gần như không có lỗi và nếu có xảy ra lỗi cũng dễ dàng phát hiện hơn.
- Có kích thước nhỏ và chi phí thấp. [14]

2.8.2. Semaphore

Khái niệm:

Semaphore giống như 1 cái hàng đợi (queue), dùng để quản lý và bảo vệ tài nguyên dùng chung (share resource). Các thread/task khác nhau khi có yêu cầu sử dụng

tài nguyên dùng chung sẽ bị tổng vào hàng đợi này. Khi nhận được semaphore token thì thread/task nào được tổng vào queue trước thì sử dụng tài nguyên trước. Sau đó nó lại release ra cho thread/task khác dùng.



Hình 2. 14. Chia sẻ tài nguyên trong Semaphore

Nguyên lý hoạt động:

Khi một tiến trình muốn truy cập vào tài nguyên được bảo vệ bởi Semaphore, nó sẽ yêu cầu Semaphore bằng cách gọi hàm Semaphore. Nếu giá trị của Semaphore là lớn hơn 0, tiến trình sẽ giảm giá trị Semaphore đi 1, và tiến trình có thể tiếp tục truy cập vào tài nguyên được bảo vệ. Nếu giá trị Semaphore đã bằng 0 trước khi tiến trình yêu cầu Semaphore, tiến trình sẽ được đưa vào hàng đợi và chờ đợi cho đến khi Semaphore có giá trị lớn hơn 0. Khi Semaphore có giá trị lớn hơn 0, tiến trình đang đợi đầu tiên trong hàng đợi sẽ được đưa vào trạng thái sẵn sàng (ready) và tiếp tục thực thi.

Khi một tiến trình đã sử dụng xong tài nguyên được bảo vệ, nó sẽ tăng giá trị Semaphore lên 1, và Semaphore sẽ thông báo cho tiến trình đang đợi đầu tiên trong hàng đợi biết rằng tài nguyên có sẵn để sử dụng.

Semaphore còn được sử dụng để giải quyết các vấn đề liên quan đến đồng bộ hóa trong hệ thống RTOS, chẳng hạn như tránh xung đột giữa các tiến trình khi truy cập vào tài nguyên cùng một lúc, hoặc đảm bảo rằng một tác vụ được thực hiện hoàn thành trước khi tác vụ khác được thực hiện. [15]

Để thực hiện khai báo FreeRTOS trên ESP32 ta sử dụng hàm `xTaskCreatePinnedToCore(`

`Task1code, /* Tên hàm sẽ xử lý trên core được chọn */`

```

"Task1", /* Tên của tác vụ */
10000, /* Kích thước của tác vụ */
NULL, /* Đối số đầu vào bạn muốn đưa vào bên trong hàm xử lý ở trên */
0, /* Mức độ ưu tiên xử lý (0 là thấp nhất) */
&Task1, /* Tác vụ bạn khai báo ở phía trên */
0); /* Chọn core muốn chạy tác vụ (0 hoặc 1) */

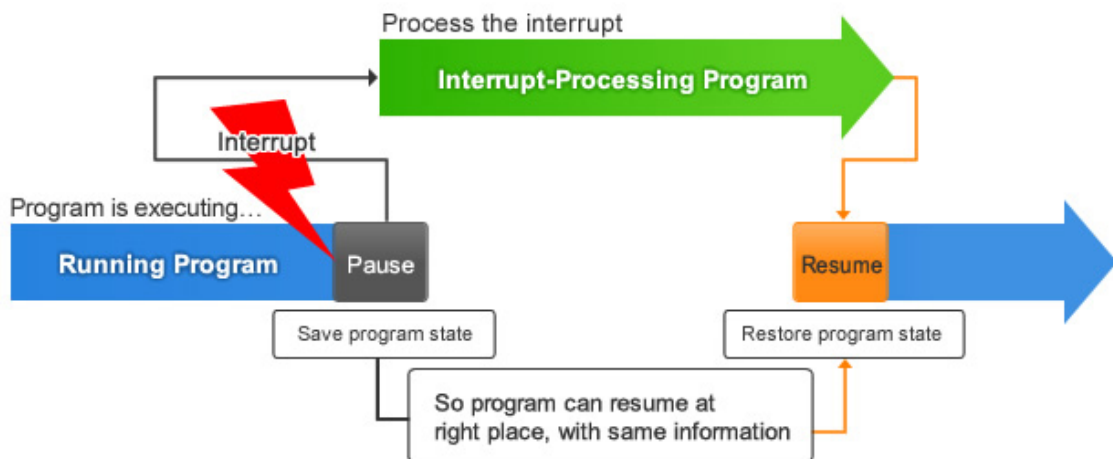
```

2.9. Ngắt

2.9.1. Khái niệm

- Tuần tự: Một chương trình C, Cpp sẽ được xử lý tuần tự theo chiều từ trên xuống dưới và sẽ bị ảnh hưởng bởi các câu lệnh rẽ nhánh và loop
- Ngắt: Chương trình sẽ bị thoát ra khỏi quá trình tuần tự, xử lý xong các lệnh trong ngắt, sau đó mới quay lại xử lý tiếp

Nói một cách đơn giản, Ngắt là một công việc được ưu tiên làm trước, khi xử lý xong mới được làm các việc khác. Các sự kiện ngắt có thể đến từ nhiều nguồn khác nhau như: ADC, IO, UART, TIMER...



Hình 2. 15. Ngắt là một sự kiện được ưu tiên xử lý trước

Ngắt ngoài EXTI chính là một sự kiện ngắt được sinh ra với nguồn từ các chân IO của ESP32.

2.9.2. Ngắt ngoài trong ESP32

Khác với các chip Arduino truyền thống, trên ESP32 mọi chân Input đều có thể được cài đặt để sử dụng với ngắt ngoài. Nó không bị giới hạn như Arduino Uno hoặc các loại tương đương là chỉ có 2 chân có thể được sử dụng làm ngắt ngoài

Ngắt ngoài có các kiểu ngắt khác nhau tương ứng cho tín hiệu xung trên chân GPIO đó, cụ thể:

Bảng 2: Phân loại các ngắt

LOW	Interrupt được kích hoạt khi chân ở mức LOW
HIGH	Interrupt được kích hoạt khi chân ở mức HIGH
CHANGE	Interrupt được kích hoạt khi chân thay đổi mức, từ HIGH sang LOW, hoặc LOW sang HIGH
FALLING	Interrupt được kích hoạt khi chân thay đổi mức, từ HIGH sang LOW
RISING	Interrupt được kích hoạt khi chân thay đổi mức, từ LOW sang HIGH

Để lập trình ngắt ngoài trên ESP32 chúng ta sử dụng hàm:

`attachInterrupt(GPIOPin, ISR, Mode);`

Đây là hàm chức năng được sử dụng để cài đặt ngắt ngoài cho 1 chân cụ thể.

