TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Thiết kế thiết bị đóng cắt 4 kênh sử dụng WiFi

Nguyễn Tuấn Anh
anh.nt173616@sis.hust.edu.vn
Ngành Kĩ thuật điện
Chuyên ngành Thiết bị điện – Điện tử

Giảng viên hướng dẫn: TS. Nguyễn Văn Ánh _____
Chữ ký của GVHD

Bộ môn: Thiết bị điện – Điện tử
Viện: Điện

HÀ NỘI, 01/2021

Phiếu giao nhiệm vụ đồ án tốt nghiệp

Lời cảm ơn

Lời đầu tiên, con xin gửi lời cảm ơn đến ông bà, bố mẹ đã dạy dỗ, chỉ bảo và nuôi nấng con thành người. Bố mẹ không chỉ là đấng sinh thành mà còn là những người thầy, người bạn yêu thương con một cách vô điều kiện. Suốt quãng thời gian là học sinh – sinh viên, bố mẹ đã không quản ngại vất vả, kiếm những đồng tiền bằng mồ hôi nước mắt cho con ăn học, không để con phải thiệt thòi so với bạn bè. Con cảm ơn bố mẹ rất nhiều.

Để đạt được thành quả như ngày hôm nay, em xin gửi lời cảm ơn đến thầy Nguyễn Văn Ánh, Bộ môn Thiết bị điện – Điện tử. Thời gian học và làm đồ án với thầy trên trường đã giúp em học hỏi được rất nhiều điều bổ ích cũng như rèn luyện được các kĩ năng giao tiếp và chia sẻ kiến thức tới mọi người.

Em cũng xin gửi lời cảm ơn đến toàn thể các thầy các thầy, các cô đã truyền đạt cho chúng em kiến thức và những kinh nghiệm quý báu bằng cả tâm huyết, giúp chúng em có hành trang tự tin bước vào đời.

Em xin cảm ơn tất cả những người bạn đã đồng hành cùng em trong suốt năm tháng Bách Khoa, luôn là chỗ dựa tinh thần vững chắc trong suốt quá trình học tập. Thời gian học ở Bách Khoa sẽ mãi là những kí ức tuyệt đẹp trong trí nhớ của em.

Trong quá trình làm đồ án tốt nghiệp, em đã cố gắng hết mình nhưng do thời gian và trình độ bản thân còn hạn chế nên đồ án tốt nghiệp của em còn nhiều thiếu sót. Em rất mong nhận được sự chia sẻ, và những ý kiến đóng góp từ thầy cô và các bạn để có thể hoàn thiện đề tài khi đưa và áp dụng thực tế.

Người thực hiện đề tài

Nguyễn Tuấn Anh

Tóm tắt nội dung đồ án

Trong những năm gần đây, số lượng các thiết bị điện thông minh với khả năng tự động hoạt động và kết nối với các thiết bị khác đang xuất hiện ngày một nhiều hơn mang đến nhiều lợi ích cho cuộc sống hàng ngày và lĩnh vực sản xuất nông nghiệp, công nghiệp cũng như dịch vụ. Với mong muốn thiết kế một hệ thống các thiết bị có khả năng kết nối với nhau từ xa qua mạng internet, cung cấp khả năng điều khiển, giám sát các thiết bị không phụ thuộc vào khoảng cách, em quyết định chọn đề tài "Thiết kế thiết bị đóng cắt 4 kênh qua WiFi". Đồ án này trình bày các nội dung đã thực hiện để xây dựng thiết bị đóng cắt từ xa và cung cấp giao diện điều khiển cũng như theo dõi trạng thái từng kênh đóng cắt của thiết bị.

Dưới đây là đề mục các chương được trình bày trong đồ án. Đồ án được chia làm 5 chương chính:

Chương 1: Giới thiệu đề tài

Chương 2: Thiết kế phần cứng

Chương 3: Thiết kế chương trình cho vi điều khiển

Chương 4: Phát triển chương trình ứng dụng điều khiển

Chương 5: Mô hình và kết quả đạt được

Trong mỗi chương sẽ có các đề mục nhỏ nhằm phân tích cụ thể và nêu ra các bước cần triển khai cho từng nhiệm vụ.

Mục lục

Danh mục hình vẽ

Danh mục bảng biểu

Danh mục phương trình

Danh mục các từ thuật ngữ và viết tắt

IoT	Internet of Things
WiFi	Wireless Fidelity
LED	Light Emitting Diode
GPIO	General Purpose Input/Output
IDE	Integrated Development Environment
RAM	Random-access memory
ROM	Read-only memory
MQTT	MQ Telemetry Transport
CoAP	Constrained Application Protocol
AMQP	Advanced Message Queuing Protocol
Publish	Xuất bản bản tin
Subscribe	Đăng kí nhận bản tin
Topic	Chủ đề của bản tin
Broker	Máy chủ
Client	Máy khách

Chương 1. Giới thiệu đề tài

1.1. Đặt vấn đề

Khoa học và công nghệ đang phát triển với tốc độ rất nhanh, đặc biệt trong lĩnh vực thiết bị điện và điện tử. Các thiết bị điện và điện tử hiện nay ngoài khả năng thực hiện các chức năng cơ bản còn có được tích hợp thêm khả năng tính toán và kết nối truyền thông nhờ sự phát triển mạnh của lĩnh vực điện tử nhúng. Với khả năng tính toán, các thiết bị có thể có các chế độ tự động, tự giám sát, … Với khả năng kết nối truyền thông và đặc biệt là các phương thức truyền thông không dây, các thiết bị điện có thể giao tiếp với nhau, từ đó cung cấp khả năng tự động hóa ở mức cao hơn cũng như khả năng điều khiển và giám sát quá trình hoạt động của các thiết bị từ xa. Tuy nhiên hiện nay các thiết bị điện còn sử dụng nhiều giao thức truyền thông không dây khác nhau như hồng ngoại, Bluetooth, Zwave, Zigbee, WiFi, … Sự thiếu đồng nhất về giao thức truyền thông này khiến cho việc kết nối giữa các thiết bị sử dụng giao thức khác nhau trở nên rất khó khăn. Trong khi đó, nhờ sự phát triển của công nghệ, cơ sở hạ tầng internet nói chung và WiFi nói riêng đã được mở rộng và nâng cấp với tốc độ truyền dữ liệu cao hơn, số lượng thiết bị cho phép kết nối vào mạng tăng được tăng từ 2^{32} thiết bị với IPv4 lên 2^{128} thiết bị nhờ IPv6. Với sự phát triển của hạ tầng internet, số lượng các thiết bị IoT (Internet vạn vật) đã bùng nổ. Các hệ thống trong nông nghiệp, công nghiệp, dịch vụ và gia đình sử dụng internet và WiFi ngày càng phổ biến, mang đến cơ hội năng cao hiệu năng, độ tin cậy, lợi ích kinh tế cũng như sự tiện nghi cho con người.

Nhận thấy sự phát triển của các thiết bị điện thông minh ở Việt Nam còn nhiều hạn chế, các thiết bị thông minh hiện nay còn sử dụng nhiều chuẩn giao tiếp không dây khác nhau, chưa được thống nhất, em quyết định thiết kế Thiết bị đóng cắt 4 kênh sử dụng WiFi.

1.2. Giới thiệu chung về thiết bị điện thông minh

Vị trí của thiết bị điện thông minh ngày càng trở nên quan trọng trong đời sống thường ngày cũng như trong các lĩnh vực sản xuất nông nghiệp, công nghiệp và dịch vụ. Các thiết bị này đem lại sự tiện lợi khi sử dụng, góp phần giúp con người có được sự tiện thoải mái, tiện nghi, tiết kiệm điện và công sức trong việc điều khiển các thiết bị điện trong nhà.

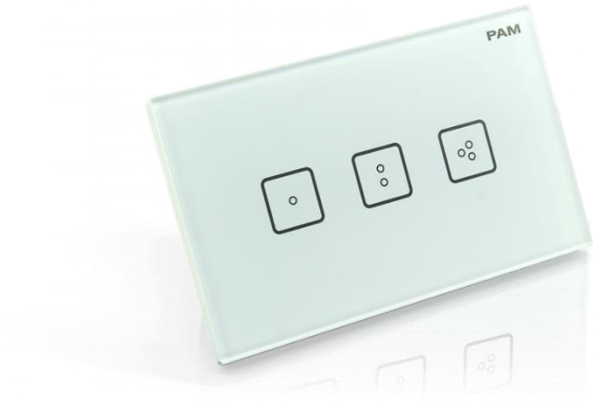
Các thiết bị điện thông minh trong nhà là các thiết bị điện mà bạn có thể điều khiển chúng thông qua những thao tác mạng internet hoặc kết nối không dây của các thiết bị bạn đang sử dụng như máy tính xách tay, điện thoại thông minh. Thậm chí các loại thiết bị điện thông minh có khả năng cảm ứng nhiệt, ánh sáng giúp cho các thao tác đóng cắt được tự động, không cần đến sự can thiệp của người dùng.

Thiết bị điện thông minh là hệ thống cho phép kết nối các thiết bị sử dụng điện trong nhà với nhau. Thiết bị mang đến nhiều cách để điều khiển, có khả năng tự động xử lý và thông báo cho người sử dụng. Ngoài ra, có khả năng tương tác được với các thông số môi trường, giúp người sử dụng có thể giám sát và điều khiển các thiết bị từ xa, đem lại sự an toàn, tiện nghi, linh hoạt, tiết kiệm.

Xu thế thiết bị điện thông minh là phát triển công nghệ vi mạch ngày càng nhỏ, công nghệ nano, tốc độ xử lý ngày càng nhanh, với chip đa nhân, khả năng kết hợp trao đổi dữ liệu với các thiết bị với nhau, các chuẩn giao tiếp dùng dây dẫn và đặc biệt là giao tiếp không dây như hồng ngoại, RF, WiFi, zigbee, …

1.3. Một số thiết bị điện thông minh trong thực tế

Các thiết bị điện thông minh phát triển với tốc độ rất nhanh trong thời gian gần đây, mang lại nhiều tiện ích và cuộc sông tiện nghi cho người sử dụng. Một số thiết bị thông minh tiêu biểu có thể kể đến như:



Hình 1. Công tắc 3 kênh điều khiển qua WiFi

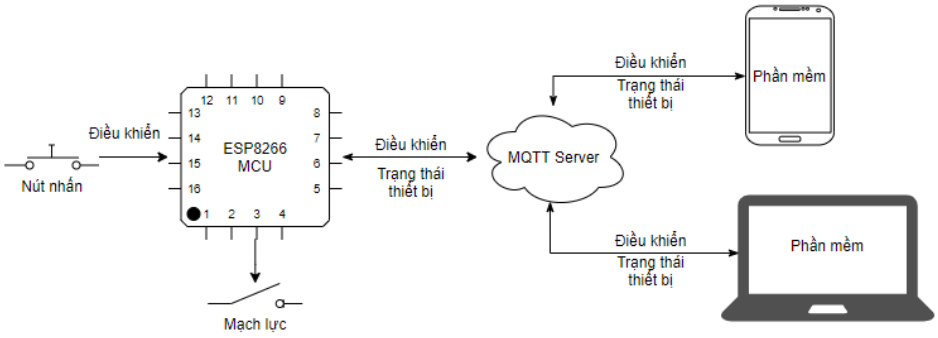
- Công tắc thông minh cho phép điều khiển tại chỗ thông qua mặt cảm ứng và điều khiển từ xa thông qua ứng dụng trên thiết bị di động

Hình 2. Ổ cắm điều khiển qua WiFi

- Ổ cắm WiFi cho phép đóng cắt cung cấp điện ở ổ cắm thông qua ứng dụng điều khiển trên điện thoại thông minh và máy tính xách tay có kết nối Internet

1.4. Mô tả tổng quan hệ thống

Hệ thống điều khiển đóng cắt 4 kênh qua WiFi gồm 3 thành phần chính: Thiết bị đóng cắt và điều khiển tại hiện trường, hệ thống máy chủ truyền dẫn dữ liệu, phần mềm điều khiển và giám sát từ xa.



Hình 3. Mô hình tổng quan của hệ thống

Phần thiết bị đóng cắt và điều khiển tại trường có chức năng đóng cắt trực tiếp các thiết bị điện thông qua Triac và các nút nhấn. Việc trao đổi dữ liệu giữa thiết bị đóng cắt, phần mềm điều khiển và máy chủ được thực hiện thông qua giao thức IoT thông dụng hiện nay là MQTT. Phía máy chủ MQTT có nhiệm vụ nhận và truyền các bản tin giữa thiết bị điều khiển và thiết bị trường.

1.5. Mục tiêu và nhiệm vụ của đề tài

Đồ án tập trung vào phát triển thiết bị đóng cắt 4 kênh sử dụng WiFi, bao gồm:

- Phát triển phần cứng thiết bị:
  - Thiết kế mạch lực
  - Thiết kế mạch nút nhấn giao tiếp người – máy
  - Thiết kế khối điều khiển
- Phát triển chương trình điều khiển cho thiết bị:
  - Phát triển chức năng điều khiển mạch lực
  - Phát triển chức năng giao tiếp người – máy
  - Phát triển trao đổi dữ liệu thiết bị và Máy chủ qua giao thức MQTT
- Phát triển ứng dụng điều khiển cho thiết bị trên điện thoại và máy tính:
  - Phát triển ứng dụng cho phép hiển thị trạng thái và điều khiển thiết bị
  - Phát triển ứng dụng trao đổi dữ liệu thiết bị và Máy chủ qua giao thức MQTT

1.6. Giải pháp

Dựa trên những yêu cầu từ đề tài, em quyết định lựa chọn các giải pháp thiết kế về phần cứng và phần mềm như sau.

1.6.1. Giải pháp phần cứng

Thiết bị được xây dựng xung quanh kit NodeMCU - vi điều khiển ESP8266 ở mạch điều khiển và mạch lực sử dụng van bán dẫn là Triac.

1.6.2. Giải pháp phần mềm

Hệ thống được xây dựng bao gồm phần điều khiển đóng cắt điện áp xoay chiều tại chỗ thông qua nút nhấn và điều khiển qua ứng dụng trên máy tính và điện thoại thông minh. Em quyết định sử dụng giao thức MQTT để truyền thông, phát

triển firmware dựa trên thư viện ESP8266 cho Arduino và phát triển ứng dụng dựa trên thư viện Kivy cho ngôn ngữ Python.

Giao thức MQTT là giao thức phổ biến trong thời đại IoT ngày nay. Đây là một giao thức gọn nhẹ, được thiết kế để kết nối các thiết bị mà có mạng băng thông thấp rất phù hợp với hệ thống điều khiển đóng cắt này.

