BÙI QUỐC TUẤN

NGÀNH CNKT ĐIỆN TVIỄTHÔNG THÔNG

BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC NGÀNH CNKT ĐIỆN TỬ VIỄN THÔNG

THIẾT KẾ HỆ THỐNG ỨNG DỤNG IoT ĐIỀU KHIỂN CÁC THIẾT BỊ TRONG NHÀ MÁY SỬ DỤNG ESP32

CBHD: TS. Phạm Xuân Thành

Sinh viên: Bùi Quốc Tuấn

Mã số sinh viên: 2018605300

Hà Nội - 2022

LỜI CẢM ƠN

Lời đầu tiên cho em xin trân thành cảm ơn TS. Phạm Xuân Thành, người đã hết lòng chỉ dẫn, truyền đạt những kiến thức chuyên môn cũng như những kinh nghiệm liên quan cho chúng em trong suốt quá trình thực hiện đồ án này.

Xin chân thành cảm ơn đến tất cả quý thầy, cô nhà trường nói chung và các thầy cô bộ môn khoa điện tử nói riêng của trường Đại Học Công Nghiệp Hà Nội đã giảng dạy, trang bị cho em những kiến thức rất bổ ích và quý báu trong suốt quá trình học tập để em có thể áp dụng nghiên cứu hoàn thành đề tài này.

Trong quá trình thực hiện đồ án, với điều kiện thời gian cũng như kinh nghiệm, kiến thức còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được những ý kiến đóng góp của thầy, cô và các bạn để em có thêm được nhiều kinh nghiệm vào công việc trong tương lai.

Cuối cùng, chúng em xin chân thành cảm ơn gia đình và bạn bè, đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên chúng em trong suốt quá trình học tập và hoàn thành đồ án này.

Em xin chân thành cảm ơn!

Hà Nội, ngày…tháng…năm 2022.

Sinh viên thực hiện đề tài

Bùi Quốc Tuấn

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC TỪ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
DC	Direct Currentic	Dòng điện một chiều
I2C	Inter-Integrated Circuit	Mạch tích hợp liên
IC	Integrated Circuit	Mạch tích hợp
LCD	Liquid Crystal Display	Màn hình tinh thể lỏng
SCL	Serial Clock Line	Dòng đồng hồ nối tiếp
SDA	Serial Data Line	Dòng dữ liệu nối tiếp
VDD	Voltage Drain Drain	Xả xả điện áp
VSS	Voltage for Substrate and Source	Điện áp cho chất nền và nguồn
GPIO	Generral– Purpose Input/Output	Chung - Đầu vào / Đầu ra Mục đích
SRAM	Static random-access memory	Bộ nhớ truy cập ngẫu nhiên tĩnh
MCU	Micro Controller Unit	Bộ điều khiển vi mô

LỜI MỞ ĐẦU

Lý do lựa chọn đề tài

Cuộc cách mạng công nghiệp 4.0 đang bùng nổ và tập trung vào công nghệ kỹ thuật số từ những thập kỷ gần đây lên một cấp độ hoàn toàn mới với sự trợ giúp của kết nối thông qua Internet vạn vật, truy cập dữ liệu thời gian thực và giới thiệu các hệ thống vật lý không gian mạng. Nó cung cấp một cách tiếp cận liên kết và toàn diện hơn cho sản xuất, đồng thời giúp kết nối vật lý với kỹ thuật số và cho phép cộng tác và truy cập tốt hơn giữa các bộ phận, đối tác, nhà cung cấp, sản phẩm và con người. Điều này cho phép các nhà máy thông minh, sản phẩm thông minh và chuỗi cung ứng cũng thông minh, và làm cho các hệ thống sản xuất và dịch vụ trở nên linh hoạt và đáp ứng khách hàng hơn. Hiện nay hầu hết các sản phẩm điện tử điều có tích hợp việc điều khiển từ xa điển hình là dùng hồng ngoại. Tuy nhiên, các sản phẩm điện tử hiện này đều hướng tới việc tích hợp IoT với giao diện điều khiển thân thiện, dễ sử dụng.

Chính vì lý do trên, đề tài em đưa ra tập trung vào lĩnh vực IOT với mục đích tìm hiểu và nghiên cứu về các ứng dụng IOT, giúp phát triển một số ứng dụng IOT trong cuộc sống hàng ngày và trong công nghiệp.

Với đề **"Thiết kế hệ thống ứng dụng IOT điều khiển các thiết bị trong nhà máy sử dụng esp32"** báo cáo của em gồm các phần sau:

Chương 1: Tổng quan về hệ thống.

Chương 2: Thiết kế mô hình.

Chương 3: Kết quả thực nghiệm.

Mục đích đối tượng phạm vi nghiên cứu

Mục đích nghiên cứu là điều khiển và giám sát các thiết bị thông qua. Từ các giá trị đó, người vận hành hay thao tác có thể đưa ra được các trường hợp, phân tích ra được các nguyên nhân lỗi.

Đối tượng nghiên bao gồm: module ESP32, giao diện điều khiển hệ thống qua internet.

Phạm vi nghiên cứu: Sử dụng kiến thức đã học, nghiên cứu thiết kế hệ thống đơn giản nhằm đáp ứng đủ yêu cầu của hệ thống, thiết kế mô hình mini mô phỏng quá trình hoạt động của hệ thống. Trong đề tài này, em sẽ thực hiện thiết kế một hệ thống IoT đơn giản bao gồm 2 phần chính:

- Phần cứng có chứa vi điều khiển ESP32 để đưa ra các tín hiệu điều khiển một thiết bị.

- Một trang web để điều khiển và hiển thị trạng thái bật/tắt của bóng đèn bất cứ khi nào chỉ cần người dùng có điện thoại hoặc máy tính có kết nối với internet.

Ý nghĩa khoa học và thực tiễn của đề tài

IoT ngành công nghiệp là sự kết hợp các máy móc thiết bị, công nghệ lưu trữ đám mây, con người lại với nhau để tối ưu quy trình sản xuất, tạo ra năng suất hiệu và hiệu suất cho ngành công nghiệp. Với một mô hình công nghiệp lớn, sẽ có hàng nghìn các thiết bị cảm biến được kết nối với máy móc, robot để xác định nhanh chóng các sai sót, lỗ hổng và những nguy cơ tìm ẩn để không làm quy trình sản xuất bị gián đoạn. Nhờ đó tối ưu hóa hoạt động máy móc thiết bị của các cơ sở sản xuất công nghiệp.

Đề tài tập trung nghiên cứu điều khiển và giám sát trạng thái hoạt động của các thiết bị.Thiết kế mô hình ứng dụng hệ thống Iot, nhằm hiểu rõ được cách một hệ thống iot vận hành trong các nhà mày, đồng thời tìm hiểu thêm được nhiều ứng dụng mà iot có thể thực hiện, sức ảnh hưởng và tâm quan trong đối với đời sống , kinh tế và xã hội.

CHƯƠNG 1:TỔNG QUAN VỀ HỆ THỐNG ỨNG DỤNG IoT ĐIỀU KHIỂN CÁC THIẾT BỊ TRONG NHÀ MÁY SỬ DỤNG ESP32

1.1 Đặt vấn đề nghiên cứu

Internet vạn vật, hay còn được gọi là IoT trong những năm gần đây đã phát triển và được ứng dụng rất nhiều trong cuộc sống hàng ngày của chúng ta. Công nghệ này mang tới cho con người sự tiện lợi, tiện nghi khi sử dụng các sản phẩm về điện tử. Trong công nghiệp, IoT mang lại sự dễ dàng trong điều khiển và giám sát các thiết bị, máy móc và quy trình tự động.

Thuật ngữ IoT đề cập đến những thứ mà chúng ta sử dụng hàng ngày cũng kết nối với internet, cho phép chúng ta kiểm soát hoặc nhận dữ liệu về từ điện thoại thông minh hoặc máy tính của mình.

Với sự phát triển của xã hội, khoa học kĩ thuật nói chung, việc tích hợp IoT vào vi điều khiển ngày càng được ứng dụng ở hầu hết các lĩnh vực. Trước thực tiễn ấy, em đã quyết định chọn đề tài **"Thiết kế hệ thống ứng dụng IOT điều khiển các thiết bị trong nhà máy sử dụng esp32"** nhằm tìm hiểu về vấn đề ứng dụng vi điều khiển ESP32 vào trong IoT, ứng dụng IoT vào trong công nghiệp.

1.2 Giới thiệu chung về đề tài

Trong đề tài này, em sẽ thực hiện thiết kế một mô hình điều khiển, giám sát từ xa bằng WiFi sử dụng ESP32. Mô hình gồm 2 phần chính là mạch điện tử và phần mềm điều khiển. Đầu vào của mạch điện tử có 6 nút nhấn để điều khiển đầu ra gồm 6 rơ le. Phần mềm sẽ đồng bộ với nút nhấn trên phần cứng để hiển thị trạng thái bật tắt của thiết bị, đồng thời chúng ta có thể bật tắt thiết bị ngay trên phần mềm.

1.3 Tổng quan về kết nối Iot trong nhà máy.

Công nghiệp 3.0 với việc sử dụng những bộ điều khiển logic và công nghệ thông tin đã giúp ngành công nghiệp tạo ra những cỗ máy tự động thông minh, hiệu quả có thể thay thế con người trong nhiều lĩnh vực. Tuy nhiên,

những cỗ máy tự động hóa ưu việt đó vẫn là những đối tượng độc lập. Vẫn có những khía cạnh mà con người vẫn còn hiện diện. VD: Vận hành và giám sát thiết bị, cung cấp những thông tin đầu vào, thực hiện bảo trì bảo dưỡng…

Tiếp nói thành tựu của công nghiệp 3.0 là sự kế tiếp của 4.0, nó tạo ra các nền tảng mà trên đó các thiết bị được kết nối, định vị và tương tác với nhau. Nền tảng IOT là một không gian có thể không giới hạn mà các đối tượng liên kết. Có rất nhiều các IoT platform khác nhau, tuy nhiên hầu hết tất cả đều có các thành phần cơ bản chung giống nhau:

- **Thiết bị kết nối:** Chúng là các loại máy móc, cảm biến hay các thiết bị kết nối khác thực hiện một hành động cụ thể: thu thập dữ liệu, kết nối với nhau, truyền và nhận dữ liệu, ...
- **Phương thức kết nối:** Dựa trên mạng viễn thông mà các thiết bị có thể kết nối, giao tiếp được với nhau và với server/cloud. Điều này phụ thuộc vào yêu cầu của dự án IoT từ đó chọn ra phương thức kết nối hiệu quả nhất.
- **Xử lý dữ liệu:** Được xử lý ở trên server/cloud. Nhận dữ liệu từ các thiết bị, từ đó phân tích và đưa ra hành động sẽ được thực hiện trong IoT platform.
- **Giao diện:** Cung cấp cho người dùng một giao diện trực quan để có thể tương tác và nhìn thấy được hoạt động của toàn bộ hệ thống.

Các IoT platform đảm bảo việc tích hợp liền mạch các phần cứng khác nhau bằng cách sử dụng một loạt các giao thức giao tiếp phổ biến (như MQTT, HTTP, CoAP, …). Sử dụng các API do IoT platform cung cấp, ta có thể tải dữ liệu IoT thu thập được vào các hệ thống phân tích, lưu trữ hoặc xử lý dữ liệu tới các thiết bị được kết nối hoặc truyền dữ liệu giữa chúng bằng việc sử dụng các loại ứng dụng người dùng khác nhau. Có thể nói IOT là một không gian kết nối vạn vật trong thế giới vật lý, và ứng dụng của nó trên toàn bộ các mặt khía

cạnh của cuộc sống. Tuy nhiên trong bài viết này chúng ta cùng xem một vài ví dụ những lợi ích của IoT trong lĩnh vực công nghiệp IIoT(Industrial Internet of Things) [1].