Hàm này có ba đối số đầu vào

- GPIOPin – Là chân GPIO được chỉ định interrupt.
- ISR – Là tên của hàm sẽ được gọi khi có interrupt
- Mode – Khai báo chế độ interrupt được sử dụng, các chế độ tương ứng với các kiểu ngắt bên trên đã nêu.

ISR: Interrupts Service Routine hay định tuyến dịch vụ, là tên hàm sẽ được gọi khi ngắt xảy ra

IRAM_ATTR khai báo cho trình biên dịch biết rằng hàm sẽ được biên dịch và đặt trong RAM (IRAM) của ESP32. Ngược lại hàm sẽ được đặt trong FLASH.

Một hàm khi được trên FLASH sẽ phải chờ Load xong mới có thể thực hiện gây tốn thời gian, do đó hàm ISR không nên được đặt trên FLASH.

Chương 3: Thiết kế hệ thống

3.1. Mô hình hệ thống.

Hệ thống giám sát từ xa nhà kho, phân xưởng nhằm phục vụ việc giám sát từ xa những biến đổi của môi trường trong kho chứa hàng hay các kho máy móc vận hành. Đặc điểm nổi bật của mô hình này chính là sử dụng các cảm biến lửa, nhiệt độ, độ ẩm và khí cháy để giám sát chất lượng môi trường từ xa.

Hệ thống được trang bị màn hình LCD, cho phép người dùng dễ dàng theo dõi biến động chất lượng trong môi trường. Đồng thời có hiển thị thời gian để có thể giám sát trực tiếp trực quan và chi tiết hơn.

Mức độ nguy hiểm của tình huống được thông báo ngay cho người dùng qua ứng dụng Blynk.

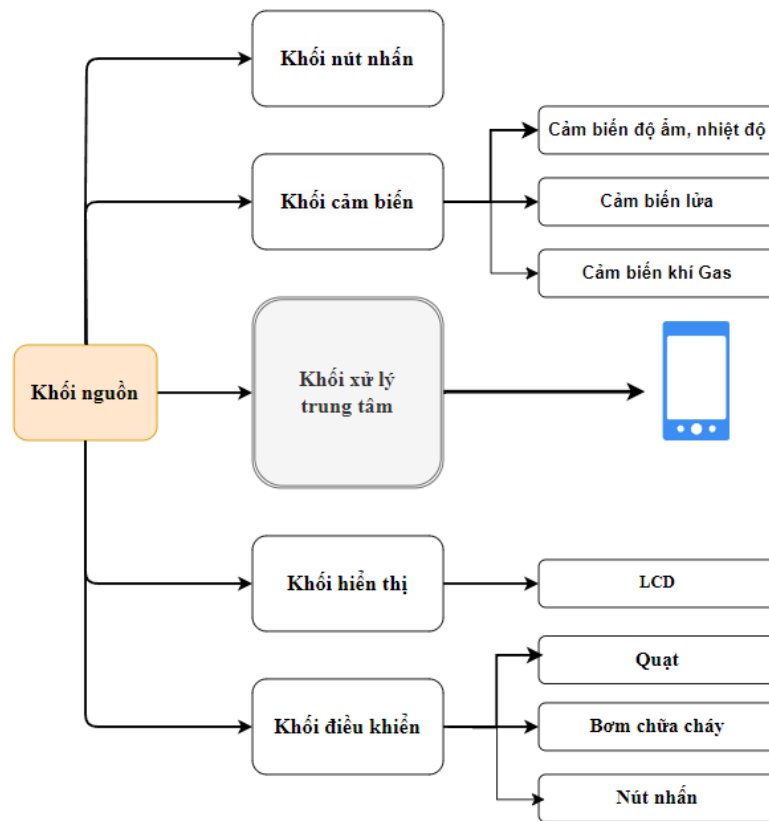
Để nhanh chóng đối phó với nguy cơ cháy nổ, hệ thống cung cấp các nút nhấn cho phép kích hoạt các thiết bị quan trọng như quạt hút khí và máy bơm nước. Nhờ đó, khi xảy ra cháy nổ, người dùng có thể kích hoạt các thiết bị này ngay lập tức để hạn chế tác động của nguy cơ và giảm thiểu thiệt hại.

Đặc biệt, người dùng có thể giám sát và điều khiển các thiết bị qua ứng dụng Blynk. Ứng dụng Blynk cung cấp một giao diện trực quan và thuận tiện, giúp người dùng dễ dàng theo dõi tình trạng hệ thống và tương tác với nó mang lại sự linh hoạt cao.

Tổng quan về hệ thống giám sát từ xa nhà kho, phân xưởng không chỉ đáp ứng nhu cầu cơ bản là giám sát và có khả năng ứng phó nhanh chóng và hiệu quả. Điều này làm tăng sự an tâm và tin tưởng của người dùng trong việc bảo vệ tài sản và tính mạng.

3.2. Thiết kế phần cứng.

3.2.1. Sơ đồ khối.



Hình 3. 1. Sơ đồ khối của hệ thống

Khối nguồn: Cung cấp nguồn điện cho các thiết bị và module để hoạt động một cách ổn định.

Khối xử lý trung tâm: Xử lý các tín hiệu do người dùng đưa vào và xử lý để đưa ra tín hiệu điều khiển các thiết bị và module khác.

Khối cảm biến: Hỗ trợ đo nồng độ khí gas môi trường, nhiệt độ, độ ẩm và cảm nhận lửa là các tín hiệu Analog và Digital sau đó chuyển đổi thành tín hiệu điện để có thể đưa vào bộ xử lý trung tâm.

Khối hiển thị: Cho phép hiển thị thông tin lên LCD 16x2.

Khối điều khiển: Điều khiển bơm chữa cháy bằng relay.

Điện thoại: Nhận thông báo được gửi từ bộ xử lý trung tâm và điều khiển các thiết bị.