Kivy là một thư viện mã nguồn mở được viết bằng Python để phát triển ứng dụng, đặc biệt là giao diện người dùng. Ứng dụng viết bằng Kivy có thể chạy trên cả Windows, Linux, OS X, iOS và Android.

1.6.3. Công cụ lập trình và môi trường phát triển

1. Công cụ để lập trình ESP8266

   Trong đồ án sử dụng Arduino IDE làm nền tảng lập trình bởi vì các lý do sau:

- Miễn phí và cộng đồng phát triển mạnh
- Hỗ trợ nhiều thư viện thuận tiên cho việc lập trình

2. Công cụ để lập trình ứng dụng điều khiển

   Trong đồ án sử dụng Python IDE làm nền tảng cho lập trình ứng dụng vì các lý do sau:

- Python có cú pháp đơn giản
- Python miễn phí và cộng đồng phát triển mạnh
- Hệ thống thư viện hỗ trợ lớn
- Hỗ trợ trên nhiều hệ điều hành

3. Công cụ thiết kế phần cứng

   Phần cứng của đồ án được thiết kế bằng phần mềm Altium Designer 17. Đây là phần mềm chuyên dụng được dùng trong lĩnh vực thiết kế mạch điện tử.

Chương 2. Thiết kế phần cứng

Chương này sẽ mô tả chi tiết các thành phần phần cứng của thiết bị và kèm với file nguyên lý. Khối điều khiển của thiết bị được xây dựng dựa trên mô-đun ESP8266 nodeMCU, bao gồm các chức năng: đọc tín hiệu nút bấm, điều khiển đóng cắt triac thông qua các IC lái triac.

2.1. Giới thiệu các linh kiện sử dụng

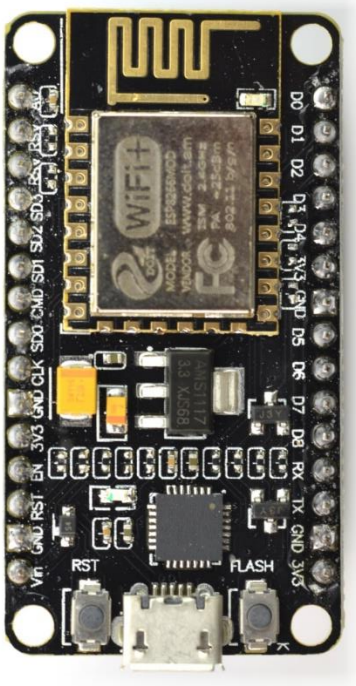
2.1.1. Vi điều khiển ESP8266

ESP8266 là dòng chip tích hợp Wi-Fi 2.4Ghz có thể lập trình được, rẻ tiền được sản xuất bởi một công ty bán dẫn Trung Quốc: Espressif Systems.

Được phát hành đầu tiên vào tháng 8 năm 2014, đóng gói đưa ra thị trường dạng Mô dun ESP-01, được sản xuất bởi bên thứ 3: AI-Thinker. Có khả năng kết nối Internet qua mạng Wi-Fi một cách nhanh chóng và sử dụng rất ít linh kiện đi kèm. Với giá cả có thể nói là rất rẻ so với tính năng và khả năng ESP8266 có thể làm được.

ESP8266 có một cộng đồng các nhà phát triển trên thế giới rất lớn, cung cấp nhiều Mô-đun lập trình mã mở giúp nhiều người có thể tiếp cận và xây dựng ứng dụng rất nhanh.

Hiện nay tất cả các dòng chip ESP8266 trên thị trường đều mang nhãn ESP8266EX, là phiên bản nâng cấp của ESP8266. Trong đồ án này sử dụng kit phát triển cho chip ESP8266 là NodeMCU.



Hình 4. Mạch phát triển nodeMCU

Thông số phần cứng:

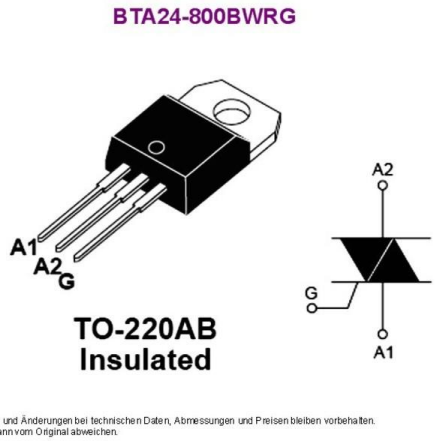
- 32-bit RISC CPU: Tensilica Xtensa LX106 running at 80 MHz
- Hổ trợ Flash ngoài từ 512KiB đến 4MiB

- 64KBytes RAM thực thi lệnh
- 96KBytes RAM dữ liệu
- 64KBytes boot ROM
- Chuẩn wifi EEE 802.11 b/g/n, Wi-Fi 2.4 GHz Tích hợp TR switch, balun, LNA, khuếch đại công suất và matching network Hỗ trợ WEP, WPA/WPA2, Open network
- Tích hợp giao thức TCP/IP
- Hổ trợ nhiều loại ăng-ten
- 16 chân GPIO
- Hổ trợ SDIO 2.0, UART, SPI, I^2C, PWM, I^2S với DMA
- 1 ADC 10-bit
- Dải nhiệt độ hoạt động rộng: -40^oC ~ 125^oC

2.1.2. Triac

Hai phần tử đóng cắt thường được dùng khi cần đóng cắt tải ở điện áp xoay chiều là rơ-le và triac. Cả triac và rơ-le đều có khả năng đóng cắt dòng điện xoay chiều tuy nhiên khi đóng cắt, rơ-le sử dụng cuộn hút và sẽ tạo ra tiếng "cạch" khi 2 tiếp điểm cơ khí chạm hoặc tách nhau ra. Trong khi đó triac đóng cắt các kênh bán dẫn nên không gây tiếng động. Đây là yếu tố chính khiến cho triac và rơ-le được ứng dụng vào những công việc khác nhau. So với rơ-le, tiếp điểm triac có tuổi thọ cao hơn do không có yếu tố cơ khí, tuổi thọ của triac có thể lên đến hàng triệu lần đóng cắt. Nhờ tuổi thọ đóng cắt lớn mà triac có thể được sử dụng để điều khiển điện áp ra thông qua điều khiển đóng mở triac bằng cách điều chế độ rộng xung. Đồng thời việc không sử dụng tiếp điểm cơ khí nên không có hồ quang khi đóng cắt, từ đó có thể được sử dụng trong môi trường đặc biệt nghiêm cấm tia lửa điện. Nhờ những ưu điểm trên của triac so với rơ-le, em quyết định lựa chọn triac nhằm tăng tuổi thọ và độ an toàn của thiết bị.

Triac là phần tử bán dẫn có cấu trúc bán dẫn gồm năm lớp, tạo nên cấu trúc p-n-p-n như ở Tiristo theo cả hai chiều giữa cực A1 và A2. Về nguyên tắc, triac hoàn toàn có thể coi tương đương với 2 tiristo đấu song song ngược. Triac có thể điều khiển mở dẫn dòng bằng cả xung dòng dương (dòng đi vào cực điều khiển) hoặc dung dòng âm (dòng đi ra khỏi cực điều khiển). Tuy nhiên xung dòng điều khiển xung dòng điều khiển âm có độ nhạy kém hơn, nghĩa là dòng chỉ có thể chạy qua triac khi điện áp giữa A1 và A2 phải lớn hơn một giá trị nhất định, lớn hơn khi dùng dòng điều khiển dương. Vì vậy trong thực tế để đảm bảo tính đối xứng của dòng điện qua triac thì sử dụng xung điều khiển âm là tốt hơn cả. Triac đặc biệt hữu ích trong các ứng dụng cần điều chỉnh điện áp xoay chiều hoặc các công tắc tơ tĩnh ở dải công suất vừa và nhỏ [1]. Sau đây là hình ảnh thực và mô hình của một triac:



Hình 5. Hình ảnh thực và mô hình Triac

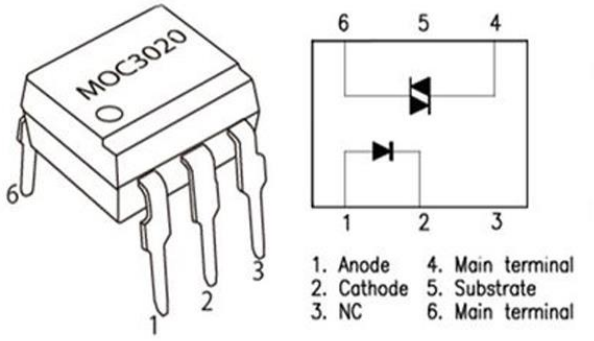
Một số thông số chính của triac:

Bảng 1. Một số thông số chính của triac [2]

V_{DRM}/V_{RRM}	Điện áp đỉnh tối đa giữa 2 cực chính của triac
I_T	Dòng điện hiệu dụng tối đa chảy qua triac
dI/dt	Tốc độ tăng tối đa của dòng điện chảy qua triac
I^2t	Bảo vệ quá dòng
I_{GT}	Dòng điện mở cực cổng triac
V_{GT}	Điện áp mở cực cổng triac
I_{GM}	Dòng điện tối đa ở cực cổng triac

2.1.3. IC lái triac

IC lái triac là IC cách ly quang được thiết kế đặc biệt để cung cấp dòng điều khiển cho triac đồng thời cung cấp phương tiện để cách ly mạch điều khiển với mạch lực. IC lái triac gồm đầu vào là 1 điốt phát quang (LED) và đầu ra là một triac với cực cổng nhạy sáng. Khi có dòng điện thuận (thường được cung cấp bởi khối vi điều khiển) chạy qua điốt phát quang, các phô tôn được sinh ra và truyền đến cực cổng của triac. Nếu dòng điện này lớn hơn dòng điện kích hoạt của điốt phát quang bên trong IC lái triac thì triac sẽ mở kênh dẫn. Triac bên trong IC lái sẽ giữ nguyên trạng thái dẫn này cho đến khi dòng điện chạy qua triac giảm xuống dưới dòng điện giữ của triac. Khi đạt đến giá trị dòng điện giữ này triac sẽ trở về trạng thái không dẫn. Ngoài ra còn có loại IC lái triac với chức năng phát hiện "điểm 0". Những IC lái này chỉ kích hoạt khi điện áp xoay chiều ở gần giá trị 0V góp phần giới hạn dòng điện tăng vọt khi mở kênh dẫn (thường thấy trong các tải trở) và hạn chế gây nhiễu điện. IC lái triac không nên được sử dụng để đóng cắt tải trực tiếp mà chỉ nên được sử dụng với vai trò là thiết bị điều khiển các triac lực. Dưới đây là hình ảnh thực tế và mô hình bên trong của một IC lái triac, nối giữa chân số 1 và chân số 2 là một điốt phát quang, nối giữa chân số 4 và chân số 6 là một triac với cực cổng nhạy sáng.



Hình 6. IC cách ly quang lái triac

Một số thông số chính của IC cách ly quang lái triac:

Bảng 2. Một số thông số chính của IC cách ly quang lái triac

V_{DRM}	Điện áp đỉnh tối đa giữa 2 cực chính của triac nhạy sáng
I_{TSM}	Dòng điện đỉnh chảy qua triac nhạy sáng
V_{TM}	Điện áp rơi trên triac
I_F	Dòng điện thuận qua LED đầu vào
V_F	Điện áp thuận trên LED đầu vào
I_R	Dòng điện rò qua LED đầu vào

2.1.4. Mô-đun chuyển đổi nguồn AC/DC

Mô-đun nguồn AC-DC Hi-Link HLK-PM01 5VDC 3W có thiết kế nhỏ gọn với vỏ bọc nhựa an toàn, chuyên nghiệp, được sử dụng để chuyển nguồn xoay chiều sang điện áp một chiều 5V với công suất tối đa 3W để cấp nguồn cho thiết bị, mô-đun được sản xuất bởi hãng Hi-Link chuyên về các mô-đun nguồn được sử dụng trong công nghiệp với độ bền, chống nhiễu tốt và độ an toàn cao.

Hình 7. Khối chuyển nguồn AC/DC 5V 3W

Thông số kĩ thuật:

- Điện áp ngõ vào: 100~240 VAC/50~60 Hz
- Điện áp ngõ ra: 5 VDC
- Công suất trung bình: 3W
- Răng cưa và nhiễu nhỏ
- Bảo vệ quá tải và ngắn mạch
- Hiệu suất 72%

2.1.5. Các linh kiện thụ động khác

Điện trở: Điện trở là một linh kiện điện tử thụ động gồm 2 tiếp điểm kết nối, thường được dùng để hạn chế cường độ dòng điện chảy trong mạch, điều chỉnh mức độ tín hiệu, dùng để chia điện áp, kích hoạt các linh kiện điện tử chủ động như transistor, tiếp điểm cuối trong đường truyền điện và có trong rất nhiều ứng dụng khác. Điện trở công suất có thể tiêu tán một lượng lớn điện năng chuyển sang nhiệt năng có trong các bộ điều khiển động cơ, trong các hệ thống phân phối điện. Các điện trở thường có trở kháng cố định, ít bị thay đổi bởi nhiệt độ và điện áp hoạt động. Biến trở là loại điện trở có thể thay đổi được trở kháng như các núm vặn điều chỉnh âm lượng. Các loại cảm biến có điện trở biến thiên như: cảm biến nhiệt độ, ánh sáng, độ ẩm, lực tác động và các phản ứng hóa học. Điện trở là loại linh kiện phổ biến trong mạng lưới điện, các mạch điện tử, Điện trở thực tế có thể được cấu tạo từ nhiều thành phần riêng rẽ và có nhiều hình dạng khác nhau, ngoài ra điện trở còn có thể tích hợp trong các vi mạch IC. Điện trở được phân loại dựa trên khả năng chống chịu, trở kháng, ... tất cả đều được các nhà sản xuất ký hiệu trên nó.

Tụ điện: Tụ điện là một loại linh kiện điện tử thụ động tạo bởi hai bề mặt dẫn điện được ngăn cách bởi điện môi. Khi có chênh lệch điện thế tại hai bề mặt, tại các bề mặt sẽ xuất hiện điện tích cùng điện lượng nhưng trái dấu. Sự tích tụ của điện tích trên hai bề mặt tạo ra khả năng tích trữ năng lượng điện trường của tụ điện. Khi chênh lệch điện thế trên hai bề mặt là điện thế xoay chiều, sự tích luỹ điện tích bị chậm pha so với điện áp, tạo nên trở kháng của tụ điện trong mạch điện xoay chiều. Về mặt lưu trữ năng lượng, tụ điện có phần giống với ắc qui. Mặc dù cách hoạt động của chúng thì hoàn toàn khác nhau, nhưng chúng đều cùng lưu trữ năng lượng điện. Ắc qui có 2 cực, bên trong xảy ra phản ứng hóa học để tạo ra electron ở cực này và chuyển electron sang cực còn lại. Tụ điện thì đơn giản hơn, nó không thể tạo ra electron - nó chỉ lưu trữ chúng. Tụ điện có khả năng nạp và xả rất nhanh. Đây là một ưu thế của nó so với ắc qui.

