|  |  |
| --- | --- |
|  |  |
| NGUYỄN HỮU DŨNG  NGUYỄN MINH QUYỀN  PHÙNG VĂN TƯỞNG | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
|  |
| ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC CNKT ĐIỆN TỬ - VIỄN THÔNG |
|  |
| **NGHIÊN CỨU, THIẾT KẾ MÔ HÌNH ĐIỀU KHIỂN THIẾT BỊ ĐIỆN TRONG NHÀ SỬ DỤNG STM32** |
|  |
|  |
| **CBHD: ThS. NGUYỄN NGỌC ANH**  **Nhóm sinh viên thực hiện: Nhóm 43**   1. **NGUYỄN HỮU DŨNG\_ Mã SV: 2019601420** 2. **NGUYỄN MINH QUYỀN\_ Mã SV: 2019601700** 3. **PHÙNG VĂN TƯỞNG\_Mã SV: 2019602132** |
| CNKT ĐIỆN TỬ - VIỄN THÔNG | **Hà Nội – 2023** |

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**--------------------------------------**

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

**NGHIÊN CỨU, THIẾT KẾ MÔ HÌNH ĐIỀU KHIỂN THIẾT BỊ ĐIỆN TRONG NHÀ SỬ DỤNG STM32**

|  |
| --- |
| **CBHD: ThS. NGUYỄN NGỌC ANH** |
| **Nhóm sinh viên thực hiện: Nhóm 43** |
| 1. **NGUYỄN HỮU DŨNG\_Mã SV: 2019601420** 2. **NGUYỄN MINH QUYỀN\_Mã SV: 2019601700** 3. **PHÙNG VĂN TƯỞNG\_Mã SV: 2019602132** |

**Hà Nội - 2023**

# LỜI CẢM ƠN

Trong thời gian thực hiện đề tài đồ án tốt nghiệp, nhóm em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của các thầy cô bộ môn Điện Tử Viễn Thông cũng như thầy cô trong khoa Điện Tử trường Đại Học Công Nghiệp Hà Nội. Đồng thời nhóm em đã được tiếp cận các trang thiết bị hiện đại của khoa để phục vụ vào mục đích nghiên cứu, học tập.

Đặc biệt nhóm em xin gửi lời cảm ơn chân thành đến thầy ThS. Nguyễn Ngọc Anh đã hướng dẫn cho chúng em những kiến thức, kỹ năng cơ bản cần có để hoàn thành đề tài nghiên cứu này.

Sau cùng chúng em xin gửi lời cảm ơn chân thành tới gia đình, bạn bè đã động viên, cổ vũ và đóng góp ý kiến trong quá trình học tập, nghiên cứu cũng như quá trình làm đồ án tốt nghiệp.

Tuy nhiên trong quá trình nghiên cứu đề tài, vì kiến thức chuyên ngành còn hạn chế nên nhóm em vẫn còn nhiều thiếu sót khi tìm hiểu, đánh giá và trình bày về đề tài. Rất mong nhận được sự quan tâm, đóng góp ý của các thầy cô để đề tài của chúng em được đầy đủ và hoàn chỉnh hơn.

Nhóm em xin chân thành cảm ơn!

|  |  |
| --- | --- |
|  | ***Hà Nội, ngày tháng năm 2023*** |
|  | **Nhóm sinh viên**  Nguyễn Hữu Dũng  Nguyễn Minh Quyền  Phùng Văn Tưởng |

# MỤC LỤC

[LỜI CẢM ƠN i](#_Toc135682178)

[DANH MỤC CÁC KÝ HIỆU, TỪ VIẾT TẮT iv](#_Toc135682180)

[DANH MỤC BẢNG BIỂU vi](#_Toc135682181)

[DANH MỤC HÌNH ẢNH vii](#_Toc135682182)

[MỞ ĐẦU 1](#_Toc135682183)

[CHƯƠNG 1. TỔNG QUAN VỀ CÔNG NGHỆ ĐIỀU KHIỂN TỪ XA 3](#_Toc135682184)

[1.1. Một số công nghệ điều khiển từ xa hiện nay 3](#_Toc135682185)

[1.1.1. Tia hồng ngoại (IR) 3](#_Toc135682186)

[1.1.2. Công nghệ điều khiển từ xa sử dụng Bluetooth 4](#_Toc135682187)

[1.1.3. Công nghệ điều khiển từ xa sử dụng Wifi 6](#_Toc135682188)

[1.2. Các hệ thống điều khiển thiết bị điện trong nhà thông minh hiện nay 7](#_Toc135682189)

[1.2.1. Hệ thống điều khiển điện thông minh [4] 7](#_Toc135682190)

[1.2.2. Hệ thống cảm biến thông minh [4] 8](#_Toc135682191)

[1.3. Kết luận phần 1 9](#_Toc135682192)

[CHƯƠNG 2. THIẾT KẾ MÔ HÌNH ĐIỀU KHIỂN THIẾT BỊ ĐIỆN TRONG NHÀ SỬ DỤNG STM32 10](#_Toc135682193)

[2.1. Phân tích yêu cầu thiết kế 10](#_Toc135682194)

[2.1.1. Mục tiêu thiết kế 10](#_Toc135682195)

[2.1.2. Điều khiện ràng buộc của thiết kế 10](#_Toc135682196)

[2.1.3. Thông số kỹ thuật 10](#_Toc135682197)

[2.1.4. Tiêu chí đánh giá sản phẩm 11](#_Toc135682198)

[2.1.5. Một số giải pháp thực hiện 11](#_Toc135682199)

[2.2. Thiết kế mô hình điều khiển thiết bị điện trong nhà 17](#_Toc135682200)

[2.2.1. Thiết kế sơ đồ nguyên lý mạch điều khiển thiết bị trong nhà 17](#_Toc135682201)

[2.2.2. Thiết kế phần mềm 30](#_Toc135682202)

[2.2.3. Thiết kế mô hình 36](#_Toc135682203)

[2.3. Kết luận chương 2 37](#_Toc135682204)

[CHƯƠNG 3. THỬ NGHIỆM VÀ ĐÁNH GIÁ 38](#_Toc135682205)

[3.1. Mạch in thủ công và mô hình thực tế 38](#_Toc135682206)

[3.2. Thử nghiệm và kiểm chứng sản phẩm 39](#_Toc135682207)

[3.3. Đánh giá sản phẩm 41](#_Toc135682208)

[3.4. Ứng dụng của sản phẩm 41](#_Toc135682209)

[3.5. Kết luận chương 3 41](#_Toc135682210)

[KẾT LUẬN 42](#_Toc135682211)

[TÀI LIỆU THAM KHẢ0 44](#_Toc135682212)

[PHỤ LỤC 1](#_Toc135682213)

# DANH MỤC CÁC KÝ HIỆU, TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **Viết tắt** | **Tiếng Anh** | **Tiếng Việt** |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| ADC | Analog Digital Converter | Bộ chuyển đổi tương tự ra số |
| AT | Attention command | Lệnh chú ý |
| CPU | Central Processing Unit | Bộ phận xử lý trung tâm |
| CAN | Controller Area Network | Điều khiển thời gian thực |
| DVD | Digital Video Disc | Đĩa quang kỹ thuật số |
| DHCP | Dynamic Host Configuration Protocol | Giao thức cấu hình máy chủ |
| HTTP | HyperText Transfer Protocol | Giao thức truyền tải siêu văn bản |
| IoT | Internet of Things | Internet vạn vật |
| IC | Integrated circuit | Chíp, vi mạch điện tử |
| I2C | Inter-Integrated Circuit | Mạch tích hợp liên |
| IFTTT | If This Then That | Nếu điều này xảy ra thì làm việc kia |
| I/O | Input/Output | Đầu vào/Đầu ra |
| IR | Infrared Radiation | Tia hồng ngoại |
| LED | Light Emitting Diodes | Đi ốt phát quang |
| LCD | Liquid-Crystal Display | Màn hình tinh thể |
| MPU | Memory Protection Unit | Bộ bảo vệ bộ nhớ |
| PWM | Pulse Width Modulation | Điều chế độ rộng xung |
| RX | Receiver | Nhận tín hiệu điều khiển |
| RTC | Real-time computing | Hệ thống tính toán thời gian thực |
| SPI | Serial Peripheral Interface | Giao diện ngoại vi nối tiếp |
| SDK | Software Development Kit | Bộ phát triển phầm mềm |
| SMT | Surface Mount Technology | Gắn kết bề mặt |
| SCL | Serial Clock | Tín hiệu xung nhịp |
| SDA | Serial Data | Dữ liệu nối tiếp |
| SSID | Service Set Identifier | Mạng cục bộ không dây |
| SRAM | Static random-access memory | Bộ nhớ truy cập ngẫu nhiên tĩnh |
| UART | Universal Asynchronous Receiver / Transmitter | Bộ tiếp nhận không đồng bộ / đồng bộ chuyển giao |
| TX | Transmitter | Truyền tín hiệu điều khiển |
| USB | Universal Serial Bus | Cổng kết nối cáp |

# DANH MỤC BẢNG BIỂU

[Bảng 2.1. Bảng so sánh 3 giải pháp 15](#_Toc135595462)

[Bảng 2.2. Bảng thông số kỹ thuật của IC PC817 24](#_Toc135595463)

[Bảng 2.3. Bảng thông tin của ESP8266 ở chế độ AP 33](#_Toc135595464)

[Bảng 3.1. Bảng kết quả thực nghiệm 39](#_Toc135595465)

# DANH MỤC HÌNH ẢNH

[Hình 1.1. Sơ đồ bộ điều khiển từ xa sử dụng IR 3](#_Toc135595508)

[Hình 1.2. Kết nối các thiết bị bằng Bluetooth 5](#_Toc135595509)

[Hình 1.3. Công tắc điều khiển thông minh 7](#_Toc135595510)

[Hình 1.4. Cảm biến ánh sáng trong ngôi nhà thông minh 8](#_Toc135595511)

[Hình 2.1. Sơ đồ khối sử dụng vi điều khiển AT89S52 11](#_Toc135595512)

[Hình 2.2. Sơ đồ khối sử dụng vi điều khiển PIC18F4520 13](#_Toc135595513)

[Hình 2.3. Sơ đồ khối sử dụng vi điều khiển STM32F103C8T6 14](#_Toc135595514)

[Hình 2.4. Sơ đồ nguyên lý khối nguồn 17](#_Toc135595515)

[Hình 2.5. Module LM2596 18](#_Toc135595516)

[Hình 2.6. Nút nhấn 4 chân 18](#_Toc135595517)

[Hình 2.7. Sơ đồ nguyên lý khối nút nhấn 19](#_Toc135595518)

[Hình 2.8. Kit STM32F103C8T6 BluePill 20](#_Toc135595519)

[Hình 2.9. Sơ đồ chân chức năng Kit STM32F103C8T6 BluePill 20](#_Toc135595520)

[Hình 2.10. Module ESP8266 21](#_Toc135595521)

[Hình 2.11. Sơ đồ chức năng module ESP8266 21](#_Toc135595522)

[Hình 2.12. Sơ đồ kết nối giao tiếp UART 22](#_Toc135595523)

[Hình 2.13. Sơ đồ nguyên lý khối xử lý và giao tiếp mạng 23](#_Toc135595524)

[Hình 2.14. Relay 5V 23](#_Toc135595525)

[Hình 2.15. IC PC817 24](#_Toc135595526)

[Hình 2.16. Sơ đồ chân của IC PC817 24](#_Toc135595527)

[Hình 2.17. Sơ đồ nguyên lý khối relay 25](#_Toc135595528)

[Hình 2.18. Màn hình LCD 1602 26](#_Toc135595529)

[Hình 2.19. Sơ đồ chân LCD 1602 26](#_Toc135595530)

[Hình 2.20. Module PCF8574 I2C 27](#_Toc135595531)

[Hình 2.21. Sơ đồ kết nối PCF8574 với LCD 28](#_Toc135595532)

[Hình 2.22. Sơ đồ nguyên lý toàn hệ thống 29](#_Toc135595533)

[Hình 2.23. Lưu đồ thuật toán của STM32F103C8T6 32](#_Toc135595534)

[Hình 2.24. Kết nối Laptop với ESP8266 33](#_Toc135595535)

[Hình 2.25. Lưu đồ giải thuật chương trình cấu hình mạng cho ESP8266 34](#_Toc135595536)

[Hình 2.26. Lưu đồ giải thuật chương trình giao tiếp Blynk server 35](#_Toc135595537)

[Hình 2.27. Thiết kế mô hình 36](#_Toc135595538)

[Hình 2.28. Bìa trắng formex 36](#_Toc135595539)

[Hình 2.29. Keo 502 36](#_Toc135595540)

[Hình 3.1. Mạch của hệ thống điều khiển thiết bị điện trong nhà 38](#_Toc135595541)

[Hình 3.2. Mô hình của đề tài 39](#_Toc135595542)

# 

# MỞ ĐẦU

* **Lý do lựa chọn đề tài:**

Ngày này, các công nghệ kết nối, điều khiển thiết bị không dây, công nghệ trí tuệ nhân tạo (AI) ngày càng phổ biến biến và phát triển mạnh mẽ ở nhiều quốc gia trên thế giới và được ứng dụng trong nhiều lĩnh vực. Trong những năm gần đây, khái niệm nhà thông minh dần dần trở nên quen thuộc và phổ biến đặc biệt đối với những cư dân sống ở các thành phố lớn. Ngôi nhà thông minh là tích hợp của các hệ thống điều khiển và giám sát môi trường như ánh sáng, nhiệt độ, độ ẩm... Ngoài ra, nhà thông minh còn được sử dụng các công nghệ hiện đại nhằm đảm bảo an ninh, tiện nghi như khóa vân tay, điều khiển từ xa các thiết bị trong nhà ...

Chính vì vậy để góp phần tiện nghi hơn cho ngôi nhà thông minh và xuất phát từ lĩnh vực thực tiễn trên và vận dụng những kiến thức đã học, nhóm em đã đưa ra đề tài **“Nghiên cứu, thiết kế mô hình điều khiển thiết bị điện trong nhà sử dụng STM32”**. Với công nghệ được áp dụng trong mô hình, người dùng có thể dễ dàng điều khiển bật tắt các thiết bị bằng nút nhấn hoặc giọng nói hoặc có thể cài đặt thời gian bật tắt các thiết bị như mong muốn.

* **Mục tiêu nghiên cứu:**

Tìm hiểu các bước cơ bản lập trình STM32F103C8T6, tìm hiểu các hoạt động của modul wifi ESP8266 và hoạt động của relay. Kết nối giữa vi điều khiển STM32F103C8T6 với ESP8266 qua giao tiếp UART.

* **Đối tượng và phạm vi đề tài:**
* Đối tượng: tìm hiểu về vi điều khiển STM32F103C8T6, ESP8266 wifi modul, PC817 và một số linh kiện khác.
* Phạm vi đề tài: nghiên cứu, thiết kế mô hình điều khiển thiết bị điện trong nhà đạt những yêu cầu đã đề ra.
* **Phương pháp nghiên cứu:** Dựa trên những kiến thức đã học của môn vi điều khiển, tổng hợp kiến thức từ những nguồn khác để phân tích và chọn lựa những cách thức ứng dụng phù hợp phục vụ cho thực hiện đề tài.
* **Ý nghĩa đề tài:** Đề tài là một cây cầu gắn kết giữa lí thuyết được học với thực hành, tạo ra các sản phẩm thực tế để bổ sung kiến thức và gia tăng tay nghề của sinh viên. Mặt khác, đề tài cũng là tài liệu tham khảo cho những nghiên cứu có liên quan và áp dụng cho thực tế.
* **Cấu trúc của báo cáo:**

Nội dung đồ án tốt nghiệp được cấu trúc gồm 3 chương:

* *Chương 1: Tổng quan về công nghệ điều khiển từ xa*

Tìm hiểu một số công nghệ điều khiển từ xa và các hệ thống điều khiển thiết bị điện trong nhà thông minh hiện nay.

* *Chương 2: Thiết kế mô hình điều khiển thiết bị điện trong nhà sử dụng STM32*

Trình bày các kiến thức liên quan tới nội dung nghiên cứu, thiết kế hệ thống. Lựa chọn phương án thiết kế rồi đưa ra sơ đồ khối, nêu chức năng nhiệm vụ của từng khối rồi lựa chọn linh kiện phù hợp để rồi thiết kế sơ đồ nguyên lý, Thiết kế phần mềm điều khiển bằng cách vẽ lưu đồ thuật toán của các chế độ hoạt động.

* *Chương 3: Thử nghiệm và đánh giá*

Trình bày về mô hình thực nghiệm và hướng dẫn sử dụng sản phẩm. Phân tích tính năng và hiệu quả của sản phẩm. Đưa ra tính ứng dụng, mức độ an toàn và tác động của sản phẩm tới môi trường, kinh tế, xã hội.

# TỔNG QUAN VỀ CÔNG NGHỆ ĐIỀU KHIỂN TỪ XA

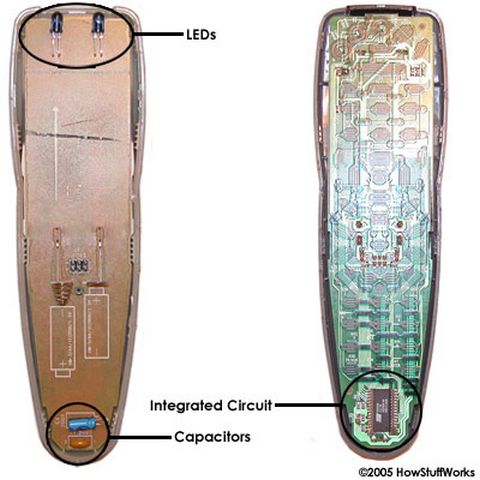
Trong một căn nhà thông minh hay Smarthome hiện nay, mọi nơi sẽ được kiểm soát bằng các thiết bị điện tử. Chúng sẽ sử dụng các cách giao tiếp riêng đề hiểu nhau như: Bluetooth, Zigbee, Z-Wave, Wifi, Matter, KNX,… Nhờ ứng dụng các công nghệ như hồng ngoại, điện thoại thông minh, IoT, công nghệ đám mây…Nhà thông minh có thể tự động giúp bạn làm những công việc trong nhà. Và công nghệ điều khiển từ xa các thiết bị điện trong nhà là một phần không thể thiếu trong một căn nhà thông minh.

Hệ thống công nghệ điều khiển thiết bị từ xa là các công nghệ cho phép người dùng từ xa có thể điều khiển và quản lý các thiết bị như máy tính, thiết bị điện tử, hệ thống thông tin, điều hòa, đèn chiếu sáng, cửa, camera... thông qua kết nối mạng internet hoặc các công nghệ truyền thông khác. Hệ thống này cho phép người dùng thực hiện các thao tác từ xa như bật/tắt thiết bị, điều chỉnh nhiệt độ, độ sáng, lịch trình hoạt động... giúp tiết kiệm thời gian, tiền bạc và năng lượng.

## Một số công nghệ điều khiển từ xa hiện nay

### Tia hồng ngoại (IR)

Một chiếc điều khiển IR sẽ gồm các bộ phận cơ bản nằm trong một hộp nối cáp kỹ thuật số như Hình 1.1 gồm: một bảng mạch tích hợp; tụ điện; đi - ốt phát quang (LED).