3.2.2. Thiết kế từng khối.

Khối điều khiển trung tâm

Hiện nay trên thị trường có rất nhiều dòng vi điều khiển khác nhau như PIC, AVR, 8051, Raspberry, Arduino...Tất cả đều có thể đáp ứng được yêu cầu đặt ra nhưng nhóm chọn ESP32 vì nó có những ưu điểm sau:

ESP32 là một trong những module IoT phổ biến nhất trên thị trường hiện nay và có nhiều ưu điểm, bao gồm:

- Hiệu suất cao: ESP32 có xung nhịp lên đến 240MHz và được trang bị 2 nhân xử lý, cho phép xử lý dữ liệu nhanh hơn và đáp ứng nhu cầu các ứng dụng IoT.
- ESP32 có khả năng kết nối Wi-Fi và Bluetooth đầy đủ, cho phép truyền dữ liệu không dây trong các ứng dụng IoT.
- ESP32 được trang bị các chế độ tiết kiệm năng lượng, cho phép tiết kiệm pin trong các ứng dụng yêu cầu pin lâu.
- ESP32 có giá thành thấp hơn so với các module IoT tương tự, giúp giảm chi phí cho các ứng dụng IoT.
- ESP32 được hỗ trợ bởi nhiều nền tảng phần mềm như Arduino IDE, ESP-IDF, MicroPython, cho phép các nhà phát triển dễ dàng phát triển ứng dụng IoT trên nền tảng này.
- ESP32 được tích hợp nhiều tính năng như GPIO, I2C, SPI, UART, ADC, DAC, PWM, RTC, touch sensor, camera interface, giúp tăng tính linh hoạt cho các ứng dụng IoT.
- Dễ sử dụng: ESP32 rất dễ sử dụng và có nhiều tài liệu và ví dụ minh họa trên Internet.

Khối cảm biến

Module cảm biến độ lửa và cảm biến khí Gas

Thông số kỹ thuật:

- Điện áp làm việc 3.3V ~ 5V
- Có chân cắm vào bảng mạch
- Sử dụng chip LM393 để so sánh, ổn định làm việc

Module cảm biến nhiệt độ độ ẩm

Thông số kỹ thuật:

- Điện áp làm việc 3.3V ~ 5V

- Có chân cắm vào bảng mạch
- Dễ giao tiếp, ổn định làm việc

Khối thiết bị ngoại vi

Module Relay 5VDC có kích thước nhỏ gọn, giá thành rẻ được sử dụng để đóng ngắt thiết bị AC hoặc DC, module sử dụng điện áp 5VDC với chỉ 3 chân kết nối dễ dàng sử dụng. Có sẵn header rất tiện dụng khi kết nối với vi điều khiển.

Thông số kỹ thuật:

- Điện áp hoạt động: 5VDC
- Relay tiêu thụ dòng khoảng 70mA.
- Điện thế đóng ngắt tối đa: AC250V ~ 10A hoặc DC30V ~ 10A.
- Tích hợp Diod chống nhiễu và đèn báo tín hiệu kích.

Khối hiển thị

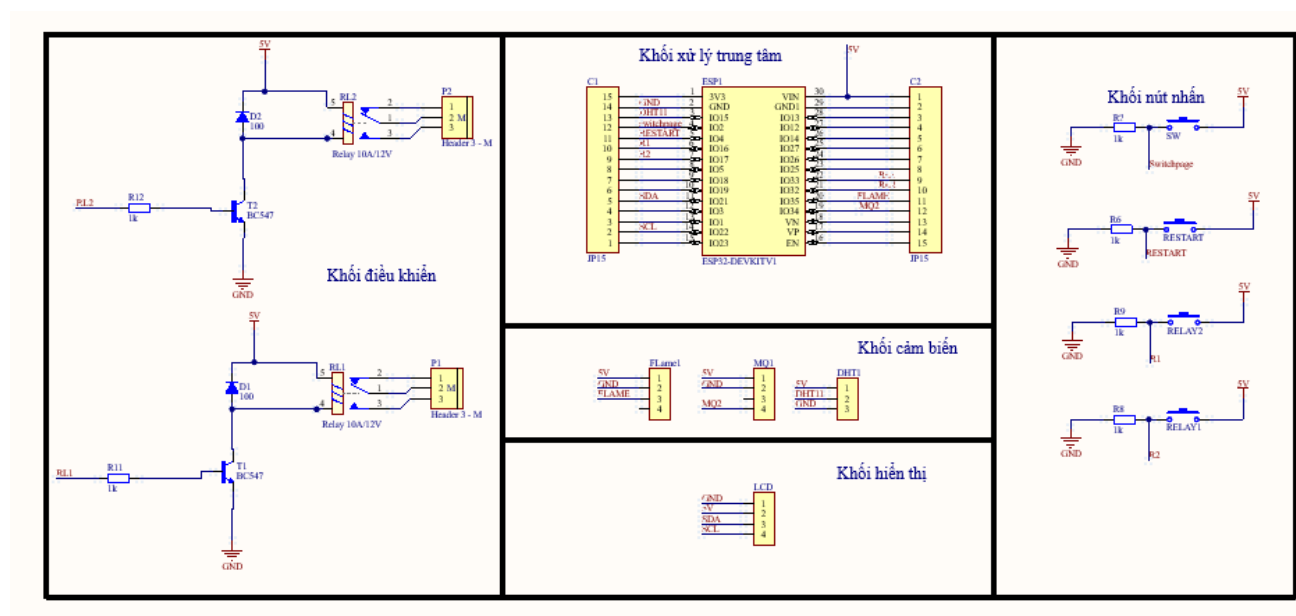
LCD 16x2 với khả năng hiển thị 16 cột 2 hàng, có thể hiển thị các ký tự một cách linh hoạt, đa dạng, trực quan bao gồm số, chữ, ký tự đồ họa, các ký tự đặc biệt... Ngoài ra LCD 16x2 có kích thước nhỏ hơn so với LCD 20x4 vì thế tốn ít tài nguyên hệ thống hơn nhờ đó giúp giảm chi phí giá thành phẩm, cùng với đó LCD đã được các nhà sản xuất tích hợp chip điều khiển (HD44780) bên trong lớp vỏ và chỉ đưa ra các chân giao tiếp cần thiết nên có thể sử dụng dễ dàng. Nhờ đó LCD 16x2 có thể đáp ứng được yêu cầu hiển thị các thông số cũng như là các chức năng hiển thị đã được lập trình trong mạch giúp người sử dụng có thể dễ dàng quan sát. Sử dụng module I2C giao tiếp với khối xử lý trung tâm

Bảng 3. Bảng tính toán các thông số điện áp

Tên linh kiện	Số lượng	Điện áp (V)	Dòng điện (A)	Công suất (W)
ESP32	1	5	0.25	1.25
Module cảm biến khí Gas MQ2	1	5	0.15	0.75
DHT11	1	5	0.2	1
Module cảm biến lửa	1	5	0.15	0.75
I2C	1	5	0	0

LCD16x2	1	5	0.05	0.25
Relay	2	5	0.07	0.7
Bơm nước	1	5	0.2	1
Quạt	1	5	0.2	1
Nút nhấn	4	5	0.1	2
Tổng cộng: 8.7W				