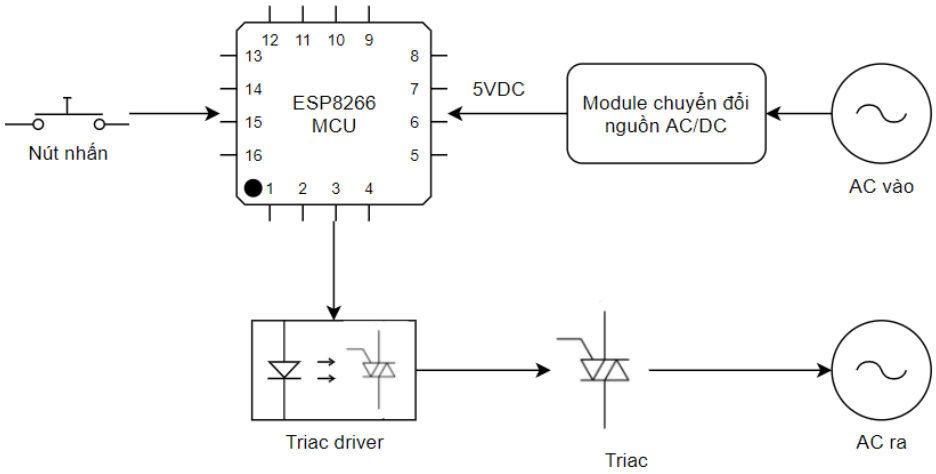
Đèn LED: LED (viết tắt của Light Emitting Diode, có nghĩa là diode phát sáng hoặc diode phát quang) là các diode có khả năng phát ra ánh sáng hay tia hồng ngoại, tử ngoại. Cũng giống như diode, LED được cấu tạo từ một khối bán dẫn loại p ghép với một khối bán dẫn loại n.

Nút bấm: Nút ấn là một loại công tắc đơn giản điều khiển hoạt động của máy hoặc một số loại quá trình. Hầu hết, các nút nhấn là nhựa hoặc kim loại. Hình

dạng của nút ấn có thể phù hợp với ngón tay hoặc bàn tay để sử dụng dễ dàng. Tất cả phụ thuộc vào thiết kế cá nhân. Nút ấn có 2 loại chính là nút nhấn thường mở hoặc nút nhấn thường đóng. Nút nhấn có ba phần: Bộ truyền động, các tiếp điểm cố định và các rãnh. Bộ truyền động sẽ đi qua toàn bộ công tắc và vào một xy lanh mỏng ở phía dưới. Bên trong là một tiếp điểm động và lò xo. Khi nhấn nút, nó chạm vào các tiếp điểm tĩnh làm thay đổi trạng thái của tiếp điểm. Trong một số trường hợp, người dùng cần giữ nút hoặc nhấn liên tục để thiết bị hoạt động. Với các nút nhấn khác, chốt sẽ giữ nút bật cho đến khi người dùng nhấn nút lần nữa. Công tắc nút nhấn sử dụng nhiều trong các ứng dụng khác nhau như máy tính, điện thoại nút nhấn và nhiều thiết bị gia dụng. Bạn có thể nhìn thấy chúng trong nhà, văn phòng và trong các ứng dụng công nghiệp ngày nay. Chúng có thể bật, tắt máy hoặc làm cho thiết bị thực hiện các hoạt động cụ thể, như trường hợp với máy tính. Trong một số trường hợp, các nút nhấn có thể kết nối thông qua liên kết cơ học, điều khiển một nút nhấn khác hoạt động. Đa số, các nút sẽ có màu sắc cụ thể để biểu thị mục đích của chúng. Ví dụ như nút nhất màu xanh thường được sử dụng để bật thiết bị hay nút nhấn màu đỏ để tắt thiết bị. Điều này tránh gây nên một sô nhầm lẫn. Nút dừng khẩn cấp thường là các nút ấn lớn, thường có màu đỏ và có đầu lớn hơn để sử dụng dễ dàng hơn.

2.2. Sơ đồ tổng quan phần cứng

Phần cứng của thiết bị được cấp nguồn bởi mô-đun chuyển đổi nguồn từ 220V xoay chiều sang 5V một chiều. Vi điều khiển ESP8266 hoạt động với nguồn cấp 3.3V một chiều nhưng trên mạch phát triển nodeMCU đã có sẵn IC hạ áp AMS1117 từ 5V xuống 3.3VDC nên chỉ cần một mô-đun nguồn duy nhất cung cấp 5VDC cho toàn bộ hệ thống.



Hình 8. Mô hình phần cứng thiết bị

2.3. Thiết kế mạch

2.3.1. Lựa chọn các linh kiện chính và giải thích

Lựa chọn triac: Với yêu cầu đóng cắt tải điện trở ở điện áp hiệu dụng 220V xoay chiều với dòng điện hiệu dụng 20A, em lựa chọn phần tử đóng cắt là triac BTA24 với điện áp tối đa giữa 2 cực chính là 600V xoay chiều, dòng điện hiệu dụng chạy qua triac tối đa là 25A, dòng điện mở triac tối đa là I_{GT} = 50mA.

Lựa chọn IC lái triac: Với yêu cầu dòng điều khiển triac tối đa I_{GT} = 50mA, em lựa chọn IC lái triac với dòng ra $I_{TM} > I_{GT}$ và điện áp ở trạng thái mở tối đa là 310V xoay chiều. Từ những yêu cầu trên em lựa chọn IC điều khiển MOC3020 với I_{TM} = 100mA và hiệu điện thế hở tại các cực chính tối đa là 400V xoay chiều.

Lựa chọn vi điều khiển: Với yêu cầu đóng cắt tại chỗ và đóng cắt qua WiFi, xử lý các tác vụ: đọc nút nhấn, gửi bản tin trạng thái, nhận bản tin điều khiển, điều khiển đóng cắt, đếm thời gian. Em quyết định lựa chọn vi điều khiển ESP8266 với khả năng kết nối internet thông qua WiFi và có thể xử lý các tác vụ trên.

Lựa chọn khối nguồn cho thiết bị: Vi điều khiển ESP8266 sử dụng điện áp một chiều 3.3V. Nhưng do sử dụng mạch phát triển nodeMCU đã có sẵn IC nguồn tuyến tính AMS1117 chuyển từ 5V một chiều sang 3.3V một chiều nên em sẽ sử dụng trực tiếp nguồn một chiều 5V. Mạch phát triển nodeMCU tiêu thụ dòng điện 40mA trung bình. Thiết bị sử dụng 5 đèn LED 5mm hiển thị tiêu thụ dòng điện một chiều 10mA ở điện áp 3.3V một chiều. Từ các thông tin trên, em có bảng tiêu thụ công suất như dưới đây:

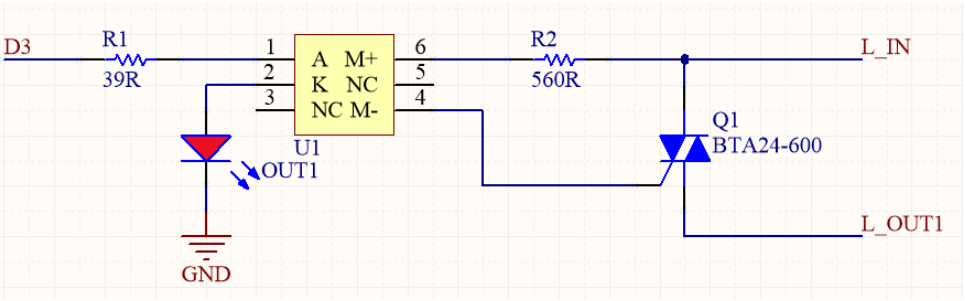
Bảng 3. Bảng tính toán công suất yêu cầu của nguồn cho thiết bị

Linh kiện	Điện áp (V)	Dòng điện (A)	Công suất (W)	Số lượng
Mạch phát triển nodeMCU	5	0.04	0.2	1
Đèn LED 5mm	3.3	0.01	0.033	5
Tổng công suất (W)	0.033			

Từ bảng liệt kê công suất trên, em quyết định lựa chọn khối chuyển nguồn từ 220V xoay chiều về 5V một chiều có công suất nhỏ nhất của hãng Hi-Link là mô-đun AC/DC 3W 5V HLK-PM01 với công suất đầu ra tối đa là 3W để đảm bảo cho thiết bị hoạt động ổn định.

2.3.2. Thiết kế khối đóng cắt

Khối đóng cắt mỗi kênh gồm 1 IC lái triac MOC3020 (U1) và 1 triac BTA24. Dưới đây là sơ đồ nguyên lí của khối đóng cắt triac.



Hình 9. Nguyên lý khối đóng cắt triac

Để kích hoạt được MOC3020 cần có dòng điện một chiều với cường độ 10mA chạy từ cực anot đến cực katot của điôt phát quang bên trong IC. Từ đó em tính được giá trị R1 để điều khiển MOC3020 như sau:

	$R_1 = \frac{U_{MCU} - U_D - U_{LED}}{I_{MCU}} = \frac{3.3 - 1.15 - 1.6}{0.01} = 55\Omega$	*PT 1*
Trong đó:	U_{MCU}: điện áp ra của chân vi điều khiển	
	U_D: điện áp rơi trên điốt phát quang cách ly	
	U_{LED}: điện áp rơi trên LED đỏ khi có dòng 10mA	
	I_{MCU}: dòng điện kích hoạt của MOC3020	

Em chọn giá trị R_1 = 39Ω để đảm bảo MOC3020 được kích hoạt hoàn toàn khi có tín hiệu điều khiển mức cao với điện áp 3.3VDC từ vi điều khiển.

Để hạn chế dòng điện chảy vào triac nội trong IC lái MOC3020 dưới mức dòng điện đỉnh chịu được của MOC3020: $I_{TSM} = 1A$. Ta tìm được giá trị R_2 tối thiểu:

	$R_2 min = \frac{V_{in(peak)}}{I_{TSM}}$ [3] $= \frac{220\sqrt{2}}{1} = 310\Omega$	*PT 2*
Trong đó:	$V_{in(peak)}$: điện áp đỉnh khi đóng cắt	
	I_{TSM}: dòng điện đỉnh của triac nội của IC lái	

Để lựa chọn giá trị R_2, ta cần xét đến sự phụ thuộc của điện áp mở triac lực vào điện trở R_2 theo phương trình sau:

	$V_{inT} = R_2 * I_{GT} + V_{GT} + V_{TM}$	*PT 3*
Trong đó:	V_{inT}: Điện áp cần đặt vào để mở triac lực	
	I_{GT}: Dòng điện cần thiết để mở triac lực	
	V_{GT}: Điện áp cần đặt vào cực cổng để mở triac lực	
	V_{TM}: Điện áp rơi đỉnh trên triac nội IC lái	

Cần lựa chọn giá trị R_2 dòng diện đỉnh chảy vào IC lái không được vượt quá dòng điện đỉnh chịu được của triac nội và giá trị điện áp cần đặt vào để mở triac lực không quá lớn. Chọn giá trị $R_2 = 560\Omega$, kiểm tra lại bằng công thức PT 3:

	$V_{inT} = R_2 * I_{GT} + V_{GT} + V_{TM}$ $= 560 * 0.0025 + 1.3 + 3 = 5.7\ (V)$	*PT 4*

Để tính công suất tối đa trên điện trở R_2, ta sử dụng giá trị dòng điện I_{GT}. Công thức tính công suất trên điện trở R_2 như sau:

	$P_{R2} = R_2 * {I_{GT}}^2$ $= 560 * 0.0025^2 = 3.5\ (mW)$	*PT 5*

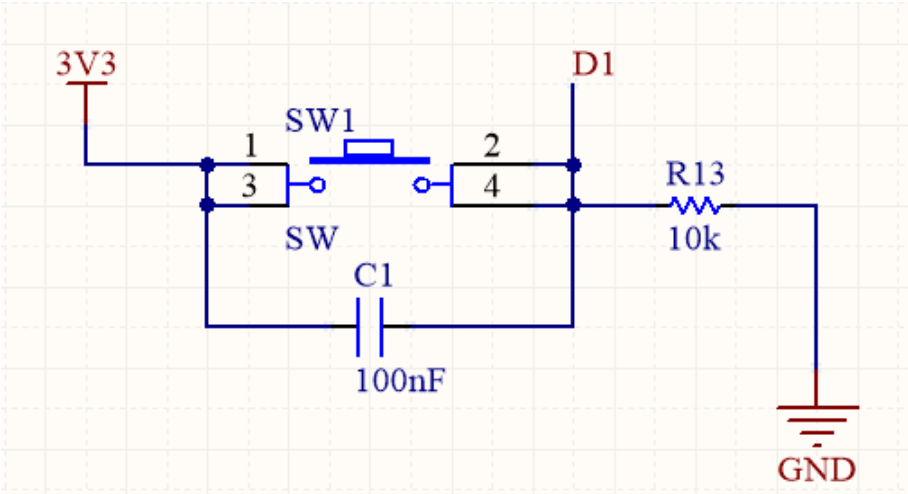
Do dòng điện chỉ chảy qua điện trở R_2 khoảng thời gian rất nhỏ mỗi nửa chu kì nên ta có thể lựa chọn điện trở $R_2 = 560\Omega$ công suất 1/4W.