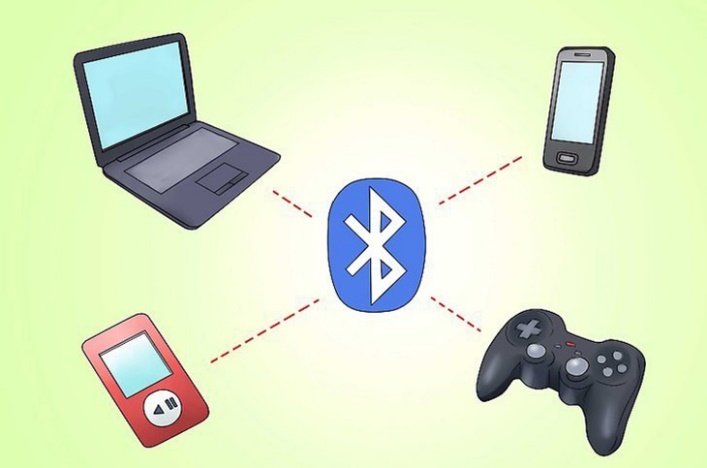
Hình 1.1. Sơ đồ bộ điều khiển từ xa sử dụng IR

Nguyên lý cơ bản của loại điều khiển từ xa này là sử dụng ánh sáng hồng ngoại của quang phổ điện từ mà mắt thường không thấy được để chuyển tín hiệu đến thiết bị cần điều khiển. Nó đóng vai trò như một bộ phát tín hiệu, sẽ phát ra các xung ánh sáng hồng ngoại mang một mã số nhị phân cụ thể. Khi ta ấn một nút phía bên ngoài thì sẽ vận hành một chuỗi các hoạt động khiến các thiết bị cần điều khiển sẽ thực hiện lệnh của nút bấm đó. [10]

* *Ưu điểm:*
* Giá thành thấp vì chúng được tích hợp vào nhiều thiết bị gia dụng.
* Dễ sử dụng: Không cần cài đặt phần mềm hay nắm vững kỹ thuật để điều khiển các thiết bị.
* Tiết kiệm năng lực: Không cần phải di chuyển điều khiển tới từng thiết bị để điều khiển, tiết kiệm thời gian, công sức.
* *Nhược điểm:*
* Phải trỏ thiết bị điều khiển vào thiết bị nhận để điều khiển. Nếu có vật cản giữa hai thiết bị thì sẽ không thể truyền tín hiệu được.
* Tầm hoạt động hạn chế: Thường chỉ trong khoảng 10-20m và có thể bị giảm tầm xa nếu ánh sáng mạnh (như ánh sáng mặt trời) chiếu vào.
* Không thể điều khiển qua tường hoặc các vật cản khác.
* Thiết bị phải được cài đặt đúng mã hồng ngoại để có thể điều khiển được.
* *Ứng dụng:*
* Điều khiển tivi, đầu đĩa DVD hoặc các thiết bị gia đình khác
* Thiết bị y tế: Một số thiết bị y tế sử dụng tia hồng ngoại để đo nhiệt độ
* Các hệ thống an ninh và cửa tự động: Tia hồng ngoại có thể được sử dụng để mở khóa cửa hoặc kích hoạt các thiết bị an ninh.

### Công nghệ điều khiển từ xa sử dụng Bluetooth

Bluetooth là sự trao đổi dữ liệu không dây tầm gần giữa các thiết bị điện tử. Bluetooth sử dụng sóng Radio tần số 2.4GHz. [7]



Hình 1.2. Kết nối các thiết bị bằng Bluetooth

Hình 1.2 mô tả công nghệ Bluetooth hỗ trợ kết nối dữ liệu qua các khoảng cách ngắn giữa các thiết bị di động như điện thoại di động, laptop, máy chơi game với nhau và với thiết bị cố định mà không cần một sợi cáp để truyền tải.

* *Ưu điểm:*
* Kết nối không dây tiện lợi: Công nghệ Bluetooth cho phép người dùng kết nối các thiết bị từ xa mà không cần dây cáp.
* Tiết kiệm năng lượng: Bluetooth tiêu thụ ít năng lượng hơn so với các công nghệ kết nối khác, giúp pin của thiết bị được sử dụng lâu hơn.
* Độ bảo mật cao: Công nghệ Bluetooth được sử dụng bởi nhiều thiết bị để truyền tải thông tin và đảm bảo an toàn.
* *Nhược điểm:*
* Khoảng cách kết nối ngắn: Công nghệ Bluetooth có khoảng cách kết nối giới hạn, nên người dùng cần ở gần thiết bị để kết nối.
* Tốc độ truyền dữ liệu chậm hơn: So với các công nghệ kết nối khác, tốc độ truyền dữ liệu của Bluetooth chậm hơn.
* Chọn mua phiên bản Bluetooth tương thích: Người dùng cần phải chọn phiên bản Bluetooth tương thích với thiết bị của mình.
* *Ứng dụng:*
* Điều khiển từ xa các thiết bị gia dụng như TV, loa, máy chiếu, quạt, đèn....
* Điều khiển robot, đồ chơi từ xa bằng Bluetooth.
* Điều khiển thiết bị y tế từ xa bằng Bluetooth.

### Công nghệ điều khiển từ xa sử dụng Wifi

Wifi là viết tắt của Wireless Fidelity là hệ thống truy cập internet không dây. Wifi là công cụ kết nối không thể thiếu trên điện thoại, laptop, máy tính bảng và một số thiết bị thông minh khác như smartwatch.

Công nghệ điều khiển thiết bị từ xa bằng Wifi là một phương pháp cho phép người dùng điều khiển và quản lý các thiết bị được kết nối trực tiếp với mạng Wifi. Cho phép người dùng từ xa tùy chỉnh và điều khiển thiết bị thông qua ứng dụng được cài đặt trên điện thoại hoặc máy tính. [3]

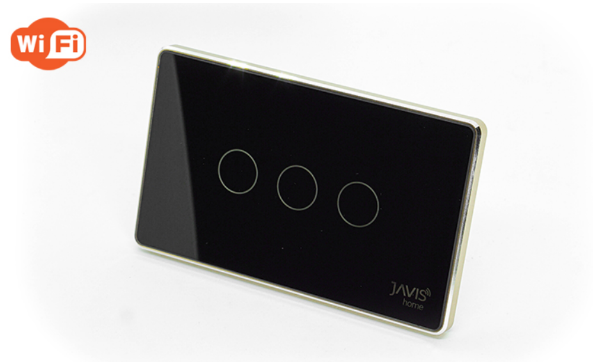
* *Ưu điểm:*
* Tiết kiệm thời gian: Người dùng có thể điều khiển các thiết bị từ xa giúp tiết kiệm thời gian và công sức.
* Tiện lợi: Người dùng có thể quản lý các thiết bị từ bất kỳ đâu, bất kỳ lúc nào chỉ cần có kết nối Wifi.
* Thông minh hơn: thường được tích hợp các tính năng thông minh, ví dụ như hẹn giờ, giọng nói,....
* *Nhược điểm:*
* Phụ thuộc vào kết nối wifi: Nếu kết nối wifi bị gián đoạn hoặc không ổn định, việc điều khiển thiết bị sẽ trở nên rất khó khăn.
* An ninh: Công nghệ điều khiển thiết bị từ xa bằng wifi có thể dễ bị tấn công bởi hacker, chủ yếu là những thiết bị có cấu hình kém hoặc không được bảo vệ đúng cách.
* Giá thành: Một số thiết bị có tính năng điều khiển từ xa wifi có giá thành cao.
* *Ứng dụng:*
* Điều khiển các thiết bị gia đình: Các thiết bị gia đình như máy lạnh, máy giặt, tủ lạnh, cửa cuốn…
* Giám sát an ninh: Các hệ thống an ninh chung cư hoặc các thiết bị giám sát nhà ở.
* Điều khiển các robot: có thể điều khiển các robot, máy bay không người lái và xe tự lái từ xa.
* Trợ lý gia đình: Các trợ lý ảo như Google Home, Amazon Echo hoặc Apple HomeKit.

## Các hệ thống điều khiển thiết bị điện trong nhà thông minh hiện nay

### Hệ thống điều khiển điện thông minh [4]

Hệ thống điều khiển điện trong ngôi nhà thông minh cũng là một phần khá quan trọng tạo nên sự thông minh của một ngôi nhà. Có thể kể đến đó là hệ thống các công tắc điện thông minh. Đây là một loại công tắc ngoài những thao tác đơn giản để bật/tắt thông thường bằng cách chạm vào phím cảm ứng mà chúng còn có khả điều khiển từ xa bằng các thiết bị smartphone, điều khiển từ xa, giọng nói thông qua kết nối internet và chúng có thể kết với với các thiết bị điện tử thông minh như bộ điều khiển trung tâm, hệ thống đèn… thông qua các chuẩn kết nối tiên tiến như Blutuool Mesh, Wifi, Zigbee…

Đây là thiết bị cũng khá quan trọng và cũng đáng để đầu tư trong các hệ thống trong ngôi nhà thông minh vì chúng có những điểm nổi bật như có thể hoạt động độc lập mà không phụ thuộc vào bộ điều khiển trung tâm (Công tắc điều khiển điện thông minh chuẩn kết nối Wifi).

Hiện nay trên thị trường có 2 loại công tắc điều khiển điện thông minh chủ yếu đó là công tắc thông minh với chuẩn kết nối wifi và công tắc thông minh với chuẩn kết nối điều khiển Blutooth Mesh (Loại này phải dùng kết hợp với hộ điều khiển trung tâm). Như Hình 1.3.

Hình 1.3. Công tắc điều khiển thông minh

### Hệ thống cảm biến thông minh [4]

Hệ thống cảm biến thông minh có nhiệm vụ như một “vệ sỹ” của ngôi nhà thông minh. Chúng tiếp nhận các dữ liệu đầu vào như sự chuyển động, ánh sáng, nhiệt độ, độ ẩm… và gửi thông tin về bộ xử lý trung tâm. Từ đây, bộ điều khiển trung tâm sẽ “ra lệnh” để điều khiển cac thiết bị hay các kịch bản mà bạn đã cài đặt sãn theo nhu cầu của mình. Các loại cảm biến trong ngôi nhà thông minh hiện nay được phân chia dựa trên chức năng hoạt động, bao gồm:

* Cảm biến chuyển động
* Cảm biến ánh sáng
* Cảm biến cửa
* Cảm biến khói
* Cảm biến bụi mịn
* Cảm biến nhiệt độ, độ ẩm
* Hệ thống cảm biến trong ngôi nhà thông minh
* Cảm biến ánh sáng trong ngôi nhà thông minh như Hình 1.4.



Hình 1.4. Cảm biến ánh sáng trong ngôi nhà thông minh

## Kết luận phần 1

Qua quá trình tìm hiểu về tổng quan về một số công nghệ điều khiển từ xa và các công nghệ điều khiển thiết bị điện trong nhà thông minh hiện nay, nhóm em quyết định chọn và hoàn thành đề tài tốt nghiệp “**Nghiên cứu, thiết kế mô hình hệ thống điều khiển thiết bị điện trong nhà sử dụng STM32**”. Đây là mô hình nhỏ có tính ứng dụng rất thực tế có thể sử dụng cho các hộ gia đình nhỏ. Nội dung thiết kế chi tiết sẽ được trình bày ở các chương tiếp theo.

# THIẾT KẾ MÔ HÌNH ĐIỀU KHIỂN THIẾT BỊ ĐIỆN TRONG NHÀ SỬ DỤNG STM32

## Phân tích yêu cầu thiết kế

### Mục tiêu thiết kế

* Thiết kế mô hình điều khiển bật, tắt 3 thiết bị điện: đèn 220V-5W, quạt 5V-1.25W có đèn Led 5V siêu sáng trắng kèm theo và Motor điện áp 3-12VDC.
* Điều khiển được thiết bị điện qua nút nhấn trên mô hình
* Điều khiển được thiết bị điện từ xa qua điện thoại thông minh
* Cài đặt thời gian bật tắt của các thiết bị theo mong muốn
* Điều khiển được thiết bị qua giọng nói

### Điều kiện ràng buộc của thiết kế

* Nguồn điện phải ổn định và đáp ứng các yêu cầu an toàn điện.
* Thiết bị điều khiển phải được thiết kế để hoạt động đúng cách với các thiết bị điện khác đồng bộ và không gây ra các vấn đề về an toàn.
* Các tín hiệu điều khiển phải được chuyển tải một cách đáng tin cậy, chính xác và đáp ứng các yêu cầu về tốc độ và khả năng truyền tải.
* Hệ thống điều khiển phải bảo vệ toàn diện cho các thiết bị điện, người sử dụng và toàn bộ ngôi nhà.
* Đảm bảo mô hình điều khiển phải tối ưu hóa tính đơn giản, dễ sử dụng và tiện ích cho người sử dụng.
* Tuân thủ các quy định về an toàn điện và môi trường, bảo đảm sự an toàn người sử dụng và bảo vệ môi trường xung quanh.

### Thông số kỹ thuật

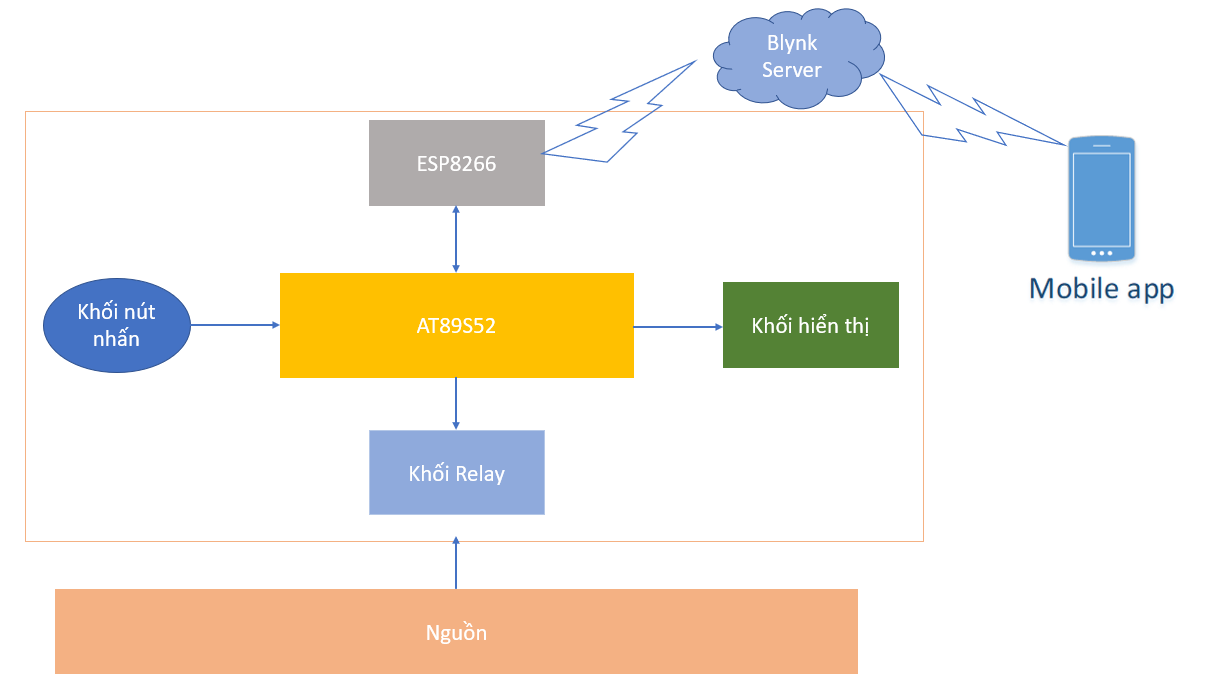
* Điện áp hoạt động: 220VAC cấp cho Đèn 220V-5W; 5V cấp cho khối relay và chân vào của STM32F103C8T6, ESP8266; 3.3V cấp cho khối nút nhấn, khối hiển thị.
* Giao thức kết nối là giao thức Wifi
* Mô hình điều khiển thiết bị điện trong nhà có thể hỗ trợ chức năng điều khiển từ xa bằng ứng dụng trên điện thoại di động.

### Tiêu chí đánh giá sản phẩm

* Mạch chạy ổn định, không bị lỗi.
* Hoạt động được trong một thời gian dài.
* Sản phẩm có tính ứng dụng thực tiễn.
* Hệ thống đơn giản, an toàn và dễ dàng với người sử dụng.
* Có độ bền và tính thẩm mỹ cao.

### Một số giải pháp thực hiện

* *Giải pháp 1: Sử dụng vi điều khiển AT89S52 giao tiếp với module ESP8266*



Hình 2.1. Sơ đồ khối sử dụng vi điều khiển AT89S52

* Chức năng các khối:
* Khối nguồn chịu trách nhiệm cấp nguồn cho các khối nút nhấn, khối relay, AT89S52 và ESP8266, LCD.
* Khối nút nhấn chịu trách nhiệm nhận lệnh điều khiển bật tắt các thiết bị từ người dùng.
* Khối relay đóng mở nguồn cho thiết bị cần điều khiển.
* Khối xử lý trung tâm AT89S52 chịu trách nhiệm giao tiếp nhận lệnh điều khiển từ khối nút nhấn, xử lý yêu cầu đóng mở relay, gửi dữ liệu tới khối hiển thị LCD và khối giao tiếp mạng ESP8266.
* Khối hiển thị sử dụng Module PCF8574 I2C để hạn chế chân của LCD và dùng LCD để hiển thị trực quan trạng thái bật tắt của các thiết bị.
* Khối giao tiếp mạng sử dụng module ESP8266 có nhiệm vụ nhận lệnh điều khiển từ khối xử lý trung tâm sau đó gửi dữ liệu lên server qua đường truyển internet thông qua wifi, đồng thời tiếp nhận lệnh điều khiển từ server để truyền tới khối xử lý trung tâm ra lệnh bật tắt thiết bị.
* Server blynk nhận dữ liệu từ ESP8266 và hiển thị trực quan trên bảng điều khiển Dashboard cung cấp khả năng giám sát từ xa.
* Ưu, nhược điểm khi sử dụng vi điều khiển AT89S52
* Ưu điểm:

+ Giá thành của AT89S52 rẻ

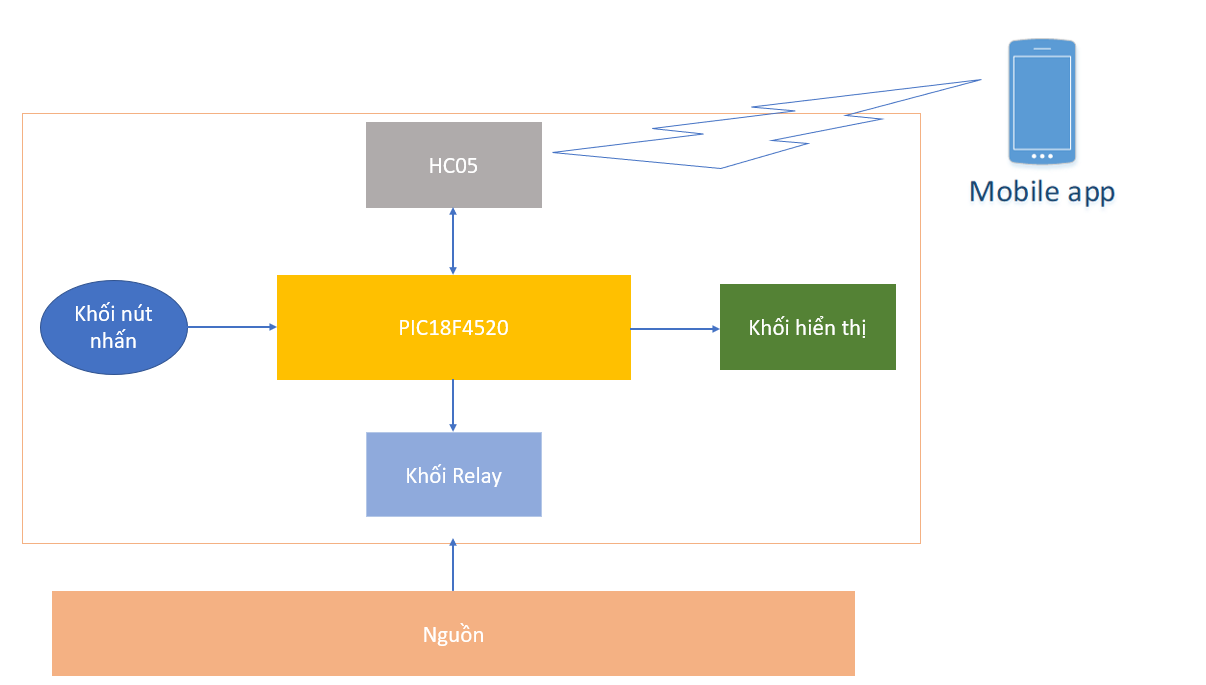
+ Dễ tiếp cận vì là vi điều khiển 8bit

* Nhược điểm:

+ Hạn chế về tài nguyên khai thác như không có bộ ADC

+ Chỉ có 1 bộ UART,…

* *Giải pháp 2: Sử dụng vi điều khiển PIC18F420 giao tiếp với module HC05*



Hình 2.2. Sơ đồ khối sử dụng vi điều khiển PIC18F4520

* Chức năng các khối:
* Khối nguồn chịu trách nhiệm cấp nguồn cho các khối nút nhấn, khối relay, PIC18F4520, HC05 và LCD.
* Khối nút nhấn chịu trách nhiệm nhận lệnh điều khiển bật tắt các thiết bị từ người dùng.
* Khối relay đóng mở nguồn cho thiết bị cần điều khiển.
* Khối xử lý trung tâm PIC18F4520 chịu trách nhiệm giao tiếp nhận lệnh điều khiển từ khối nút nhấn, xử lý yêu cầu đóng mở relay, gửi dữ liệu tới khối hiển thị LCD và kết nối với module HC05 qua giao tiếp UART.
* Khối hiển thị sử dụng Module PCF8574 I2C để hạn chế chân của LCD và dùng LCD để hiển thị trực quan trạng thái bật tắt của các thiết bị.
* Module HC05 phát ra Bluetooth để Smartphone kết nối sau đó điều khiển thiết bị.
* Mobile App là một thiết bị điện tử như smartphone sử dụng hệ điều hành Android có khả năng kết nối Bluetooth, Internet và tải về ứng dụng MIT AI2 Companion trên CH Play.
* Ưu, nhược điểm khi sử dụng vi điều khiển PIC18F4520
* Ưu điểm:

+ Sử dụng cho các bài toán có độ phức tạp cao

+ Được ứng dụng rộng rãi trong công nghiệp

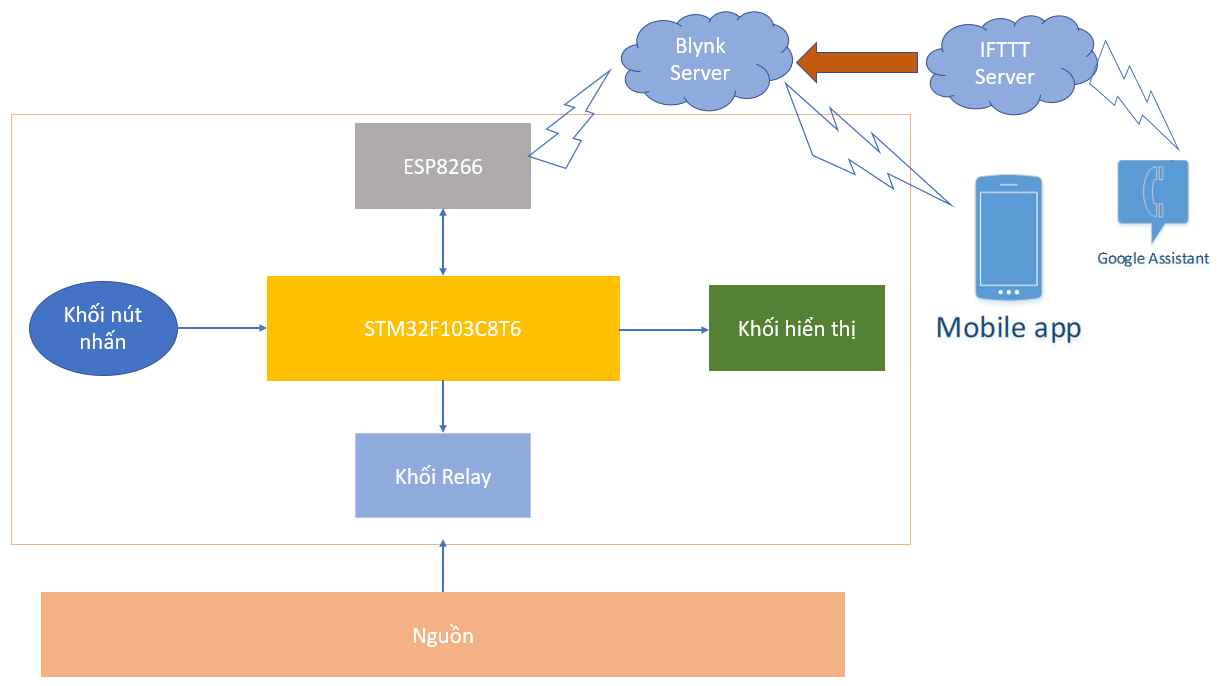
+ Đã được hỗ trợ nhiều tài nguyên hơn so với AT89S52

* Nhược điểm:

+ Giá thành khá cao so với các dòng vi điều khiển tương tự

+ Tốc độ xử lý chậm hơn so với các dòng vi điều khiển hiện nay

* *Giải pháp 3: Sử dụng vi điều khiển STM32F103C8T6 giao tiếp với module ESP8266*



Hình 2.3. Sơ đồ khối sử dụng vi điều khiển STM32F103C8T6

* Chức năng các khối:
* Khối nguồn chịu trách nhiệm cấp nguồn cho các khối nút nhấn, khối relay, STM32F103C8T6 và ESP8266.
* Khối nút nhấn chịu trách nhiệm nhận lệnh điều khiển bật tắt các thiết bị từ người dùng.
* Khối relay đóng mở nguồn cho thiết bị cần điều khiển.
* Khối xử lý trung tâm STM32 chịu trách nhiệm giao tiếp nhận lệnh điều khiển từ khối nút nhấn, xử lý yêu cầu đóng mở relay, gửi dữ liệu tới khối hiển thị LCD và khối giao tiếp mạng ESP8266.
* Khối hiển thị sử dụng Module PCF8574 I2C để hạn chế chân của LCD và dùng LCD để hiển thị trực quan trạng thái bật tắt của các thiết bị.
* Khối giao tiếp mạng sử dụng module ESP8266 có nhiệm vụ nhận lệnh điều khiển từ khối xử lý trung tâm sau đó gửi dữ liệu lên server qua đường truyển internet thông qua wifi, đồng thời tiếp nhận lệnh điều khiển từ server để truyền tới khối xử lý trung tâm ra lệnh bật tắt thiết bị.
* Server blynk nhận dữ liệu từ ESP8266 và hiển thị trực quan trên bảng điều khiển Dashboard cung cấp khả năng giám sát từ xa.
* Server IFTTT nhận lệnh giọng nói từ Google Assistant trên điện thoại sau đó gửi lệnh đến server blynk để bật tắt thiết bị.
* Ưu điểm:
* Sử dụng cho các bài toàn có độ phức tạp cao
* Được ứng dụng rộng rãi trong công nghiệp
* Dòng vi điều khiển có nhiều ứng dụng thực tế
* Tốc độ xử lý nhanh, đã được cải thiện so với những dòng vi điều khiển cùng tầm giá
* Giá thành rẻ
* Đã được hỗ trợ nhiều tài nguyên hơn so với AT89S52 cũng như PIC18F4520
* Lựa chọn và đánh giá giải pháp tối ưu

Từ những ưu và nhược điểm của 3 giải pháp nhóm em đã rút ra bảng so sánh giữa 3 giải pháp như bảng 2.3

Bảng 2.1. Bảng so sánh 3 giải pháp

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu chí** | **Giải pháp 1** [5] | **Giải pháp 2** [6] | **Giải pháp 3** [2] |
| Dung lượng | 8KB Flash, 256byte SRAM | 32KB Flash, 2KB SRAM | 64KB Flash, 20KB SRAM |
| Công nghệ sử dụng | Wifi | Bluetooth | Wifi |
| Chi phí | Khoảng 400.000VND | Khoảng 485.000VND | Khoảng 445.000VND |
| Tần số hoạt động vi điều khiển | Tối đa 24MHz | Tối đa 40MHz | Tối đa 72MHz |
| Ứng dụng | được dùng trong các ứng dụng nhỏ và đơn giản, chẳng hạn như các ứng dụng nhúng | ứng dụng rộng rãi trong các thiết bị điện tử trong công nghiệp và các sản phẩm tiêu dùng | được ứng dụng trong hệ thống điều khiển và các sản phẩm điện tử cao cấp |

Từ những số liệu so sánh 3 giải pháp ở Bảng 2.1 và ưu nhược điểm của ba giải pháp mà nhóm đã nghiên cứu. Về tổng quan, giải pháp 3 sử dụng STM32F103C8T6 có tốc độ xử lý nhanh hơn đáng kể so với AT89S52 và PIC18F4520. PIC18F4520 có dung lượng bộ nhớ Flash lớn hơn AT89S52. AT89S52 có dung lượng bộ nhớ thấp nhất trong số ba vi điều khiển này. Giải pháp 3 còn được ứng dụng trong các hệ thống điều khiển và các sản phẩm điện tử cao cấp. Vậy nên nhóm lựa chọn giải pháp 3 để áp dụng vào mô hình điều khiển thiết bị điện trong nhà.

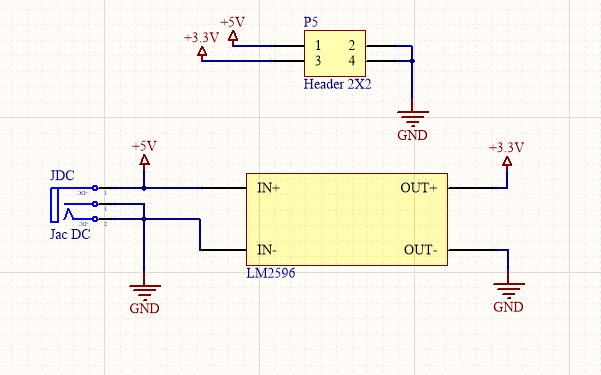
## Thiết kế mô hình điều khiển thiết bị điện trong nhà

### Thiết kế sơ đồ nguyên lý mạch điều khiển thiết bị trong nhà

Theo yêu cầu của bài toán đặt ra, nhóm đã lựa chọn giải pháp 3 sử dụng vi điều khiển STM32 giao tiếp với module ESP8266 do vậy sơ đồ khối của hệ thống và các chức năng các khối đã được trình bày ở Hình 2.3 của mục 2.1.3 trang 14.

#### Thiết kế khối nguồn

Trong mô hình điều khiển thiết bị điện trong nhà có sử dụng 3 nguồn:



**B**

**A**

Hình 2.4. Sơ đồ nguyên lý khối nguồn

Nguồn 220VAC để cung cấp riêng cho đèn 220V-5W. Trong sơ đồ nguyên lý khối nguồn có sử dụng một Adapter nhiệm vụ chuyển từ 220VAC xuống 5VDC để cung cấp cho chân Vin ESP8266, STM32 và khối relay. Sau đó dùng module LM2596 để hạ áp xuống 3.3VDC như Hình 2.4B để cung cấp cho nút nhấn và khối hiển thị. Header 2X2 nối với các nguồn 5V và 3.3V dự phòng cho các mục đích sử dụng khác nếu có trong tương lai như Hình 2.4A.

LM2596 là một IC hạ áp được sử dụng phổ biến. Phiên bản điều chỉnh có thể nhận điện áp đầu vào từ 4,5V đến 40V và chuyển đổi thành nguồn điện áp liên tục 3.3V.



Hình 2.5. Module LM2596

Hình 2.5 là hình dạng thực tế của module LM2596 gồm IC LM2596 và biến trở sử dụng để điều chỉnh điện áp đầu ra.

Đặc tính thông số kỹ thuật của LM2596: [8]

* Có bộ điều chỉnh 3.3V, bộ điều chỉnh 5V, bộ điều chỉnh 12V và bộ điều chỉnh biến đổi
* Điện áp đầu vào: 4,5V đến 40V
* Dải điện áp đầu ra có thể điều chỉnh: 1.23V đến 38.5V
* Dòng điện đầu ra liên tục: 3A
* Dòng đầu ra đỉnh: 6.9A
* Tần số chuyển mạch: 150KHz

#### Thiết kế khối nút ấn

Trong đề tài sẽ sử dụng 3 nút nhấn để điều khiển bật tắt của 3 thiết bị. Nút nhấn đó có hình dạng thực tế như Hình 2.6

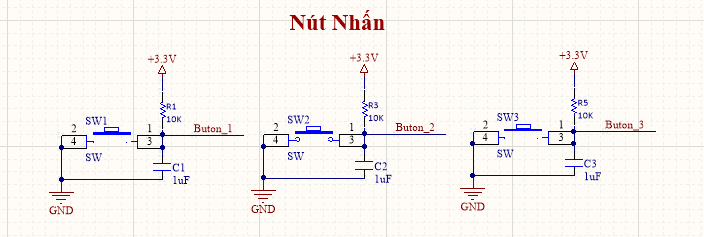


Hình 2.6. Nút nhấn 4 chân

Thông số kỹ thuật của nút nhấn: [9]

* Số chân: 4 chân.
* Dòng điện chịu đựng: 2A.
* Màu sắc: đen.
* Hình dạng: vuông.
* Kích thước: 6 x 6 x 8mm.

Sau khi tìm hiểu được các thông số kỹ thuật sơ đồ chân của nút nhấn thì đề tài sẽ thiết kế sơ đồ nguyên lý khối nút nhấn được thể hiện ở Hình 2.7



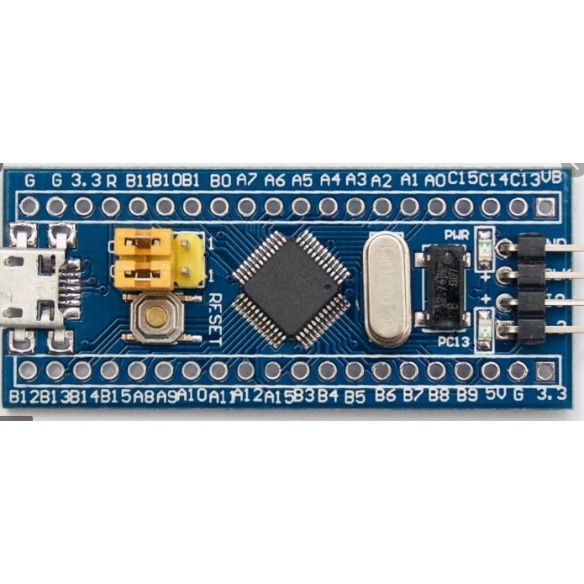
Hình 2.7. Sơ đồ nguyên lý khối nút nhấn

Trong đó lần lượt các nút nhấn SW1, SW2, SW3 sẽ điều khiển bật tắt lần lượt đèn, quạt và motor quay.

Khối nút nhấn được cấp nguồn +3.3V, tín hiệu đầu ra nối đến chân của vi điều khiển. Các điện trở R1, R3, R5 nối lên VCC để tạo mức logic 1 mặc định. Khi người dùng ấn nút trạng thái đầu ra sẽ ở mức logic 0. Đồng thời khi nút nhấn được nhấn xuống, tụ 1uF sẽ được nạp, làm tăng thời gian đáp ứng của mạch, giúp tránh tình trạng nhiễu.

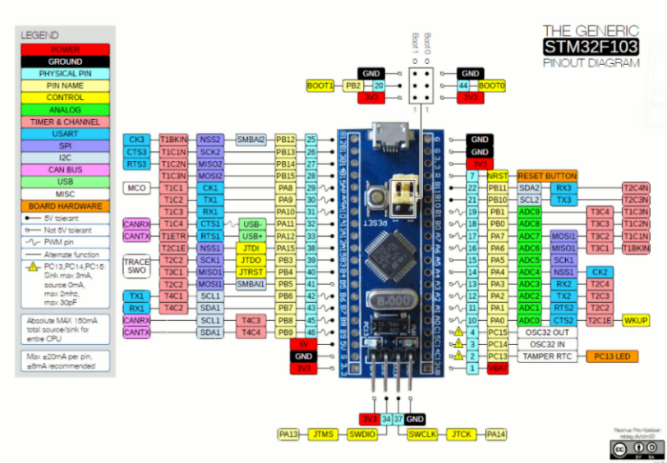
#### Thiết kế khối xử lý và giao tiếp mạng

Trong mọi hệ thống điều khiển cũng như trong hệ thống nhúng, khối xử lý luôn là bộ phận rất quan trọng. Khối này sẽ xử lý các tín hiệu được đưa vào và đưa ra các tín hiệu ra để điều khiển hệ thống hoạt động như mong muốn. Trong hệ thống điều khiển thiết bị điện trong nhà, khối xử lý sẽ được sử dụng bằng vi điều khiển STM32F103C8T6.



Hình 2.8. Kit STM32F103C8T6 BluePill

Hình 2.8 là hình ảnh thực tế của Kit STM32F103C8T6



Hình 2.9. Sơ đồ chân chức năng Kit STM32F103C8T6 BluePill

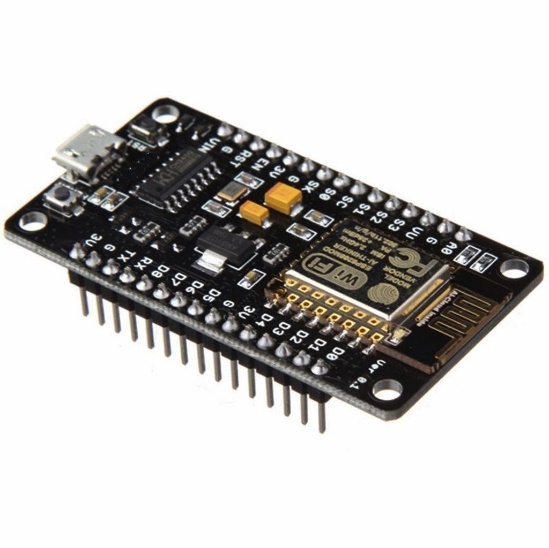
Từ sơ đồ chức năng ở Hình 2.9 có thể thấy Kit STM32F103C8T6 có các giao tiếp: CAN, I2C, SPI, UART,...

Thông số kỹ thuật của STM32F103C8T6: [1]

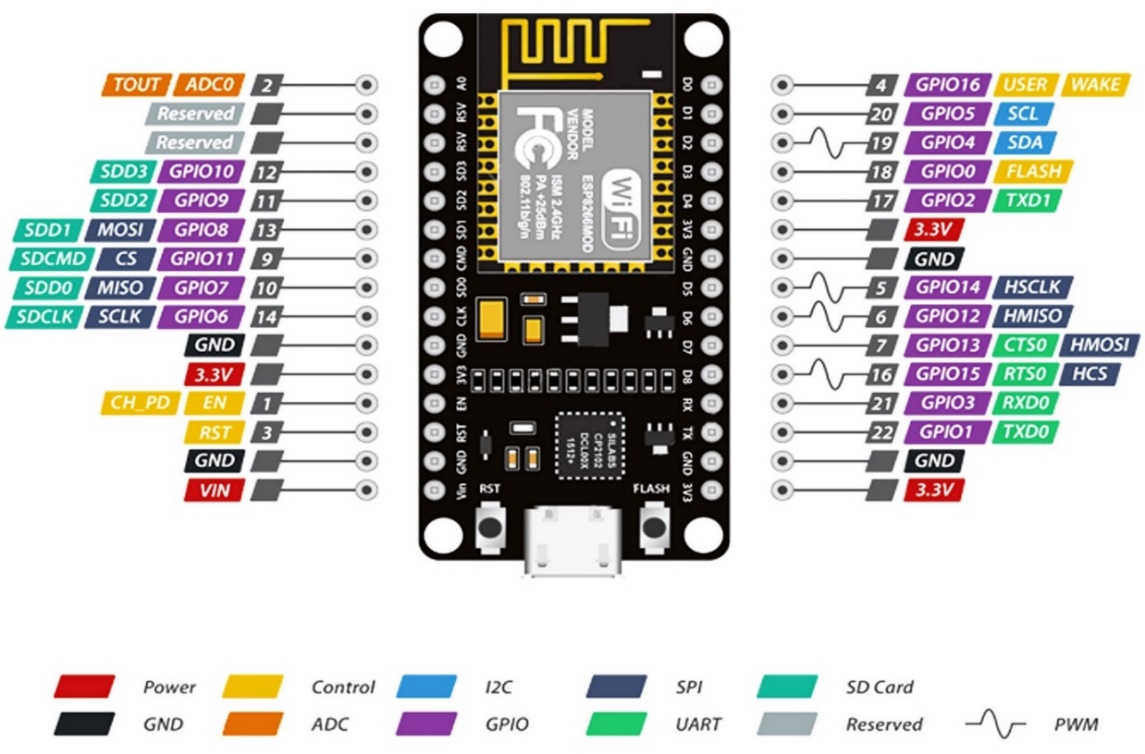
* Điện áp cấp 5VDC qua cổng Micro USB sẽ được chuyển đổi thành 3.3VDC qua IC nguồn và cấp cho Vi điều khiển chính.
* Tích hợp sẵn thạch anh 8Mhz.
* Tần số hoạt động lên đến 48MHz
* Bộ nhớ Flash: 32kB
* RAM: 4kB
* Kích thước: 53.34 x 15.24mm.

Để điều khiển được thiết bị điện trong nhà từ xa thì hệ thống sử dụng công nghệ Wifi vì có thể bật tắt bằng nút nhấn từ xa, bật tắt bằng cài đặt thời gian và có thể điều khiển bật tắt qua giọng nói.

ESP8266 là một chip của Espressif Systems có tích hợp công nghệ Wi-Fi. Sử dụng firmware tập lệnh AT tiêu chuẩn bằng cách kết nối nó với bất kỳ bộ vi điều khiển nào sử dụng UART nối tiếp hoặc trực tiếp làm bộ vi điều khiển hỗ trợ Wifi, bằng cách lập trình một chương trình cơ sở mới sử dụng SDK được cung cấp. Hình 2.10 là hình ảnh thực tế của ESP8266.