Kiểm soát và cấu hình các thiết bị từ xa

Hình 1.1. Kiểm soát và cấu hình các thiết bị từ xa

Không gian IOT được tạo ra, nơi mà các thiết bị & thành phần (máy móc, cảm biến, máy tính, CSDL…) trong chuỗi sản xuất được kết nối. Dữ liệu của các thành phần này được cập nhật Realtime lên đám mây.

Các ứng dụng người dùng được thiết kế có thể cập nhật dữ liệu để mô phỏng toàn bộ tiến trình theo thời gian thực. Như vậy toàn bộ các hoạt động diễn ra trong nhà máy được giám sát ở bất cứ nơi đâu mà không cần thiết phải có mặt trong nhà máy.

Và khi có yêu cầu thay đổi thì thông qua ứng dụng và đám mây, người sử dụng lại có thể tương tác ngược lại toàn bộ thành phần nhà máy cái mà đã được kết nối và định vị trong không gian (Dừng dây chuyền A, vận hành dây chuyền B…vv).

Đưa ra thông tin nhanh chóng và chính xác.

Khi tất cả các thành phần trong chuỗi sản xuất được kết nối, thì các dữ liệu được đồng bộ và liên kết với nhau. Đầu ra của thành phần này chính là đầu vào của thành phần kia, dữ liệu được trao đổi liền mạch, chính xác và tức thì.

Xử lý sự cố.

Trong các máy móc hoạt động trong nhà máy, phát sinh sự cố là điều không thể tránh khỏi, sự cố có thể đến từ nhiều nguyên nhân, có thể do chủ quan. Từ phía bản thân các thiết bị trong dây chuyền. Hoặc sự cố khách quan đến từ phía người sử dụng vận hành dây chuyền đó. Vấn đề đối với đội ngũ kỹ thuật là làm sao để phát hiện sự cố và xử lý kịp thời để tránh làm giám đoạn sản xuất.

Bảo trì dự đoán.

Nội dung phía trên liên quan tới việc xử lý sự cố đối với máy móc. Tuy nhiên việc ngăn chặn sự cố xảy ra, việc bảo trì bảo dưỡng thiết bị là hoạt động bắt buộc trong mỗi nhà máy. Thông thường bảo dưỡng thường diễn ra định kỳ đối với từng loại hạng mục.

Công nghệ 4.0 đề cập đến rất nhiều dữ liệu có thể được thu thập thông qua các cảm biến và được sử dụng để đưa ra quyết định vận hành, sửa chữa, bảo trì và tiến đến sự thông minh hóa hệ thống. IOT cùng điện toán đám mây giúp ngành tự động hóa công nghiệp đạt đến tầm cao mới hơn cho phép chúng ta thu thập và phân tích dữ liệu theo những cách không thể có trước đây và các công nghệ đó được thể hiện ở mọi khía cạnh của sản xuất. Từ sản xuất, đến bảo trì, đến tiếp thị, và thậm trí sau đó đến những sản phẩm cuối cùng mà chúng ta tạo ra. Tuy nhiên sự kết hợp công nghệ này đòi hỏi cơ sở hạ tầng mới, bao gồm những ứng dụng phần cứng và phần mền, cũng như một hệ điều hành. Các nhà sản xuất sẽ phải đối phó với dòng dữ liệu lớn bắt đầu chảy vào và phân tích nó theo thời gian thực khi nó phát triển theo thời gian [2].

1.4 Kết luận chương 1

Tìm hiểu tình hình nghiên cứu trong và ngoài nước của hệ thống diều khiển thiết bị trong nhà máy. Nắm được tầm quan trọng của hệ thống, ứng dụng cũng như ảnh hưởng của hệ thống đối với sự phát triển công nghiệp của nước ta. Từ đó đưa ra ý tưởng thiết kế hệ thống, các bước để hoàn thành đề tài. Tiếp đó mô tả trực quan về hệ thống, sơ đồ khối giúp người đọc hình dung ra được nguyên lý và những việc phải làm khi thiết kế mạch và lập trình cho hệ thống.

CHƯƠNG 2:THIẾT KẾ MÔ HÌNH ỨNG DỤNG IoT ĐIỀU KHIỂN CÁC THIẾT BỊ TRONG NHÀ MÁY SỬ DỤNG ESP32

2.1 `Thiết kế sơ đồ khối của mô hình

2.1.1 Yêu cầu đặt ra.

Bài toán:

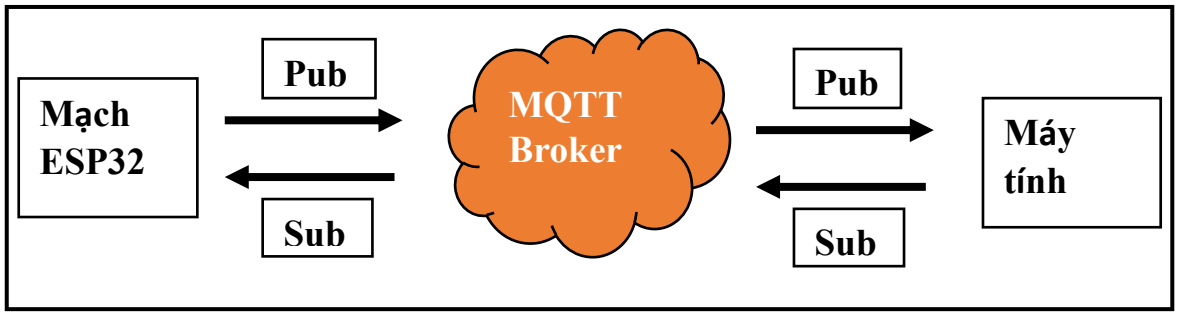
Đề tài tập trung nghiên cứu, thiết kế mạch và chế tạo hệ thống điều khiển thiết bị thông qua Internet.

Hệ thống có 2 chế độ: Chế độ bằng tay và chế độ tự động.

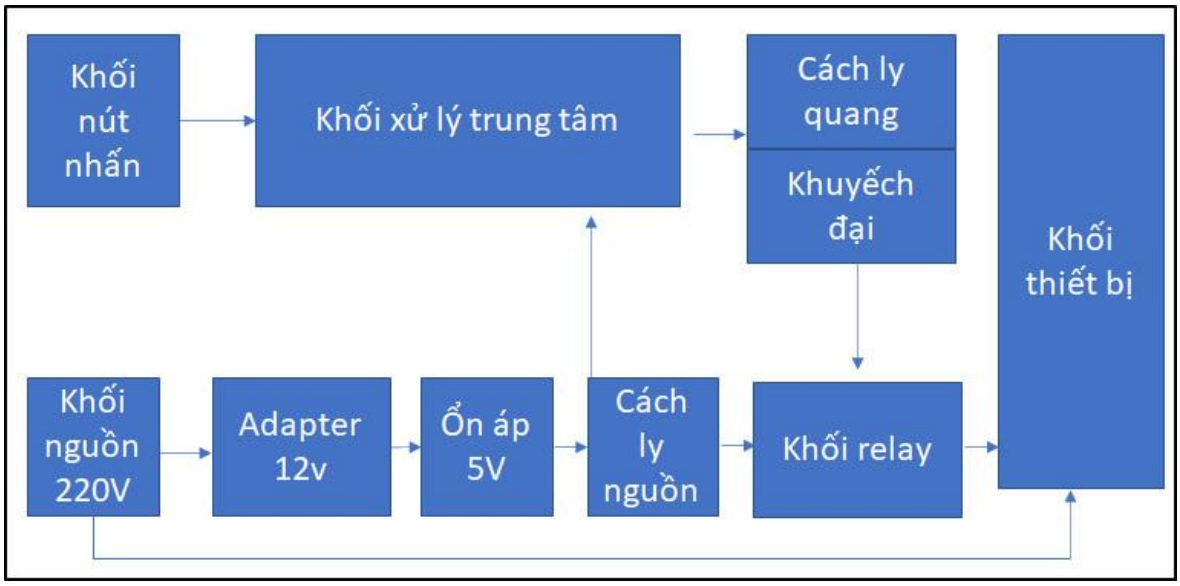
- Chế độ bằng tay: Sử dụng các nút nhấn bật/ tắt thiết bị.
- Chế độ tự động: Thiết kế giao diện trên Web điều khiển thông qua mạng wifi được kết nối với hệ thống.

2.1.2 Sơ đồ khối của mô hình

Từ yêu cầu bài toán, thiết kế sơ đồ khối của hệ thống đảm bảo chức năng cơ bản, điều khiển thiết bị thông qua hệ thống IOT, giám sát được trạng thái của các thiết bị thông qua giao diện Web.



Hình 2.1. Sơ đồ nhận dữ liệu giữa ESP32 và điện thoại/máy tính



Hình 2.2. Sơ đồ khối của hệ thống

Mạch điều khiển các thiết bị bao gồm 8 khối chính:

- **Khối nút nhấn:** Bao gồm 6 nút nhấn để bật tắt 6 thiết bị tương ứng.
- **Khối xử lý trung tâm:** Chứa vi điều khiển ESP32, xử lý các tín hiệu gửi đi, nhận về.
- **Khối nguồn và ổn áp 5V:** Mạch được cấp nguồn từ 5 tới 12V, được ổn áp qua mạch ổn áp có IC LM7805.
- **Khối cách ly nguồn:** Cách ly độc lập thành 2 nguồn riêng biệt không chung GND, tránh tất cả mọi nhiễu từ nguồn điều khiển phía relay.
- **Khối cách ly quang:** Cách ly tín hiệu của mạch điều khiển với relay, tránh bị nhiễu từ relay.
- **Khối relay:** điều khiển thiết bị 1 chiều hoặc xoay chiều thông qua tiếp điểm.
- **Khối thiết bị:** nhận tín hiệu từ relay và điều khiển thiết bị.

2.2 Sơ đồ nguyên lý của hệ thống

2.2.1 Khối nguồn và ổn áp nguồn

a) Khối nguồn

Hình 2.3. Nguồn Adapter 12V

Mạch sử dụng 12V-2A Delta chuyên dụng.