2.3.3. Thiết kế khối đọc tín hiệu nút bấm

Khối đọc nút nhấn gồm:

- Nút nhấn nhả
- Tụ điện 100nF chống dội nút nhấn
- Điện trở 10K Ohm kéo xuống đất

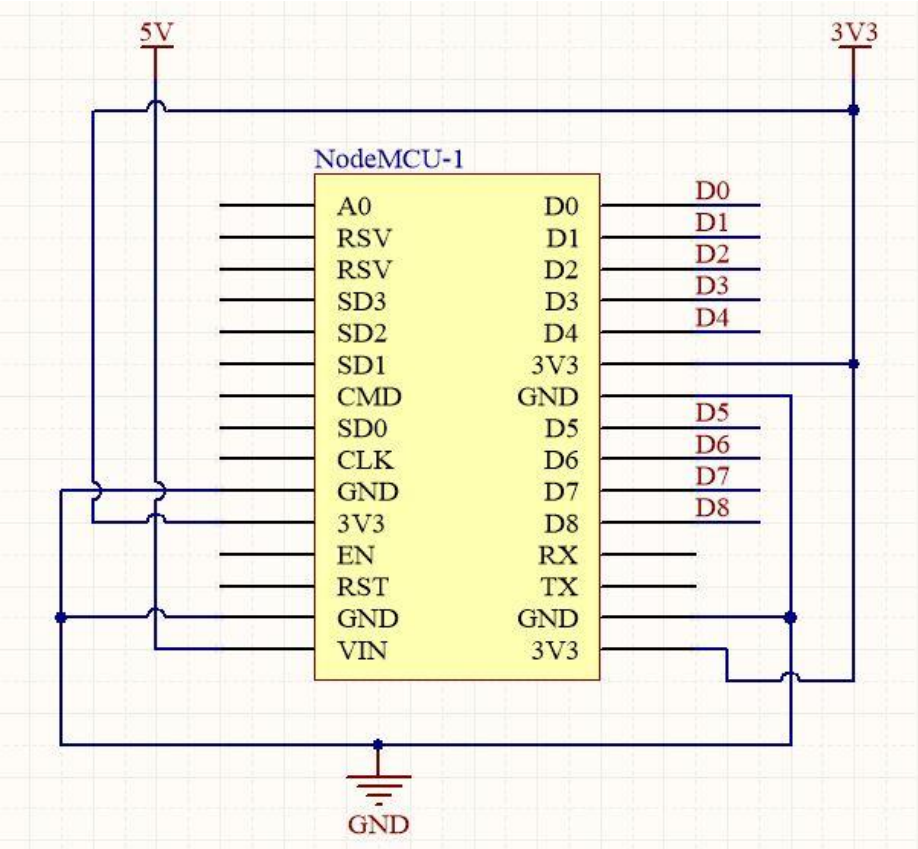
Nút nhấn được thiết kế hoạt động ở chế độ tích cực mức cao. Khi không nhấn vi điều khiển sẽ đọc chân nút nhấn có mức logic thấp. Khi nút nhấn được nhấn vi điều khiển sẽ đọc chân nút nhấn có mức logic cao và xử lý tương ứng.



Hình 10. Sơ đồ nguyên lý khối nút nhấn

2.3.4. Thiết kế khối điều khiển

Khối điều khiển sử dụng vi điều khiển ESP8266 làm trung tâm. Mạch phát triển nodeMCU cho phép cấu hình các chân GPIO ở các chế độ khác nhau. Ở đây sử dụng 4 chân GPIO để đọc các nút nhấn, 4 chân GPIO để điều khiển các kênh đóng cắt đầu ra và 1 chân GPIO để điều khiển LED hiển thị trạng thái thiết bị. Dưới đây là sơ đồ nguyên lý khối điều khiển và sơ đồ cấu hình các chân của nodeMCU:



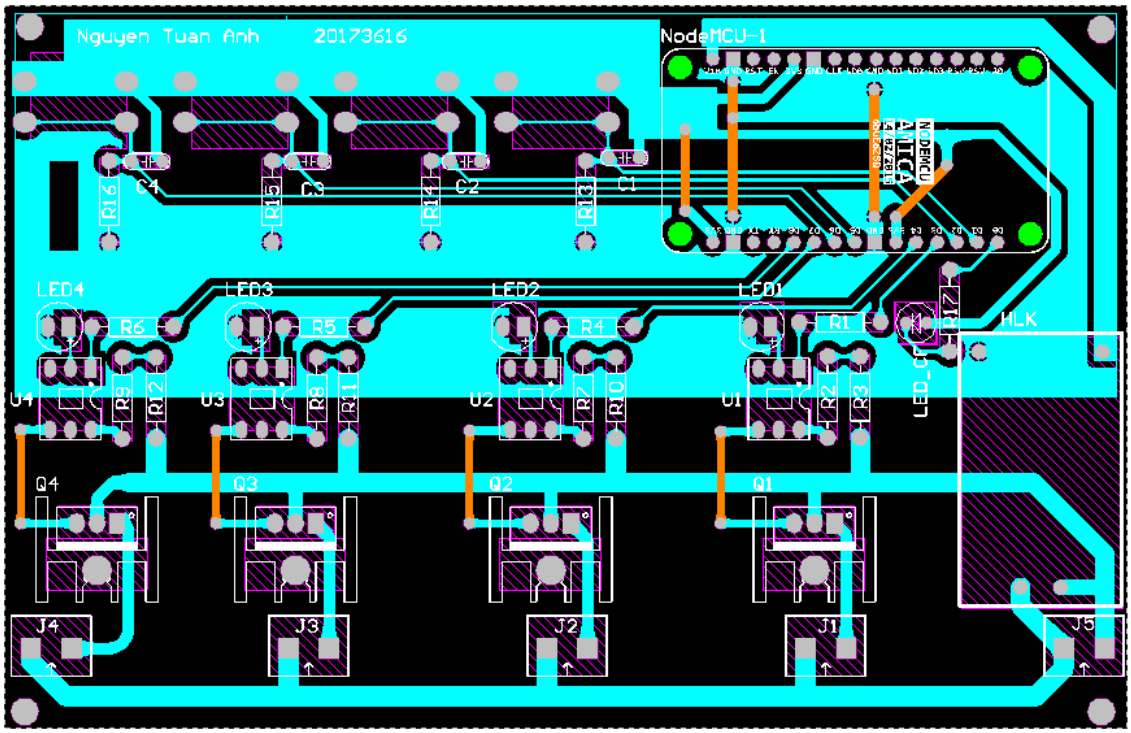
Hình 11. Sơ đồ nguyên lý khối điều khiển

Bảng 4. Sơ đồ cấu hình chân GPIO cho nodeMCU

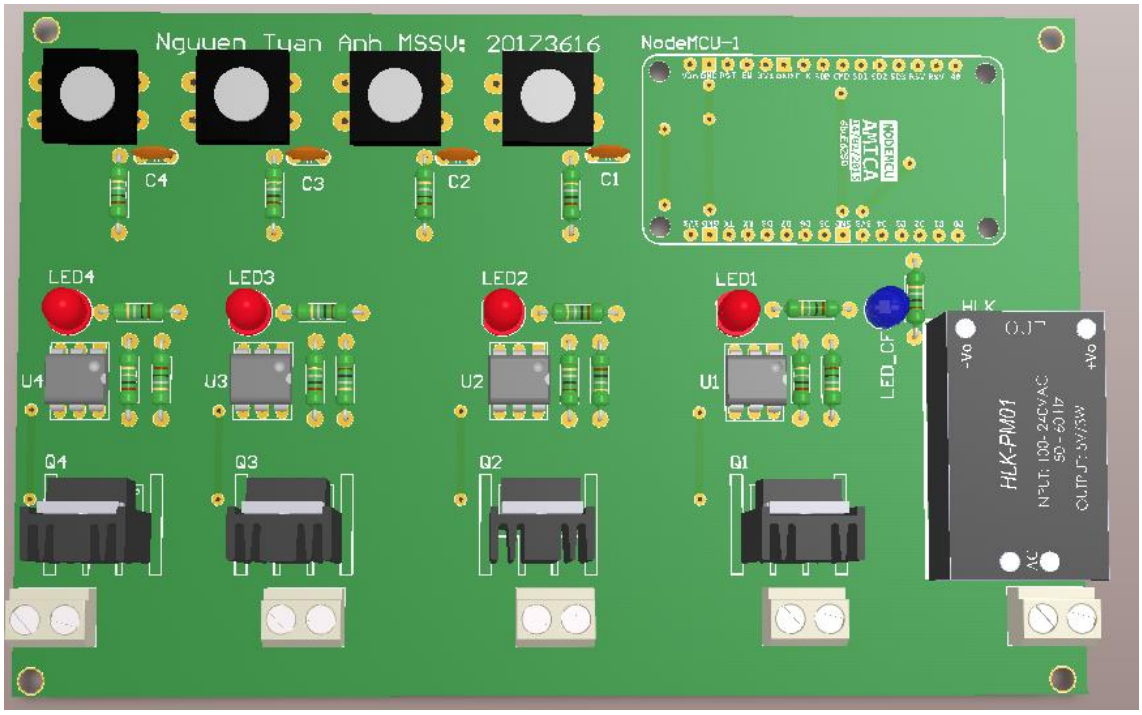
Nội dung	**Chân GPIO sử dụng**
Nút nhấn 1	D1
Nút nhấn 2	D2
Nút nhấn 3	D5
Nút nhấn 4	D6
Kênh điều khiển triac 1	D3
Kênh điều khiển triac 2	D4
Kênh điều khiển triac 3	D7
Kênh điều khiển triac 4	D8
Đèn chỉ thị	D0

2.3.5. Thiết kế mạch in

Mạch in được thiết kế sử dụng phần mềm Altium Designer 17. Kết quả đạt được mạch in 2 lớp như sau:



Hình 12. Ảnh chụp 2 chiều của mạch in.



Hình 13. Ảnh chụp 3 chiều của mạch in

Chương 3. Phát triển firmware

3.1. Giới thiệu một số giao thức truyền thông sử dụng trong thiết bị điện thông minh

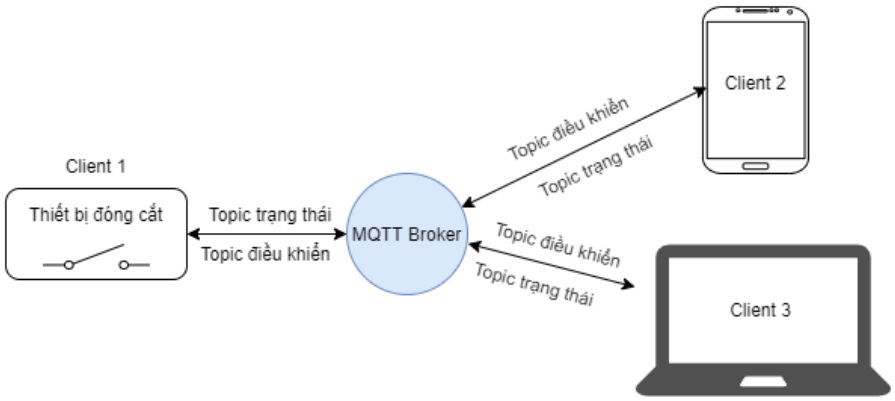
Trong IoT việc giao tiếp giữa các thiết bị với nhau là một trong các yếu tố cơ bản, từ đó dẫn đến yêu cầu phải có một giao thức chung để các thiết bị với phần cứng khác nhau có thể giao tiếp với nhau. Các giao thức truyền dữ liệu phổ biến trong lĩnh vực thiết bị điện thông minh hiện nay là CoAP, MQTT, AMQP, …

- Giao thức CoAP: Giao thức được thiết kế cho các thiết bị nhúng, về cơ bản giao thức CoAP tương tự như HTTP nhưng các gói tin nhỏ hơn, chủ yếu là giao thức 1-1 để truyền trạng thái thông tin giữa máy khách và máy chủ.
- Giao thức AMQP: Giao thức AMQP được thiết kế với các gói tin được định hướng, xếp hàng và định tuyến có độ tin cậy và bảo mật cao.
- Giao thức MQTT: MQTT là giao thức truyền thông theo mô hình xuất bản/đăng kí (Publish/Subscribe).

3.2. Giới thiệu giao thức truyền thông MQTT

MQTT là giao thức truyền thông theo mô hình Publish/Subscribe, sử dụng băng thông thấp, có độ tin cậy cao, có khả năng hoạt động trong môi trường đường truyền không ổn định.

3.2.1. Mô hình của giao thức MQTT



Hình 14. Mô hình giao thức MQTT

Mô hình gồm 2 phần chính là là Máy chủ (với vai trò máy chủ) và Máy khách (máy khách):

- Máy chủ đóng vai trò máy chủ đảm nhận vai trò nhận và chuyển các bản tin. Máy chủ có thể cài đặt trên các máy tính nhúng, hoặc sử dụng Máy chủ trên đám mây
- Máy khách có thể là các thiết bị chấp hành, các cảm biến, đám mây nhận dữ liệu, các thiết bị điện thông minh, …
- Máy khách của MQTT gửi và nhận bản tin dựa trên mô hình Topic. Để nhận bản tin của 1 Topic, Máy khách sẽ Subscribe (đăng kí) Topic đó và khi có bản tin được Publish (xuất bản) đến Topic đó Máy chủ sẽ tự động gửi bản tin đến Máy khách.

- Máy chủ nhận tin nhắn từ các Máy khách gửi đến các Topic, chuyển tiếp bản tin hoặc lưu lại. Ngoài ra Máy chủ cũng đảm nhận thêm một số tính năng khác như bảo mật, lưu trữ thông tin các máy khách đã kết nối đến máy chủ
- QoS: Mức độ tin cậy khi gửi bản tin. Có 3 mức QoS: 0,1 và 2
  - QoS 0: Máy chủ/Máy khách gửi bản tin đúng 1 lần
  - QoS 1: Máy chủ/Máy khách gửi bản tin cho đến khi có xác nhận từ bên nhận
  - QoS 2: Máy chủ/Máy khách gửi bản tin và đảm bảo có xác nhận từ cả 2 phía
- Retain: Bản tin có flag Retain sẽ được Máy chủ lưu lại tại Topic và chuyển đến các Máy khách mới Subscribe Topic đó.
- Birth/Death/LWT:
  - Birth là bản tin được gửi tới các Máy khách khi có thiết bị mới kết nối
  - Death là bản tin được gửi khi có thiết bị mất kết nối
  - LWT là bản tin được cài đặt bởi Máy khách, sẽ được gửi tới Topic chỉ định khi Máy khách đó mất kết nối
- Mức độ bảo mật: 1 lớp xác thực bằng ID và mật khẩu khi các Máy khách kết nối với Máy chủ. Mức độ bảo mật có thể tăng thêm bằng các giải pháp bảo mật ở tầng mạng.

3.2.2. Lựa chọn giao thức MQTT

Từ mô hình của giao thức MQTT như trên chúng ta có thể thấy một số ưu điểm và nhược điểm của giao thức MQTT:

- Ưu điểm:
  - Yêu cầu băng thông thấp
  - Thích hợp với các thiết bị nhúng
  - Chi phí triển khai thấp
  - Dễ dàng nâng cấp hệ thống
  - Các máy khách dễ kết nối lại khi bị mất kết nối
  - Có các tính năng giúp duy trì kết nối và phát hiện khi các node bị mất kết nối
- Nhược điểm:
  - Sử dụng Máy chủ nên khi Máy chủ gặp sụ cố sẽ cả hệ thống sẽ mất kết nối. Để đảm bảo kết nối thông suốt cần có Máy chủ dự phòng
  - Số lượng Máy khách bị giới hạn bởi cấu hình của Máy chủ
  - Cần sử dụng các biện pháp bên ngoài để tăng tính bảo mật cho hệ thống

Có thể thấy đây là một giao thức gọn nhẹ, được thiết kế để kết nối các thiết bị sử dụng mạng internet băng thông thấp rất phù hợp với các thiết bị điện thông minh.