Hình 2.10. Module ESP8266



Hình 2.11. Sơ đồ chức năng module ESP8266

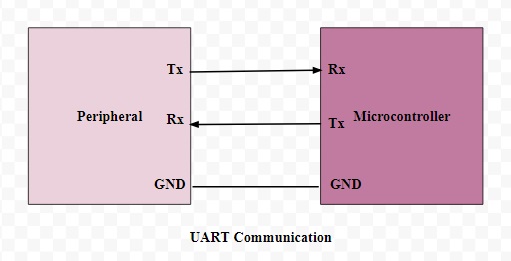
Sơ đồ chức năng ở Hình 2.11 cho thấy module ESP8266 cũng có các giao tiếp Analog, SPI, UART, PWM, …

Thông số kỹ thuật của ESP8266: [11]

* Bộ vi điều khiển: CPU RISC 32-bit Tensilica Xtensa LX106
* Điện áp hoạt động: 3.3V
* Điện áp đầu vào: 7-12V
* Chân I/O kỹ thuật số: 16
* Chân đầu vào tương tự (ADC): 1
* Bộ nhớ Flash: 4 MB
* SRAM: 64 KB
* Tốc độ đồng hồ: 80 MHz

Có thể thấy vi điều khiển STM32F103C8T6 với module ESP8266 đều có giao tiếp UART, vậy nên hệ thống sẽ chọn giao tiếp UART để kết nối khối xử lý và giao tiếp mạng.

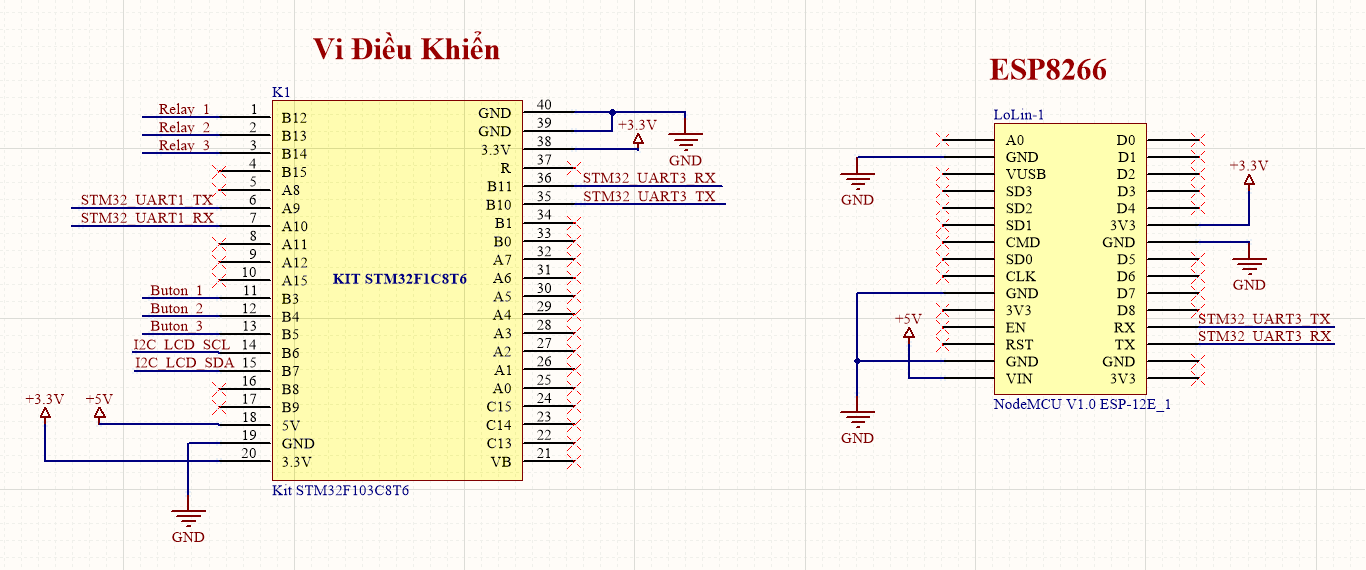
Tên đầy đủ của UART là “Universal Asynchronous Receiver / Transmitter”, và nó là một vi mạch sẵn có trong một vi điều khiển nhưng không giống như một giao thức truyền thông (I2C & SPI). Chức năng chính của UART là truyền dữ liệu nối tiếp. [12]



Hình 2.12. Sơ đồ kết nối giao tiếp UART

Dựa vào Hình 2.12 thì có thể thấy cách kết nối vi điều khiển với thiết bị ngoại vi tương đối đơn giản, chân TX của ngoại vi sẽ nối chân RX của vi điều khiển và chân RX của ngoại vi sẽ nối chân TX của vi điều khiển.

Sau khi tìm hiểu được các thông số kỹ thuật, sơ đồ chức năng của STM32F103C8T6 và ESP8266 và sơ đồ kết nối của giao tiếp UART thì đề tài sẽ thiết kế khối xử lý và giao tiếp mạng được thể hiện ở Hình 2.13



Hình 2.13. Sơ đồ nguyên lý khối xử lý và giao tiếp mạng

Kit STM32F103C8T6 gửi dữ liệu và nhận dữ liệu từ ESP8266 thông qua giao tiếp UART được nối từ các chân TX3 (PIN B10) và RX3 (PIN B11) của STM32 tới các chân TX0 và RX0 trên kit ESP8266. Ngoài ra STM32 sử dụng thêm UART1 để Debug.

#### Thiết kế khối Relay

Mô hình hệ thống sẽ điều khiển đèn 220V, quạt 5V, motor 5V. Vì điện áp nằm trong dải từ 5V đến 220V do vậy theo tính toán nhóm sẽ sử dụng Relay có điện áp tiếp điểm 250VAC. Do mạch được cung cấp nguồn 5VDC vậy nên nhóm đã sử dụng Relay có điện áp cuộn hút 5V và có 3 chân ra NO/NC và COM. Hình ảnh thực tế của Relay như ở Hình 2.14.



Hình 2.14. Relay 5V

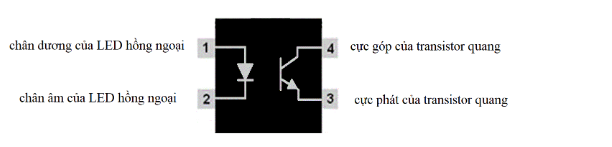
Thông số kỹ thuật của Relay: [13]

* Điện áp cuộn dây: 5V
* Điện áp tiếp điểm: 10A – 250VAC
* Số chân: 5 chân
* Kích thước: 19mm x 15.5mm x 15mm.

Trong khối Relay còn có thêm opto PC817 cách ly quang để cách ly mạch với cuộn hút. Hình ảnh thực tế được thể hiện ở Hình 2.15.



Hình 2.15. Opto PC817



Hình 2.16. Sơ đồ chân của opto PC817

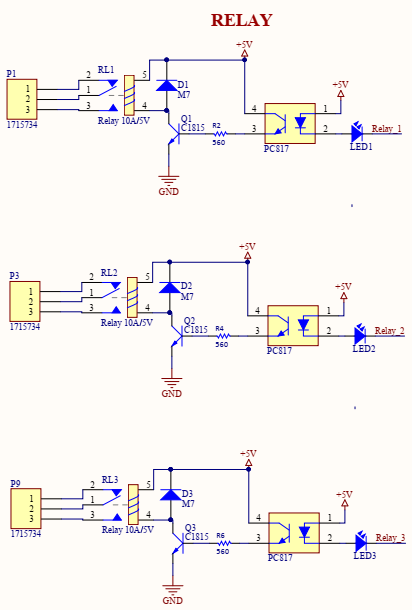
PC817 là một opto được sử dụng rất phổ biến, chứa một LED hồng ngoại và một transistor quang được thể hiện như Hình 2.16. Được sử dụng để cách ly hai mạch với nhau.

Thông số kỹ thuật của PC817: [14]

Bảng 2.2. Bảng thông số kỹ thuật của PC817

|  |  |
| --- | --- |
| Loại gói: | Dip 4 chân và SMT |
| Loại transistor: | NPN |
| Dòng cực góp tối đa (IC): | 50mA |
| Điện áp cực góp - cực phát tối đa: | 80V |
| Điện áp bão hòa cực góp - cực phát: | 0,1 đến 0,2 |
| Điện áp cực phát - cực gốc tối đa: | 6V |
| Công suất tiêu tán cực góp tối đa (Pc): | 200 mW |

Sau khi tìm hiểu về thông số và nguyên lý hoạt động của Relay và PC817, có thể thiết kế sơ đồ nguyên lý khối relay như ở Hình 2.17.



Hình 2.17. Sơ đồ nguyên lý khối relay

Module relay sử dụng các relay có cuộn hút 5V, tín hiệu điều khiển từ STM32 được cách ly bằng các opto PC817. Các transistor dùng để dẫn dòng đi qua cuộn dây, R2, R4, R6 để phân cực cho transistor.

Khi vi điều khiển xuất ra mức logic 0, khi đó sẽ có dòng qua LED làm LED phát sáng đồng thời chân 3 và chân 4 của PC817 sẽ được kích dẫn, dòng 5V sẽ vào cực B của transistor C1815, lúc đó chân C vs chân E nối thông. Lúc này sẽ có dòng điện từ +5V chảy qua cuộn hút của Relay kích hoạt tiếp điểm Relay.

Khi tắt dòng điện cho cuộn dây thì cuộn dây sẽ phóng điện ngược lại, nên cần sử dụng các Diode D1, D2, D3 để bảo vệ không cho dòng điện phóng ngược lại mạch.

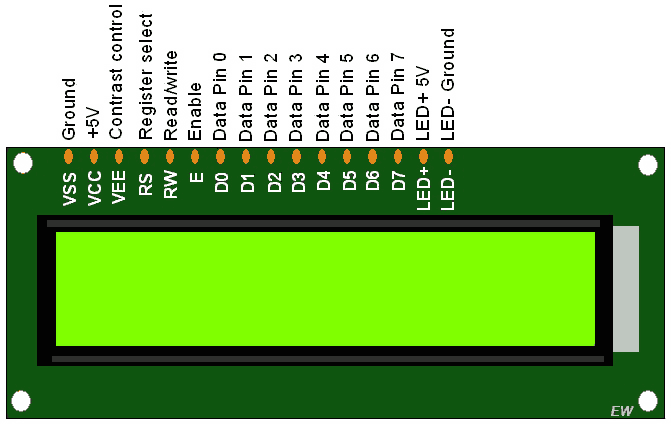
#### Thiết kế khối hiển thị

Trong hệ thống điều khiển thiết bị điện trong nhà sử dụng khối hiển thị bao gồm LCD và module PCF8574 I2C.



Hình 2.18. Màn hình LCD 1602

LCD có rất nhiều dạng phân biệt theo kích thước từ vài kí tự đến hàng chục ký tự, từ một hàng đến vài chục hàng. Trong đồ án sử dụng LCD 1602 có nghĩa là có 2 hàng, mỗi hàng có 16 ký tự được mô tả ở Hình 2.18. Sơ đồ chân của LCD được mô tả ở Hình 2.19.

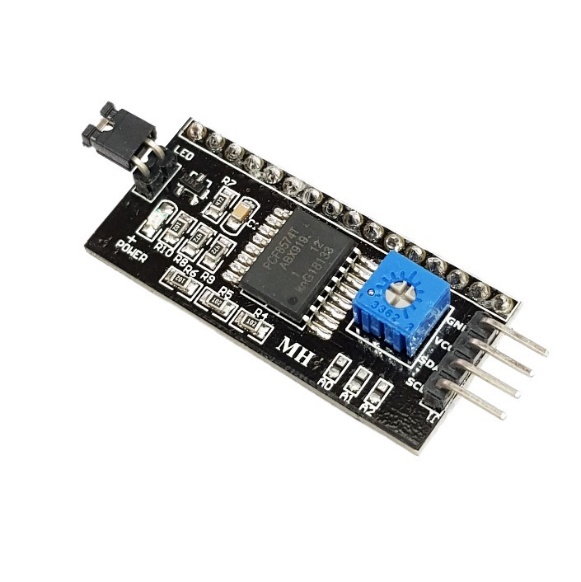


Hình 2.19. Sơ đồ chân LCD 1602

Thông số kỹ thuật của LCD: [15]

* Điện áp MAX: 7V
* Điện áp MIN: - 0,3V
* Hoạt động ổn định: 2.7-5.5V
* Điện áp ra mức cao: > 2.4
* Điện áp ra mức thấp: < 0.4V
* Dòng điện cấp nguồn: 350uA - 600uA
* Nhiệt độ hoạt động: - 30 - 75 độ C

LCD có quá nhiều nhiều chân gây khó khăn trong quá trình đấu nối và chiếm dụng nhiều chân trên vi điều khiển. Để khắc phục tình trạng này thì hệ thống sử dụng **Module PCF8574 I2C** để giao tiếp với vi điều khiển thông qua giao tiếp I2C.



Hình 2.20. Module PCF8574 I2C

Hình ảnh thực tế của PCF8574 được thể hiện ở Hình 2.20. Thay vì phải mất 6 chân vi điều khiển để kết nối với LCD 1602 (RS, EN, D7, D6, D5 và D4) thì module **PCF8574** chỉ cần tốn 2 chân (SCL, SDA) để kết nối.

Module **PCF8574** hỗ trợ các loại LCD sử dụng driver HD44780(LCD 16×2, LCD 20×4, …) và tương thích với hầu hết các vi điều khiển hiện nay.

Thông số kỹ thuật của PCF8574: [15]

* Điện áp hoạt động: 2.5-6V DC.
* Hỗ trợ màn hình: LCD1602,1604,2004 (driver HD44780).
* Giao tiếp: I2C.
* Địa chỉ mặc định: 0X27 (có thể điều chỉnh bằng ngắn mạch chân A0/A1/A2).
* Tích hợp Jump chốt để cung cấp đèn cho LCD hoặc ngắt.
* Tích hợp biến trở xoay điều chỉnh độ tương phản cho LCD.

PCF8574 giao tiếp với STM32 qua giao tiếp IC2, chân SCL nối chân SCL (PIN PB6) và chân SDA nối chân SDA (PIN PB7) của STM32F103C8T6.

Từ các thông số kĩ thuật và các ưu điểm của module PCF8574 thì hệ thống thiết kế khối hiển thị như Hình 2.21.

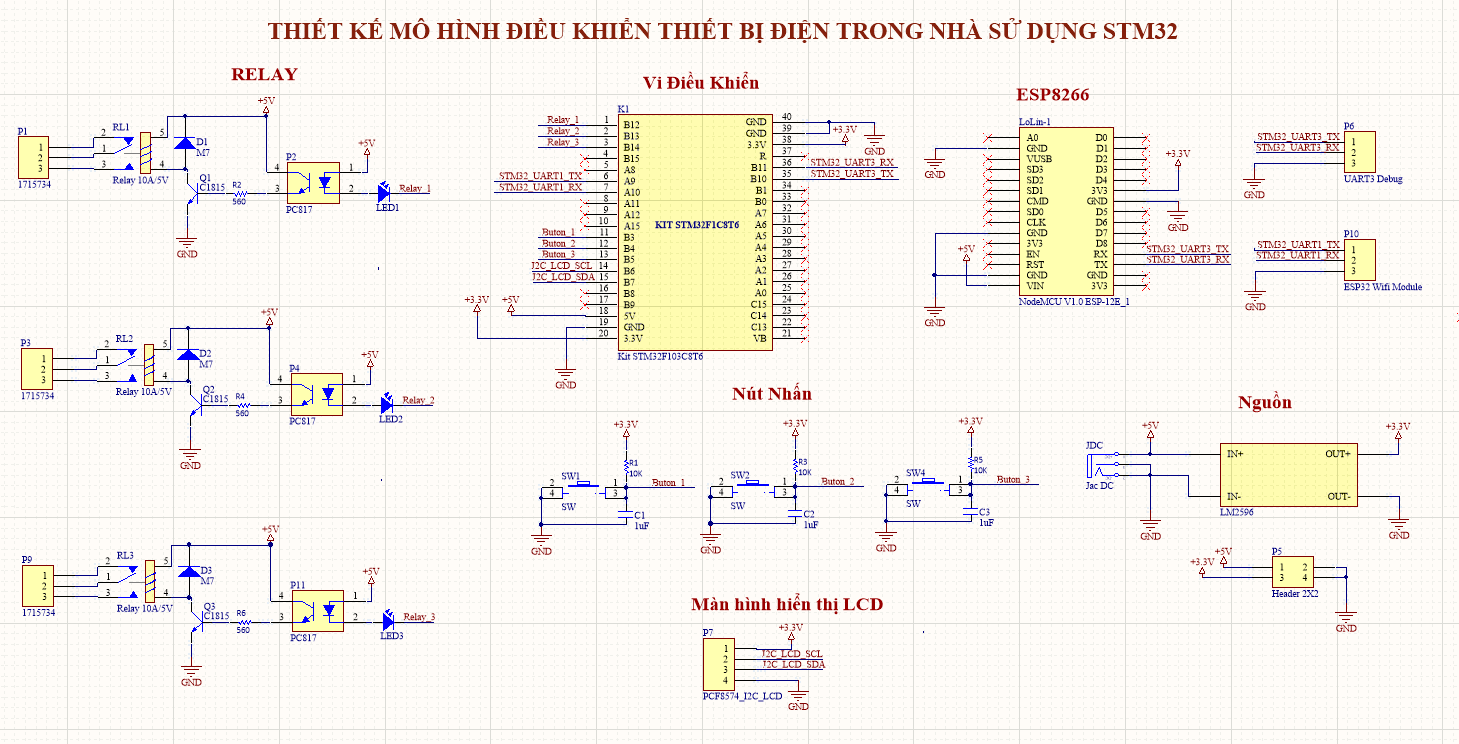


Hình 2.21. Sơ đồ kết nối PCF8574 với LCD

#### Sơ đồ nguyên lý và sơ đồ mạch in của hệ thống điều khiển thiết bị điện trong nhà

* Sơ đồ nguyên lý

Từ sơ đồ khối được thể hiện ở Hình 2.3 trang 14, sơ đồ nguyên lý và cách hoạt động của từng khối, có thể dễ dàng thiết kế được sơ đồ nguyên lý của toàn hệ thống cụ thể như Hình 2.22.



Hình 2.22. Sơ đồ nguyên lý toàn hệ thống

* Sơ đồ mạch in

Sau khi hoàn thành sơ đồ nguyên lý, nhóm đã thiết kế được sơ đồ mạch in như Hình 2.23.



Hình 2.23. Sơ đồ mạch in

### Thiết kế phần mềm

#### Thiết kế phần mềm khối vi điều khiển STM32

Thiết kế đường truyền giao tiếp giữa STM32 và ESP8266 sử dụng giao thức UART. Theo đó các lệnh điều khiển sẽ được đóng gói thành các chuỗi dữ liệu với định dạng như sau:

|S[a]:[b]-[c]E|

Trong đó các tham số lần lượt là:

1. : Mã lệnh chỉ định loại gói tin.
   * a = 1 thể hiện loại gói tin là dữ liệu,
   * a = 0 thể hiện loại gói tin là lệnh điều khiển
2. : Số thứ tự của relay cần điều khiển
3. : Lệnh muốn điều khiển:
   * c = 1 tương đương bật thiết bị.
   * c =0 tương đương tắt thiết bị.

Ví dụ:

* Để bật thiết bị 1 thì chuỗi dữ liệu truyền từ STM32 sang ESP8266 sẽ là:

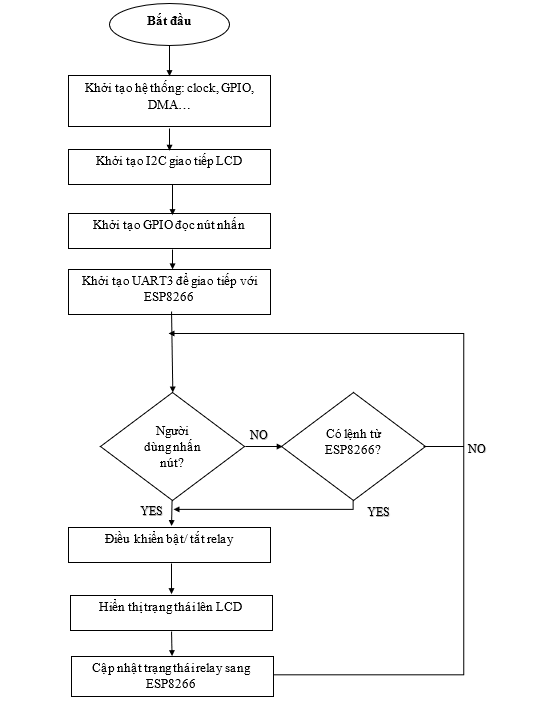
"|S[0]:[1]-[1]E|"

* Để tắt thiết bị 1 thì chuỗi dữ liệu truyền từ STM32 sang ESP8266 sẽ là:

"|S[0]:[1]-[0]E|"

* Lưu đồ thuật toán

Khi được cấp nguồn vi điều khiển sẽ khởi động giao tiếp I2C, UART3 để giao tiếp với module PCF8574, module ESP8266 và khỏi tạo GPIO để đọc các trạng thái của nút nhấn. Khi đọc được giá trị nút nhấn từ nhấn nút vật lý hoặc có lệnh từ ESP8266 thì vi điều khiển STM32F103C8T6 sẽ xuất tín hiệu ra để điều khiển các relay đóng mở theo yêu cầu. Sau đó cập nhật trạng thái các thiết bị lên LCD thông qua PCF8574 và cập nhật trạng thái sang ESP8266. Lưu đồ thuật toán được thể hiện ở Hình 2.24.