3.2.3. Sơ đồ nguyên lý hệ thống



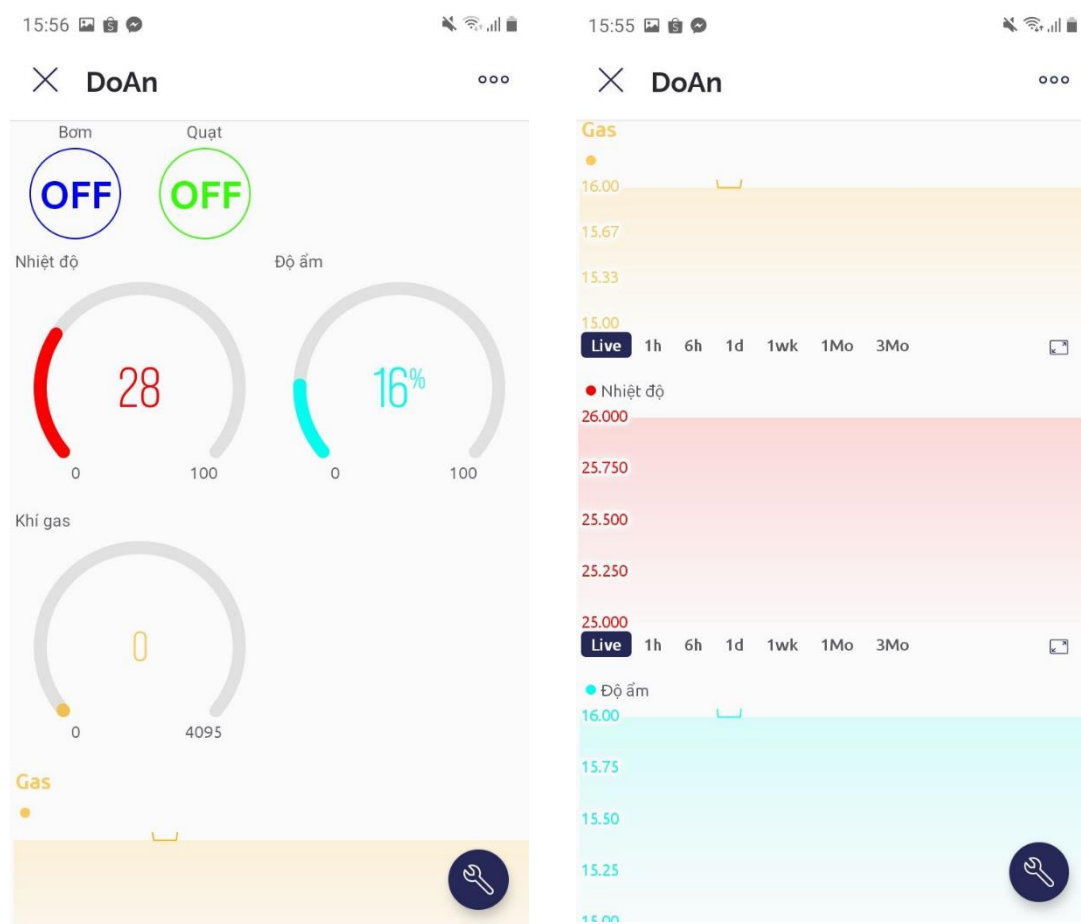
Hình 3. 2. Sơ đồ nguyên lý của hệ thống

3.3. Thiết kế phần mềm

3.3.1. Phần mềm lập trình Arduino IDE

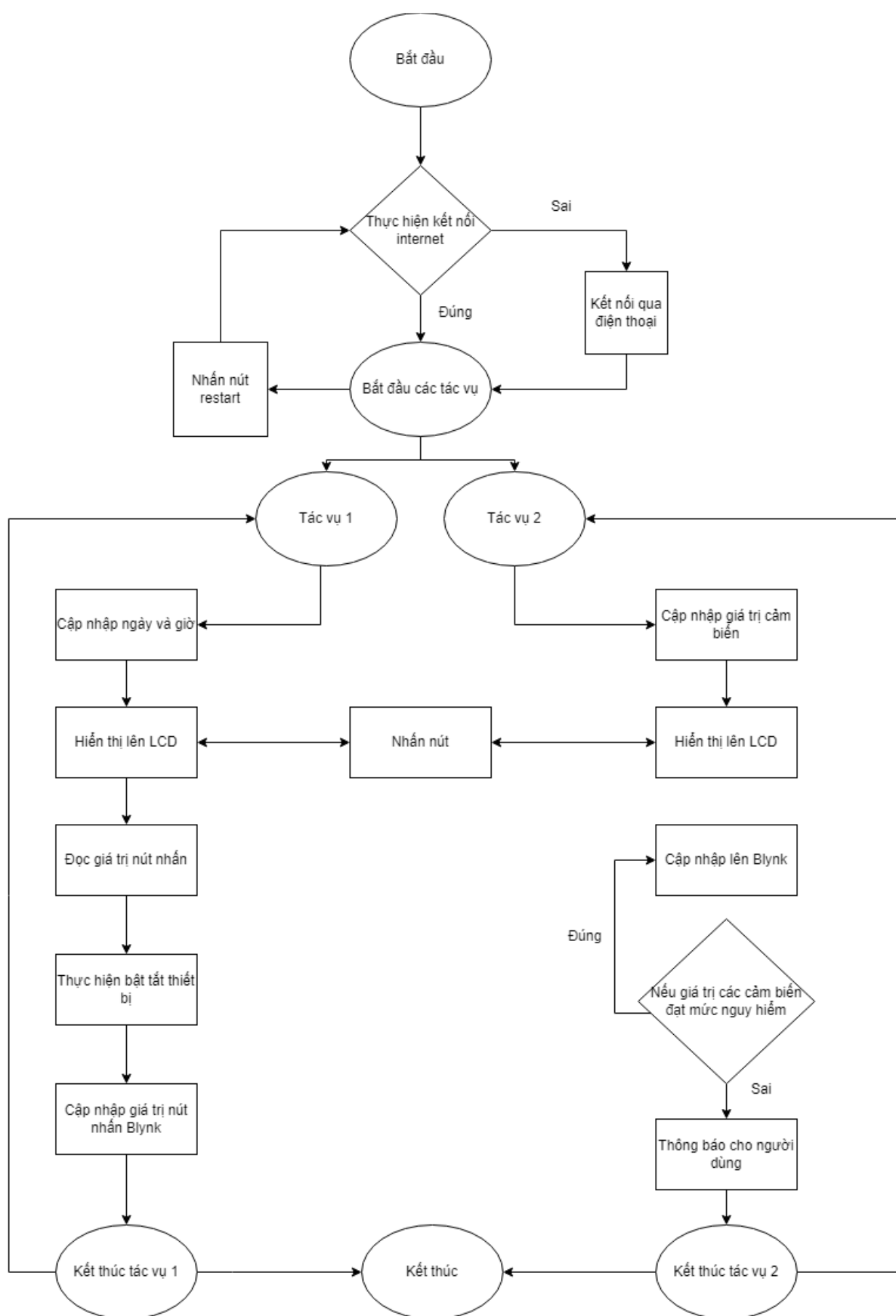
(Code chương trình ở phần phụ lục)