3.2.3. Thông tin máy chủ và quy định bản tin MQTT

- Thông tin máy chủ MQTT

Bảng 5. Thông tin máy chủ MQTT sử dụng

Máy chủ	broker.hivemq.com
Cổng	1883

- Quy định bản tin MQTT: Bản tin MQTT được quy định để tạo sự thống nhất giữa các thiết bị, giúp công việc xử lý truyền thông dễ dàng hơn.

Bảng 6. Quy định bản tin MQTT

Nội dung	Topic	Payload
Trạng thái thiết bị đóng cắt	status/datnta	/1/x/2/x/3/x/4/x - x là trạng thái của từng kênh. x = 1: kênh đang dẫn. x = 0: kênh mở ra VD: /1/0/2/1/3/1/4/0: kênh 1 và 4 đang mở, kênh 2 và 3 đang dẫn.
Lệnh điều khiển thiết bị đóng cắt	cmd/datnta	/kênh đóng cắt/điều khiển/thời gian - Kênh đóng cắt: 1, 2, 3, 4 - Điều khiển: 0 hoặc 1 tương ứng là mở hoặc dẫn - Thời gian: thời gian dẫn nếu lệnh điều khiển là 1, thời gian trễ mở kênh nếu lệnh điều khiển là 0. Đơn vị tính bằng giây. VD: /1/1/60: điều khiển kênh 1 dẫn trong 60 giây rồi mở ra. /1/1/00: điều khiển kênh 1 dẫn không giới hạn thời gian.

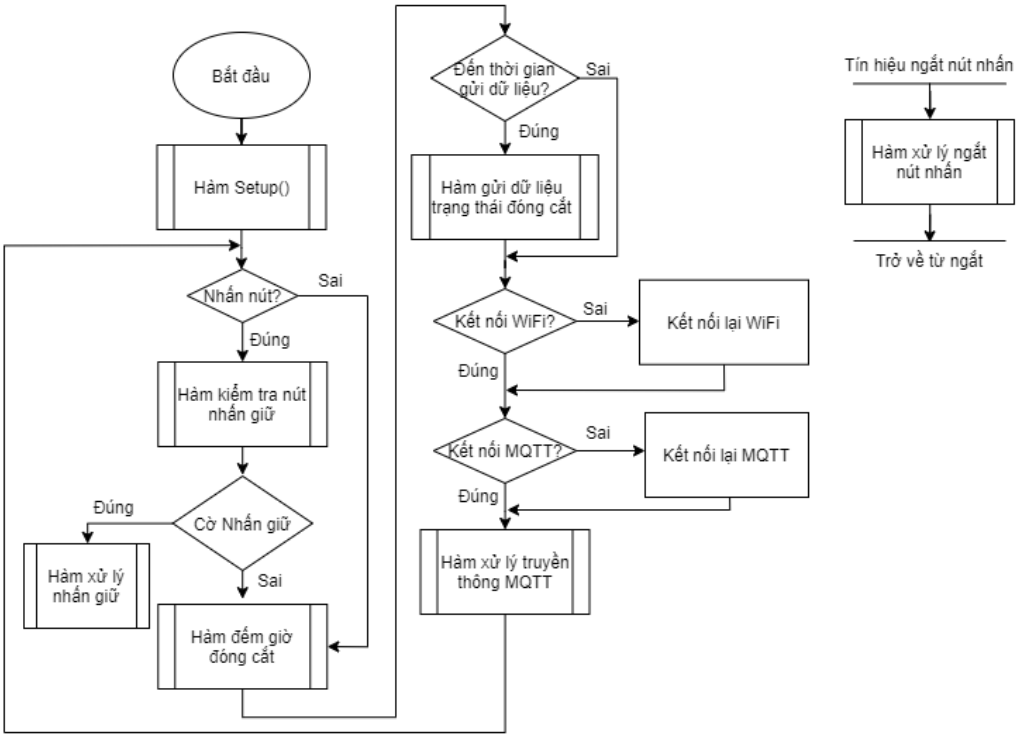
3.3. Lưu đồ thuật toán và chương trình cho Vi điều khiển

Lưu đồ thuật toán hoạt động của vi điều khiển: Khi thiết bị được cấp nguồn, hàm Setup sẽ được gọi để cầu hình các chân GPIO vào cho các nút nhấn, các chân GPIO ra để điều khiển triac và đèn hiển thị trạng thái thiết bị, kiểm tra bộ nhớ để ra quyết định cấu hình WiFi hoặc kết nối với WiFi có sẵn. Sau khi kết thúc quá trình khởi tạo ban đầu, vi điều khiển vào vòng lặp vô tận với chu trình lần lượt như sau:

- Kiểm tra nút nhấn giữ
- Theo dõi thời gian cho chế độ đóng cắt có hẹn giờ

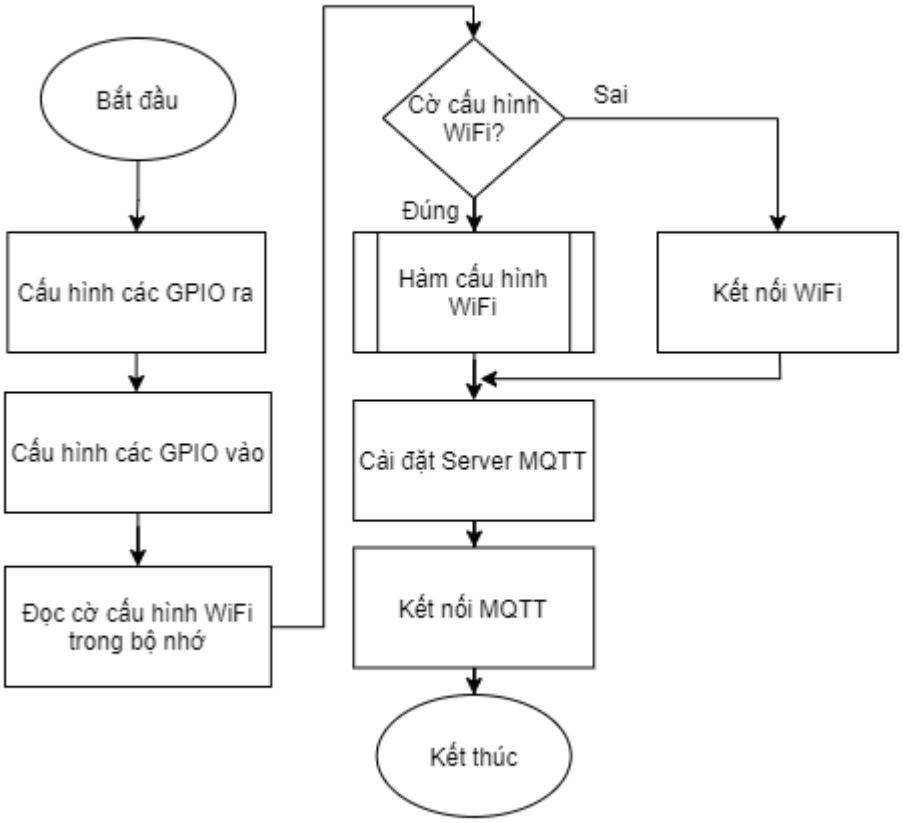
- Gửi dữ liệu về trạng thái đóng cắt
- Kiểm tra kết nối WiFi
- Kiểm tra kết nối với máy chủ MQTT

Ngoài ra chương trình cho vi điều khiển cũng sử dụng ngắt để đọc các nút nhấn nhận tín hiệu từ người sử dụng.



Hình 15. Lưu đồ thuật toán cho vi điều khiển

- Hàm Setup: Hàm làm nhiệm vụ cấu hình các chân GPIO vào và ra cho thiết bị. Đọc bộ nhớ để xác định kết nối vơi WiFi đã lưu hay đưa thiết bị vào chế độ cấu hình WiFi để người sử dụng nhập WiFi mới. Cài đặt thông tin máy chủ MQTT và kết nối đến máy chủ MQTT.



Hình 16. Lưu đồ thuật toán hàm cấu hình ban đầu

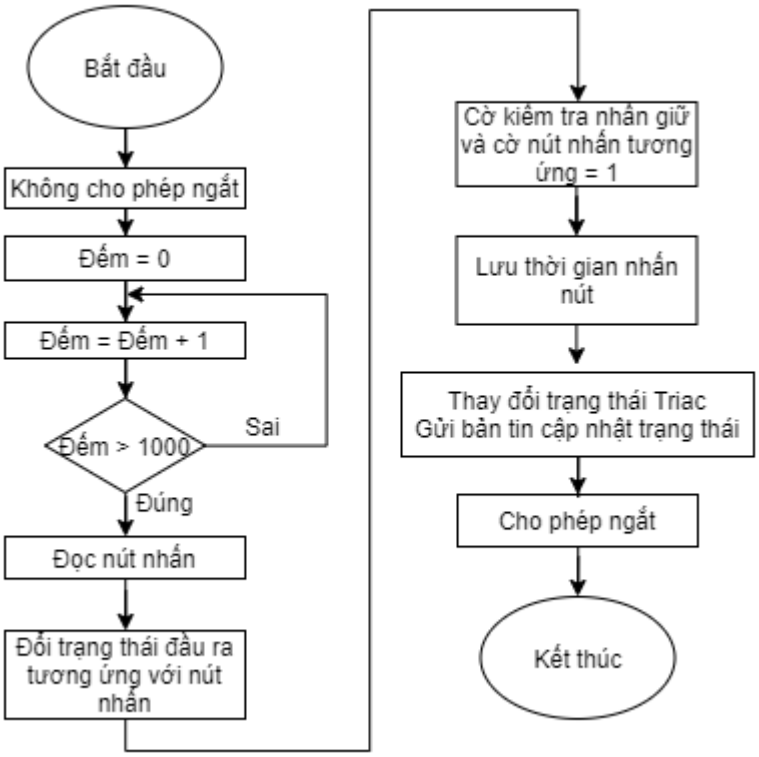
Chương trình Setup:

```
  // put your setup code here, to run once:
  pinMode(out1, OUTPUT);
  digitalWrite(out1, LOW);
  pinMode(out2, OUTPUT);
  digitalWrite(out2, LOW);
  pinMode(out3, OUTPUT);
  digitalWrite(out3, LOW);
  pinMode(out4, OUTPUT);
  digitalWrite(out4, LOW);
  Serial.begin(115200);
  pinMode(ledConfig, OUTPUT);
  digitalWrite(ledConfig, HIGH);
  pinMode(bt1, INPUT);
  attachInterrupt(digitalPinToInterrupt(bt1), isrPressed,
RISING);
  pinMode(bt2, INPUT);
  attachInterrupt(digitalPinToInterrupt(bt2), isrPressed,
RISING);
  pinMode(bt3, INPUT);
  attachInterrupt(digitalPinToInterrupt(bt3), isrPressed,
RISING);
```

```
  pinMode(bt4, INPUT);
  attachInterrupt(digitalPinToInterrupt(bt4), isrPressed,
RISING);

  EEPROM.begin(512);
  char check = char(EEPROM.read(0));
  #ifdef DEBUG
  Serial.print("check = ");
  Serial.print(check);
  #endif
  EEPROM.write(0, 0);
  EEPROM.commit();
  EEPROM.end();
  if(check == 'y')
  {
    digitalWrite(ledConfig, LOW);
    WiFi.disconnect(true);
    WiFiManager wifiManager;
    wifiManager.setConfigPortalTimeout(180);
    wifiManager.setCaptivePortalEnable(false);
    wifiManager.setBreakAfterConfig(true);
    wifiManager.setConnectRetries(5);
    wifiManager.startConfigPortal("DATN-NTA-20173616");
    ESP.restart();
  }
  else
  {
    setupWifi(10);
  }
  digitalWrite(ledConfig, HIGH);
  client.setServer(mqttServer, 1883);
  client.setCallback(callback);
  reconnect(2);
}
```

- Hàm xử lý ngắt nút nhấn làm nhiệm vụ đọc các nút nhấn, khởi động hàm thay đổi trạng thái đầu ra và hàm gửi bản tin cập nhật trạng thái.



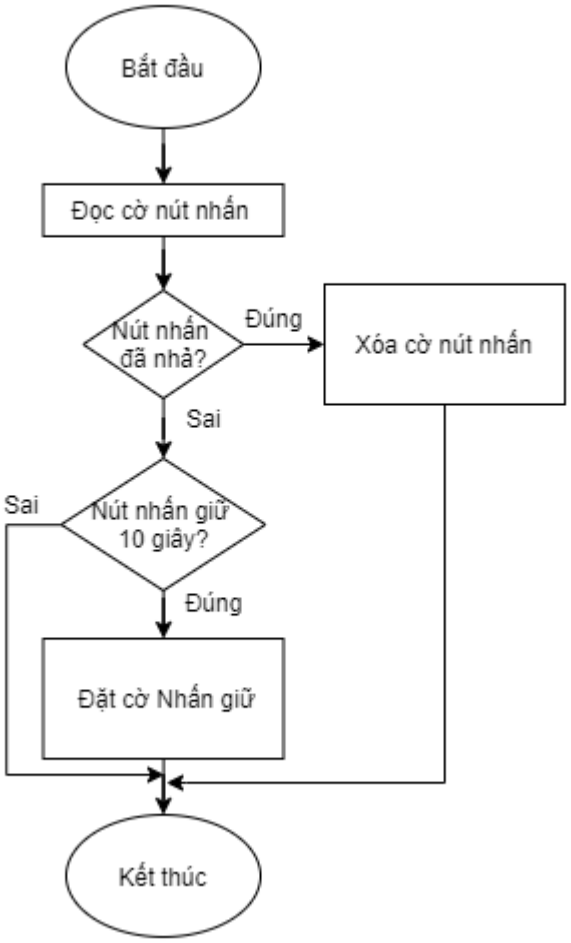
Hình 17. Lưu đồ thuật toán hàm ngắt nút nhấn

Chương trình xử lý ngắt nút nhấn:

```
ICACHE_RAM_ATTR void isrPressed(void){
  noInterrupts();
  delayMicroseconds(1000);
  if(digitalRead(bt1) == 1)
  {
    outState[0] = !outState[0];
    switchTime[0] = 0;
    buttonFlag = 1;
  }
  if(digitalRead(bt2) == 1)
  {
    outState[1] = !outState[1];
    switchTime[0] = 0;
    buttonFlag = 2;
  }
  if(digitalRead(bt3) == 1)
  {
    outState[2] = !outState[2];
    switchTime[2] = 0;
    buttonFlag = 3;
  }
  if(digitalRead(bt4) == 1)
  {
    outState[3] = !outState[3];
    switchTime[3] = 0;
    buttonFlag = 4;
```

```
  }
  changeOutput();
  longPressTime = millis();
  checkLongPress = true;
  interrupts();
}
```

- Hàm kiểm tra nút nhấn giữ: Hàm làm nhiệm vụ xác định trạng thái của nút nhấn bằng cách kiểm tra xem nút nhấn đã được người dùng nhả tay mỗi chu kì lặp của chương trình trong 10 giây. Nếu nút nhấn được giữ 10 giây sẽ đặt cờ để chương trình xử lý.