Hình 2.24. Lưu đồ thuật toán của STM32F103C8T6

#### Thiết kế phần mềm khối giao tiếp mạng ESP8266

ESP8266 sẽ có 2 chế độ hoạt động chính là:

* Chế độ “Wifi Access Point” để phát ra wifi cho phép các client kết nối tới.
* Chế độ “Wifi Station” để kết nối tới các AP khác, cụ thể trong trường hợp này là để kết nối tới AP ra mạng internet

Theo yêu cầu của đề tài, ESP8266 cần kết nối tới blynk server qua internet do đó ESP8266 cần hoạt động ở chế độ Wifi Station để kết nối tới router wifi có sẵn kết nối internet. Để làm được điều này cần phải cung cấp thông tin tên SSID và mật khẩu của mạng Wifi cho ESP8266, ngoài ra để kết nối tới blynk server cần có thêm thông tin blynk authen token.

Cơ chế cấu hình các thông tin cho ESP8266 được thiết kế như sau:

Mặc định khi khởi động ESP8266 sẽ thiết lập chế độ Wifi Station, sau đó kiểm tra bộ nhớ EEPROM xem đã có thông tin SSID, password, blynk token hay chưa, nếu chưa có thông tin hoặc đã có thông tin nhưng kết nối tới internet không thành công thì ESP8266 sẽ tự động chuyển sang chế độ “Wifi Access Point”.

Ở chế độ AP, ta cài đặt để ESP8266 phát ra Wifi với thông tin như sau:

Bảng 2.3. Bảng thông tin của ESP8266 ở chế độ AP

|  |  |  |
| --- | --- | --- |
| SSID | “DoAnTotNghiep” | |
| Mật khẩu | “1234567890” |
| IP mặc định | “192.168.4.1” |

Để thực hiện cài đặt người dùng sẽ kết nối máy tính tới wifi này, lúc này máy tính sẽ được DHCP server trên ESP8266 cấp phát địa chỉ IP. Địa chỉ IP được cấp phát này sẽ được máy tính sử dụng để giao tiếp với ESP8266 được mô tả ở Hình 2.25.

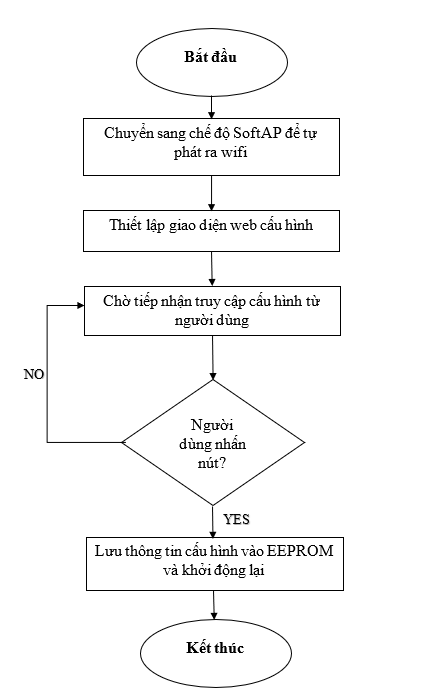


Hình 2.25. Kết nối Laptop với ESP8266

Trên ESP8266 ta sẽ thiết lập 1 máy chủ HTTP server để cung cấp giao diện hiển thị và cấu hình các thông tin cho ESP8266.

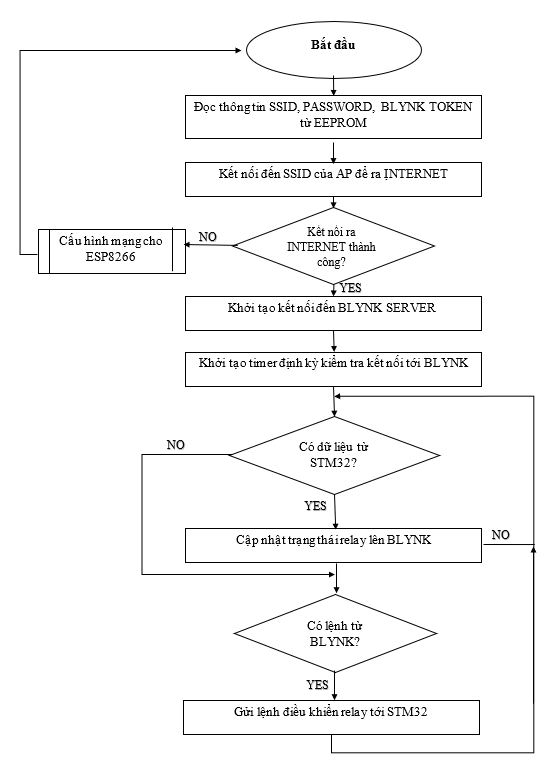
Lúc này trên máy tính ta mở trình duyệt, gõ địa chỉ Web của ESP8266 là <http://192.168.4.1> để cấu hình thông tin cần thiết sau đó lưu và khởi động lại.

Nếu các thông tin được cấu hình hợp lệ ESP8266 sẽ chuyển chế độ hoạt động về Wifi Station để kết nối tới Blynk server. Hình 2.26 dưới đây đã thể hiện được cách kết nối và hoạt động của ESP8266.



Hình 2.26. Lưu đồ giải thuật chương trình cấu hình mạng cho ESP8266

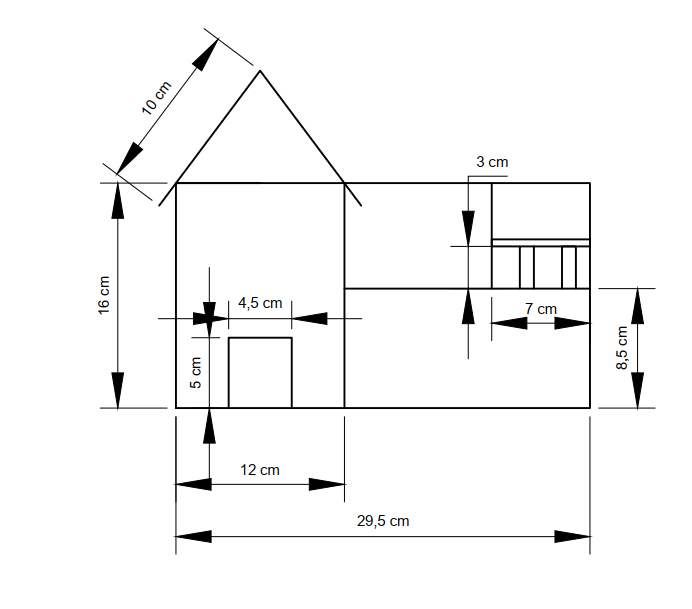
#### Giao tiếp Blynk Sever



Hình 2.27. Lưu đồ giải thuật chương trình giao tiếp Blynk server

Lưu đồ hình 2.27 đã thể hiện được cách hoạt động của Blynk Server. Trước tiên sẽ đọc thông tin SSID, PASSWORD, BLYNK TOKEN để kết nối đến SSID của AP để ra INTERNET.

### Thiết kế mô hình



Hình 2.28. Thiết kế mô hình

Mô hình thiết kế mặt trước sẽ có hình dáng của một ngôi nhà như Hình 2.28. Ngôi nhà có chiều sâu 21 cm, chiều ngang 29.5 cm. Ngoài ra mặt trước ngôi nhà thiết kế thêm cửa nhà cao 5 cm rộng 4.5 cm.

|  |  |
| --- | --- |
| Tổng hợp 87+ hình về giấy formex bìa mô hình - daotaonec  Hình 2.29. Bìa trắng formex | Keo 502 có dán Gỗ được không? Sử dụng keo 502 dán Gỗ thế nào?  Hình 2.30. Keo 502 |

Đề tài sử dụng bìa trắng formex như Hình 2.29 để làm nguyên liệu xây dựng mô hình và dùng keo 502 như Hình 2.30 để dính các tầm bìa trắng lại theo ý muốn.

## Kết luận chương 2

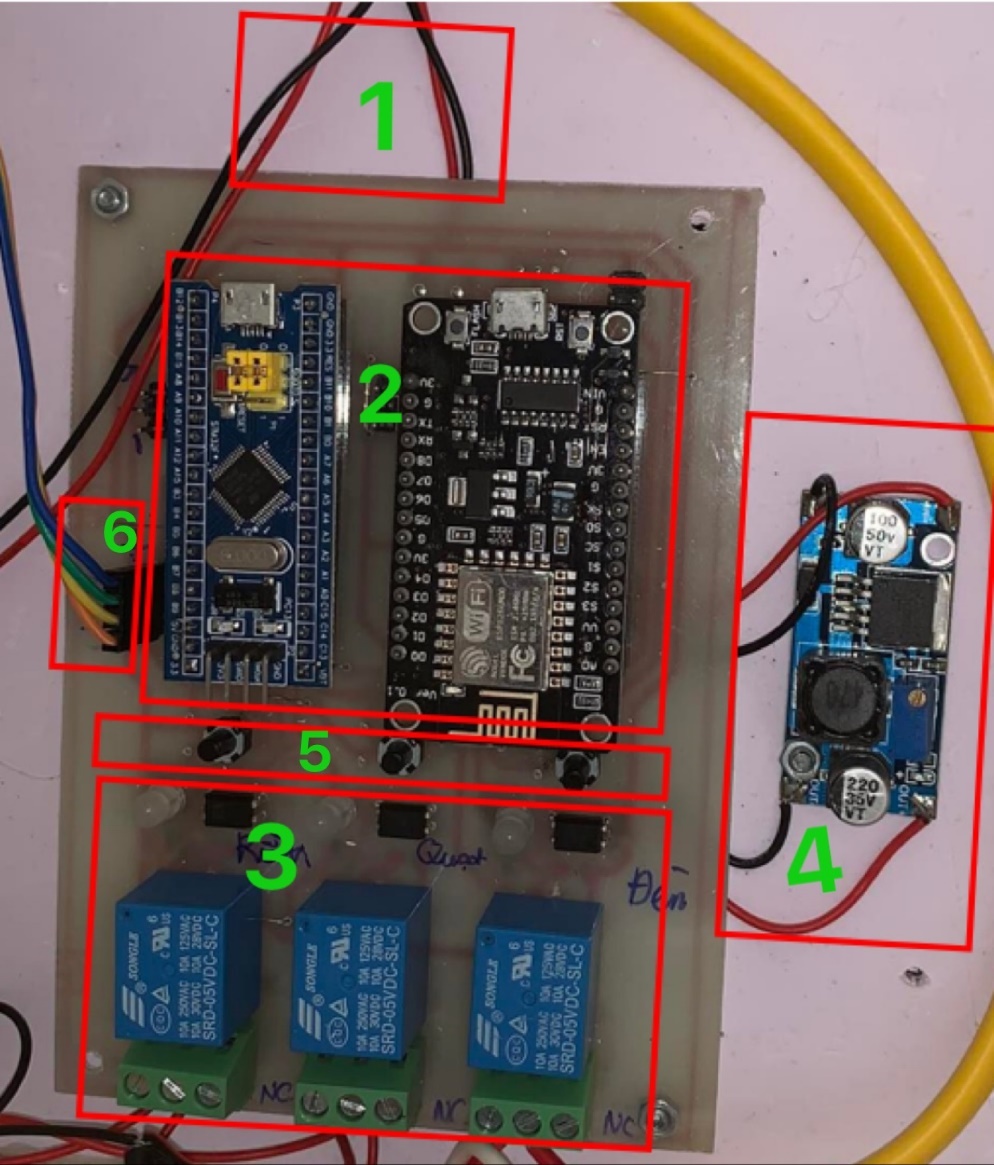
Chương 2 đã phân tích được yêu cầu bài toán và đưa ra các giải pháp thực hiện và chọn được giải pháp phù hợp. Chương 2 cũng trình bày thiết kế được phần cứng, phần mềm của hệ thống và thiết kế mô hình. Sau khi thiết kế được mô hình của đề tài thì nhóm sẽ tiến hành các thử nghiệm thực tế, trình bày bảng đánh giá hiệu năng khi tiến hành đo thực nghiệm ở Chương 3.

# THỬ NGHIỆM VÀ ĐÁNH GIÁ

## Mạch in thủ công và mô hình thực tế

* Mạch thực tế

Trong quá trình thực hiện đề tài nhóm đã thiết kế được mạch của hệ thống điều khiển thiết bị điện trong nhà được thể hiện ở Hình 3.1.



Hình 3.1. Mạch của hệ thống điều khiển thiết bị điện trong nhà

1 - Dây cấp nguồn 5V cho mạch

2 - Khối xử lý và giao tiếp mạng: gồm STM32F103C8T6 và module ESP8266

3 - Khối Relay: gồm 3 relay và 3 PC817

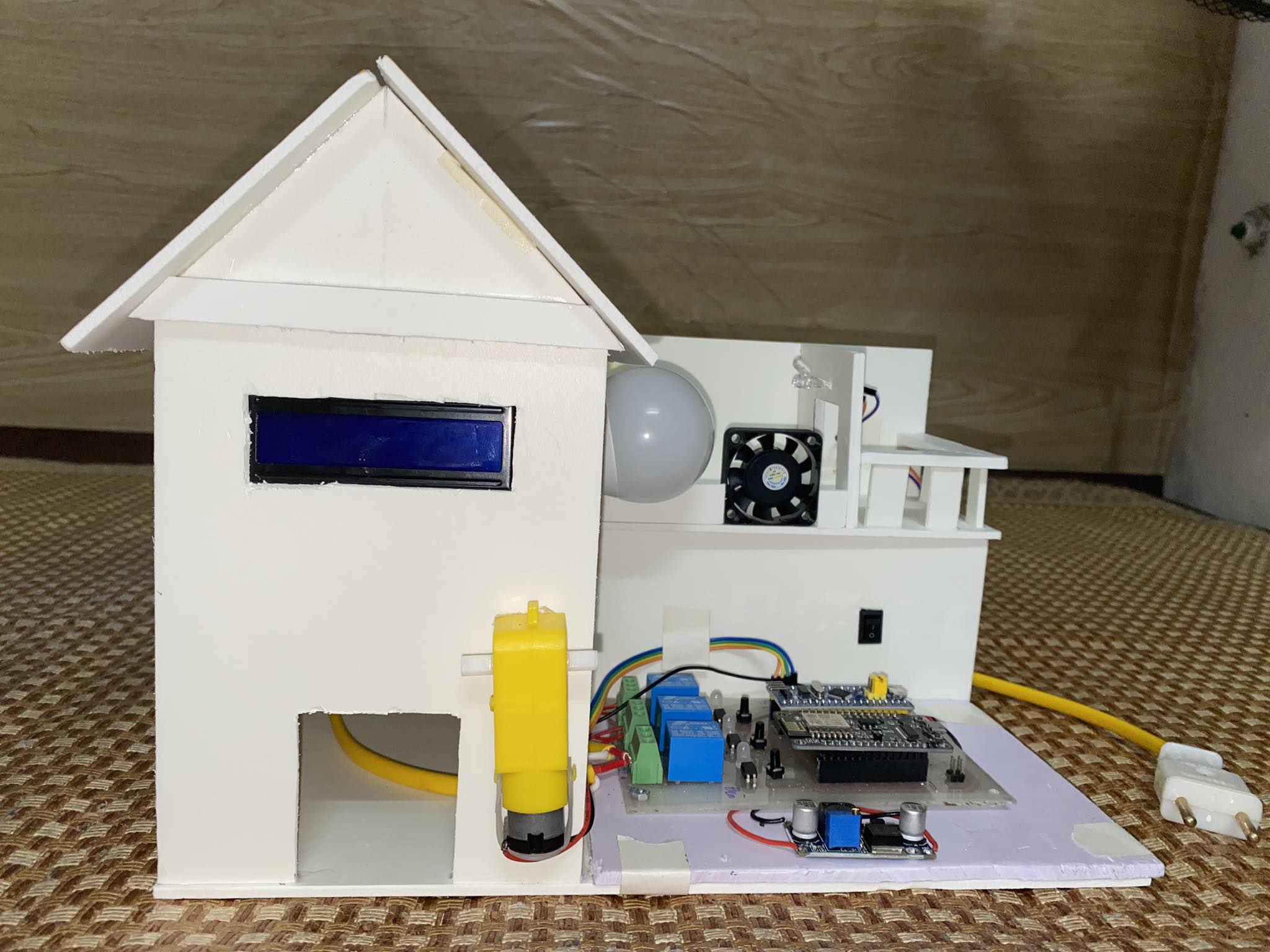
4 - Module LM2596 dùng để hạ áp từ 5V xuống 3.3V

5 - Ba nút nhấn để điều khiển bật tắt 3 thiết bị

6 - Dăm kết nối với Module PCF8574 I2C để kết nối với LCD

* Mô hình sản phẩm

Sau khi hoàn chỉnh mạch thực tế. nhóm đã thiết kế được mô hình hệ thống điều khiển thiết bị điện trong nhà như Hình 3.2 dưới đây.



Hình 3.2. Mô hình của đề tài

## Thử nghiệm và kiểm chứng sản phẩm

Sau quá trình hoàn thiện mô hình, nhóm đưa mô hình vào thực nghiệm và được kết quả như Bảng 3.1.

Bảng 3.1. Bảng kết quả thực nghiệm

|  |  |  |  |
| --- | --- | --- | --- |
| **Lần thực nghiệm** | **Vấn đề gặp phải** | **Cách khắc phục** | **Kết quả** |
| 1 | Relay để điều khiển thiết bị 3 là motor không hoạt động | Hàn lại đường dây dẫn nguồn cho Relay | Relay hoạt động, Motor hoạt động theo lệnh điều khiển |
| 2 | ESP8266 không thể kế nối đươc với Wifi có Internet được phát bởi Smartphone. | Nhấn nút Reset ESP8266 và kết nối lại với Wifi. | ESP8266 kết nối được với Wifi |
| 3 | Sử dụng bật điều khiển từ xa qua smartphone. Bật được đèn sau đó bật quạt và động cơ thì quạt và động cơ không hoạt động | Bật 3 thiết bị lần lượt cách nhau khoảng 5s đến 6s. | Mô hình đã hoạt động bật được 3 thiết bị lần lượt bằng nút nhấn từ xa |
| 4 | Khi bật thiết bị bằng nút nhấn trên mạch của hệ thống, thiết bị hoạt động nhưng chế độ hoạt động không được đồng bộ lên App Blynk IoT | Do mạng kém chưa cập nhật được trạng thái hoạt động của thiết bị. Thoát giao diện điều khiển ra màn hình chính sau đó quay lại. | Đồng bộ được khi điều khiển bằng nút nhấn trên mạch của hệ thống, trạng thái được cập nhật lên App Blynk IoT |
| 5 | LCD được cấp nguồn nhưng không hiển thị các thiết bị và trạng thái của thiết bị lên màn hình | Các chân kết nối SCL, SDA của PCF8574 I2C bị lỏng, cắm các kết nối ổn định | LCD hiển thị tên các thiết bị và trạng thái của các thiết bị. |

Sau khi tiến hành thực nghiệm mô hình, nhóm đã hoàn thiện sản phẩm hoạt động chính xác, đúng với yêu cầu đã đề ra điều khiển các thiết bị ổn định được bằng 4 cách: điều khiển bằng nút nhấn trên mạch, điều khiển bằng nút nhấn từ xa, điều khiển bằng cài đặt thời gian và điều khiển bằng giọng nói. Các trạng thái hoạt động được hiển thị trên LCD và Smartphone.