3.3.2. Tương tác qua Blynk



Hình 3. 3. Giao diện tương tác qua Blynk trên điện thoại

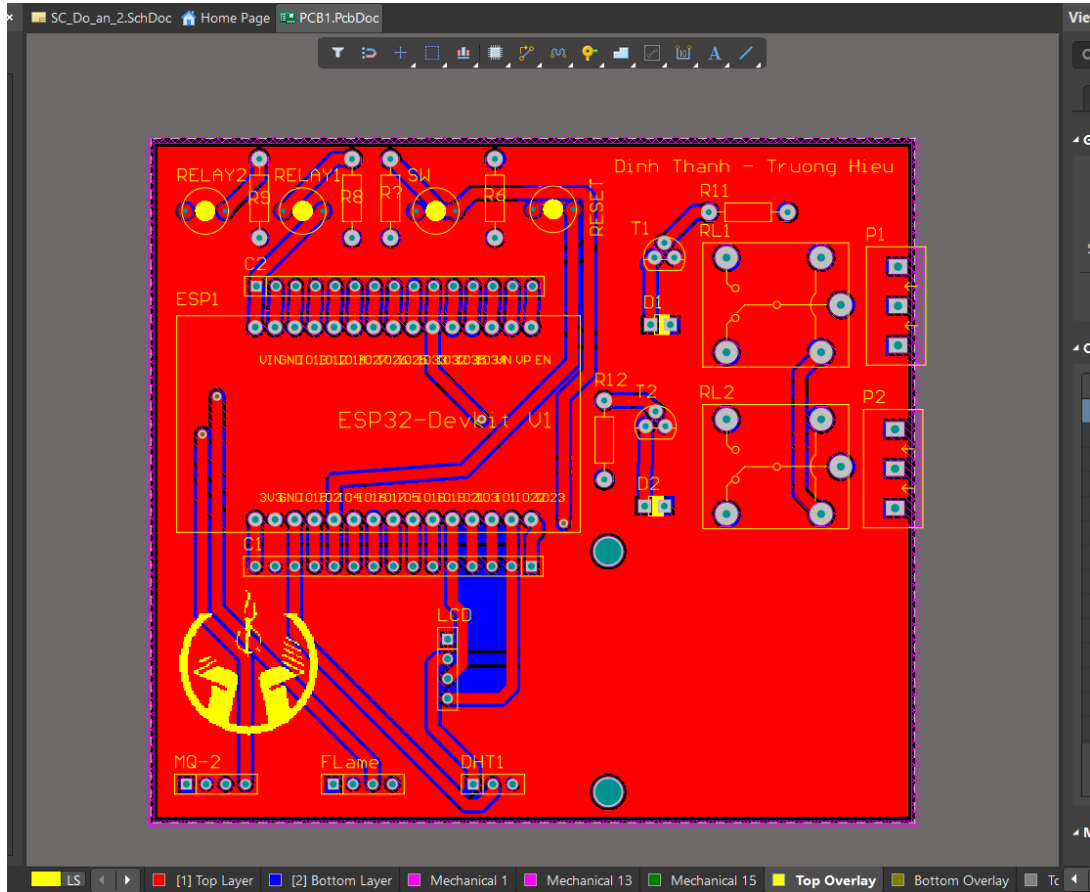
3.3.3. Lưu đồ giải thuật hệ thống



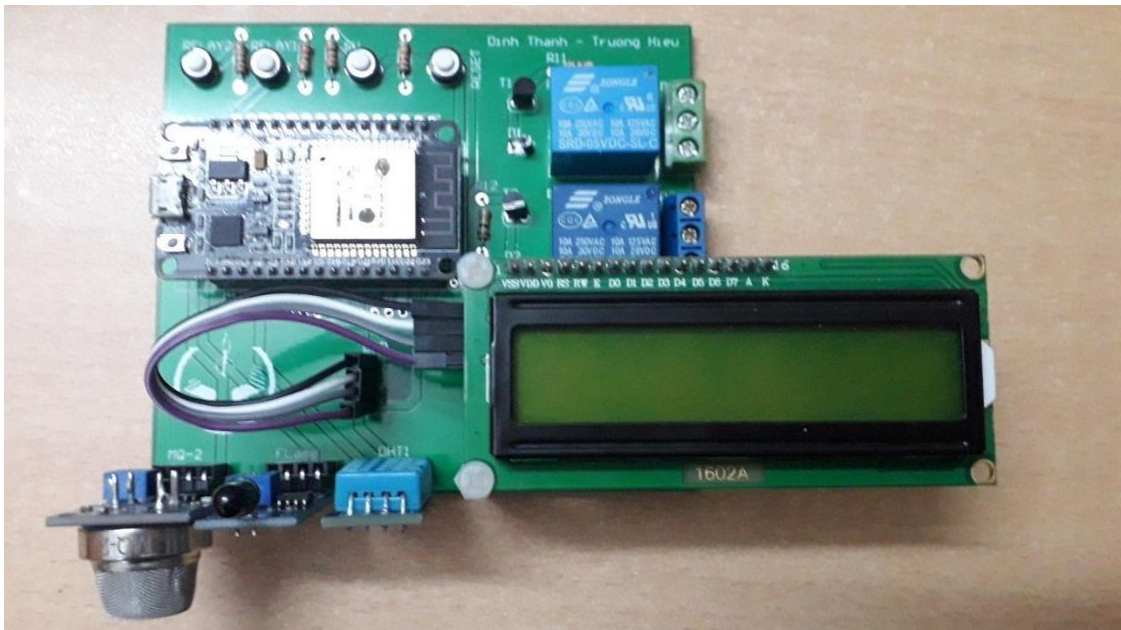
Hình 3. 4. Lưu đồ giải thuật hệ thống

Chương 4: Kết quả đánh giá

4.1. Kết quả mô hình thi công



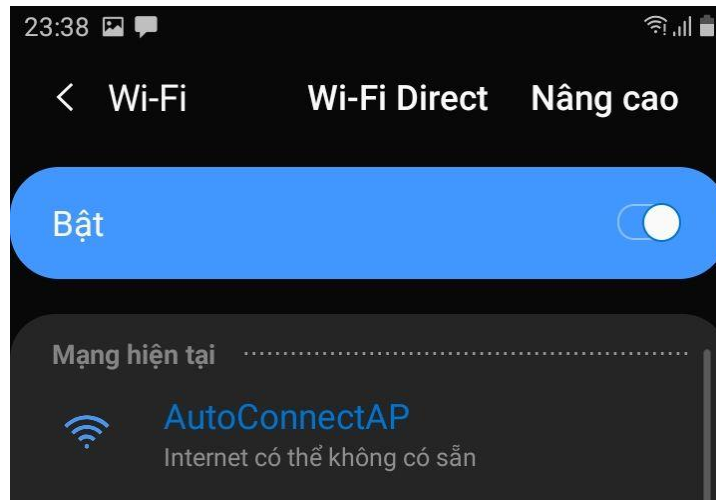
Hình 4. 1. Thiết kế PCB



Hình 4. 2. Phần cứng hệ thống

4.2. Hoạt động của hệ thống

Để có thể kết nối wifi mới, vào cài đặt và chọn kết nối vào wifi có tên là “AutoConnectAP”