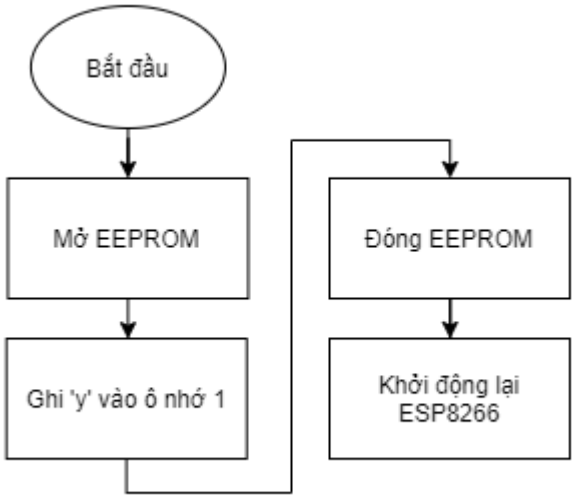
Hình 18. Lưu đồ thuật toán hàm kiểm tra nút nhấn giữ

Chương trình kiểm tra nút nhấn giữ:

```
if(checkLongPress == true)
{
  #ifdef DEBUG
  Serial.print("checkLongPress");
  #endif
  switch(buttonFlag)
  {
    case 1:
      if(digitalRead(bt1) == 0) checkLongPress = false;
      break;
    case 2:
      if(digitalRead(bt2) == 0) checkLongPress = false;
```

```
        break;
      case 3:
        if(digitalRead(bt3) == 0) checkLongPress = false;
        break;
      case 4:
        if(digitalRead(bt4) == 0) checkLongPress = false;
        break;
      default:
        break;
    }
    if(millis() - longPressTime > _10sec)
    {
      longPressed = true;
    }
  }
  if(longPressed == true)
  {
    longPress();
  }
```

- Hàm xử lý nhấn giữ: Khi người sử dụng thực hiện thao tác nhấn giữ. Chương trình sẽ mở bộ nhớ không bị mất đi khi mất điện, ghi kí tự ‘y’ vào để thông báo là lần khởi động sau thiết bị sẽ ở chế độ cấu hình WiFi và khởi động lại vi điều khiển.

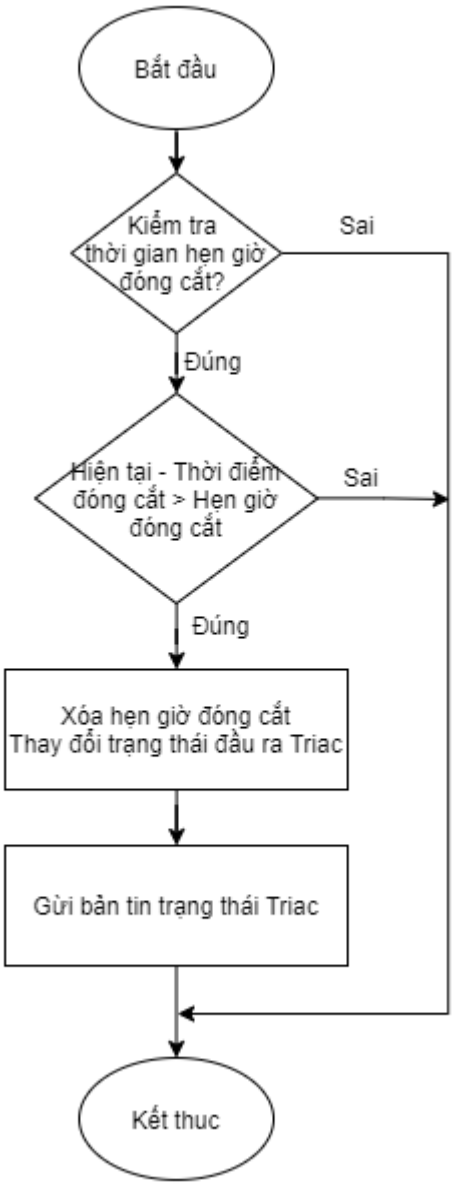


Hình 19. Lưu đồ thuật toán hàm xử lý nút nhấn giữ

Chương trình xứ lý nhấn giữ:

```
void longPress(void)
{
  longPressed = false;
  checkLongPress = false;
  EEPROM.begin(512);
  EEPROM.write(0, 'y');
  EEPROM.commit();
  EEPROM.end();
  ESP.restart();
}
```

- Hàm hẹn giờ đóng cắt: Khi người sử dụng chọn đóng cắt theo thời gian hàm này sẽ liên tục theo dõi thời gian hoạt động của từng kênh và ra lệnh đóng cắt khi thời điểm hẹn giờ đến.



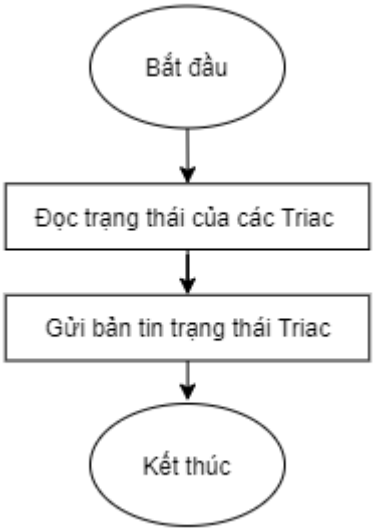
Hình 20. Lưu đồ thuật toán hàm hẹn giờ đóng cắt

Chương trình đóng cắt theo hẹn giờ:

```
void switchTimming(void)
{
  for(int i = 0; i < 4; i++)
  {
    if(switchTime[i] != 0)
    {
      if(millis() - switchTimeRef[i] > switchTime[i]*1000)
      {
        switchTime[i] = 0;
        outState[i] = 0;
        changeOutput();
```

```
        }
      }
    }
}
```

- Hàm gửi dữ liệu trạng thái đóng cắt: Hàm đọc trạng thái của các kênh đóng cắt, xây dựng bản tin trạng thái và đưa bản tin trạng thái tới máy chủ MQTT.



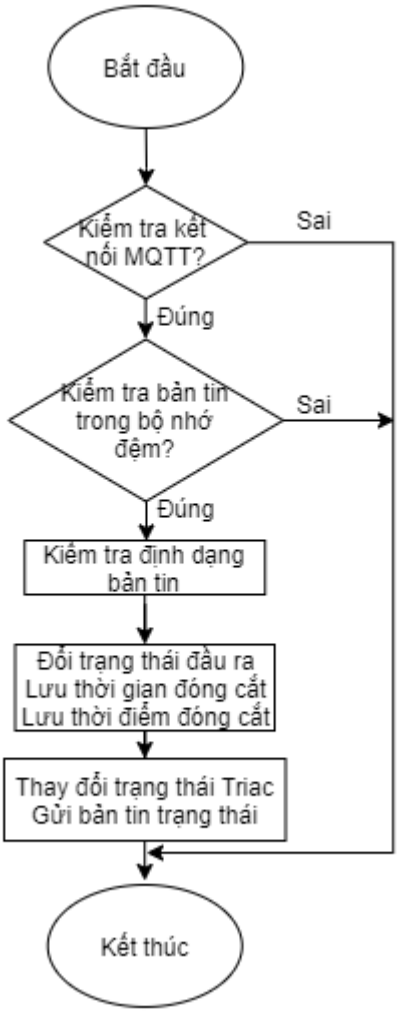
Hình 21. Lưu đồ thuật toán hàm gửi dữ liệu trạng thái đóng cắt

Chương trình gửi dữ liệu trạng thái đóng cắt:

```
void updateState(void)
{
  String mqttPubString;
  mqttPubString = "/1/" + String(outState[0]);
  mqttPubString += "/2/" + String(outState[1]);
  mqttPubString += "/3/" + String(outState[2]);
  mqttPubString += "/4/" + String(outState[3]);

client.publish("status/datnta",(char*)mqttPubString.c_str()
);
}
```

- Hàm xử lý truyền thông MQTT: Hàm làm nhiệm vụ xử lý các bản tin điều khiển gửi đến thiết bị. Đầu tiên hàm sẽ kiểm tra kết nối tới máy chủ, sau đó kiểm tra bộ nhớ đệm xem có bản tin không. Nếu có bản tin thì kiểm tra định dạng bản tin điều khiển. Thực hiện các chức năng đóng cắt và lưu thời gian đóng cắt theo nội dung bản tin điều khiển. Gửi bản tin phản hồi về trạng thái đầu ra hiện tại.



Hình 22. Lưu đồ thuật toán hàm xử lý truyền thông MQTT

Chương trình xử lý truyền thông MQTT:

```
boolean PubSubClient::loop() {
    if (connected()) {
        unsigned long t = millis();
        if ((t - lastInActivity > this->keepAlive*1000UL)
|| (t - lastOutActivity > this->keepAlive*1000UL)) {
            if (pingOutstanding) {
                this->_state = MQTT_CONNECTION_TIMEOUT;
                _client->stop();
                return false;
            } else {
                this->buffer[0] = MQTTPINGREQ;
                this->buffer[1] = 0;
                _client->write(this->buffer,2);
                lastOutActivity = t;
```

```
                lastInActivity = t;
                pingOutstanding = true;
            }
        }
        if (_client->available()) {
            uint8_t llen;
            uint16_t len = readPacket(&llen);
            uint16_t msgId = 0;
            uint8_t *payload;
            if (len > 0) {
                lastInActivity = t;
                uint8_t type = this->buffer[0]&0xF0;
                if (type == MQTTPUBLISH) {
                    if (callback) {
                        uint16_t tl = (this-
>buffer[llen+1]<<8)+this->buffer[llen+2]; /* topic length
in bytes */
                        memmove(this->buffer+llen+2,this-
>buffer+llen+3,tl); /* move topic inside buffer 1 byte to
front */
                        this->buffer[llen+2+tl] = 0; /* end
the topic as a 'C' string with \x00 */
                        char *topic = (char*) this-
>buffer+llen+2;
                        // msgId only present for QOS>0
                        if ((this->buffer[0]&0x06) ==
MQTTQOS1) {
                            msgId = (this-
>buffer[llen+3+tl]<<8)+this->buffer[llen+3+tl+1];
                            payload = this-
>buffer+llen+3+tl+2;
                            callback(topic,payload,len-
llen-3-tl-2);

                            this->buffer[0] = MQTTPUBACK;
                            this->buffer[1] = 2;
                            this->buffer[2] = (msgId >> 8);
                            this->buffer[3] = (msgId &
0xFF);
                            _client->write(this->buffer,4);
                            lastOutActivity = t;

                        } else {
                            payload = this-
>buffer+llen+3+tl;
                            callback(topic,payload,len-
llen-3-tl);
                        }
                    }
                } else if (type == MQTTPINGREQ) {
                    this->buffer[0] = MQTTPINGRESP;
                    this->buffer[1] = 0;
                    _client->write(this->buffer,2);
                } else if (type == MQTTPINGRESP) {
                    pingOutstanding = false;
                }
            } else if (!connected()) {
                // readPacket has closed the connection
                return false;
            }
```

```
        }
        return true;
    }
    return false;
}

void callback(char* topic, byte* payload, unsigned int
length)
{
  char timeBuffer[2];
  int timeTemp;
  timeBuffer[0] = (char)payload[5];
  timeBuffer[1] = (char)payload[6];
  timeTemp = atoi(timeBuffer);
  if((char)payload[0] == '/')
  {
    if((char)payload[1] == '1')
    {
      switchTime[0] = timeTemp;
      switchTimeRef[0] = millis();
      if((char)payload[3] == '1')
      {
        outState[0] = 1;
      }
      else if(switchTime[0] == 0)
      {
        outState[0] = 0;
      }
    }
    else if ((char)payload[1] == '2')
    {
      switchTime[1] = timeTemp;
      switchTimeRef[1] = millis();
      if((char)payload[3] == '1')
      {
        outState[1] = 1;
      }
      else if(switchTime[1] == 0)
      {
        outState[1] = 0;
      }
    }
    else if ((char)payload[1] == '3')
    {
      switchTime[2] = timeTemp;
      switchTimeRef[2] = millis();
      if((char)payload[3] == '1')
      {
        outState[2] = 1;
      }
      else if(switchTime[2] == 0)
      {
        outState[2] = 0;
      }
    }
    else if ((char)payload[1] == '4')
    {
      switchTime[3] = timeTemp;
      switchTimeRef[3] = millis();
      if((char)payload[3] == '1')
```

```
        {
          outState[3] = 1;
        }
        else if(switchTime[3] == 0)
        {
          outState[3] = 0;
        }
      }
      changeOutput();
      updateState();
    }
}
```

Chương 4. Phát triển ứng dụng điều khiển

4.1. Giới thiệu ngôn ngữ Python và thư viện Kivy

4.1.1. Ngôn ngữ Python

Python là một ngôn ngữ lập trình thông dịch, hướng đối tượng bậc cao ngữ nghĩa động. Python hỗ trợ các thư viện dưới dạng các mô-dun và gói. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng. Cấu trúc của Python cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu. Một số đặc điểm chính của Python:

- Ngữ pháp đơn giản, dễ đọc
- Vừa hướng thủ tục, vừa hướng đối tượng
- Hỗ trợ mô-đun và hỗ trợ gói
- Xử lý lỗi bằng ngoại lệ
- Kiểu dữ liệu động ở mức cao
- Có các bộ thư viện chuẩn và các mô-đun ngoài, đáp ứng tất cả các nhu cầu lập trình
- Có khả năng tương tác với các mô-đun khác viết trên C/C++
- Có thể nhúng vào ứng dụng như một giao tiếp kịch bản

Nhờ những đặc điểm như dễ đọc, dễ học và có cộng đồng hỗ trợ lớn, ngôn ngữ Python ngày càng trở nên phổ biến trong lĩnh vực lập trình. Những chương trình viết bằng ngôn ngữ Python cũng cho phép chạy trên nhiều nền tảng hệ điều hành khác nhau nên được ưu tiên lựa chọn để thiết kế ứng dụng điều khiển trong đồ án này.

4.1.2. Thư viện Kivy cho Python

Kivy là một mô-đun thư viện cho Python có mã nguồn mở hỗ trợ phát triển ứng dụng có giao diện. Kivy hỗ trợ thiết kế chương trình đa nền tảng. Một chương trình sử dụng mô-đun Kivy có thể chạy trên Linux, Windows, OS X, Android và iOS. Kivy được viết bằng Python, do đó nó rất nhanh và có độ chính xác cao. Ngoài Pyhton thì Kivy cũng tự xây dựng ngôn ngữ của riêng mình là ngôn ngữ Kivy, có thể tìm thấy ở các tệp có đuôi ".kv". Với ngôn ngữ Kivy thì chúng ta có thể tạo giao diện người dùng với số ít mã lệnh, tách riêng phần giao diện với phần logic tính toán chình của chương trình.