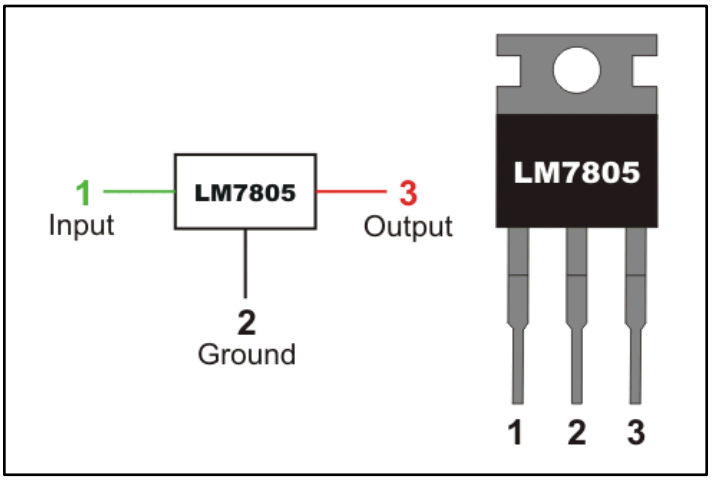
Thông số kỹ thuật [8]:

Bảng 2.1 Thông số kỹ thuật nguồn Adapter 12V

Điện áp đầu vào	100-240V-0.5A
Tần số hoạt động	12V-2A
Model	ADP-24ZB
Kích thước jack đầu ra	5.5 x 2.1mm
Chiều dài dây cắm từ ổ điện tới Adapter	44 cm
Chiều dài dây từ Adapter tới jack 12V	48 cm
Chiều dài sợi dây nối dài	1.5m

b) IC LM7805 [9]

LM 7805 là IC điều chỉnh điện áp dương đầu ra 5V. Nó là IC của dòng ổn áp dương LM78xx, được sản xuất và đóng gói dạng TO-220 và các gói khác. IC này được sử dụng rộng rãi trong các thiết bị thương mại và giáo dục. Nó cũng được sử dụng bởi cho nhiều mạch điện tử thông dụng do giá rẻ, dễ sử dụng và không cần nhiều linh kiện bên ngoài. IC có nhiều tính năng tích hợp lý tưởng để sử dụng trong nhiều ứng dụng điện tử như dòng điện đầu ra 1.5A, chức năng bảo vệ quá tải, bảo vệ quá nhiệt, dòng điện tĩnh thấp…

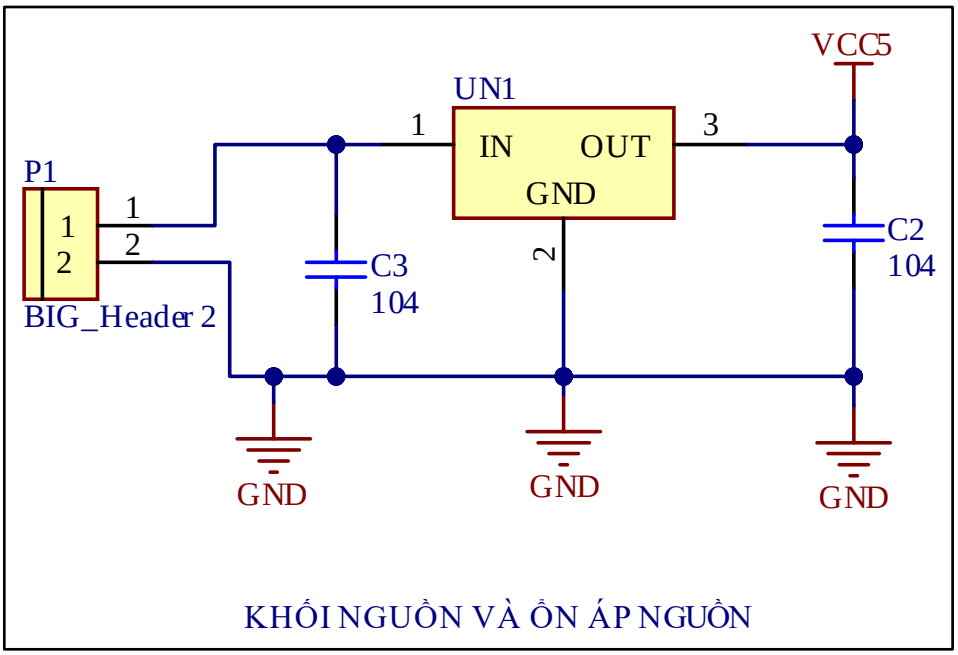


Hình 2.4. Sơ đồ chân LM7805.

Bảng 2.2 Thông số kỹ thuật IC LM7805

Điện áp vào lớn nhất	20V
Điện áp vào nhỏ nhất	7V
Kiểu đóng vỏ	TO-220
Nhiệt độ hoạt Dòng đầu ra động lớn nhất	85°C
Nhiệt độ hoạt động nhỏ nhất	-20°C
Dòng đầu ra	1. 5V 5A
Điện áp ổn định	5V

Mạch ổn áp sử dụng IC LM7805

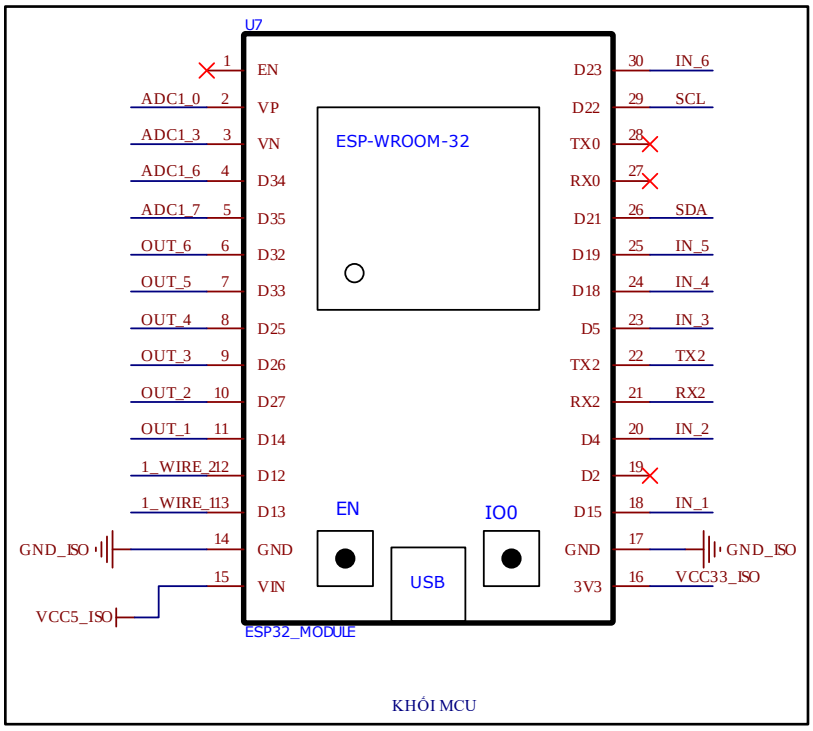


Hình 2.5. Mạch ổn áp sử dụng IC LM7805

- Điện áp đầu vào được cung cấp qua 1 Header 2 chân qua và điện áp 5V ở ngõ ra sẽ được lấy ở VCC5.
- Tụ C3 và C2 để lọc điện áp cấp cho tải tiêu thụ lấy từ chân Vo của IC 7805, tụ C2 có các dụng cung cấp điện áp tạm thời cho tải khi điện áp tải đột ngột bị sụt áp, tụ C3 trở kháng lớn, C2 có tác dụng lọc nhiễu điện áp đầu ra (nhiễu là các điện áp không mong muốn làm cho dạng sóng điện áp ngõ ra có hình răng cưa).
- IC 7805 dễ toả nhiệt nên để mạch hoạt động ổn định và lâu dài, chúng ta nên gắn thêm tản nhiệt cho IC.
- Mạch ổn áp này phù hợp để cấp nguồn cho các mạch điện tử vận hành với điện áp 5V và dòng điện 1A.

2.2.2 Khối xử lý trung tâm

Module ESP32 Dev Kit



Hình 2.6. Mạch nguyên lý khối xử lý trung tâm

Các thông số kỹ thuật [3]:

- Module ESP32 có lõi kép, điều này có nghĩa là nó có 2 bộ vi xử lý.
- Được tích hợp Wi-Fi và bluetooth.
- Nó chạy các chương trình 32 bit.
- Xung nhịp (clock frequency) có thể lên đến 240MHz và nó có RAM 512 kB.
- Loại module này có 30 hoặc 36 chân, mỗi hàng có 15 chân.
- Nó cũng có sẵn nhiều loại ngoại vi như: cảm ứng điện dung (capacitive touch), ADC, DAC, UART, SPI, I2C và nhiều hơn nữa.
- Nó được tích hợp với cảm biến hiệu ứng Hall (Hall effect sensor) và cảm biến nhiệt độ.

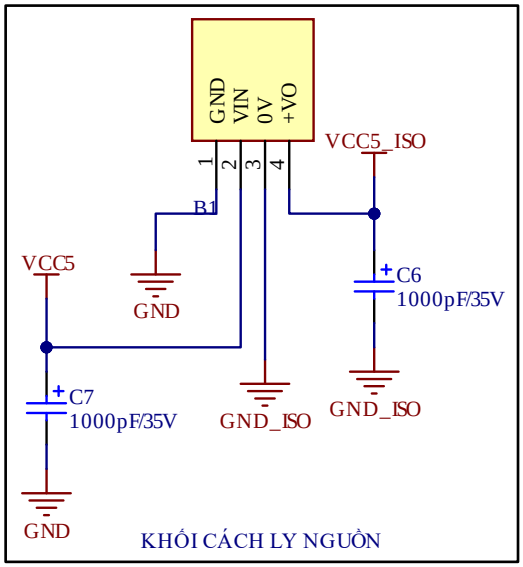
Môi trường lập trình:

Module ESP32 có thể được lập trình trên nhiều môi trường lập trình khác nhau như:

- Arduino IDE.
- Espressif IDF (IoT Development Framework).
- MicroPython.

2.2.3 Khối cách ly nguồn

IC cách ly nguồn B0505s



Hình 2.7. Mạch nguyên lý khối cách ly nguồn

IC cách ly dùng cho các mạch nhạy cảm cần cách ly như mạch điều khiển, mạch âm thanh, mạch thu phát tín hiệu...

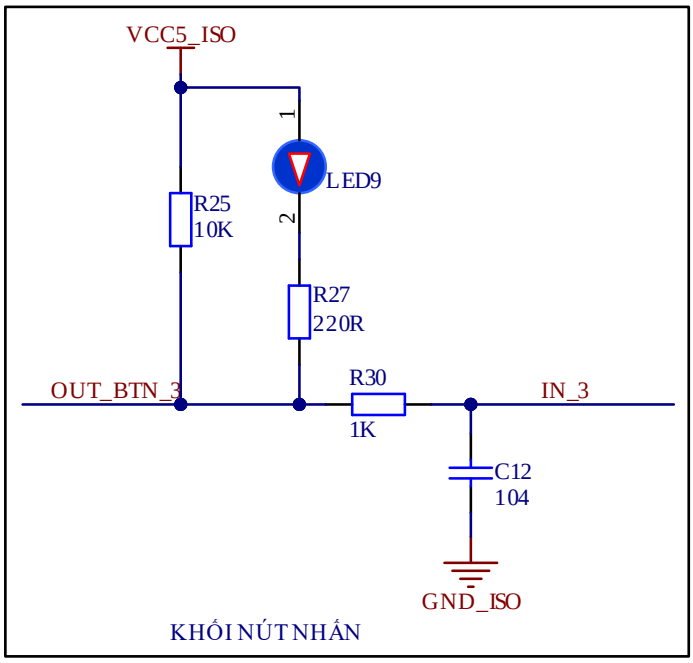
Bảng 2.3 Thông số kỹ thuật khối cách ly nguồn [4]

Hiệu suất của mạch	80%
Điện áp vào	5VDC (4.5~5.5V)
Điện áp ra	5VDC
Công suất ra	2W
Dải nhiệt độ	(-40°C to +105°C)
Mạch cách ly hoàn toàn cả GND và VCC	

IC được sử dụng để cách ly nguồn là B0505s. Các tụ C3 và C4 cũng là các tụ lọc đầu vào và đầu ra của IC B0505S.

Giá trị của tụ được chọn theo datasheet của nhà sản xuất yêu cầu.

2.2.4 Khối nút nhấn



Hình 2.8. Mạch nguyên lý khối nút nhấn

Khối nút nhấn được gắn một điện trở kéo lên có giá trị 10K, ngoài ra còn có các Led 3mm báo tín hiệu. Ngoài ra mạch còn được trang bị thêm mạch lọc thông thấp RC nhằm tránh việc nút nhấn bị nhiễu khi nhấn.

Sử dụng nút nhấn DS-316 để điều khiển các thiết bị trong hệ thống [5].