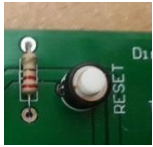


Hình 4. 3. Kết nối tới wifi hệ thống

Sau khi chọn, vào trình duyệt nhập địa chỉ 192.168.4.1 để hiển thị giao diện lựa chọn wifi để kết nối. Sau đó chọn Configure Wifi và thực hiện chọn 1 wifi để kết nối. Nhập đúng mật khẩu và đợi khi màn hình hiển thị lại. Sau khi kết nối thành công wifi “AutoConnectAP” cũng sẽ không còn và tự động thoát ra khỏi trang web chọn wifi.



Hình 4. 4. Thực hiện nhập mật khẩu

Để có thể chọn 1 wifi khác có thể nhấn nút Reset trên phần cứng , và thực hiện nhập mật khẩu wifi như các bước trên.



Nút SW trên phần cứng dùng để chuyển màn hình hiển thị thời gian và hiển thị giá trị các cảm biến.



2 nút RELAY1 và RELAY2 dùng để kích hoạt các thiết bị bơm và quạt. Các nút nhấn bật tắt các thiết bị cũng sẽ được đồng bộ với 2 nút nhấn



trong app Blynk và ngược lại.



Hình 4. 5. LCD hiển thị khi chưa kết nối

Hệ thống sử dụng thư viện RTC (Read Time Clock) để sử dụng internet cập nhập giờ trực tiếp theo múi giờ hiển thị lên LCD ở tác vụ thứ nhất được thực hiện ở core 1.



Hình 4. 6. LCD hiển thị ngày và giờ

Các giá trị của cảm biến được cập nhập và hiển thị lên LCD ở tác vụ 2 thực hiện ở core 2.



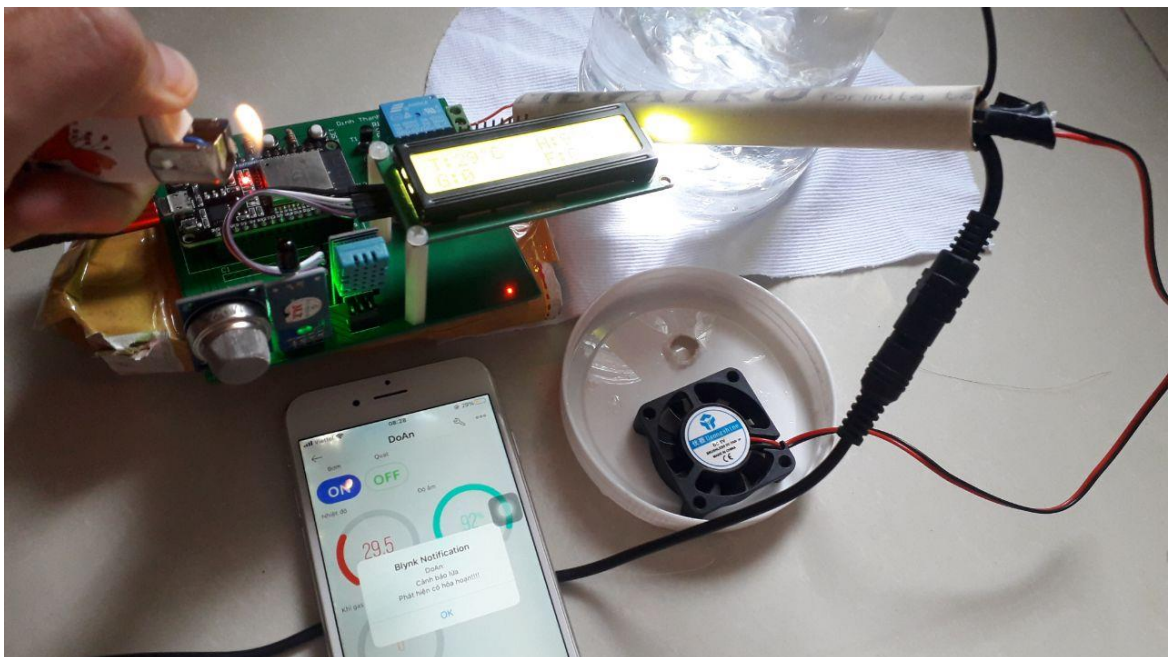
Hình 4. 7. LCD hiển thị giá trị các cảm biến

Khi hệ thống phát hiện khí gas, hệ thống sẽ gửi thông báo về điện thoại qua Blynk



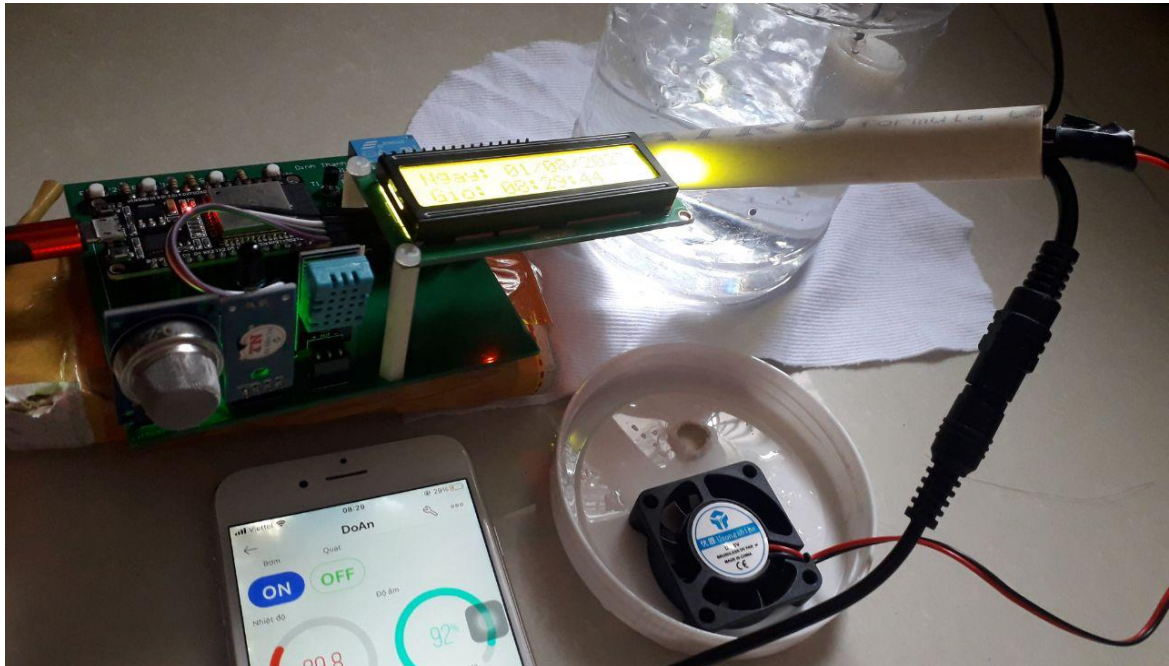
Hình 4. 8. Cảnh báo khi có khí gas rò rỉ