Đồ án này thiết kế ứng dụng với mô-đun Kivy do tính đa nền tảng của mô-đun này. Chương trình ứng dụng điều khiển có thể được sử dụng trên máy tính xách tay hay điện thoại thông minh với kết nối internet.

4.2. Lưu đồ thuật toán và chương trình cho ứng dụng điều khiển

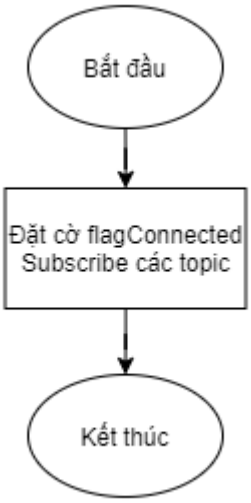
4.2.1. Chương trình logic chính

Chương trình logic chính của ứng dụng điều khiển được thiết kế bằng ngôn ngữ Python với 4 đối tượng gồm lớp logic chính của ứng dụng "DATNApp", lớp giao diện dạng lưới "MyGrid" và 2 lớp cửa sổ thông báo "PopupTurnOn" và "PopupTurnOff" được xây dựng trên các lớp cha:

- Lớp cha ứng dụng: App

- Lớp cha làm khung giao diện: Widget
- Lớp cha giao diện lưới: GridLayout
- Lớp cha giao diện thả nổi: Floatlayout

- Lớp logic chính của ứng dụng: Lớp logic chính của ứng dụng “DATNApp” được xây dựng trên lớp cha “App” với các hàm riêng để xử lý truyền thông MQTT: hàm “onConnect” được gọi khi thiết bị kết nối thành công với máy chủ; hàm “onDisconnect” được gọi khi thiết bị mất kết nối với máy chủ; hàm “onMessage” được gọi khi có bản tin mới gửi đến thiết bị; hàm “sendCmd” được gọi khi thiết bị gửi bản tin điều khiển; hàm “mqttReconnect” được gọi để thiết bị kết nối lại với máy chủ MQTT.
  - Hàm “onConnect” đặt cờ thông báo thiết bị đã kết nối với máy chủ MQTT và đăng kí (subscribe) các chủ để (topic) trạng thái của thiết bị trường.

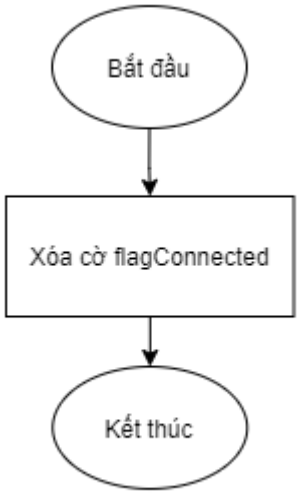


Hình 23. Lưu đồ thuật toán hàm "onConnect"

Chương trình của hàm “onConnect”:

```
def onConnect(self, client, userdata, flags, rc):
    global flagConnected
    flagConnected = True
    for topic in topicSubscribe:
        client.subscribe(topic)
```

- Hàm “onDisconnect” xóa cờ đã kết nối của thiết bị để thông báo cho hàm “mqttReconnect” tiến hành kết nối lại với máy chủ



Hình 24. Lưu đồ thuật toán hàm "onDisconnect"

Chương trình của hàm “onDisconnect”:

```
def onDisconnect(self, client, userdata, rc):
    global flagConnected
    flagConnected = False
```

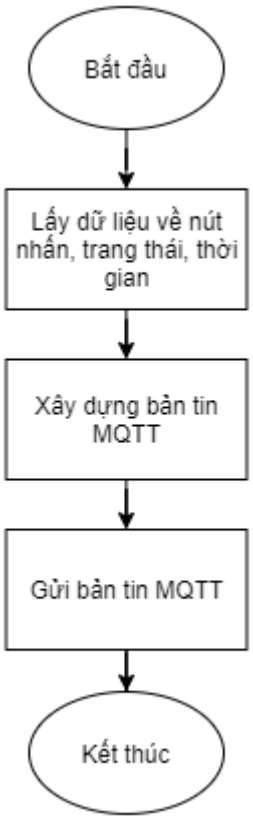
- Hàm “onMessage” định dạng lại bản tin MQTT cho phù hợp và truyền vào hàm “update” của lớp giao diện lưới để thay đổi thông tin hiển thị.

Hình 25. Lưu đồ thuật toán hàm xử lý bản tin MQTT nhận được

Chương trình của hàm “onMessage”:

```
def onMessage(self, client, userdata, message):
    data = message.payload.decode("utf-8")
    if data[0] == '/':
        self.root.update(data)
```

- Hàm “sendCmd” xây dựng bản tin điều khiển và tiến hành gửi bản tin tới máy chủ MQTT

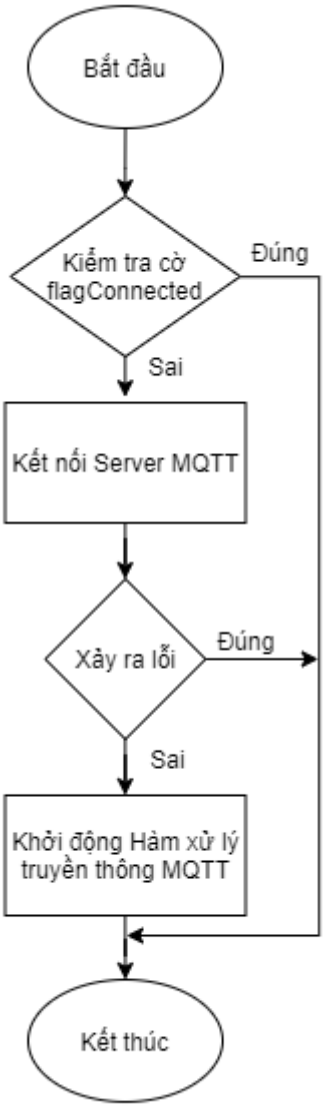


Hình 26. Lưu đồ thuật toán hàm gửi bản tin điều khiển

Chương trình của hàm “sendCmd”:

```
def sendCmd(self, switchNumber, switchType,
switchTime):
    strTemp = "/" + str(switchNumber) + "/"
    if buttonState[int(switchNumber) - 1] == 0:
        strTemp += '1'
    else:
        strTemp += '0'
    strTemp += '/' + str(switchTime)
    self.client.publish(topicPublish, strTemp)
```

- Hàm “mqttReconnect” được gọi mỗi 30 giây khi ứng dụng đang hoạt động để tiến hành kết nối lại với máy chủ nếu mất kết nối.



Hình 27. Lưu đồ thuật toán hàm kết nối lại với máy chủ MQTT

Chương trình của hàm “mqttReconnect”:

```
def mqttReconnect(self):
    global flagConnected
    if flagConnected == False:
        try:
            self.client.connect(mqttMáy chủ, mqttPort,
60)
            self.client.loop_start()
        except:
            pass
```

- Lớp giao diện dạng lưới: Lớp giao diện dạng lưới “MyGrid” của ứng dụng được xây dựng trên lớp cha “Widget” với các hàm riêng để cập nhật và xử lý thao tác từ người dùng: hàm “update” được gọi để thay đổi trạng thái các nút nhấn trên màn hình hiển thị; hàm “showPopup” được gọi để hiển thị cửa sổ

thông báo khi người dùng nhấn nút; hàm "buttonConfigWifi" được gọi khi người dúng bấm vào nút Cấu hình WiFi.

- Hàm "update" nhận dữ liệu từ bản tin MQTT và thay đổi giao diện các nút nhấn tương ứng

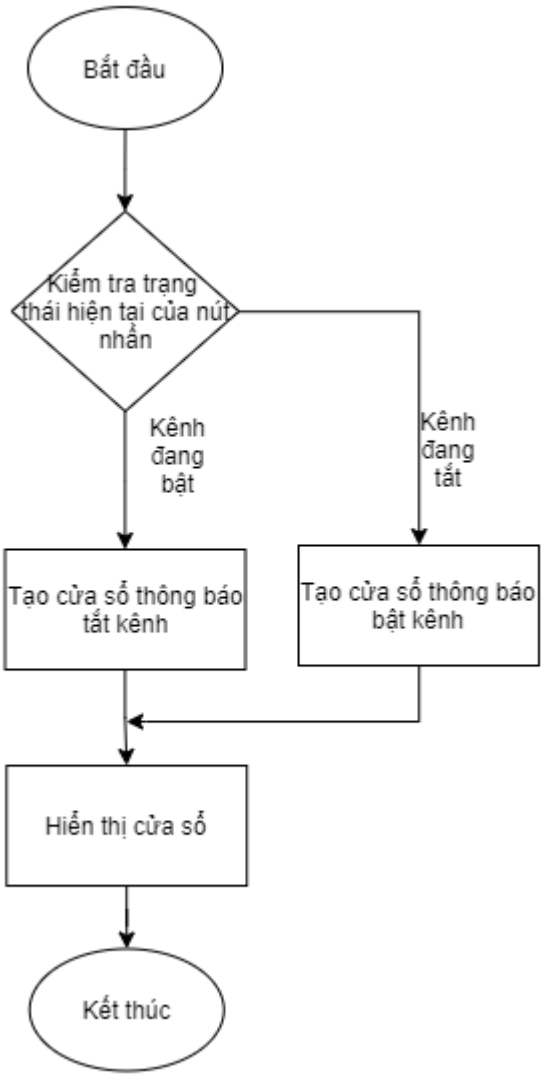
Hình 28. Lưu đồ thuật toán hàm thay đổi giao diện thiết bị điều khiển

Chương trình của hàm "update":

```
def update(self, data):
    buttonUpdateState = [data[3], data[7], data[11],
data[15]]
    if buttonUpdateState[0] == '1':
        self.button1.text = "Cong tac 1 Bat"
        buttonState[0] = 1
    elif buttonUpdateState[0] == '0':
        self.button1.text = "Cong tac 1 Tat"
        buttonState[0] = 0
    if buttonUpdateState[1] == '1':
        self.button2.text = "Cong tac 2 Bat"
        buttonState[1] = 1
    elif buttonUpdateState[1] == '0':
        self.button2.text = "Cong tac 2 Tat"
        buttonState[1] = 0
    if buttonUpdateState[2] == '1':
        self.button3.text = "Cong tac 3 Bat"
        buttonState[2] = 1
    elif buttonUpdateState[2] == '0':
        self.button3.text = "Cong tac 3 Tat"
        buttonState[2] = 0
    if buttonUpdateState[3] == '1':
        self.button4.text = "Cong tac 4 Bat"
        buttonState[3] = 1
    elif buttonUpdateState[3] == '0':
        self.button4.text = "Cong tac 4 Tat"
```

```
        buttonState[3] = 0
```

- Hàm “showPopup” hiển thị một cửa sổ thông báo tương ứng với nút người dùng vừa nhấn.



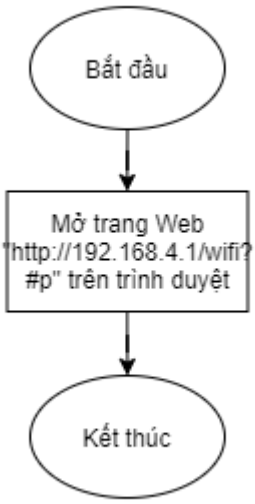
Hình 29. Lưu đồ thuật toán hàm hiển thị cửa sổ thông báo

Chương trình của hàm “showPopup”:

```
def showPopup(self, switchNumber):
    global _switchNumber, popupWindow
    _switchNumber = int(switchNumber)
    if buttonState[_switchNumber - 1] == 0:
        show = PopupTurnOn()
        popupWindow = Popup(title = "Bat cong tac",
content = show,
                            size_hint = (None, None),
size = (500,500))
    elif buttonState[_switchNumber - 1] == 1:
        show = PopupTurnOff()
        popupWindow = Popup(title = "Tat cong tac",
content = show,
```

```
                                    size_hint = (None, None),
size = (500,500))
```

- Hàm “buttonConfigWifi” mở một trình duyệt cho phép người dùng cập nhật mạng WiFi cho thiết bị.



Hình 30. Lưu đồ thuật toán hàm mở trang web cấu hình WiFi

Chương trình của hàm “buttonConfigWifi”:

```
def buttonConfigWifi(self):
    webbrowser.open('http://192.168.4.1/wifi?#p')
```

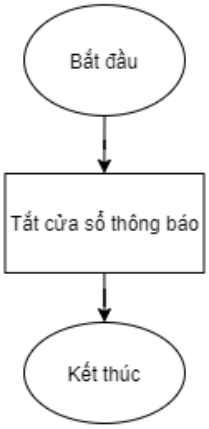
- Lớp giao diện cửa sổ thông báo: Lớp giao diện cửa sổ thông báo được xây dựng trên lớp cha "FloatLayout" với các hàm riêng để xử lý thao tác từ người dùng: hàm "switch" được gọi để xử lý thao tác nhấn nút; hàm "close" được gọi khi người dùng chọn nút đóng cửa sổ.
  - Hàm "switch" xử lý thao tác của người dùng và chuyển thành các biến tương ứng để truyền cho hàm gửi bản tin điều khiển.

Hình 31. Lưu đồ thuật toán hàm xử lý nút bấm ở cửa sổ thông báo

Chương trình của hàm "switch":

```
def switch(self, switchType, switchTime):
    global  _switchNumber
    DATNApp().sendCmd(_switchNumber, switchType,
switchTime)
    popupWindow.dismiss()
```

  - Hàm "close" xử lý thao tác người dúng bấm nút "Bỏ qua"



Hình 32. Lưu đồ thuật toán hàm đóng cửa sổ thông báo

Chương trình của hàm “close”:

```
def close(self):
    popupWindow.dismiss()
```

4.2.2. Chương trình giao diện

Chương trình giao diện thiết kế bằng mô-đun Kivy được chia làm 2 phần gồm có phần giao diện hiển thị dạng lưới ở trạng thái bình thường và giao diện cửa sổ thông báo khi người dùng sử dụng các nút nhấn:

- Giao diện dạng lưới: Giao diện của ứng dụng được xây dựng trên mô hình dạng lưới. Cửa sổ làm việc được chia thành các ô nhỏ, mỗi ô được xây dựng với những chức năng khác nhau. Ô thông tin ở trên cùng được xây dựng bằng thuộc tính “Label”, chuyên dùng để viết văn bản và không cho phép người dùng thay đổi. Các ô nút nhấn được xây dựng bằng thuộc tính “Button”. Thuộc tính này cho phép người dùng tương tác với ứng dụng điều khiển với các thao tác như nhấn, nhấn giữ. Khi người dùng tương tác với nút nhấn, các hàm điều khiển sẽ được kích hoạt tương ứng. Ở trong đồ án này, khi người dùng nhả tay hoặc chuột điều khiển ra khỏi nút nhấn, hàm điều khiển trong mục “on_release” của nút nhấn sẽ được gọi.