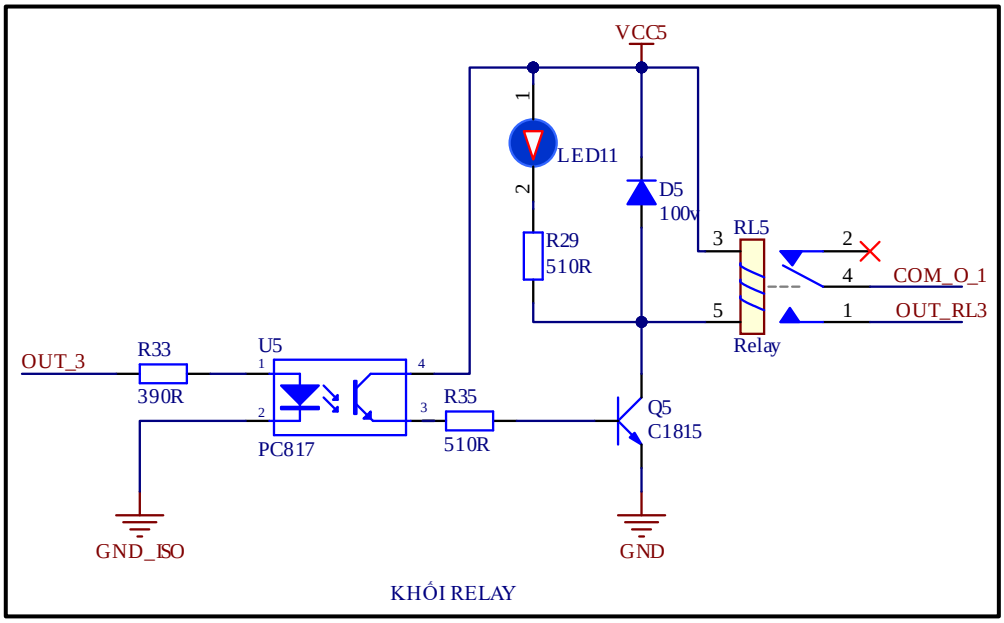
Thông số kỹ thuật:

Bảng 2.4 Thông số kỹ thuật DS-316

Hoạt động	Nhấn nhả / Nhấn giữ
Đường kính	10mm
Màu núm chỉnh	Đỏ / Xanh
Điện áp định mức	250 V AC
Dòng định mức	0.5A

Hình 2.9. Nút nhấn DS-316

2.2.5 Khối relay [6]



Hình 2.10. Mạch nguyên lý khối Relay

Khối Relay được điều khiển bởi các chân GPIO của vi điều khiển ESP32. Mạch sử dụng 6 relay để điều khiển các phụ tải.

Khối Relay được cách ly với vi điều khiển qua Opto quang PC817, đồng thời nguồn điều khiển cuộn hút Relay được lấy trực tiếp từ khối nguồn, trong khi đó nguồn của vi điều khiển được lấy từ khối cách ly nguồn.

Mục đích dùng Opto quang kết hợp với dùng cách ly nguồn nhằm đảm bảo chống nhiễu tối đa và an toàn cho mạch điều khiển khi có sự cố từ thiết bị mà Relay điều khiển.Ngoài ra, mạch sử dụng thêm các diode (1N4148) mắc song song và mắc ngược với cuộn hút của Relay. Mục đích của việc làm này nhằm ngăn chặn sự tăng đột biến điện áp lớn phát sinh khi nguồn điện bị ngắt (gọi là điện áp ngược) bảo vệ cho Relay và mạch điều khiển.

Do tín hiệu lấy ra từ vi điều khiển và lấy ra từ Opto quang không đủ, mạch sử dụng thêm các mạch khuyếch đại với transistor C1815 mắc phân cực cố định nhằm tăng dòng điều khiển cho cuộn hút của Relay.

Hình 2.11. Relay 5V 10A

Ứng dụng:

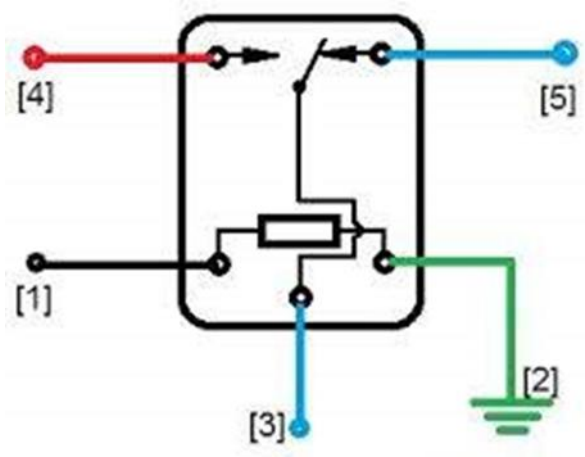
- Nhìn chung, công dụng của relay là "dùng một năng lượng nhỏ để đóng cắt nguồn năng lượng lớn hơn".
- Relay được dùng khá thông dụng trong các ứng dụng điều khiển động cơ và chiếu sáng.
- Khi cần đóng cắt nguồn năng lượng lớn, relay thường được ghép nối tiếp. Nghĩa là một relay nhỏ điều khiển một relay lớn hơn, và relay lớn sẽ điều khiển nguồn công suất.

Bảng 2.5 Thông số kỹ thuật Relay 5V 10A

Dòng AC max	10 A
Dòng AC min	6A
Diameter, PCB hole	1.3 mm
Length / Height, external	22 mm
Material, contact	Silver alloy
Nhiệt độ hoạt động	- 45 °C to 75 °C

Thời gian tác động	10 ms
Thời gian nhả hãm	5 ms
Điện áp điều khiển cuộn dây	5 V

Sơ đồ chân:

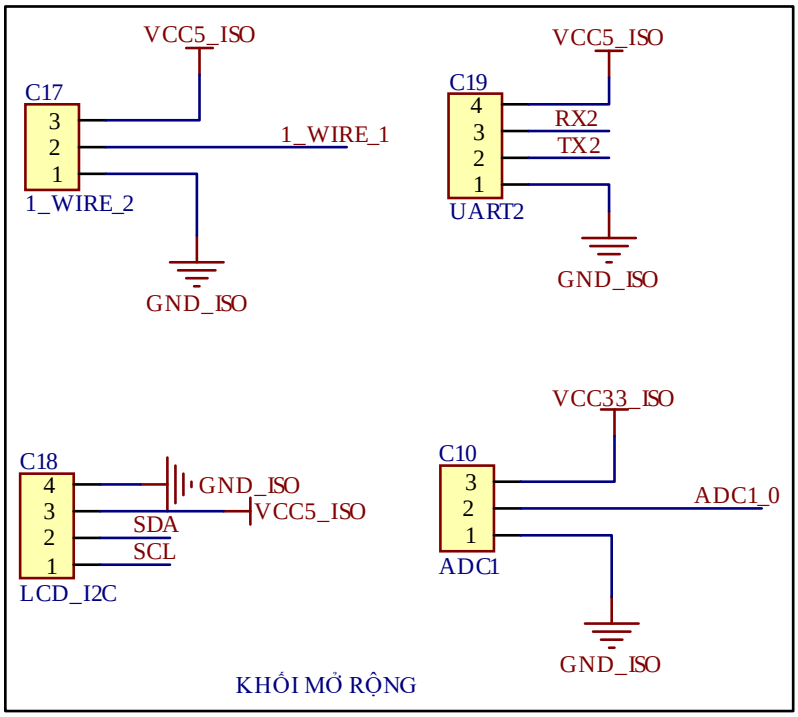


Hình 2.12. Sơ đồ chân Relay

- Chân 1 và chân 2 được nối vào cuộn hút, khi có điện vào cuộn hút sẽ hút tiếp điểm chuyển từ vị trí 4 xuống tiếp điểm 5.
- Chân 3: đặt điện áp (nếu là loại Relay 5V thì đặt 5V DC vào đây).
- Chân 4, chân 5: tiếp điểm.

2.2.6 Khối mở rộng

Mạch được thiết kế để có thể mở rộng thêm một số ngoại vi như sau:



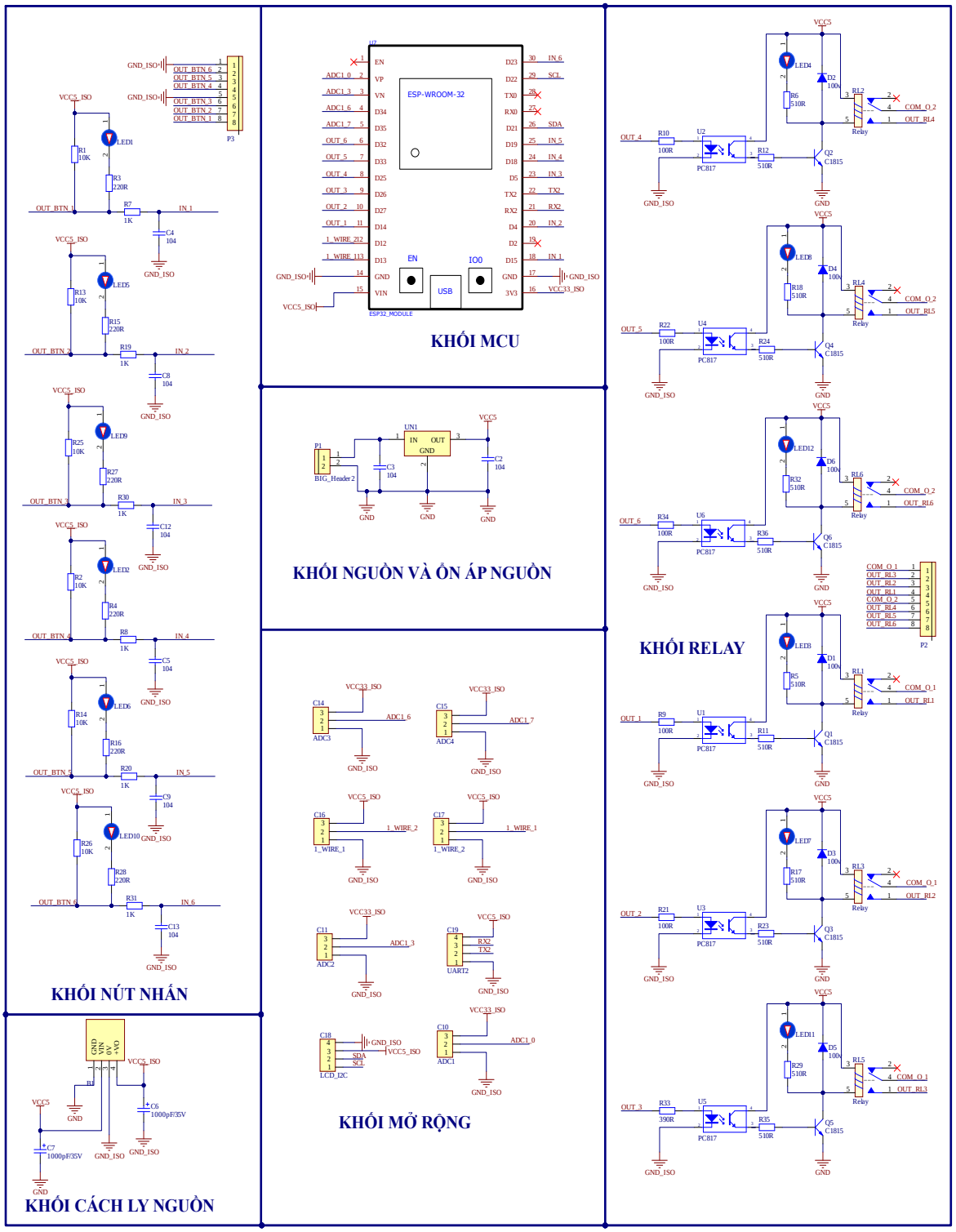
Hình 2.13. Mạch nguyên lý khối mở rộng

Giao tiếp One-Wire: Được sử dụng cho việc giao tiếp với các cảm biến nhiệt độ và độ ẩm như: DHT22, DS1307…

ADC: Bộ chuyển đổi tương tự - số, trên ESP32 được tích hợp một bộ ADC có độ phân giải là 12 bit thích hợp cho việc giao tiếp với các ngoại vi có sử dụng giao tiếp ADC.