Khi hệ thống phát hiện có hỏa hoạn, hệ thống cũng sẽ gửi thông báo về điện thoại đồng thời kích hoạt máy bơm chữa cháy và máy bơm chỉ có thể tắt thủ công hoặc tắt từ xa qua Blynk để đảm bảo có thể dập tắt hỏa hoạn triệt để.



Hình 4. 9. Cảnh báo khi có hỏa hoạn

Máy bơm chữa cháy và quạt thông gió cũng có thể điều khiển thủ công qua điện thoại hoặc nút nhấn trên hệ thống. Khi sử dụng nút nhấn thì hệ thống cũng sẽ tự đồng bộ trạng thái giá trị của thiết bị với ứng dụng điện thoại.



Hình 4. 10. Điều khiển máy bơm chữa cháy



Chương 5: Kết luận và hướng phát triển

5.1. Kết luận

Sau quá trình nghiên cứu và thực hiện đề tài "Hệ thống cảnh báo cháy và rò rỉ khí gây cháy", nhóm nghiên cứu đã đạt được kết quả hiệu quả với mức độ hoạt động của mô hình dao động từ 60% đến 75%. Trong quá trình nghiên cứu và thực hiện đề tài, nhóm đã tích lũy được nhiều kiến thức mới và củng cố kiến thức đã có để hoàn thành đề tài này. Tuy nhiên, do đây là một đề tài tập trung vào việc giám sát từ xa kho, phân xưởng hoặc có thể sử dụng ở phạm vi nhỏ hơn là trong dân dụng cần phải chú trọng ưu tiên sự ổn định và chính xác cao dẫn đến nhiều khó khăn trong khi thiết kế hệ thống. Nhưng nhờ sự hướng dẫn từ giảng viên và tài liệu tham khảo, nhóm đã vượt qua được các yêu cầu của đề tài.

Mặc dù sản phẩm đã được hoàn thành, nhóm vẫn nhận thấy rằng còn nhiều thiếu sót và cần được điều chỉnh và cải tiến hơn. Nhóm nhận thức rằng việc nghiên cứu và phát triển một hệ thống giám sát về vấn đề môi trường nhiệt độ, độ ẩm, hay rò rỉ khí báo cháy là một quá trình liên tục và đòi hỏi sự phối hợp và phản hồi liên tục từ phía người dùng và chuyên gia trong lĩnh vực. Nhóm cam kết tiếp tục nỗ lực để nâng cao hiệu suất và chất lượng của sản phẩm trong tương lai.

Đề tài cơ bản đáp ứng được những yêu cầu đặt ra tuy nhiên để sản phẩm hoàn thiện được hơn nữa thì đòi hỏi cần được cải tiến và nghiên cứu thêm.

5.2. Hướng phát triển

Về chức năng, thiết bị được nghiên cứu chỉ dừng lại ở các chức năng cơ bản: đọc nồng độ khí gas, đọc cảm biến lửa, cảm biến nhiệt độ, độ ẩm hiển thị được ngay giờ song song với các giá trị nồng độ trên LCD điều khiển giám sát hoặc nhận cảnh báo thông qua ứng dụng Blynk IoT.

Một số chức năng nên bổ sung thêm là:

- Cảm biến chuyển động để cảnh báo người còn trong khu vực hoả hoạn.
- Phát triển mô hình với quy mô lớn hơn, cảm biến tốt hơn với đa dạng các loại khí gây cháy nổ.
- Hoàn thiện mô hình và tối ưu một cách kinh tế hơn
- Bổ sung để điều khiển thêm các thiết bị mở cửa tự động khi có hoả hoạn, servo để có thể đẩy các cửa thoát hiểm.

Tài liệu tham khảo

- [1] "ESP32 Series Datasheet 2021," 1 Overview, p. 8.
- [2] "ESP-IDF Get Started," [Online]. Available: docs.espressif.com. [Accessed 21 3 2023].
- [3] "“Third-Party Platforms That Support Espressif Hardware”," [Online]. Available: <http://espressif.com/en/support/download/sdk>. [Accessed 25 3 2023].
- [4] T. Mattison, "AWS IoT on Mongoose OS – Part 1," [Online]. Available: <https://aws.amazon.com/blogs/apn/aws-iot-on-mongoose-os-part-1/>. [Accessed 26 3 2023].
- [5] Google, "Google Cloud IoT Partners," [Online]. Available: <https://cloud.google.com/iot/partners/>. [Accessed 26 3 2023].
- [6] "ESP32 Series Datasheet 2021".
- [7] "Sơ đồ chân ESP32," [Online]. Available: <https://dientutuonglai.com/so-do-chan-esp32.html>. [Accessed 27 3 2023].
- [8] "Cảm Biến Khí Gas (LPG/CO/CH4) MQ-2," [Online]. Available: <https://nshopvn.com/product/cam-bien-khi-gas-mq-2/>. [Accessed 27 3 2023].
- [9] "MODULE CẢM BIẾN KHÍ GAS MQ2," [Online]. [Accessed 27 3 2023].
- [10] "Cảm Biến Phát Hiện Lửa (Flame Sensor)," [Online]. Available: <https://nshopvn.com/product/cam-bien-phat-hien-lua-flame-sensor/>. [Accessed 27 3 2023].
- [11] "Màn hình LCD 1602 Xanh Lá," [Online]. Available: <https://nshopvn.com/product/man-hinh-lcd-1602-xanh-la/>. [Accessed 27 3 2023].
- [12] "Relay là gì? Nguyên lý hoạt động của relay có thể bạn chưa biết," [Online]. Available: <https://thietbikythuat.com.vn/relay-la-gi-nguyen-ly-hoat-dong-cua-relay/>. [Accessed 5 25 2023].
- [13] "Cảm Biến Nhiệt Độ Và Độ Ẩm DHT11," [Online]. Available: <https://dientutuonglai.com/cam-bien-nhiet-do-va-do-am-dht11.html>. [Accessed 4 7 2023].