Mô hình giao diện dạng lưới có dạng như sau:

Thông tin	
Nút nhấn 1	Nút nhấn 2
Nút nhấn 3	Nút nhấn 4
Nút cấu hình WiFi	

Hình 33. Hình mẫu cho giao diện dạng lưới

Chương trình xây dựng giao diện dạng lưới:

```
<MyGrid>:
    button1: button1
    button2: button2
    button3: button3
    button4: button4
    configWifi: configWifi
    infoLabel1: infoLabel1
    infoLabel2: infoLabel2
    infoLabel3: infoLabel3
    GridLayout:
        cols: 1
        size: root.width, root.height

        Label:
            id: infoLabel1
            size_hint: 1, 0.05
            text: "Sinh vien thuc hien: NGUYEN TUAN
ANH_MSSV: 20173616"
        Label:
            id: infoLabel2
            size_hint: 1, 0.05
            text: "Giang vien huong dan: TS. NGUYEN VAN
ANH"
        Label:
            id: infoLabel3
```

```
            size_hint: 1, 0.05
            text: "De tai: Thiet ke thiet bi dong cat 4
kenh qua WiFi"
        GridLayout:
            cols: 2
            Button:
                id: button1
                text: "Cong tac 1"
                font_size: 50
                on_release: root.showPopup(1)
            Button:
                id: button2
                text: "Cong tac 2"
                font_size: 50
                on_release: root.showPopup(2)
            Button:
                id: button3
                text: "Cong tac 3"
                font_size: 50
                on_release: root.showPopup(3)
            Button:
                id: button4
                text: "Cong tac 4"
                font_size: 50
                on_release: root.showPopup(4)
        Button:
            size_hint: 1, 0.2
            id: configWifi
            text: "Cau hinh WiFi"
            font_size: 50
            on_release: root.buttonConfigWifi()
```

- Giao diện cửa sổ thông báo: Giao diện cửa sổ thông báo của ứng dụng được xây dựng trên thuộc tính "Popup" của thư viện Kivy. Giao diện gồm trường thông tin và các trường nút nhấn tương tự như giao diện chính. Khi người dùng nhả tay trên màn hình cảm ứng hoặc nhả chuột trên máy tính, hàm tương ứng trường "on_release" của chương trình sẽ được gọi. Mô hình giao diện cửa sổ có dạng như sau:

Thông tin
Nút nhấn 1
Nút nhấn 2
Nút nhấn 3

Hình 34. Hình mẫu cho cửa sổ thông báo

Chương trình giao diện cho cửa sổ thông báo:

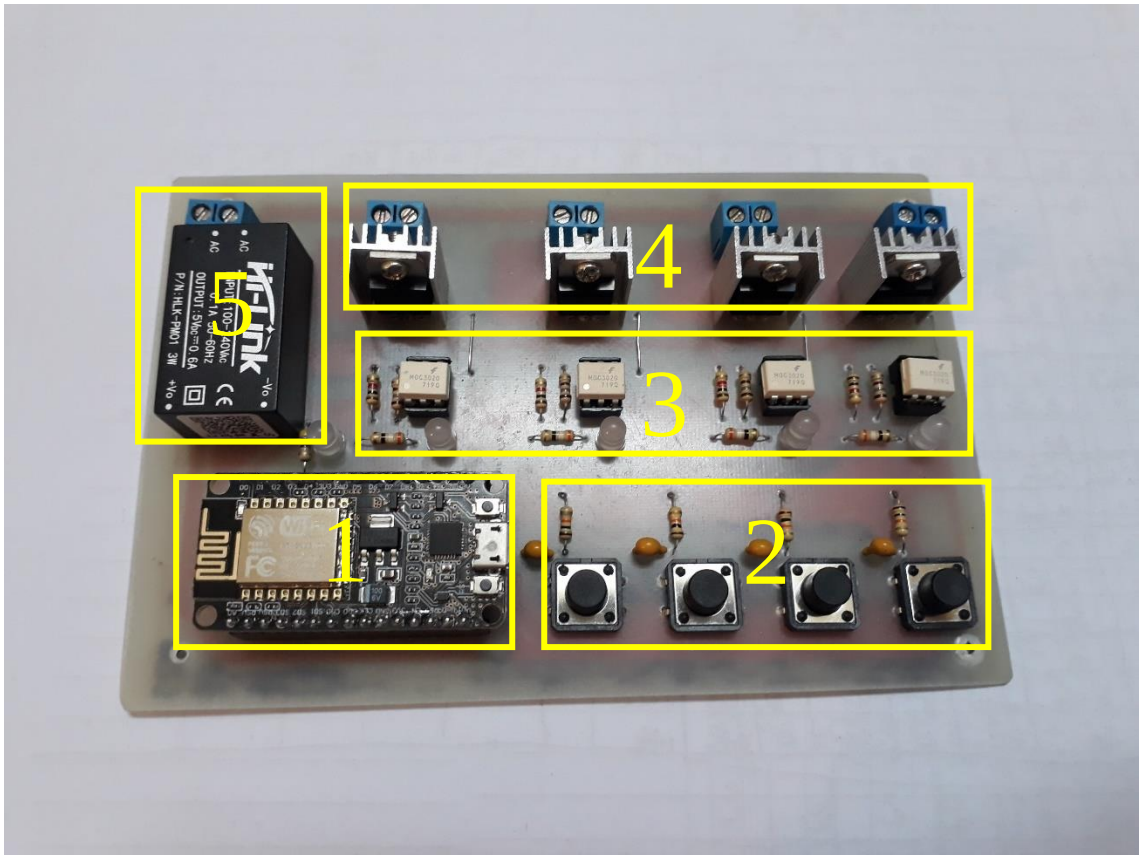
```
<PopupTurnOn>:
    Button:
        text: "Bat"
        size_hint: 1, 0.25
        pos_hint: {"x":0, "y":0.75}
        on_release: root.switch(1, 00)
    Button:
        text: "Bat 30 giay"
        size_hint: 1, 0.25
        pos_hint: {"x":0, "y":0.5}
```

```
            on_release: root.switch(1, 30)
        Button:
            text: "Bat 1 phut"
            size_hint: 1, 0.25
            pos_hint: {"x":0, "y":0.25}
            on_release: root.switch(1, 60)
        Button:
            text: "Bo qua"
            size_hint: 1, 0.25
            pos_hint: {"x":0, "y":0}
            on_release: root.close()
<PopupTurnOff>:
    Button:
        text: "Tat"
        size_hint: 1, 0.25
        pos_hint: {"x":0, "y":0.75}
        on_release: root.switch(0, 00)
    Button:
        text: "Tat sau 30 giay"
        size_hint: 1, 0.25
        pos_hint: {"x":0, "y":0.5}
        on_release: root.switch(0, 30)
    Button:
        text: "Tat sau 1 phut"
        size_hint: 1, 0.25
        pos_hint: {"x":0, "y":0.25}
        on_release: root.switch(0, 60)
    Button:
        text: "Bo qua"
        size_hint: 1, 0.25
        pos_hint: {"x":0, "y":0}
        on_release: root.close()
```

Chương 5. Mô hình và kết quả đạt được

5.1. Kết quả thiết kế phần cứng thiết bị

Sau đây là hình ảnh thực tế của thiết bị:



Hình 35. Sản phẩm thực tế

Chú thích:

1. Mạch phát triển nodeMCU
2. Khối nút nhấn
3. Khối cách ly quang giữa mạch điều khiển và mạch lực
4. Khối triac đóng cắt
5. Khối chuyển đổi nguồn AC/DC 5V – 3W

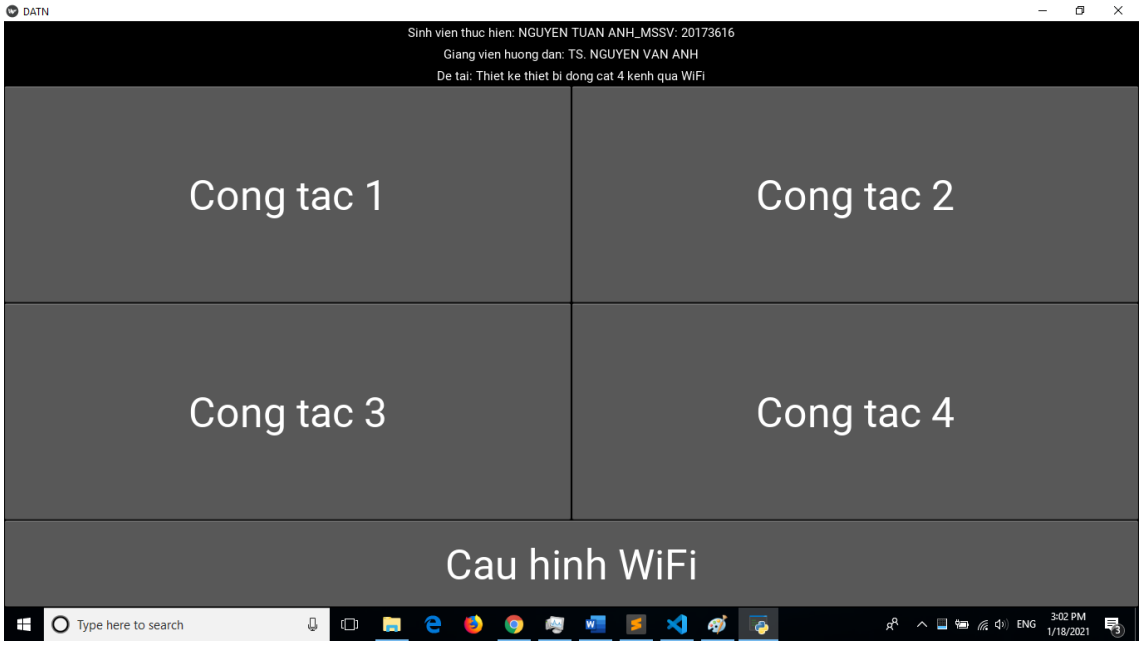
5.2. Kết quả thiết kế ứng dụng điều khiển

Hình ảnh thực tế ứng dụng điều khiển trên điện thoại Android:

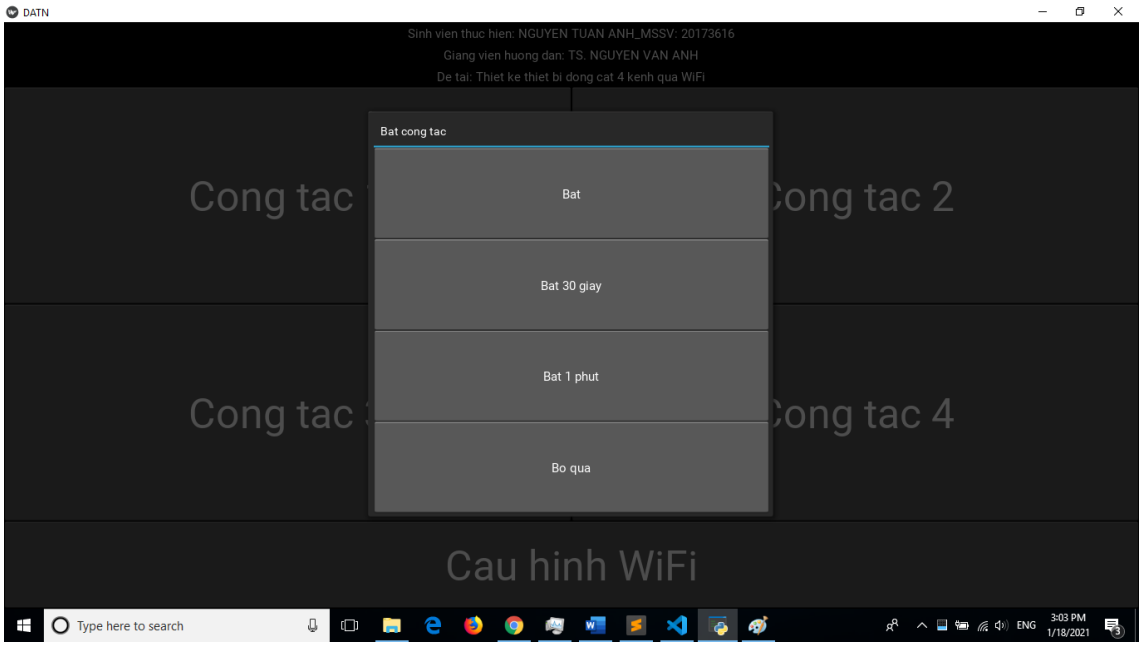
Hình 36. Giao diện thực tế của ứng dụng điều khiển trên điện thoại

Hình 37. Giao diện thực tế của ứng dụng trên điện thoại khi điều khiển thiết bị

Hình ảnh thực tế ứng dụng điều khiển trên máy tính xách tay:



Hình 38. Giao diện thực tế của ứng dụng điều khiển trên máy tính xách tay



Hình 39. Giao diện thực tế của ứng dụng trên máy tính xách tay khi điều khiển thiết bị

5.3. Kết luận

Trong đồ án này đã thiết kế một thiết bị có khả năng đóng cắt 4 kênh ra tại hiện trường và từ xa thông qua kết nối internet với khả năng đóng cắt dòng điện với tải điện trở là 20A ở hiệu điện thế là 220V xoay chiều, có khả năng tạo trễ cắt điện ở 4 kênh ra. Xây dựng được ứng dụng điều khiển và hiển thị trạng thái của 4 kênh đóng cắt.

Bên cạnh đó còn một vài hạn chế về loại tải đầu ra, giao diện điều khiển thiết bị chưa thực sự thân thiện với người dùng.

5.4. Định hướng phát triển

Khi quyết định thực hiện đề tài này, em mong muốn sẽ triển khai được một hệ thống thông minh với số lượng các thiết bị lớn hơn và đa dạng thiết bị hơn không chỉ thực hiện công việc đóng cắt mà còn tích hợp các các thiết bị cảm biến để giám sát môi trường xung quanh, giúp cho các thiết bị có thể tương tác với nhau từ đó nâng cao mức độ tự động hóa cho hệ thống.

Tài liệu tham khảo

[1] T. T. Minh, Giáo trình điện tử công suất, Nhà xuất bản Giáo dục Việt Nam, 2012.

[2] "www.electronics-notes.com," [Online]. Available: https://www.electronics-notes.com/articles/electronic_components/scr/understanding-triac-specifications-datasheet-parameters.php.

[3] F. Semiconductor, "Applications of Random Phase Crossing Triac Drivers".