## Đánh giá sản phẩm

Mô hình hoạt động tốt đúng, đủ yêu cầu đã đề ra. Về kỹ thuật và ứng dụng, mô hình hoạt động chính xác. Còn một số điểm hạn chế: kết cấu mô hình và lắp đặt các thiết bị linh kiện chưa được chắc chắn, chưa tối ưu được tất cả các chức năng linh kiện…

Do hạn chế về mặt thời gian và kiến thức nên sản phẩm còn nhiều thiếu xót. Rất mong nhận được những góp ý của thầy cô giáo cùng các bạn sinh viên để sản phẩm ngày càng được hoàn thiện hơn.

## Ứng dụng của sản phẩm

Sản phẩm điều khiển thiết bị điện trong nhà từ xa: Với đồ án này, người dùng có thể điều khiển thiết bị điện trong nhà từ xa thông qua smartphone hoặc máy tính bảng. Điều này giúp tiết kiệm thời gian và tiện lợi cho người dùng.

Tạo lịch trình điều khiển: Người dùng có thể tạo lịch trình điều khiển cho các thiết bị điện trong nhà. Ví dụ như thiết lập hệ thống điều khiển đèn, quạt thông gió bật/tắt theo giờ.

## Kết luận chương 3

Trong chương 3, đã trình bày về mô hình thực nghiệm và những cách khắc phục với một số vấn đề gặp phải trong quá trình thực nghiệm, đánh giá tổng quan về mô hình sản phẩm và đưa ra ứng dụng của sản phẩm. Các yếu tố được đề cập ở trên phản ánh sự tiện ích và cần thiết của hệ thống điều khiển thiết bị điện đối với một ngôi nhà thông minh.

Về cơ bản, hệ thống được đưa vào chạy thực tế, hoạt động tương đối ổn định, đúng như mục tiêu đã đặt ra trước đó. Hệ thống được thiết kế đơn giản, dễ dàng sử dụng.

# KẾT LUẬN

**Kết luận**

Sau 9 tuần nỗ lực, nhóm đã hoàn thành đồ án với đề tài **“Nghiên cứu, thiết kế mô hình điều khiển thiết bị điện trong nhà sử dụng STM32”**, đúng với thời gian và yêu cầu đã được quy định. Thông qua đề tài lần này nhóm đã học được rất nhiều kiến thức bổ ích, cũng rút ra được rất nhiều kinh nghiệm có giá trị qua việc tìm hiểu một số tài liệu sẵn có, tài liệu trên mạng internet và sự hướng dẫn chỉ bảo của giáo viên hướng dẫn nên nhóm đã thu được một số kết quả nhất định: Củng cố được các 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 Designer, Arduino, STM32Cube, …Nghiên cứu và tìm hiểu tổng quan về vi điều khiển STM32F106C8T6, ngoại vi và lập trình STM. Hiểu được quy trình và nguyên lý hoạt động của hệ thống điều khiển thiết bị điện trong nhà.

Trong quá trình thực hiện đề tài, từ những mục tiêu ban đầu cũng như từng bước hoàn thiện mô hình hệ thống. Nhóm có một số nhận xét về ưu điểm sau đây: Về phần cứng nhóm đã thiết kế và thi công hoàn chỉnh mạch, hệ thống được đưa vào chạy thực tế, hoạt động tương đối ổn định, thiết kế đơn giản, dễ dàng sử dụng. Về phần mềm: Thiết kế, xây dựng được lưu đồ thuật toán điều khiển, viết chương trình hoàn chỉnh cho mạch hoạt động theo yêu cầu đặt ra.

Tuy nhiên, bên cạnh những ưu điểm thì vẫn còn tồn tại một số nhược điểm cần khắc phục như: Tính thẩm mĩ chưa cao, ít chức năng sử dụng, chưa tối ưu được nguồn điện cho hệ thống, chưa đưa được nhiều các thiết bị điện vào đề tài.

**Hướng phát triển**

Phát triển tính năng AI: Các hệ thống điều khiển thiết bị điện thông minh có thể được cải tiến bằng việc tích hợp tính năng trí tuệ nhân tạo (AI). Tính năng này sẽ giúp hệ thống tự động học hỏi và cải thiện khả năng dự đoán, giúp người dùng tiết kiệm thời gian và nâng cao hiệu quả sử dụng.

Mở rộng ứng dụng cho các thiết bị khác: Ngoài điều khiển các thiết bị điện trong nhà, đề tài này có thể được mở rộng để điều khiển các thiết bị khác như thiết bị âm thanh, thiết bị giải trí, thiết bị y tế, và các thiết bị khác trong nhà.

Tối ưu hóa hiệu suất tiêu thụ năng lượng: Để giúp người dùng tiết kiệm chi phí và bảo vệ môi trường, hệ thống điều khiển thiết bị điện thông minh có thể được cải tiến bằng cách tối ưu hóa hiệu suất tiêu thụ năng lượng. Các tính năng tiết kiệm năng lượng có thể được tích hợp để giảm thiểu lượng điện tiêu thụ, đồng thời cải thiện hiệu quả sử dụng thiết bị.

Tích hợp hệ thống an ninh: Để nâng cao tính an toàn cho người dùng, hệ thống điều khiển thiết bị điện thông minh có thể được tích hợp với hệ thống an ninh trong nhà. Điều này sẽ giúp người dùng dễ dàng quản lý mọi hoạt động trong nhà của mình.

Cải tiến giao diện người dùng: Để giúp người dùng sử dụng hệ thống điều khiển thiết bị điện thông minh dễ dàng hơn, giao diện người dùng có thể được cải tiến để tăng tính thẩm mỹ và dễ sử dụng hơn.

# TÀI LIỆU THAM KHẢ0

|  |  |
| --- | --- |
| [1] | Nguyễn Thanh Hải, Phạm Văn Hà (2019), *Giáo trình cơ sở lập trình nhúng*, Trường Đại học Công nghiệp Hà Nội. |
| [2] | Warren Gay (2018), *Beginning STM32: Developing with FreeRTOS, libopencm3 and GCC 1st ed*, Edition NXB Apress. |
| [3] | DienMayXanh (2020), *Wifi là gì? Có những chuẩn Wifi nào?,* từ https://www.dienmayxanh.com/kinh-nghiem-hay/wifi-la-gi-596329#., truy cập ngày 08/05/2023 |
| [4] | RangDong (2019), *Hệ thống thiết bị thông minh cơ bản trong mô hình nhà thông minh hiện nay*, Công Ty Cổ Phần Bóng Đèn Phích Nước Rạng Đông, từ https://rangdong.com.vn/6-he-thong-thiet-bi-thong-minh-co-ban-trong-mo-hinh-nha-thong-minh-hien-nay-n865.html, truy cập ngày 08/05/2023 |
| [5] | Đức Huy (2018), *Vi điều khiển AT89S52*, DHE electronic automation, từ https://dientuduchuy.com/products/at89s52-24pu-dip40, truy cập ngày 11/04/2023 |
| [6] | Huỳnh Nhật Tùng (2018), *Vi điều khiển PIC18F4520*, Nhật Tùng linhkien & Mobile, từ https://dientunhattung.com/product/pic18f4520-vi-dieu-khien/, truy cập ngày 11/04/2023 |
| [7] | DienMayXanh (2020), *Kết nối Bluetooth?*, sieuthi DienMayXanh , từ https://www.dienmayxanh.com/kinh-nghiem-hay/ket-noi-bluetooth-la-gi-595540#, truy cập ngày 08/05/2023 |
| [8] | Alldatasheet.com (2020), *LM2596 product details datasheet*, từ https://www.alldatasheet.com/datasheet-pdf/pdf/1614705/TECHPUBLIC/LM2596.html, truy cập ngày 11/04/2023 |
| [9] | TGDienTu (2017), *Nút nhấn 4 chân 6x6x8mm*, IC đây rồi Thế giới Điện Tử, từ https://icdayroi.com/nut-nhan-4-chan-6x6x15mm, truy cập ngày 13/05/2023 |
| [10] | Khoa học (2018), *Điều khiển từ xa hoạt động như thế nào*, Báo Đất Việt, từ https://khoahoc.tv/dieu-khien-tu-xa-hoat-dong-nhu-the-nao-28786, truy cập ngày 09/05/2023 |
| [11] | Tiến Dũng (2021) , *Giới thiệu về ESP8266, ESP8266 là một chip tích hợp công nghệ Wifi*, từ https://dungat.com/blog/posts/gioi-thieu-ve-esp8266, truy cập ngày 13/05/2023 |
| [12] | Advance Cad (2020), *Khái niệm cơ bản về truyền thông UART*, Trung tâm công nghệ Advance Cad, từ https://advancecad.edu.vn/khai-niem-co-ban-ve-truyen-thong-uart-so-do-khoi-ung-dung/, truy cập ngày 09/05/2023 |
| [13] | 3CE (2022) , *Role là gì, chức năng và cấu tạo*, Công ty TNHH Điện - Điện tử 3C, từ https://3ce.vn/ro-le-la-gi-chuc-nang-va-cau-tao/#, truy cập ngày 13/05/2023 |
| [14] | ĐTTL (2020), *Tìm hiểu opto PC817, thông số kỹ thuật*, Điện tử tương lai, từ https://dientutuonglai.com/tim-hieu-opto-pc817.html, truy cập ngày 13/05/2023 |
| [15] | Huỳnh Nhật Tùng (2018) , *I2C LCD GIAO TIẾP ARDUINO, I2C LCD PCF8574 và LCD1602 và ARDUINO*, Nhật Tùng linhkien va Mobile, https://huynhnhattung.com/i2c-lcd-giao-tiep-arduino-i2c-lcd-pcf8574-lcd1602-arduino/, truy cập ngày 13/05/2023 |

# PHỤ LỤC

**Bảng giá linh kiện**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên linh kiện** | **Số lượng** | **Giá** |
| 1 | Kit STM32F103C8T6 | 1 | 70.000 VNĐ / chiếc |
| 2 | Module ESP8266 | 1 | 65.000 VNĐ / chiếc |
| 3 | Module LM2596 | 1 | 18.000 VNĐ / chiếc |
| 4 | Module PCF8574 I2C | 1 | 20.000 VNĐ / chiếc |
| 5 | Màn hình LCD 1602 | 1 | 39.000 VNĐ / chiếc |
| 6 | Nguồn Adapter 5V | 1 | 28.500 VNĐ / chiếc |
| 7 | Nút nhấn 5V 4 chân | 3 | 500 VNĐ / chiếc |
| 8 | Relay 5V | 3 | 7.500 VNĐ / chiếc |
| 9 | PC817 | 3 | 2.000 VNĐ / chiếc |
| 10 | Đèn 220V | 1 | 20.000 VNĐ / chiếc |
| 11 | Quạt 5V | 1 | 9.500 VNĐ / chiếc |
| 12 | Động cơ vàng giảm tốc | 1 | 14.000 VNĐ / chiếc |

Bảng giá của các linh kiện trên là các linh kiện đã được sử dụng trong đề tài, nhóm đã tham khảo tại http://linhkienviet.vn/

**Hướng dẫn sử dụng sản phẩm**

Điều kiện để sử dụng sản phẩm: smartphone hệ điều hành Android để điều khiển từ xa, Wifi có tốc độ Internet ổn định.

Mô hình sử dụng nguồn điện 220VAC. Để sử dụng được mô hình này thì ta sẽ làm trình tự theo các bước sau đây:

Đầu tiên hãy cấp nguồn điện 220VAC cho mô hình và bấm công tắc để bật nguồn. Sau khi cấp nguồn ta nhìn thấy đèn báo màu đỏ ở vi điều khiển, đèn báo xanh ở LM2596 và LCD hiển thị trạng thái các thiết bị như Hình 1.

(Chú ý: nếu đèn đỏ không sáng, đèn xanh không sáng hoặc 2 đèn không sáng thì xem lại nguồn cấp cho mô hình)

|  |  |
| --- | --- |
| Không có mô tả. | Không có mô tả. |

Hình 1. Mô hình khi bật đèn

Tiếp theo mô hình sẽ có 4 cách để điều khiển bật tắt của 3 thiết bị

* *Cách 1: Sử dụng nút nhấn trên mạch trong mô hình*

**Bước 1:** Để bật đèn thì nhấn vào nút đèn ở mô hình,

**Bước 2:** Để tắt đèn thì nhấn tiếp tục vào nút đèn.

(Muốn bật tắt quạt thông gió và động cơ cũng tương tự như bật tắt đèn. Ngoài ra, động cơ sẽ tự dừng hoạt động sau thời gian đã cài đặt.)

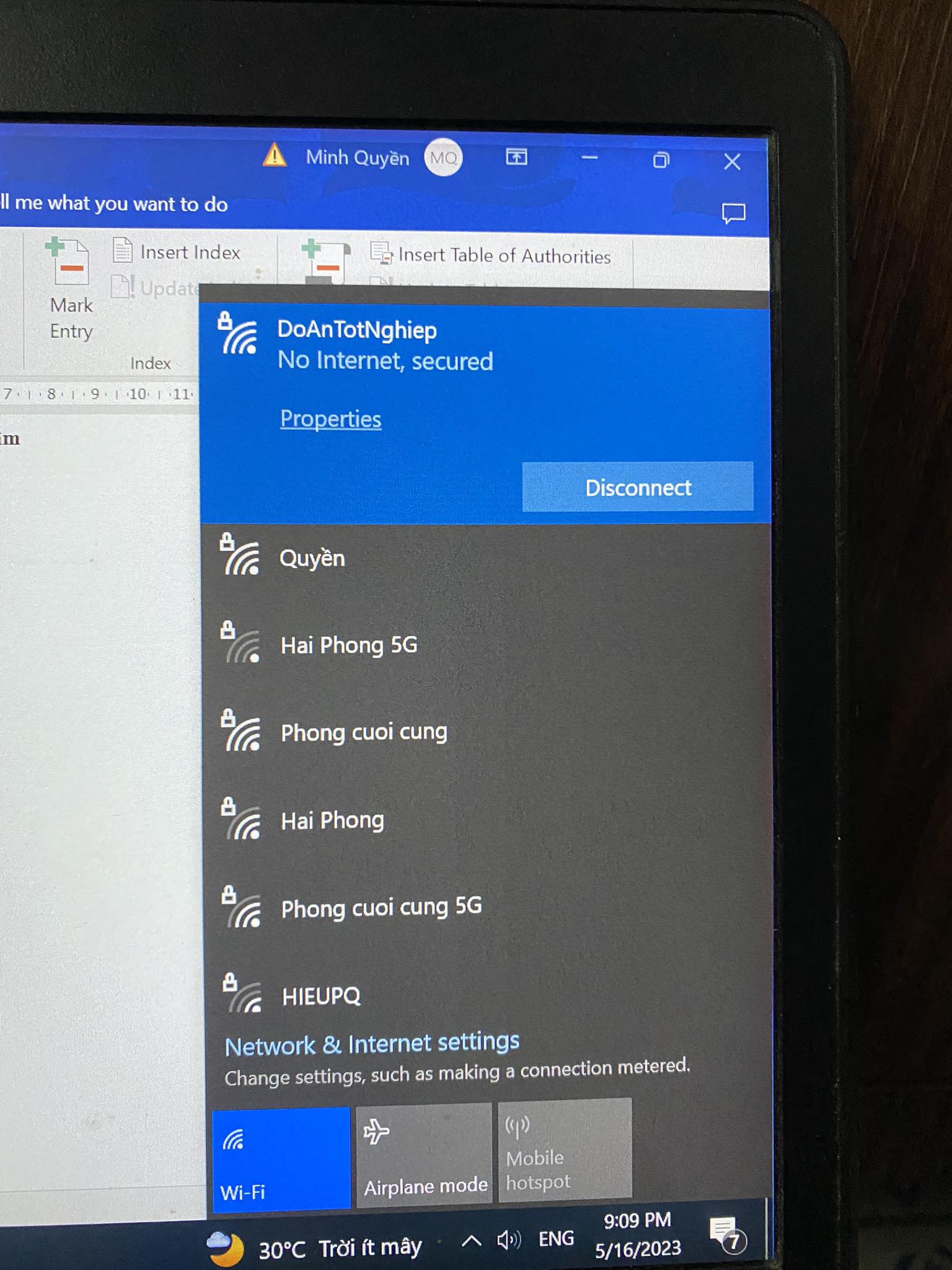
Khi đèn được bật thì đèn sẽ sáng, đồng thời LCD sẽ hiện TB1: ON

* *Cách 2: Sử dụng App Blynk IoT để điều khiển bật tắt bằng nút nhấn*

Để sử dụng được chức năng này thì điện thoại phải được cài đặt App Blynk IoT

**Bước 1:** Kết nối với Wifi mà ESP8266 phát ra

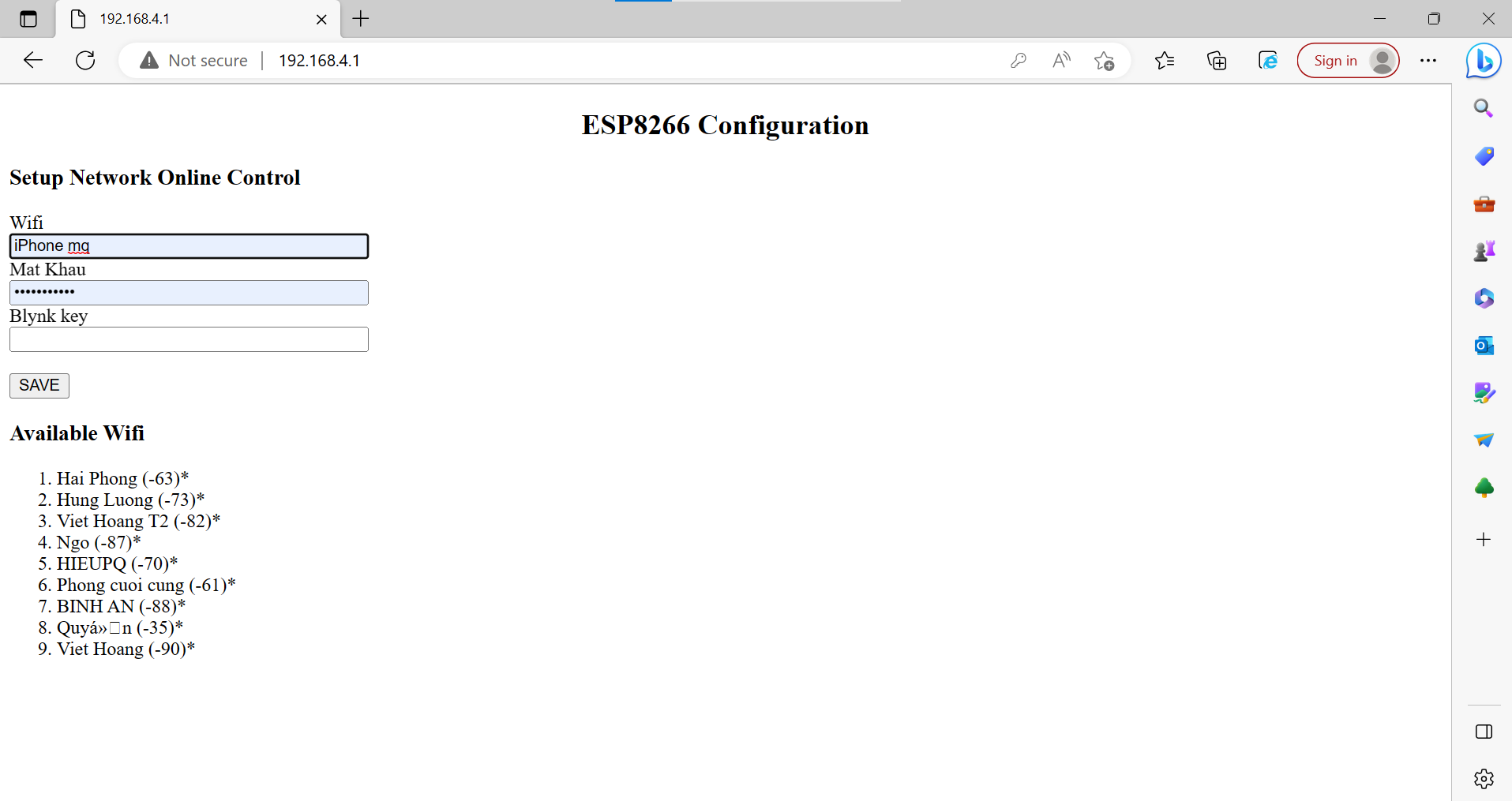
Sau khi cấp nguồn, sử dụng Smartphone hoặc Laptop kết nối đến Wifi DoAnTotNghiep, mật khẩu: 1234567890. Sau khi kết nối thành công sẽ hiển thị đã kết nối như Hình 2.