Phụ lục

```
#include <WiFi.h>
#include "time.h"
#include "sntp.h"
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <WiFiManager.h>
#include <Button2.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <EEPROM.h>
#include <BlynkSimpleEsp32.h> // Thêm thư viện Blynk
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL61-vlcK7W"
#define BLYNK_TEMPLATE_NAME "DoAn2"
#define BLYNK_AUTH_TOKEN "ZYkdM9D7GR6fcplqione0iFsbWopARjq"
boolean bt1_state = LOW;
boolean bt2_state = LOW;
boolean bt3_state = LOW;
#define DHT_PIN 15
#define DHT_TYPE DHT11
#define bt 4
#define r1 16
#define r2 17
#define MQ_PIN 34
#define FLAME_PIN 35
#define Relay1 33
#define Relay2 32
LiquidCrystal_I2C lcd(0x27, 16, 2);
DHT dht(DHT_PIN, DHT_TYPE);
TaskHandle_t Task1;
TaskHandle_t Task2;
SemaphoreHandle_t lcdSemaphore;
volatile bool switchPage = false;
volatile int currentPage = 1;
unsigned long lastDebounceTime = 0;

const char* ntpServer1 = "pool.ntp.org";
const char* ntpServer2 = "time.nist.gov";
const long gmtOffset_sec = 25200;
const int daylightOffset_sec = 0;

BlynkTimer timer;
```

```

WiFiManager wifiManager;
Button2 resetButton(bt);
byte customChar[] = { B00100, B01010, B00100, B00000, B00000, B00000, B00000,
B00000 };
void setup() {
    lcd.init();
    lcd.backlight();
    dht.begin();
    EEPROM.begin(512);
    lcd.setCursor(2, 0);
    lcd.print("DISCONNECT!!");
    Serial.begin(9600);
    lcd.createChar(0, customChar);
    delay(3000);
    resetButton.setLongClickHandler([](Button2& btn) {
        wifiManager.resetSettings();
        delay(1000);
        ESP.restart();
    });
    wifiManager.autoConnect("Hệ thống nhà kho");

    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
    }
    configTime(gmtOffset_sec, daylightOffset_sec, ntpServer1, ntpServer2);

    while (!time(nullptr)) {
        delay(1000);
    }

    lcdSemaphore = xSemaphoreCreateMutex();
    xTaskCreatePinnedToCore(task1, "Task1", 10000, NULL, 1, &Task1, 1);
    xTaskCreatePinnedToCore(task2, "Task2", 10000, NULL, 1, &Task2, 0);
    attachInterrupt(
        digitalPinToInterrupt(2),
        switchPageButtonInterrupt,
        RISING);
    Blynk.begin(BLYNK_AUTH_TOKEN, WiFi.SSID().c_str(), WiFi.psk().c_str());
    pinMode(Relay1, OUTPUT);
    pinMode(Relay2, OUTPUT);
    pinMode(r1, INPUT_PULLUP);
    pinMode(r2, INPUT_PULLUP);
    pinMode(FLAME_PIN, INPUT);
    timer.setInterval(100L, sendSensor);
}

BLYNK_WRITE(V0) {
    int p = param.asInt();
    digitalWrite(Relay1, p);
}

```

```

}
BLYNK_WRITE(V1) {
    int p = param.asInt();
    digitalWrite(Relay2, p);
}

void sendSensor() {
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();
    int mqValue = analogRead(MQ_PIN);
    int flameValue = digitalRead(FLAME_PIN);
    // You can send any value at any time.
    // Please don't send more that 10 values per second.
    Blynk.virtualWrite(V3, temperature);
    Blynk.virtualWrite(V4, humidity);
    Blynk.virtualWrite(V5, mqValue);
    if (mqValue > 500) {
        Blynk.logEvent("canhbaogas", "Phát hiện rò rỉ khí gas!!!!");
    }
    if (flameValue == 0) {
        Blynk.logEvent("canhbaolua", "Phát hiện có hỏa hoạn!!!!");
    }
}

void loop() {
    resetButton.loop();
}

void check_button() {
    if (digitalRead(r1) == HIGH) {
        if (bt1_state == LOW) {
            digitalWrite(Relay1, !digitalRead(Relay1));
            Blynk.virtualWrite(V0, digitalRead(Relay1));
            bt1_state = HIGH;
            delay(200);
        }
    } else {
        bt1_state = LOW;
    }
    if (digitalRead(r2) == HIGH) {
        if (bt2_state == LOW) {
            digitalWrite(Relay2, !digitalRead(Relay2));
            Blynk.virtualWrite(V1, digitalRead(Relay2));
            bt2_state = HIGH;
            delay(200);
        }
    } else {
        bt2_state = LOW;
    }
}
}

```

```

void task1(void* parameter) {
    for (;;) {
        check_button();
        if (currentPage == 1) {
            display1();
        }
        delay(1000);
    }
}

void task2(void* parameter) {
    for (;;) {
        if (currentPage == 2) {
            display2();
        }
        Blynk.run();
        timer.run();
        delay(1000);
    }
}

void switchPageButtonInterrupt() {
    unsigned long currentMillis = millis();
    if (currentMillis - lastDebounceTime > 200) {
        switchPage = true;
    }
    lastDebounceTime = currentMillis;

    currentPage = (currentPage == 1) ? 2 : 1;
}

void display1() {
    time_t now = time(nullptr);
    struct tm* timeinfo;
    timeinfo = localtime(&now);
    xSemaphoreTake(lcdSemaphore, portMAX_DELAY);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Ngày: ");
    lcd.print(getFormattedDate(timeinfo));

    lcd.setCursor(0, 1);
    lcd.print("Gio: ");
    lcd.print(getFormattedTime(timeinfo));

    xSemaphoreGive(lcdSemaphore);
}

void display2() {
    int temperature = dht.readTemperature();
    int humidity = dht.readHumidity();
    int mqValue = analogRead(MQ_PIN);

```

```

int flameValue = digitalRead(FLAME_PIN);
xSemaphoreTake(lcdSemaphore, portMAX_DELAY);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("T:");
lcd.print(temperature);
lcd.write(0);
lcd.print("C");

lcd.setCursor(9, 0);
lcd.print("H:");
lcd.print(humidity);
lcd.print("%");

lcd.setCursor(0, 1);
lcd.print("G:");
lcd.print(mqValue);

lcd.setCursor(9, 1);
lcd.print("F:");
lcd.print(flameValue);

xSemaphoreGive(lcdSemaphore);
}
String getFormattedDate(struct tm* timeinfo) {
    char buffer[20];
    sprintf(buffer, "%02d/%02d/%04d",
            timeinfo->tm_mday, timeinfo->tm_mon + 1, timeinfo->tm_year + 1900);
    return String(buffer);
}
String getFormattedTime(struct tm* timeinfo) {
    char buffer[20];
    sprintf(buffer, "%02d:%02d:%02d",
            timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);

    return String(buffer);
}

```