I2C: Giao tiếp sử dụng 2 dây SCL và SDA để giao tiếp. I2C được sử dụng chủ yếu cho màn hình LCD hoặc các loại cảm biến có chuẩn giao tiếp I2C.

2.2.7 Sơ đồ nguyên lý toàn mạch



Hình 2.14: Sơ đồ nguyên lý toàn mạch

2.3 Xây dựng phần mềm điều khiển

2.3.1 Xây dựng lưu đồ thuật toán

Xử lý sự kiện nhận về từ MQTT Broker.

Tín hiệu nhận được sẽ được xử lý thông qua một hàm callback().

Hàm callback() này có chức năng như một ngắt nhận, khi nhận được tín hiệu.

ESP32 sẽ xử lý dữ liệu và bật tắt thiết bị theo dữ liệu nhận về tương ứng.

Dữ liệu Webserver gửi về với các dữ liệu tương ứng như sau [7]:

Gửi “11” nếu muốn bật đèn 1, “10” nếu muốn tắt đèn 1

Gửi “21” nếu muốn bật đèn 1, “20” nếu muốn tắt đèn 2

Gửi “31” nếu muốn bật đèn 1, “30” nếu muốn tắt đèn 3

Gửi “41” nếu muốn bật đèn 1, “40” nếu muốn tắt đèn 4

Gửi “51” nếu muốn bật đèn 1, “50” nếu muốn tắt đèn 5

Gửi “61” nếu muốn bật đèn 1, “60” nếu muốn tắt đèn 6

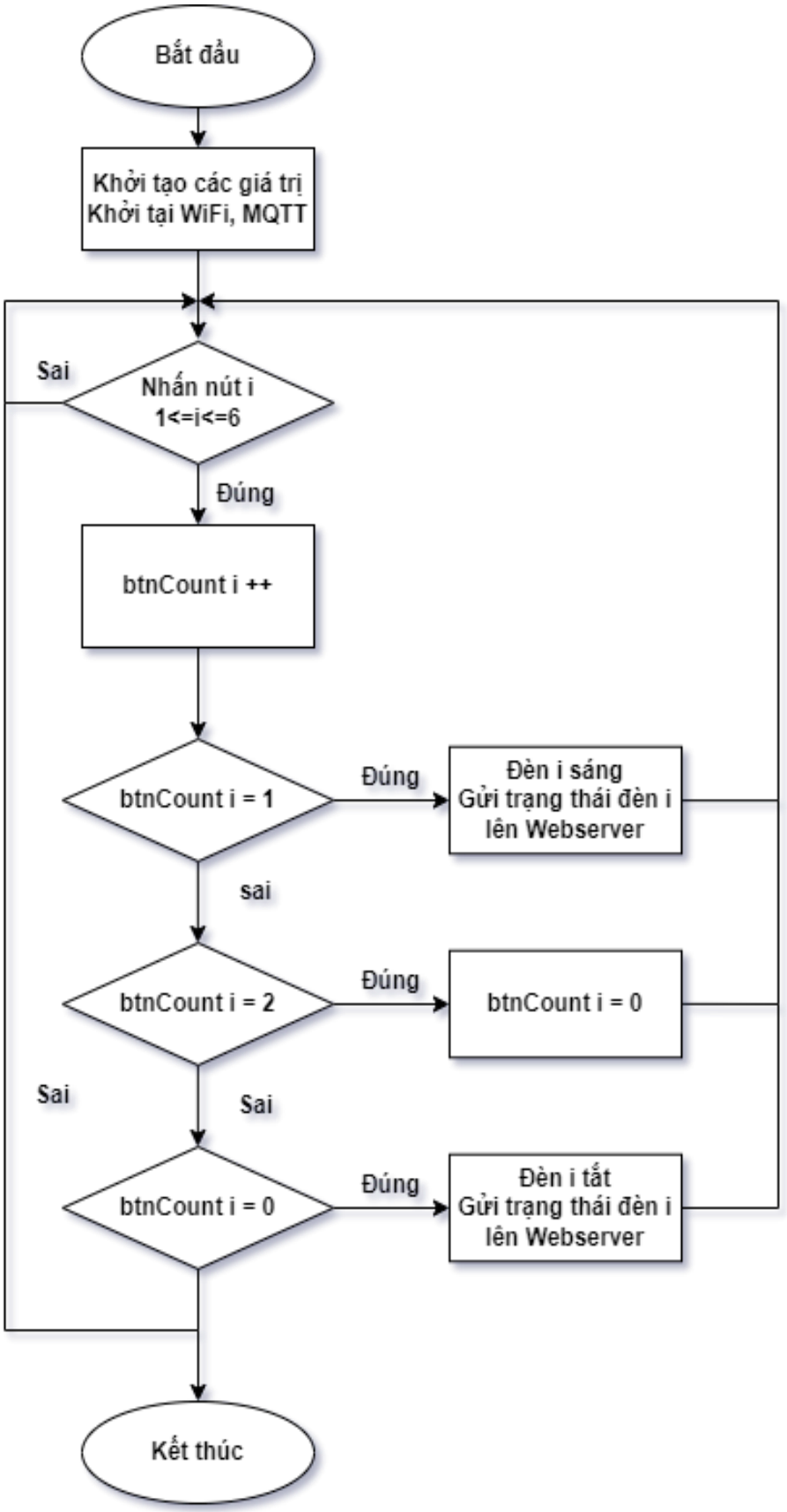
Xử lý sự kiện nhấn nút

Khi nhấn nút:

Dựa vào sự thay đổi của biến “btnCount” để bật rơ le hoặc tắt rơ le.

Quá trình được mô tả như hình 2.16

- Nếu btnCount = 0 thì tắt rơ le.
- Nếu btnCount = 1 thì bật rơ le.
- Nếu btnCount = 2 thì reset lại trạng thái của btnCount về 0.



Hình 2.15. Lưu đồ thuật toán sử lý sự kiện nút nhấn

2.3.2 Phần mềm điều khiển

Sử dụng HTML, CSS và JavaScript để lập trình phần Webserver.

Sử dụng HTML dựng khung giao diện:

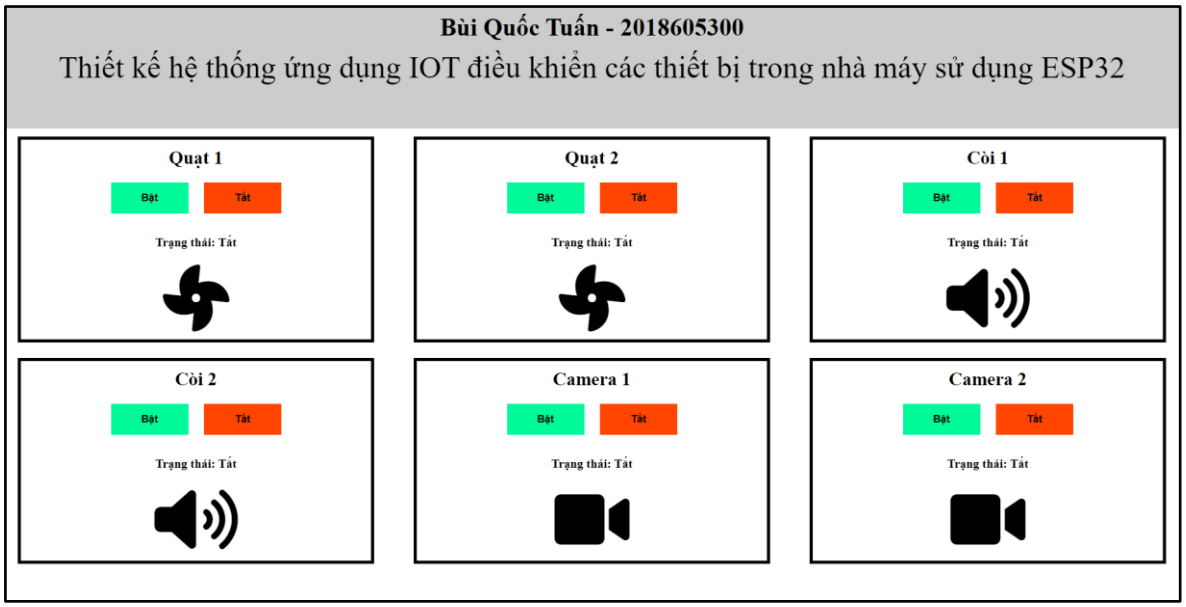
Tạo một file index.html và tạo khung chương trình:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Sử dụng thêm một số thẻ để tạo các thành phần khác như: nút nhấn, đèn hiển thị, các thiết bị khác.

Sử dụng CSS để tạo kiểu cho HTML (Tô màu, chỉnh font chữ đậm nhạt…).

Giao diện điều khiển điều khiển hệ thống:



Hình 2.16. Giao diện điều khiển hệ thống trên Web Server

Giao diện gồm 6 khối điều khiển các thiết bị:

Mỗi khối có 2 nút nhấn bật/tắt thiết bị. Trạng thái của thiết bị được hiển thị tại mỗi khối.

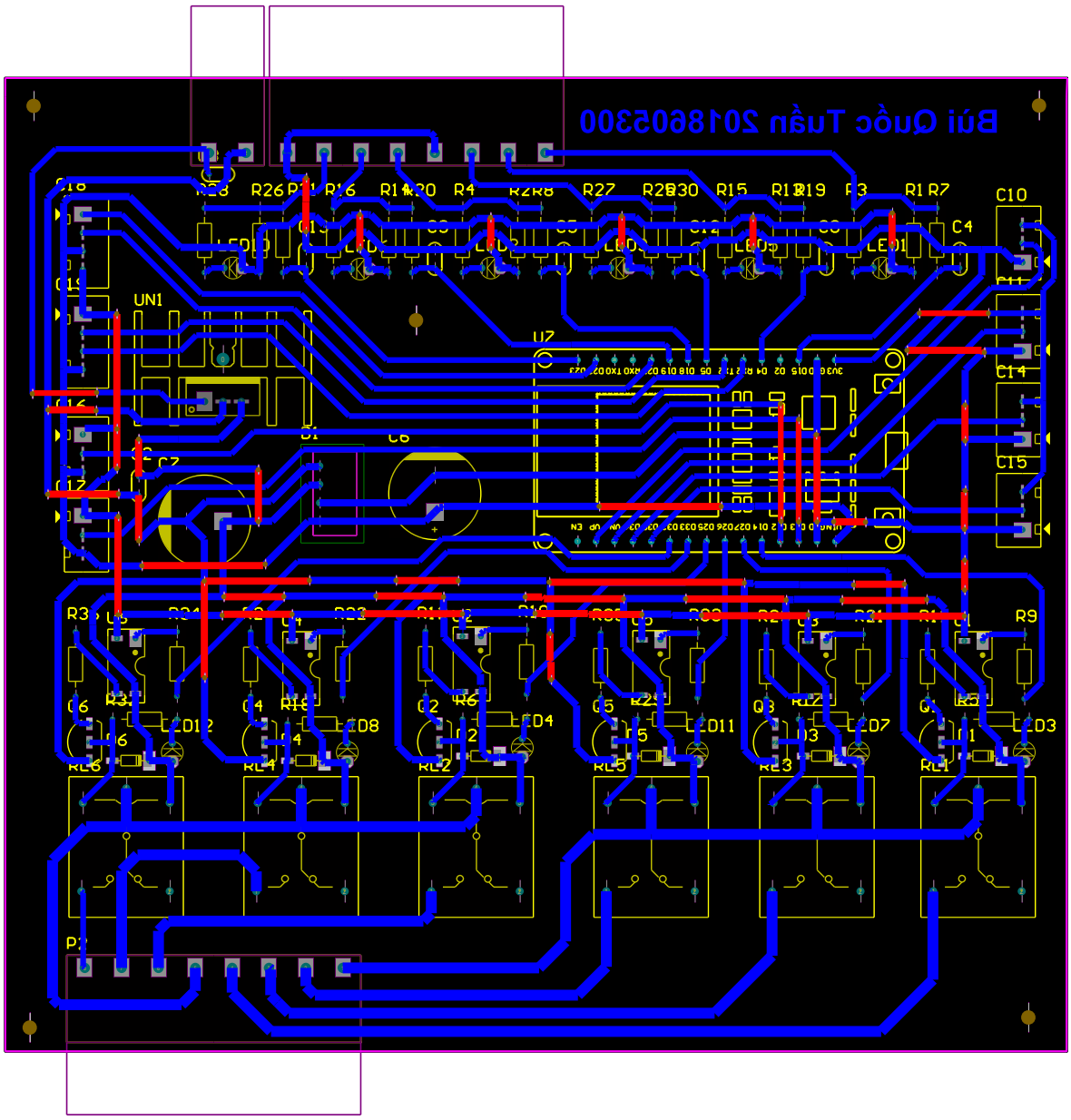
2.4 Thiết kế phần cứng

Đặt luật cho mạch in:

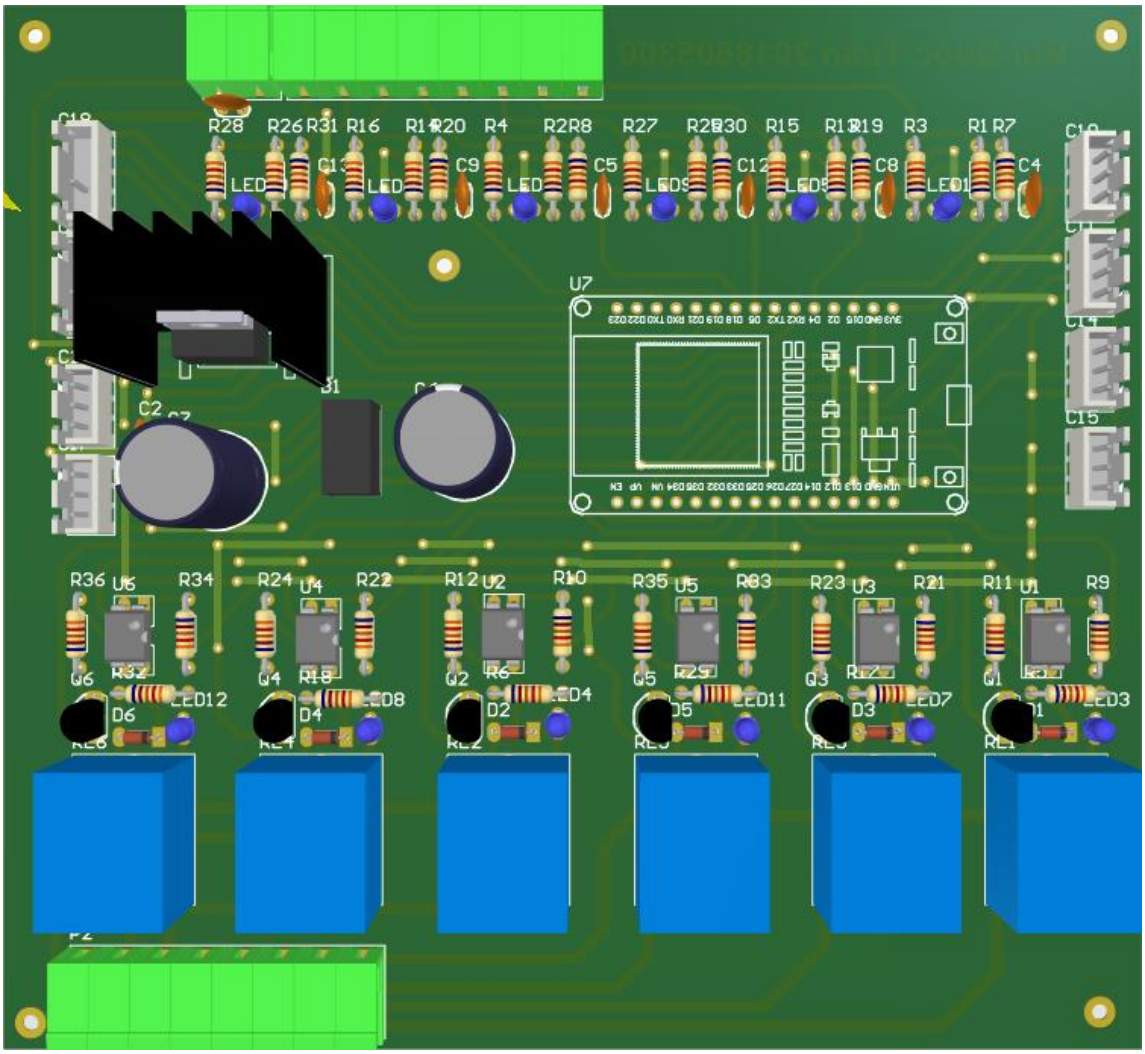
Mạch in của hệ thống được thiết kế trên phần mềm Altium Designer với các thông số như sau:

- Khoảng cách giữa các đường mạch: 1mm.
- Độ rộng đường GND_ISO: 1mm.
- Độ rộng đường COM_O_1: 1.5 mm.
- Độ rộng đường COM_O_2: 1.5 mm.
- Độ rộng đường VCC_ISO: 1mm.
- Độ rộng đường GND: 1mm.

Sơ đồ mạch in



Hình 2.17. Sơ đồ mạch in PCB 2D



Hình 2.18. Sơ đồ mạch in PCB 3D

2.5 Kết luận chương 2

Chương 2 nói về việc sử dụng các thiết bị, các phần mềm và các mô-đun đã dùng trong quá trình hoàn thành sản phẩm và các kiến trúc tổng quan phục vụ cho việc giải bài toán được đặt ra.

Kết thúc chương, em đã rút ra nhiều kết luận:

Tìm hiểu các bước thực hiện và giải quyết một bài toán ứng dụng cụ thể.

Thiết kế mạch và lập trình ứng dụng vào mạch thực tế.

Khắc phục lỗi, sự cố của sản phẩm.

Thử nghiệm và đánh giá kết quả sản phẩm.

CHƯƠNG 3:KẾT QUẢ THỰC NGHIỆM

3.1 Phân tích tính năng và hiệu quả sử dụng của sản phẩm

Điều khiển, giám sát thiết bị hệ thống đèn, từng đèn thông quan giao tiếp không dây. Quản lý trạng thái đèn, điều khiển trực tiếp, thiết lập ngữ cảnh theo lịch trình. Phần mềm điều khiển thiết bị thân thiện, dễ sử dụng có thể chạy trên nhiều nền tảng (ứng dụng công nghệ điện toán đám mây). Phần mềm quản lý dưới dạng sơ đồ, giúp việc quản lý được trực quan hơn và dễ dàng hơn rất nhiều. Thống kê tỉ lệ đèn lỗi, hỏng và thời gian hoạt động của hệ thống. Từ các báo cáo thông kê từ phần mềm giúp bạn kiểm soát tốt sự hoạt động cũng như có những điều chỉnh phù hợp với tình hình thực tế. Có thể mở rộng tích hợp với các nền tảng thông minh khác. Phân tích tính ứng dụng, mức độ an toàn và tác động của sản phẩm thiết kế tới môi trường, kinh tế và xã hội.

3.2 Phân tích tính ứng dụng, mức độ an toàn và tác động của sản phẩm thiết kế tới môi trường, kinh tế và xã hội.

Với các công nghệ IoT ngày càng tinh vi đang trở nên phổ biến hiện nay, các công ty không chỉ có thể theo dõi luồng sản phẩm hoặc kiểm tra các tài sản hữu hình, mà còn có thể quản lý hiệu suất làm việc của từng thiết bị máy móc và hệ thống, ví dụ như là theo dõi và quản lý một dây chuyền lắp rắp toàn bộ các bộ phận của robot hoặc máy móc nào đó. Các thiết bị cảm biến cũng có thể được nhúng trong cơ sở hạ tầng cơ sở, ví dụ như, bộ cảm biến từ tính đặt trên đường có thể đếm chính xác số lượng các loại phương tiện xe chạy qua, có thể hiệu chỉnh theo thời gian thực thời gian tín hiệu giao thông. Quan trọng không kém các cảm biến và các thiết bị truyền động này là các kết nối thông tin liên lạc dữ liệu để truyền dữ liệu này và các chương trình mã hóa, bao gồm các phân tích dữ liệu lớn, làm cho dữ liệu trở nên có ý nghĩa. Hơn nữa, các ứng dụng của IoT tính đến cả các thiết lập hệ điều khiển khép kín trong các hoạt động có thể tự động kích hoạt dựa trên các dữ liệu do thiết bị cảm biến đóng gói.

Dưới đây là các giải pháp được coi là tốt nhất mà hệ thống IoT đem lại cho cuộc sống con người ngày nay.

Thiết kế sản phẩm cung cấp độ tin cậy và bảo mật.

Sử dụng các giao thức xác thực và bảo mật mạnh mẽ.

Tắt các dịch vụ không cần thiết.

Đảm bảo các dịch vụ và trung tâm quản lý Internet và IoT được bảo mật.

Các thuật toán tiết kiệm năng lượng được thiết kế để hệ thống hoạt động lâu hơn.

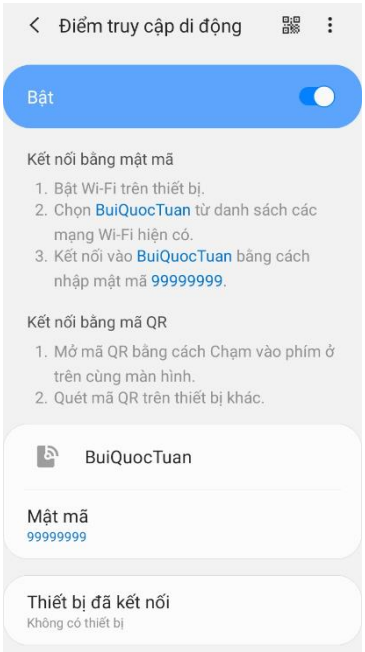
3.3 Hướng dẫn sử dụng sản phẩm thiết kế.

Bước 1: Dùng thiết bị phát Wifi chia sẻ mạng với thông tin như sau (hình :

Tên wifi: “BuiQuocTuan” ;

Mật khẩu: “99999999”.

Bước này là kết nối wifi với module ESP32 trên mô hình. Nhằm mở giao diện điều khiển trên Wed.