Hình 2. Kết nối Wifi phát ra từ ESP8266

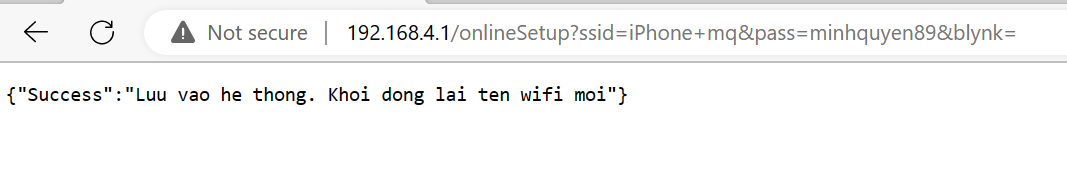
**Bước 2:** Kết nối ESP8266 với Wifi có Internet

- Vào trình duyệt Web và search địa chỉ Web của ESP8266 là: “192.168.4.1” và nhập Wifi có kết nối mạng như Hình 3.



Hình 3. Kết nối mô hình với Wifi

- Khi kết nối thành công thì trang Web sẽ hiển thị như Hình 4.



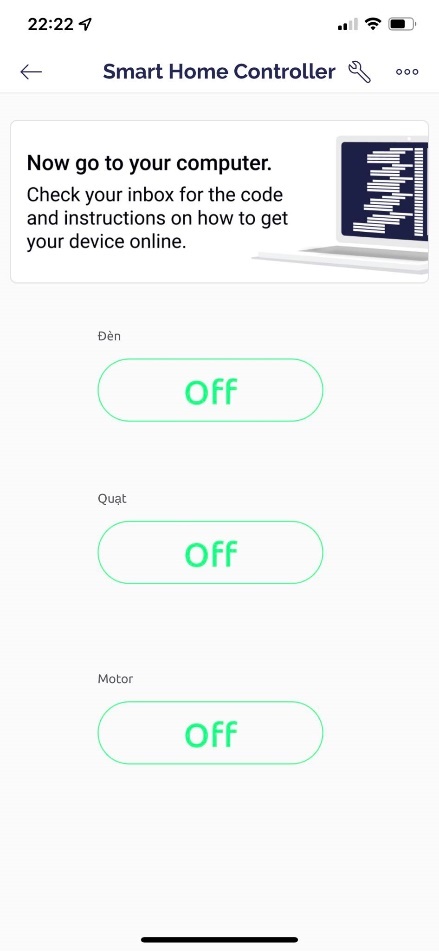
Hình 4. Lưu Wifi vào hệ thống

- Tiếp theo nhấn nút RST trên ESP8266 để reset, khi nhấn nút RST thì ESP8266 sẽ có led nháy đèn màu đỏ.

- Nếu Wifi DoAnTotNghiep không còn hiển thị thì mô hình đã sẵn sàng hoạt động điều khiển qua App Blynk IoT.

**Bước 3:** Đăng nhập vào App Blynk IoT

**Bước 4:** Tại giao diện App, chọn Smart Home Controller để điều khiển bằng nút nhấn. Có 3 nút nhấn để bật tắt thiết bị như Hình 5. Có thể điều khiển thiết bị từ xa thông qua thông qua Wifi.



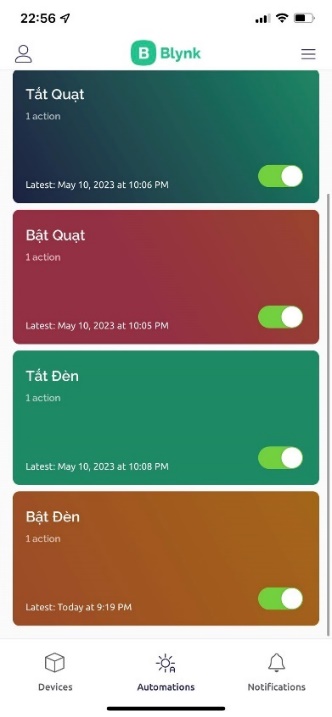
Hình 5. Giao diện App Blynk IoT

**Bước 5:** Điều khiển thiết bị bật tắt theo ý muốn. Trạng thái của thiết bị cũng sẽ được cập nhật trên LCD.

* *Cách 3: Điều khiển bật tắt bằng cài đặt thời gian.*

Làm theo cách 2 các **Bước 1, 2, 3**

**Bước 4:** Tại giao diện App, chọn Automations sẽ có giao diện như Hình 6.



Hình 6. Giao diện cài thời gian

**Bước 5:** Chọn thiết bị và cài đặt trạng thái theo ý muốn.

* *Cách 3: Điều khiển thiết bị bằng giọng nói*

Làm theo cách 2 các **Bước 1, 2**

**Bước 3:** Cài đặt Ứng dụng Google Home trên CH Play và đăng nhập.

**Bước 4:** Vào Google Home, sau đó vào Cài đặt và bật Trợ lý Google

**Bước 5:** Điều khiển bật tắt theo thiết bị mong muốn như Hình 7.

Các lệnh được sử dụng trong mô hình: Activate turn on light: bật đèn; activate turn off light: tắt đèn; Activate turn on fan: bật quạt; activate turn off fan: tắt quạt; Activate turn on curtain: bật động cơ; activate turn off curtain: tắt động cơ.

|  |  |
| --- | --- |
| Không có mô tả. | Không có mô tả. |

Hình 7. Bật/tắt đèn bằng giọng nói

**Thông tin liên hệ:**

Nếu có bất cứ thắc mắc hoặc câu hỏi gì về sản phẩm vui lòng liên hệ:

* Người thiết kế 1: Nguyễn Hữu Dũng
* Số điện thoại: 0975455147
* Email: dung2312001@gmail.com
* Người thiết kế 2: Nguyễn Minh Quyền
* Số điện thoại: 0868448960
* Email: minhquyen8901@gmail.com
* Người thiết kế 3: Phùng Văn Tưởng
* Số điện thoại: 0961968446
* Email: phungvantuong283@gmail.com