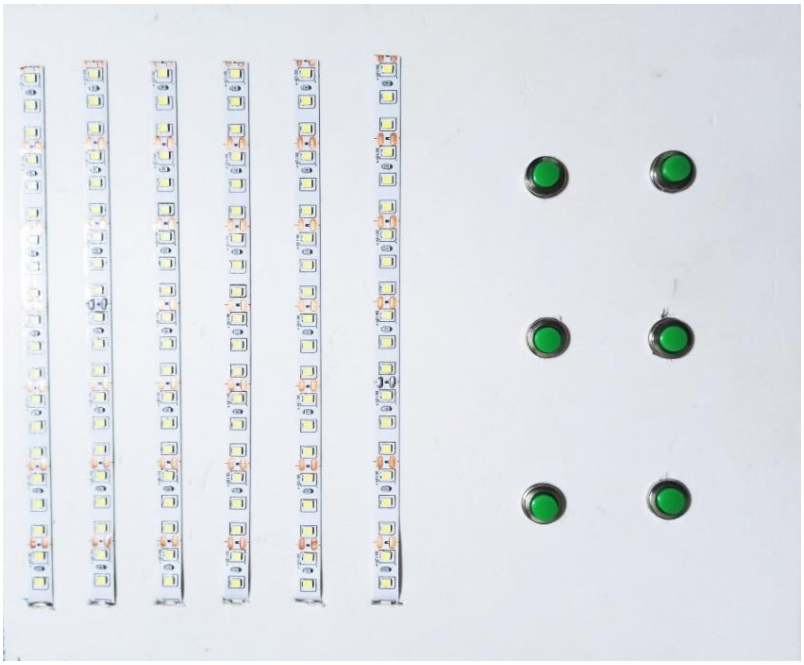


Hình 3.1. Giao diện kết nối internet của ESP32

Bước 2: Cấp nguồn 12V qua Jack DC 5.5 cho mô hình.

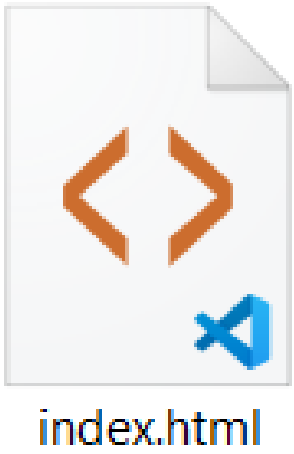
Sử dụng 2 nguồn 12V. Một nguồn là cấp cho toàn mô hình. Một nguồn cấp cho khối chấp hành.

Bước 3: Bật tắt các thiết bị bằng nút nhấn có trên bộ điều khiển.



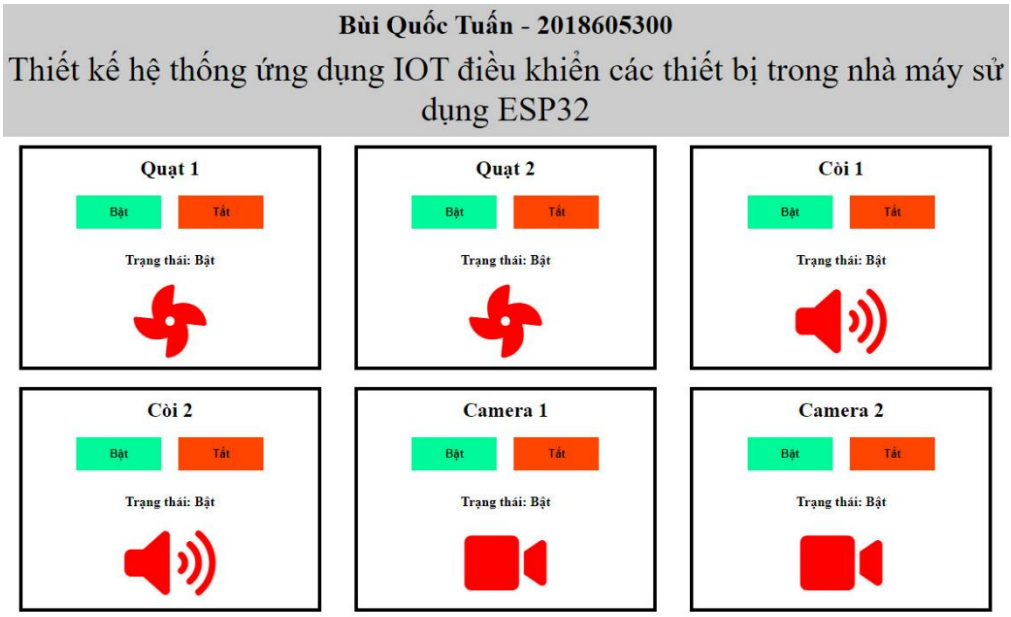
Hình 3.2. Điều khiển hệ thống bằng nút nhấn

Bước 4: Mở file index.html chứa giao diện của wedsite điều khiển và giám sát hệ thống.



Hình 3.3: File Index.html

Bước 5: Điều khiển hệ thống bằng cách nhấn các nút nhấn có trên giao diện, đồng thời quan sát được trạng thái của từng thiết bị trên giao diện.



Hình 3.4: Giao diện điều khiển và giám sát qua website

KẾT LUẬN

Kết quả đạt được

Qua quá trình thực hiện làm đồ án, em đã trình bày các cơ sở lý thuyết liên quan và chạy thành công hệ thống **"Thiết kế hệ thống ứng dụng IOT điều khiển các thiết bị trong nhà máy sử dụng esp32"**. Từ việc tìm hiểu công nghệ phát triển của hệ thống cho tới các bước thiết kế hoàn thành mô hình.

Cũng cố kiến thức, áp dụng vào việc thiết kế hệ thống. Sử dụng thành thạo các phần mềm hỗ trợ trong việc hoàn thành đề tài như: Altium… Nghiên cứu và tìm hiểu về các hệ thống điều khiển và giám sát trên thực tế, ưu điểm và nhược điểm của từng hệ thống. Nghiên cứu và tìm hiểu về vi điều khiển ESP32, các ngoại vi và lập trình ESP32. Tìm hiểu về chuẩn truyền thông không dây WiFi, ứng dụng nó vào trong việc điều khiển, truyền nhận tín hiệu và giám sát. Nắm được các hệ thống điều khiển và giám sát từ xa hoạt động. Xây dựng được giao diện điều khiển và giám sát. Thiết kế và vận hành thành công mạch giám sát và điều khiển thiết bị qua WiFi sử dụng vi điều khiển ESP32.

Ưu điểm của hệ thống

- Tiết kiệm chi phí vận hành (giảm hoàn toàn nhân công vận hành).
- Kiểm soát thông minh và giám sát hệ thống từ trung tâm.
- Cảnh báo sự cố đèn hỏng tức thì và cô lập riêng từng đèn bị hỏng để sửa chữa.
- Phần mềm có thể tương thích với PC, Laptop và các sản phẩm khác.
- Hệ thống hoạt động liên tục 24/7. Hệ thống thân thiện với người sử dụng, không đòi hỏi cao về kiến thức vận hành. Hệ thống được thiết kế module hóa và có khả mở rộng nâng cấp dễ dàng.

Nhược điểm của hệ thống.

- Chưa thể hiện được lợi ích của hệ thống Iot trong công nghiệp.

- Điều khiển ít thiết bị, mới sử dụng đèn thay thế, chưa thể hiện tính công nghiệp trong mô hình.
- Giao diện điều khiển còn đơn giản, chưa thể giám sát được nhiều trạng thái của thiết bị cũng như hệ thống.
- Chưa có hệ thống phân tích bảo trì dự đoán tình trạng của thiết bị công nghiệp.

Hướng phát triển đề tài.

- Tạo giao diện phong phú và đa dạng hơn.
- Giám sát và điều khiển các thiết bị không chỉ ON/OFF mà cả các tín hiệu nhiều dạng khác nhau.
- Phát triển đa nền tảng (Android, IOS…).
- Điều khiển và giám sát thêm các thiết bị khác, các loại thiết bị có công suất lớn.
- Phát triển đa nền tảng: đề tài không chỉ giới hạn ở Web mà còn có thể phát triển để phù hợp với điện thoại Android và IOS.
- Phát triển các chức năng cấu hình ngay trên điện thoại.
- Phát triển nhiều giao thức hơn nữa thay vì chỉ sử dụng MQTT để trong tất cả các trường hợp đều có thể bật tắt các thiết bị.

Thông qua quá trình làm đồ án, em đã được vận dụng những kiến thức chuyên ngành trong 4 năm học. Qua đó đã giúp cho em rèn luyện được kỹ năng, cách tiếp cận với các vấn đề, các bài toán thực tế phức tạp tại các doanh nghiệp, nhà máy khi ra trường làm việc.

Việc xây dựng mô hình đã đáp ứng được yêu cầu đặt ra, tuy nhiên do trình độ và kinh nghiệm thực tiễn còn hạn chế nên không thể tránh khỏi sai sót và thiếu hoàn chỉnh. Rất mong được đón nhận sự đóng góp ý kiến từ thầy cô và các bạn.

TÀI LIỆU THAM KHẢO

[1] Intech. 2017, “Kết nối Iot trong nhà máy”, tháng 4 năm 2022.

< https://intech-group.vn/ket-noi-iot-trong-nha-may.htm >.

[2] Evans, D. The Internet of Things: How the Next Evolution of the Internet is Changing Everything; Cisco Internet Business Solutions Group: San Jose, CA, USA, 2011.

[3] Wikipedia.2021,“ESP32”, tháng 4 năm 2022.

< https://vi.wikipedia.org/wiki/ESP32 >.

[4] Linh kiện Việt. 2018, “IC cách ly nguồn B5050S”, tháng 4 năm 2022.

< http://linhkienviet.vn/module-nguon-cach-ly-b0505s-2w-dc-dc-5v-5v >.

[5] The gioi IC. 2021. “Nút nhấn DS-316”. tháng 4 năm 2022.

< https://www.thegioiic.com/products/ds-316-nut-nhan-nha-10mm-2-chan-mau-do>.

[6] Linh kiện 888. 2021, “Relay 5v 10A”, tháng 4 năm 2022.

< https://linhkien888.vn/relay-5v-10a-5-chan-srd-05vdc-sl-c >.

[7] BùiVănVinh(2020), Mô đun: Lập trình vi điều khiển, Trường Cao đẳng Kỹ thuật Công Nghệ, Vũng Tàu.

[8] Điện tử DAT. 2021, “Nguồn 12V”, tháng 4 năm 2022.

< https://www.dientudat.com/mach-chuyen-nguon-ac-dc-12v-2a >.

[9] Machdientu. 2021. “LM7805”. tháng 4 năm 2022.

< https://machdientu.org/datasheet-7805-va-huong-dan-su-dung-ung-dung-7805#gsc.tab=0 >.

PHỤ LỤC

CODE ARDUINO

```
#include <Arduino.h>
#include <PubSubClient.h>
#include <WiFi.h>
//#include <Button.h>
#include <ArduinoJson.h>
//define pin to use
//input
#define BTN_01 15
#define BTN_02 4
#define BTN_03 5
#define BTN_04 18
#define BTN_05 19
#define BTN_06 23
//output
#define OUT_03 14
#define OUT_02 27
#define OUT_01 26
#define OUT_04 25
#define OUT_05 33
#define OUT_06 32
```

```
//define wifi
#define ssid "tuốn"
#define password "88888888"
//define mqtt
#define mqtt_server "broker.hivemq.com"
#define mqtt_user ""
#define mqtt_pwd ""
const uint16_t mqtt_port = 1883;
String mqtt_topic_pub = "buiquoctuan/relayInfor";
String mqtt_topic_sub = "buiquoctuan/relayControl";
String mqtt_topic_pub_now = "buiquoctuan/relayInforNow";
//user variable
long lastMsg = 0;
char msg[50];
int value = 0;
int dataInt = 0;
String Data = "";
String ChuoiSendWebJson = "";
float nhietdo = 0;
char inforInverterBuff[256];
unsigned long last = 0, bien = 0;
//define
```

```
int button1PressCount = 0;

int button2PressCount = 0;

int button3PressCount = 0;

int button4PressCount = 0;

int button5PressCount = 0;

int button6PressCount = 0;

WiFiClient espClient;

PubSubClient client(espClient);

//array of pin

const int inputPinArr[6] = {BTN_01, BTN_02, BTN_03, BTN_04, BTN_05,
BTN_06};

const int outputPinArr[6] = {OUT_01, OUT_02, OUT_03, OUT_04,
OUT_05, OUT_06};

//function define

void pinInit(int pinNumberInput, int pinNumberOnput);

void buttonInit();

void callback(char* topic, byte* payload, unsigned int length);

void setupWifi();

void reconnect();

//task init

void taskInit();

//button 1 task

void handleButton1Task();
```

```
void handleButton1(void *parameter);
//button 2 task
void handleButton2Task();
void handleButton2(void *parameter);
//button 3 task
void handleButton3Task();
void handleButton3(void *parameter);
//button 4 task
void handleButton4Task();
void handleButton4(void *parameter);
//button 5 task
void handleButton5Task();
void handleButton5(void *parameter);
//button 6 task
void handleButton6Task();
void handleButton6(void *parameter);
//mqtt handle
void handleMQTTTask();
void handleMQTT(void *parameter);
void setup() {
  Serial.begin(115200);
  client.setServer(mqtt_server, mqtt_port);
```

```
  client.setCallback(callback);
  pinInit(6, 6);
  taskInit();
  setupWifi();}
void loop() {
  if (!client.connected()) {
    reconnect();}
  client.loop();}
void pinInit(int pinNumberInput, int pinNumberOnput){
  for (int i = 0; i < pinNumberInput; i ++){
    pinMode(inputPinArr[i], INPUT);}
  for (int i = 0; i < pinNumberOnput; i ++){
    pinMode(outputPinArr[i], OUTPUT); }}
void taskInit(){
  handleButton1Task();
  handleButton2Task();
  handleButton3Task();
  handleButton4Task();
  handleButton5Task();
  handleButton6Task();
  handleMQTTTask();}
void handleButton1Task(){
```

```
  xTaskCreatePinnedToCore(  // Use xTaskCreate() in vanilla FreeRTOS
    handleButton1,  // Function to be called
    "handleButton1",   // Name of task
    2048,         // Stack size (bytes in ESP32, words in FreeRTOS)
    NULL,         // Parameter to pass to function
    1,           // Task priority (0 to configMAX_PRIORITIES - 1)
    NULL,         // Task handle
    1);}
void handleButton2Task(){
  xTaskCreatePinnedToCore(  // Use xTaskCreate() in vanilla FreeRTOS
    handleButton2,  // Function to be called
    "handleButton2",   // Name of task
    2048,         // Stack size (bytes in ESP32, words in FreeRTOS)
    NULL,         // Parameter to pass to function
    1,           // Task priority (0 to configMAX_PRIORITIES - 1)
    NULL,         // Task handle
    1);}
void handleButton3Task(){
  xTaskCreatePinnedToCore(  // Use xTaskCreate() in vanilla FreeRTOS
    handleButton3,  // Function to be called
    "handleButton3",   // Name of task
    2048,         // Stack size (bytes in ESP32, words in FreeRTOS)
```

```
    NULL,        // Parameter to pass to function
    1,           // Task priority (0 to configMAX_PRIORITIES - 1)
    NULL,        // Task handle
    1);}
void handleButton4Task(){
  xTaskCreatePinnedToCore(  // Use xTaskCreate() in vanilla FreeRTOS
    handleButton4,  // Function to be called
    "handleButton4",   // Name of task
    2048,        // Stack size (bytes in ESP32, words in FreeRTOS)
    NULL,        // Parameter to pass to function
    1,           // Task priority (0 to configMAX_PRIORITIES - 1)
    NULL,        // Task handle
    1);}
void handleButton5Task(){
  xTaskCreatePinnedToCore(  // Use xTaskCreate() in vanilla FreeRTOS
    handleButton5,  // Function to be called
    "handleButton5",   // Name of task
    2048,        // Stack size (bytes in ESP32, words in FreeRTOS)
    NULL,        // Parameter to pass to function
    1,           // Task priority (0 to configMAX_PRIORITIES - 1)
    NULL,        // Task handle
    1);}
```

```
void handleButton6Task(){
  xTaskCreatePinnedToCore(  // Use xTaskCreate() in vanilla FreeRTOS
    handleButton6,  // Function to be called
    "handleButton6",   // Name of task
    2048,         // Stack size (bytes in ESP32, words in FreeRTOS)
    NULL,         // Parameter to pass to function
    1,            // Task priority (0 to configMAX_PRIORITIES - 1)
    NULL,         // Task handle
    1);}
void handleMQTTTask(){
  xTaskCreatePinnedToCore(  // Use xTaskCreate() in vanilla FreeRTOS
    handleMQTT,  // Function to be called
    "handleMQTT",   // Name of task
    2048,         // Stack size (bytes in ESP32, words in FreeRTOS)
    NULL,         // Parameter to pass to function
    1,            // Task priority (0 to configMAX_PRIORITIES - 1)
    NULL,         // Task handle
    1);}
void handleButton1(void *parameter) {
  while (1) {
    while (digitalRead(BTN_01) == HIGH);
    while (digitalRead(BTN_01) == LOW);
```

```
    button1PressCount ++;
    if (button1PressCount == 2){
      button1PressCount = 0;}
    if (button1PressCount == 0){
      digitalWrite(OUT_01, LOW);
      client.publish(mqtt_topic_pub_now.c_str(), "10");}
    else if (button1PressCount == 1){
      digitalWrite(OUT_01, HIGH);
      client.publish(mqtt_topic_pub_now.c_str(), "11");}}}
//handle button 2
void handleButton2(void *parameter) {
  while (1) {
    while (digitalRead(BTN_02) == HIGH);
    while (digitalRead(BTN_02) == LOW);
    button2PressCount ++;
    if (button2PressCount == 2){
      button2PressCount = 0;}
    if (button2PressCount == 0){
      digitalWrite(OUT_02, LOW);
      client.publish(mqtt_topic_pub_now.c_str(), "20");}
    else if (button2PressCount == 1){
      digitalWrite(OUT_02, HIGH);
```

```
      client.publish(mqtt_topic_pub_now.c_str(), "21");}}}
//handle button 3
void handleButton3(void *parameter) {
  while (1) {
    while (digitalRead(BTN_03) == HIGH);
    while (digitalRead(BTN_03) == LOW);
    button3PressCount ++;
    if (button3PressCount == 2){
      button3PressCount = 0;}
    if (button3PressCount == 0){
      digitalWrite(OUT_03, LOW);
      client.publish(mqtt_topic_pub_now.c_str(), "30");}
    else if (button3PressCount == 1){
      digitalWrite(OUT_03, HIGH);
      client.publish(mqtt_topic_pub_now.c_str(), "31");}}}
void handleButton4(void *parameter) {
  while (1) {
    while (digitalRead(BTN_04) == HIGH);
    while (digitalRead(BTN_04) == LOW);
    button4PressCount ++;
    if (button4PressCount == 2){
      button4PressCount = 0;}
```

```
    if (button4PressCount == 0){
      digitalWrite(OUT_04, LOW);
      client.publish(mqtt_topic_pub_now.c_str(), "40");}
    else if (button4PressCount == 1){
      digitalWrite(OUT_04, HIGH);
      client.publish(mqtt_topic_pub_now.c_str(), "41");}}}
void handleButton5(void *parameter) {
  while (1) {
    while (digitalRead(BTN_05) == HIGH);
    while (digitalRead(BTN_05) == LOW);
    button5PressCount ++;
    if (button5PressCount == 2){
      button5PressCount = 0;}
    if (button5PressCount == 0){
      digitalWrite(OUT_05, LOW);
      client.publish(mqtt_topic_pub_now.c_str(), "50"); }
    else if (button5PressCount == 1) {
      digitalWrite(OUT_05, HIGH);
      client.publish(mqtt_topic_pub_now.c_str(), "51");}}}
void handleButton6(void *parameter) {
  while (1) {
    while (digitalRead(BTN_06) == HIGH);
```

```
    while (digitalRead(BTN_06) == LOW);
    button6PressCount ++;
    if (button6PressCount == 2){
      button6PressCount = 0;}
    if (button6PressCount == 0){
      digitalWrite(OUT_06, LOW);
      client.publish(mqtt_topic_pub_now.c_str(), "60");}
    else if (button6PressCount == 1){
      digitalWrite(OUT_06, HIGH);
      client.publish(mqtt_topic_pub_now.c_str(), "61");}}}
void handleMQTT(void *parameter) {
  while (1){
    DynamicJsonDocument doc(1024);
    doc["den1"]   =  button1PressCount;
    doc["den2"] =  button2PressCount;
    doc["den3"]   = button3PressCount;
    doc["den4"] = button4PressCount;
    doc["den5"]   = button5PressCount;
    doc["den6"] = button6PressCount;
    serializeJson(doc, inforInverterBuff);
    client.publish(mqtt_topic_pub.c_str(), inforInverterBuff);
    vTaskDelay(3000/portTICK_PERIOD_MS);}}
```

```
//mqtt function
void reconnect(){
  while (!client.connected()){
    String clientId = String(random(0xffff), HEX);
    if (client.connect(clientId.c_str(), mqtt_user, mqtt_pwd)){
      Serial.println("Connected MQTT ngoinhaiot.com");
      client.subscribe(mqtt_topic_sub.c_str());
      digitalWrite(16, HIGH);}
    else{
      Serial.println("Không thể kết nối MQTT ngoinhaiot.com");
      digitalWrite(16, LOW);
      delay(5000);}}}
//call back mqtt
void callback(char* topic, byte* payload, unsigned int length){
  Data = "";
  for (int i = 0; i < length; i++){
    Data += (char)payload[i]; // abcde}
  dataInt = Data.toInt();
  Serial.println(dataInt);
  ///button 1
  if (dataInt == 11){
    button1PressCount = 1;
```

```
  digitalWrite(OUT_01, HIGH);
  client.publish(mqtt_topic_pub_now.c_str(), "11");}
else if (dataInt == 10){
  button1PressCount = 0;
  digitalWrite(OUT_01, LOW);
  client.publish(mqtt_topic_pub_now.c_str(), "10");}
///button 2
else if (dataInt == 21){
  button2PressCount = 1;
  digitalWrite(OUT_02, HIGH);
  client.publish(mqtt_topic_pub_now.c_str(), "21");}
else if (dataInt == 20){
  button2PressCount = 0;
  digitalWrite(OUT_02, LOW);
  client.publish(mqtt_topic_pub_now.c_str(), "20");}
///button 3
else if (dataInt == 31){
  button2PressCount = 1;
  digitalWrite(OUT_03, HIGH);
  client.publish(mqtt_topic_pub_now.c_str(), "31");}
else if (dataInt == 30){
  button3PressCount = 0;
```

```
  digitalWrite(OUT_03, LOW);
  client.publish(mqtt_topic_pub_now.c_str(), "30");}
///button 4
else if (dataInt == 41){
  button4PressCount = 1;
  digitalWrite(OUT_04, HIGH);
  client.publish(mqtt_topic_pub_now.c_str(), "41");}
else if (dataInt == 40){
  button4PressCount = 0;
  digitalWrite(OUT_04, LOW);
  client.publish(mqtt_topic_pub_now.c_str(), "40");}
///button 5
else if (dataInt == 51){
  button5PressCount = 1;
  digitalWrite(OUT_05, HIGH);
  client.publish(mqtt_topic_pub_now.c_str(), "51");}
else if (dataInt == 50){
  button5PressCount = 0;
  digitalWrite(OUT_05, LOW);
  client.publish(mqtt_topic_pub_now.c_str(), "50");}
///button 6
else if (dataInt == 61){
```

```
    button6PressCount = 1;
    digitalWrite(OUT_06, HIGH);
    client.publish(mqtt_topic_pub_now.c_str(), "61");}
  else if (dataInt == 60){
    button6PressCount = 0;
    digitalWrite(OUT_06, LOW);
    client.publish(mqtt_topic_pub_now.c_str(), "60");}}
//set up wifi
void setupWifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");}
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());}
```