**Code của chương trình**

|  |  |
| --- | --- |
| #include "main.h"  /\* Private includes --------------------------------\*/  /\* USER CODE BEGIN Includes \*/  /\* USER CODE END Includes \*/  /\* Private typedef ----------------------------\*/  /\* USER CODE BEGIN PTD \*/  // nghĩa mã bản tin  typedef enum  {  DATA, CMD  } MSG\_TYPE;  typedef enum  {  OFF = 0, ON  } State\_t;  typedef enum  {  FALSE, TRUE  } bool\_t;  //kiểu dữ liệu đại diện cho 1 thiết bị  typedef struct Device\_ts  {  uint8\_t id;  State\_t enState; // trạng thái on-off  uint16\_t relayGpio; //chân GPIO điều khiển relay  BUTTON\_Name \*ctrlButton; //nút nhấn điều khiển thiết bị  } Device\_ts;  /\* USER CODE END PTD \*/  /\* Private define ------------------------------------------\*/  /\* USER CODE BEGIN PD \*/  #define UART\_MSG\_MAX\_SIZE 64  #define NUM\_OF\_DEVICE 3  /\* USER CODE END PD \*/  /\* Private macro -----------------------------------\*/  /\* USER CODE BEGIN PM \*/  /\* Private function prototypes ---------------------------\*/  void lcdDisplayDeviceState();  uint8\_t turnOnDevice(Device\_ts \*dev);  uint8\_t turnOffDevice(Device\_ts \*dev);  /\* USER CODE END PM \*/  /\* Private variables --------------------------------------\*/  //Khởi tạo bộ xử lý I2C  I2C\_HandleTypeDef hi2c1;  //Khởi tạo bộ xử lý timer  TIM\_HandleTypeDef htim4;  //Khởi tạo bộ xử lý UART  UART\_HandleTypeDef huart1;  UART\_HandleTypeDef huart3;  /\* USER CODE BEGIN PV \*/  //Khởi tạo bộ xử lý I2C - LCD  CLCD\_I2C\_Name LCD1;  //Khởi tạo 3 nút nhấn điều khiển thiết bị  BUTTON\_Name Button1, Button2, Button3;  uint8\_t Rx\_data[2], Transfer\_cplt;  char Rx\_Buffer[100];  bool\_t enUpdateDisplayFlag = FALSE;  //Khai báo danh sách các thiết bị  Device\_ts stDevices[NUM\_OF\_DEVICE] =  {  { 1, OFF, RELAY1\_Pin, &Button1 },  { 2, OFF, RELAY2\_Pin, &Button2 },  { 3, OFF, RELAY3\_Pin, &Button3 }, };  /\*\*  \* @brief Retargets the C library printf function to the USART.  \* @param None  \* @retval None  \*/  #if defined(\_\_GNUC\_\_)  int \_write(int fd, char \*ptr, int len)  {  HAL\_UART\_Transmit(&huart1, (uint8\_t\*) ptr, len, HAL\_MAX\_DELAY);  return len;  }  #elif defined(\_\_ICCARM\_\_)#include "LowLevelIOInterface.h"  size\_t \_\_write(int handle,  const unsigned char \* buffer, size\_t size) {  HAL\_UART\_Transmit( & huart1, (uint8\_t \* ) buffer, size, HAL\_MAX\_DELAY);  return size;  }  #elif defined(\_\_CC\_ARM)  int fputc(int ch, FILE \* f) {  HAL\_UART\_Transmit( & huart1, (uint8\_t \* ) & ch, 1, HAL\_MAX\_DELAY);  return ch;  }  #endif  //Khởi tạo tài nguyên hệ thống  void user\_setup(void)  {  printf("start program \r\n");  //khoi tao LCD  CLCD\_I2C\_Init(&LCD1, &hi2c1, 0x4e, 20, 4);  printf("Init LCD Done \r\n");  //khoi tao nut nhan  BUTTON\_Init(&Button1, GPIOB, BUT1\_Pin);  BUTTON\_Init(&Button2, GPIOB, BUT2\_Pin);  BUTTON\_Init(&Button3, GPIOB, BUT3\_Pin);  printf("Init Button Done \r\n");  //Hien thi trang thai thiet bi  lcdDisplayDeviceState();  HAL\_UART\_Receive\_IT(&huart3, Rx\_data, 1); //Kich hoat ngat uart khi nhan duoc 1 byte  }  //Ham ngat uart  void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart)  {  static uint8\_t Rx\_indx = 0;  if (huart->Instance == USART3) //UART hien tai la uart3  {  if (Rx\_indx == 0)  {  memset(Rx\_Buffer, 0, sizeof(Rx\_Buffer));  } //xoa Rx\_Buffer truoc khi nhan du lieu  if (Rx\_data[0] != 13)//Neu du lieu nhan duoc khac voi 13 (13 la ma cua phim enter)  {  Rx\_Buffer[Rx\_indx++] = Rx\_data[0]; //them du lieu vao Rx\_Buffer  }  else //Neu phim enter duoc an  {  Rx\_indx = 0; //Dua index ve 0  Transfer\_cplt = 1;//Qua trinh truyen hoan tat, du lieu da san sang de doc  }  HAL\_UART\_Receive\_IT(&huart3, Rx\_data, 1);//Kich hoat UART nhan du lieu ngat moi luc  }  }  //Loại bỏ ký tự xuống dòng và khoảng trấng ở đầu chuỗi  char\* trimLstr(char \*str)  {  while (\*str == '\n' || \*str == ' ')  {  str++;  }  return str;  }  //Nhận và xử lý lệnh điều khiển từ esp8266  bool\_t user\_uartRecvDeviceState(char \*rxData)  {  int idx;  MSG\_TYPE msgType = DATA;  int devId, devState;  rxData = trimLstr(rxData);  // printf("rxData=[%s]\r\n", rxData);  if (3 != sscanf(rxData, "|S[%d]:[%d]-[%d]E|", (int\*) &msgType, &devId, &devState))  {  printf("Parse rx failed\r\n");  return FALSE;  }  for (idx = 0; idx < NUM\_OF\_DEVICE; idx++)  {  //xác định thiết bị cần điều khiển  if (devId == stDevices[idx].id)  {  //thực hiện on-off theo lệnh  if (ON == devState)  {  turnOnDevice(&stDevices[idx]);  }  else  {  turnOffDevice(&stDevices[idx]);  }  break;  }  }  return TRUE;  }  //Gửi trạng thái thiết bị đến esp8266  bool\_t user\_uartSendDeviceState(Device\_ts \*dev)  {  char strData[UART\_MSG\_MAX\_SIZE] =  { 0 };  MSG\_TYPE msgType = CMD;  snprintf(strData, sizeof(strData), "|S[%d]:[%d]-[%d]E|", msgType, dev->id,  dev->enState);  //gửi dữ liệu qua UART3  if (HAL\_OK  != HAL\_UART\_Transmit(&huart3, (uint8\_t\*) strData, strlen(strData),  HAL\_MAX\_DELAY))  {  return TRUE;  }  return FALSE;  }  //thực hiện bật thiết bị  uint8\_t turnOnDevice(Device\_ts \*dev)  {  HAL\_GPIO\_WritePin(GPIOB, dev->relayGpio, GPIO\_PIN\_RESET);  dev->enState = ON;  user\_uartSendDeviceState(dev);  enUpdateDisplayFlag = TRUE;  if (3 == dev->id) //rem cua  {  HAL\_TIM\_Base\_Start\_IT(&htim4);  }  printf("Turn On device %d\r\n", dev->id);  return TRUE;  }  //thực hiện tắt thiết bị  uint8\_t turnOffDevice(Device\_ts \*dev)  {  HAL\_GPIO\_WritePin(GPIOB, dev->relayGpio, GPIO\_PIN\_SET);  dev->enState = OFF;  user\_uartSendDeviceState(dev);  enUpdateDisplayFlag = TRUE;  if (3 == dev->id) //rem cua  {  HAL\_TIM\_Base\_Stop\_IT(&htim4);  }  printf("Turn Off device %d\r\n", dev->id);  return TRUE;  }  // Đảo trạng thái thiết bị  uint8\_t toggleDevice(Device\_ts \*dev)  {  if (OFF == dev->enState)  {  turnOnDevice(dev);  }  else  {  turnOffDevice(dev);  }  return TRUE;  }  //Hiển thị trạng thái thiết bị lên LCD  void lcdDisplayDeviceState()  {  char buffer[16] =  { 0 };  char buffer2[16] =  { 0 };  //xóa màn hình  CLCD\_I2C\_Clear(&LCD1);  //hiển thị lại trạng thái mới  snprintf(buffer, sizeof(buffer), "TB1:%s TB2:%s",  (ON == stDevices[0].enState) ? "ON" : "OFF",  (ON == stDevices[1].enState) ? "ON" : "OFF");  CLCD\_I2C\_SetCursor(&LCD1, 0, 0);  CLCD\_I2C\_WriteString(&LCD1, buffer);  // memset(buffer, 0, sizeof(buffer));  snprintf(buffer2, sizeof(buffer2), "TB3:%s",  (ON == stDevices[2].enState) ? "ON" : "OFF");  CLCD\_I2C\_SetCursor(&LCD1, 0, 1);  CLCD\_I2C\_WriteString(&LCD1, buffer2);  }  // dung dong co rem sau 1 khoang thoi gian  void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)  {  static uint8\_t periodCnt = 0;  if (htim->Instance == TIM4)  {  periodCnt++;  if (120 == periodCnt)  {  if (stDevices[2].enState == ON)  {  turnOffDevice(&stDevices[2]);  }  HAL\_TIM\_Base\_Stop\_IT(&htim4);  periodCnt = 0;  }  }  }  //CHương trình vòng lặp xử lý chính  void user\_main\_loop(void)  {  int idx;  uint16\_t butStat;  char buffer[105] =  { 0 };  //scan nut nhan  for (idx = 0; idx < NUM\_OF\_DEVICE; idx++)  {  butStat = BUTTON\_Read(stDevices[idx].ctrlButton);  if (SINGLE\_CLICK == butStat)  {  //Thực hiện lệnh khi có nhấn nút  toggleDevice(&stDevices[idx]);  butStat = NO\_CLICK;  }  }  //Hien thi trang thai thiet bi  if (enUpdateDisplayFlag)  {  lcdDisplayDeviceState();  enUpdateDisplayFlag = FALSE;  }  if (Transfer\_cplt)  {  strcpy(buffer, Rx\_Buffer);  //nhận lệnh điều khiển từ ESP8266  user\_uartRecvDeviceState(buffer);  Transfer\_cplt = 0; //Reset lai bien tranfer\_complete  HAL\_Delay(500);  }  }  /\* USER CODE END PV \*/  /\* Private function prototypes -----------------------------------------------\*/  void SystemClock\_Config(void);  static void MX\_GPIO\_Init(void);  static void MX\_TIM4\_Init(void);  static void MX\_USART1\_UART\_Init(void);  static void MX\_USART3\_UART\_Init(void);  static void MX\_I2C1\_Init(void);  /\* USER CODE BEGIN PFP \*/  /\* USE CODE END PFP \*/  /\* Private user code ---------------------------------------------------------\*/  /\* USER CODE BEGIN 0 \*/  /\* USER CODE END 0 \*/  /\*\*  \* @brief The application entry point.  \* @retval int  \*/  int main(void)  {  /\* USER CODE BEGIN 1 \*/  /\* USER CODE END 1 \*/  /\* MCU Configuration---------------------------\*/  /\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/  HAL\_Init();  /\* USER CODE BEGIN Init \*/  /\* USER CODE END Init \*/  /\* Configure the system clock \*/  SystemClock\_Config();  /\* USER CODE BEGIN SysInit \*/  /\* USER CODE END SysInit \*/  /\* Initialize all configured peripherals \*/  MX\_GPIO\_Init();  MX\_TIM4\_Init();  MX\_USART1\_UART\_Init();  MX\_USART3\_UART\_Init();  MX\_I2C1\_Init();  /\* USER CODE BEGIN 2 \*/  user\_setup();  /\* USER CODE END 2 \*/  /\* Infinite loop \*/  /\* USER CODE BEGIN WHILE \*/  while (1)  {  //vòng lặp chương trình chính  user\_main\_loop();  /\* USER CODE END WHILE \*/  /\* USER CODE BEGIN 3 \*/  }  /\* USER CODE END 3 \*/  }  /\*\*  \* @brief System Clock Configuration  \* @retval None  \*/  void SystemClock\_Config(void)  {  RCC\_OscInitTypeDef RCC\_OscInitStruct =  { 0 };  RCC\_ClkInitTypeDef RCC\_ClkInitStruct =  { 0 };  /\*\* Initializes the RCC Oscillators according to the specified parameters  \* in the RCC\_OscInitTypeDef structure.  \*/  RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;  RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;  RCC\_OscInitStruct.HSEPredivValue = RCC\_HSE\_PREDIV\_DIV1;  RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;  RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;  RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;  RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL9;  if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)  {  Error\_Handler();  }  /\*\* Initializes the CPU, AHB and APB buses clocks  \*/  RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK | RCC\_CLOCKTYPE\_SYSCLK  | RCC\_CLOCKTYPE\_PCLK1 | RCC\_CLOCKTYPE\_PCLK2;  RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;  RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;  RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;  RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;  if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != HAL\_OK)  {  Error\_Handler();  }  }  /\*\*  \* @brief I2C1 Initialization Function  \* @param None  \* @retval None  \*/  static void MX\_I2C1\_Init(void)  {  /\* USER CODE BEGIN I2C1\_Init 0 \*/  /\* USER CODE END I2C1\_Init 0 \*/  /\* USER CODE BEGIN I2C1\_Init 1 \*/  /\* USER CODE END I2C1\_Init 1 \*/  hi2c1.Instance = I2C1;  hi2c1.Init.ClockSpeed = 100000;  hi2c1.Init.DutyCycle = I2C\_DUTYCYCLE\_2;  hi2c1.Init.OwnAddress1 = 0;  hi2c1.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;  hi2c1.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;  hi2c1.Init.OwnAddress2 = 0;  hi2c1.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;  hi2c1.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;  if (HAL\_I2C\_Init(&hi2c1) != HAL\_OK)  {  Error\_Handler();  }  /\* USER CODE BEGIN I2C1\_Init 2 \*/  /\* USER CODE END I2C1\_Init 2 \*/  }  /\*\*  \* @brief TIM4 Initialization Function  \* @param None  \* @retval None  \*/  static void MX\_TIM4\_Init(void)  {  /\* USER CODE BEGIN TIM4\_Init 0 \*/  /\* USER CODE END TIM4\_Init 0 \*/  TIM\_ClockConfigTypeDef sClockSourceConfig =  { 0 };  TIM\_MasterConfigTypeDef sMasterConfig =  { 0 };  /\* USER CODE BEGIN TIM4\_Init 1 \*  /\* USER CODE END TIM4\_Init 1 \*/  htim4.Instance = TIM4;  htim4.Init.Prescaler = 72;  htim4.Init.CounterMode = TIM\_COUNTERMODE\_UP;  htim4.Init.Period = 0xffff - 1;  htim4.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;  htim4.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;  if (HAL\_TIM\_Base\_Init(&htim4) != HAL\_OK)  {  Error\_Handler();  }  sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;  if (HAL\_TIM\_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL\_OK)  {  Error\_Handler();  }  sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;  sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;  if (HAL\_TIMEx\_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL\_OK)  {  Error\_Handler();  }  /\* USER CODE BEGIN TIM4\_Init 2 \*/  /\* USER CODE END TIM4\_Init 2 \*/  }  /\*\*  \* @brief USART1 Initialization Function  \* @param None  \* @retval None  \*/  static void MX\_USART1\_UART\_Init(void)  {  /\* USER CODE BEGIN USART1\_Init 0 \*/  /\* USER CODE END USART1\_Init 0 \*/  /\* USER CODE BEGIN USART1\_Init 1 \*/  /\* USER CODE END USART1\_Init 1 \*/  huart1.Instance = USART1;  huart1.Init.BaudRate = 115200;  huart1.Init.WordLength = UART\_WORDLENGTH\_8B;  huart1.Init.StopBits = UART\_STOPBITS\_1;  huart1.Init.Parity = UART\_PARITY\_NONE;  huart1.Init.Mode = UART\_MODE\_TX\_RX;  huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;  huart1.Init.OverSampling = UART\_OVERSAMPLING\_16;  if (HAL\_UART\_Init(&huart1) != HAL\_OK)  {  Error\_Handler();  }  /\* USER CODE BEGIN USART1\_Init 2 \*/  /\* USER CODE END USART1\_Init 2 \*/  }  /\*\*  \* @brief USART3 Initialization Function  \* @param None  \* @retval None  \*/  static void MX\_USART3\_UART\_Init(void)  {  /\* USER CODE BEGIN USART3\_Init 0 \*/  /\* USER CODE END USART3\_Init 0 \*/  /\* USER CODE BEGIN USART3\_Init 1 \*/  /\* USER CODE END USART3\_Init 1 \*/  huart3.Instance = USART3;  huart3.Init.BaudRate = 115200;  huart3.Init.WordLength = UART\_WORDLENGTH\_8B;  huart3.Init.StopBits = UART\_STOPBITS\_1;  huart3.Init.Parity = UART\_PARITY\_NONE;  huart3.Init.Mode = UART\_MODE\_TX\_RX;  huart3.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;  huart3.Init.OverSampling = UART\_OVERSAMPLING\_16;  if (HAL\_UART\_Init(&huart3) != HAL\_OK)  {  Error\_Handler();  }  /\* USER CODE BEGIN USART3\_Init 2 \*/  /\* USER CODE END USART3\_Init 2 \*/  }  /\*\*  \* @brief GPIO Initialization Function  \* @param None  \* @retval None  \*/  static void MX\_GPIO\_Init(void)  {  GPIO\_InitTypeDef GPIO\_InitStruct =  { 0 };  /\* GPIO Ports Clock Enable \*/  \_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();  \_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();  \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();  /\*Configure GPIO pin Output Level \*/  HAL\_GPIO\_WritePin(GPIOB, RELAY1\_Pin | RELAY2\_Pin | RELAY3\_Pin,  GPIO\_PIN\_SET);  /\*Configure GPIO pins : RELAY1\_Pin RELAY2\_Pin RELAY3\_Pin \*/  GPIO\_InitStruct.Pin = RELAY1\_Pin | RELAY2\_Pin | RELAY3\_Pin;  GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;  GPIO\_InitStruct.Pull = GPIO\_NOPULL;  GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;  HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);  /\*Configure GPIO pins : BUT1\_Pin BUT2\_Pin BUT3\_Pin \*/  GPIO\_InitStruct.Pin = BUT1\_Pin | BUT2\_Pin | BUT3\_Pin;  GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;  GPIO\_InitStruct.Pull = GPIO\_NOPULL;  HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);  }  /\* USER CODE BEGIN 4 \*/  /\* USER CODE END 4 \*/  /\*\*  \* @brief This function is executed in case of error occurrence.  \* @retval None  \*/  void Error\_Handler(void)  {  /\* USER CODE BEGIN Error\_Handler\_Debug \*/  /\* User can add his own implementation to report the HAL error return state \*/  \_\_disable\_irq();  while (1)  {  }  /\* USER CODE END Error\_Handler\_Debug \*/  }  #ifdef USE\_FULL\_ASSERT  /\*\*  \* @brief Reports the name of the source file and the source line number  \* where the assert\_param error has occurred.  \* @param file: pointer to the source file name  \* @param line: assert\_param error line source number  \* @retval None  \*/  void assert\_failed(uint8\_t \*file, uint32\_t line)  {  /\* USER CODE BEGIN 6 \*/  /\* User can add his own implementation to report the file name and line number,  ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/  /\* USER CODE END 6 \*/  }  #endif /\* USE\_FULL\_ASSERT \*/ | #define BLYNK\_TEMPLATE\_ID "TMPL6GiBfhBf-"  #define BLYNK\_TEMPLATE\_NAME "Smarthome System"  //token đăng nhâp blynk  #define BLYNK\_AUTH\_TOKEN "63KlKWwKQ2r9fKaVEGfvZjmOWG4KewIr"  #define BLYNK\_PRINT Serial // Comment this out to disable prints and save space  #include <ESP8266WiFi.h>  #include "EEPROM.h"  #include <BlynkSimpleEsp8266.h>  #include <ESP8266WebServer.h>  /\*===================================== BEGIN Global definition =================================\*/  char auth[] = BLYNK\_AUTH\_TOKEN;  BlynkTimer timer;  // khai báo máy chủ web  ESP8266WebServer server(80);  //tên wifi và pass mặc định của esp8266  const char \*ssid = "DoAnTotNghiep";  const char \*passphrase = "1234567890";  String availAP\_list;  String webPage\_Content;  int statusCode;  int wifiFlag = 0;  /\*\*\*\*\*\*\*\*\*\*\*\* RELAY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  typedef struct relay  {    const int virtual\_Pin;    bool state;  } relay;  //loại dữ liệu  typedef enum  {    DATA, CMD  } MSG\_TYPE;  #define RELAY\_OFF\_LEVEL LOW  #define RELAY\_ON\_LEVEL  HIGH  #define NUM\_OF\_RELAY 3  #define VPIN\_SWITCH\_1 V0  #define VPIN\_SWITCH\_2 V1  #define VPIN\_SWITCH\_3 V2  enum CONTROL\_DEVICE  {    SWITCH\_1 = 0,    SWITCH\_2,    SWITCH\_3,  };  //danh sách các thiết bị  relay relay\_list[NUM\_OF\_RELAY] = {      {.virtual\_Pin = VPIN\_SWITCH\_1, .state = false},      {.virtual\_Pin = VPIN\_SWITCH\_2, .state = false},      {.virtual\_Pin = VPIN\_SWITCH\_3, .state = false},  };  /\*===================================== END Global definition =================================\*/  /\*===================================== BEGIN WIFI Handle =================================\*/  //kiểm tra kết nối wifi đã ra được internet chưa  bool testWifi(void)  {    int c = 0;    Serial.println("Xin vui long doi ket noi WIFI");    while (c < 20)    {      if (WiFi.status() == WL\_CONNECTED)      {        return true;      }      delay(1000);      Serial.print(WiFi.status());      c++;    }    Serial.println("");    Serial.println("Thoi gian ket noi cham, Mo AP");    return false;  }  //quét danh sách các wifi sẵn có  void get\_AvailAP\_List()  {    WiFi.mode(WIFI\_STA);    WiFi.disconnect();    delay(100);    int n = WiFi.scanNetworks();    Serial.println("Tim hoan tat");    if (n == 0)    {      Serial.println("no networks found");    }    else    {      Serial.print(n);      Serial.println(" networks found");      for (int i = 0; i < n; ++i)      {        // Print SSID and RSSI for each network found        Serial.print(i + 1);        Serial.print(": ");        Serial.print(WiFi.SSID(i));        Serial.print(" (");        Serial.print(WiFi.RSSI(i));        Serial.print(")");        Serial.println((WiFi.encryptionType(i) == ENC\_TYPE\_NONE) ? " " : "\*");        delay(10);      }    }    Serial.println("");    availAP\_list = "<ol>";    for (int i = 0; i < n; ++i)    {      // Print SSID and RSSI for each network found      availAP\_list += "<li>";      availAP\_list += WiFi.SSID(i);      availAP\_list += " (";      availAP\_list += WiFi.RSSI(i);      availAP\_list += ")";       availAP\_list += (WiFi.encryptionType(i) == ENC\_TYPE\_NONE) ? " " : "\*";      availAP\_list += "</li>";    }    availAP\_list += "</ol>";  }  //Khởi động chế độ softAP cho esp8266 để phát wifi cấu hình  void setupAP(void)  {    get\_AvailAP\_List();    delay(100);    Serial.println("softap");    Serial.println(ssid);    Serial.println(passphrase);    WiFi.softAP(ssid, passphrase, 6);    launchWeb(1);  }  /\*===================================== END WIFI Handle =================================\*/  /\*===================================== BEGIN WEB Handle =================================\*/  //khởi chạy web cấu hình cho esp8266  void launchWeb(int webtype)  {    Serial.println("");    Serial.println("WiFi ket noi");    Serial.print("Dia chi IP: ");    Serial.println(WiFi.localIP());    Serial.print("SoftAP IP: ");    Serial.println(WiFi.softAPIP());    createWebServer(webtype);    // Start the server    server.begin();    Serial.println("May chu bat dau");  }  //Thiết lập nội dung giao diện web cấu hình  void getWebContent()  {    webPage\_Content = "<!DOCTYPE HTML>\r\n<html>";    webPage\_Content += "<script type=\"text/javascript\">";    webPage\_Content += "function init(){}";    webPage\_Content += "</script>";    webPage\_Content += "<body onload=\"init();\"><center><h2>ESP8266 Configuration</h2></center>";    webPage\_Content += "<form method=\"get\" action=\"/onlineSetup\"><h3>Setup Network Online Control</h3>";    webPage\_Content += "<div>Wifi</div>";    webPage\_Content += "<div><input name=\"ssid\" size=\"40\"></div>";    webPage\_Content += "<div>Mat Khau</div>";    webPage\_Content += "<div><input type=\"password\" name=\"pass\" size=\"40\"></div>";    webPage\_Content += "<div>Blynk key</div>";    webPage\_Content += "<div><input name=\"blynk\" size=\"40\"></div><br>";    webPage\_Content += "<div><input type='submit' value=\"SAVE\"></div>";    webPage\_Content += "<h3>Available Wifi</h3>";    webPage\_Content += "<p>";    webPage\_Content += availAP\_list;    webPage\_Content += "</p></body>";    webPage\_Content += "</html>";    Serial.print("Get web content done");  }  // nhận cài đặt từ người dùng và lưu vào EEPROM  void onlineSetup\_handler()  {    String qsid = server.arg("ssid");    String qpass = server.arg("pass");    String qblynk = server.arg("blynk");    Serial.println("onlineSetup\_handler 1");    if (qsid.length() > 0 && qpass.length() > 0)    {      Serial.println("onlineSetup\_handler 2");      EEPROM.begin(512);      Serial.println("clearing eeprom");      for (int i = 0; i < 128; ++i)      {        EEPROM.write(i, 0);      }      EEPROM.commit();      Serial.println(qsid);      Serial.println("");      Serial.println(qpass);      Serial.println("");      Serial.println(qblynk);      Serial.println("");      Serial.println("writing eeprom ssid:");      for (int i = 0; i < qsid.length(); ++i)      {        EEPROM.write(i, qsid[i]);        Serial.print("Wrote: ");        Serial.println(qsid[i]);      }      Serial.println("writing eeprom pass:");      for (int i = 0; i < qpass.length(); ++i)      {        EEPROM.write(32 + i, qpass[i]);        Serial.print("Wrote: ");        Serial.println(qpass[i]);      }      Serial.println("writing eeprom blynk:");      for (int i = 0; i < qblynk.length(); ++i)      {        EEPROM.write(96 + i, qblynk[i]);        Serial.print("Wrote: ");        Serial.println(qblynk[i]);      }      EEPROM.commit();      EEPROM.end();      // Chop den xanh sau khu lam xong      pinMode(13, OUTPUT);      digitalWrite(13, LOW);      digitalWrite(13, HIGH);      delay(500);      digitalWrite(13, LOW);      webPage\_Content = "{\"Success\":\"Luu vao he thong. Khoi dong lai ten wifi moi\"}";      statusCode = 200;    }    else    {      webPage\_Content = "{\"Error\":\"404 not found\"}";      statusCode = 404;      Serial.println("Sending 404");    }    server.send(statusCode, "application/json", webPage\_Content);  }  //xóa eeprom, xóa các cấu hình đã lưu  void restoreDefault\_handler()  {    webPage\_Content = "<!DOCTYPE HTML>\r\n<html>";    webPage\_Content += "<h2>XSwitch</h2><p>Clearing the EEPROM</p></html>";    server.send(200, "text/html", webPage\_Content);    Serial.println("clearing eeprom");    for (int i = 0; i < 128; ++i)    {      EEPROM.write(i, 0);    }    EEPROM.commit();  }  void mainPage\_handler()  {    Serial.println("mainPage\_handler");    getWebContent();    server.send(200, "text/html", webPage\_Content);  }  // Tạo web server  void createWebServer(int webtype)  {    if (webtype == 1)    {      server.on("/", mainPage\_handler);      server.on("/onlineSetup", onlineSetup\_handler);      server.on("/restoreDefault", restoreDefault\_handler);    }  }  /\*===================================== END WEB Handle =================================\*/  /\*===================================== BEGIN RELAY Handle =================================\*/  // Gửi trạng thái các thiết bị lên blynk server  void updateDeviceStateToCloud(int devId, int devState){    switch (devId)    {    case 1:      Blynk.virtualWrite(VPIN\_SWITCH\_1, devState);      break;    case 2:      Blynk.virtualWrite(VPIN\_SWITCH\_2, devState);      break;    case 3:      Blynk.virtualWrite(VPIN\_SWITCH\_3, devState);      break;    default:      break;    }    relay\_list[devId - 1].state = devState;  }  //Xử lý lệnh điều khiển từ stm32  bool processDataFromStm32(const char \*input){    MSG\_TYPE msgType;    int devId, devState;    if (3 != sscanf(input, "|S[%d]:[%d]-[%d]E|", &msgType, &devId, &devState))    {      Serial.println("Fail to parse rx data");      return false;    }    updateDeviceStateToCloud(devId, devState);    return true;  }  // gửi lệnh bật thiết bị tới stm32  bool uart\_sendTurnOnCmd(int relayId)  {    char strData[64] = { 0 };    MSG\_TYPE msgType = CMD;    snprintf(strData, sizeof(strData), "|S[%d]:[%d]-[%d]E|", msgType, relayId + 1, 1);    Serial.println(strData);    return true;  }  // gửi lệnh tắt thiết bị tới stm32  bool uart\_sendTurnOffCmd(int relayId)  {    char strData[64] = { 0 };    MSG\_TYPE msgType = CMD;    snprintf(strData, sizeof(strData), "|S[%d]:[%d]-[%d]E|", msgType, relayId + 1, 0);    Serial.println(strData);    return true;  }  //thực thi lệnh bật thiết bị từ blynk  void relayOn(int relayId)  {    if(RELAY\_OFF\_LEVEL == relay\_list[relayId].state)    {      // turn on relay      uart\_sendTurnOnCmd(relayId);      relay\_list[relayId].state = RELAY\_ON\_LEVEL;    }  }  //thực thi lệnh tắt thiết bị từ blynk  void relayOff(int relayId)  {    if(RELAY\_ON\_LEVEL == relay\_list[relayId].state)    {      // turn off relay      uart\_sendTurnOffCmd(relayId);    relay\_list[relayId].state = RELAY\_OFF\_LEVEL;    }  }  //Xử lý lệnh điều khiển từ blynk  void relay\_control(int relayId, int cmd)  {    if (cmd == RELAY\_ON\_LEVEL)    {      relayOn(relayId);    }    else if (cmd == RELAY\_OFF\_LEVEL)    {      relayOff(relayId);    }    else    {      Serial.println("Relay control error, unknown command");    }  }  //Khởi tạo trạng thái thiết bị  void initControlDevice(void)  {    for (int relay\_idx = 0; relay\_idx < NUM\_OF\_RELAY; relay\_idx++)    {      relayOff(relay\_idx);    }  }  /\*===================================== END RELAY Handle =================================\*/  /\*===================================== BEGIN Blynk Handle =================================\*/  int blynk\_control\_cmd = 0;  //Hàm nhận lệnh từ blynk cho thiết bị 1  BLYNK\_WRITE(VPIN\_SWITCH\_1)  {    int blynk\_control\_cmd = param.asInt();    relay\_control(SWITCH\_1, blynk\_control\_cmd);  }  //Hàm nhận lệnh từ blynk cho thiết bị 2  BLYNK\_WRITE(VPIN\_SWITCH\_2)  {    int blynk\_control\_cmd = param.asInt();    relay\_control(SWITCH\_2, blynk\_control\_cmd);  }  //Hàm nhận lệnh từ blynk cho thiết bị 3  BLYNK\_WRITE(VPIN\_SWITCH\_3)  {    int blynk\_control\_cmd = param.asInt();    relay\_control(SWITCH\_3, blynk\_control\_cmd);  }  //Thực thi khi đã kết nối thành công tới blynk server  BLYNK\_CONNECTED() {      // Blynk.syncAll();      initBlynkWidgetState();  }  //Gửi dữ liệu khởi tạo đến blynk  void initBlynkWidgetState(){    // Update Button state    Blynk.virtualWrite(VPIN\_SWITCH\_1, RELAY\_OFF\_LEVEL);    Blynk.virtualWrite(VPIN\_SWITCH\_2, RELAY\_OFF\_LEVEL);    Blynk.virtualWrite(VPIN\_SWITCH\_3, RELAY\_OFF\_LEVEL);  }  //Kiểm tra trạng thái kết nối tới blynk  void checkBlynkStatus()  { // called every 3 seconds by SimpleTimer    bool isconnected = Blynk.connected();    if (isconnected == false)    {      wifiFlag = 1;      // digitalWrite(wifiLed, LOW); //Turn off WiFi LED    }    if (isconnected == true)    {      wifiFlag = 0;      // digitalWrite(wifiLed, HIGH); //Turn on WiFi LED    }  }  /\*===================================== END Blynk Handle =================================\*/  void setup()  {    Serial.begin(115200);    initControlDevice();    Serial.println("Startup");  // Khởi tạo eeprom    EEPROM.begin(512);    Serial.println("EEPROM Begined");    delay(10);    // đọc thông tin  ssid, pass and blynk token từ eeprom    Serial.println("Reading EEPROM ssid");    String esid;    for (int i = 0; i < 32; ++i)    {      esid += char(EEPROM.read(i));    }    Serial.print("SSID: ");    Serial.println(esid.c\_str());    esid.trim();    Serial.println("Reading EEPROM pass");    String epass = "";    for (int i = 32; i < 96; ++i)    {      epass += char(EEPROM.read(i));    }    Serial.print("PASS: ");    Serial.println(epass.c\_str());    epass.trim();    Serial.println("Reading EEPROM blynk");    String eblynk = "";    for (int i = 96; i < 128; ++i)    {      eblynk += char(EEPROM.read(i));    }    Serial.print("BLYNK: ");    Serial.println(eblynk.c\_str());    eblynk.trim();    if (esid.length() > 1)    {      //Kết nối tới bộ phát wifi đã lưu      WiFi.begin(esid.c\_str(), epass.c\_str());      if (testWifi())      {        launchWeb(1);        WiFi.disconnect();        char \*auth\_ = new char[eblynk.length() + 1];        //Nếu kết nối wifi thành công thì kết nối tới blynk server        eblynk.toCharArray(auth\_, eblynk.length() + 1);        Blynk.begin(auth, esid.c\_str(), epass.c\_str(), "blynk.cloud", 80);          EEPROM.end();        return;      }    }    //Nếu kết nối wifi chưa thành công thì tự phát ra wifi để cấu hình    setupAP();    EEPROM.end();  }  void loop()  {    // đợi request truy nhập web cấu hình của người dùng    server.handleClient();    if (WiFi.status() == WL\_CONNECTED)    {      //đợi lệnh và xử lý lệnh từ blynk      Blynk.run();    }    //Đợi lệnh và xử lý lệnh từ stm32      if(Serial.available()){      String serialRxData;      serialRxData = Serial.readString();      serialRxData.trim();      processDataFromStm32(serialRxData.c\_str());    }  } |