# LỜI CẢM ƠN

Trước tiên em xin bày tỏ lòng biết ơn chân thành và sâu sắc nhất của mình tới Ths. Dương Thị Hằng người đã hướng dẫn tận tình và hiệu quả, thường xuyên động viên chúng em trong quá trình hoàn thiện đề tài. Người đã dành cho em sự ưu ái nhất trong thời gian học tập, nghiên cứu cũng như quá trình hoàn thành thực tập tốt nghiệp.

Em xin cảm ơn các Thầy giáo, Cô giáo trong khoa Điện Tử trường Đại học Công Nghiệp Hà Nội cùng tất cả thành viên lớp Điện tử 04 – K13 đã tạo điều kiện và đóng góp ý kiến để em hoàn thành tốt đồ án tốt nghiệp.

Mặc dù em đã cố gắng để hoàn thành thực tập nhưng do kiến thức cũng như khả năng còn hạn hẹp nên quá trình thực hiện đề tài còn có sai sót. Rất mong nhận được sự góp ý và chỉ bảo của quý thầy cô.

Hà Nội, Ngày… tháng… năm 2022

Sinh viên thực hiện

Nguyễn Văn Vinh

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc104024037)

[MỤC LỤC 2](#_Toc104024038)

[DANH MỤC HÌNH ẢNH, BẢNG BIỂU 4](#_Toc104024039)

[LỜI MỞ ĐẦU 6](#_Toc104024040)

[CHƯƠNG 1: TỔNG QUAN VỀ MÔ HÌNH HỆ THỐNG ĐÈN VÀ RÈM CỬA THÔNG MINH SỬ DỤNG ESP32 8](#_Toc104024041)

[1.1 Tổng quan về vấn đề nghiên cứu 8](#_Toc104024042)

[1.1.1 Tình hình nghiên cứu ngoài nước. 8](#_Toc104024043)

[1.1.2 Tình hình nghiên cứu trong nước 10](#_Toc104024044)

[1.2 Đặt vấn đề 14](#_Toc104024045)

[1.3 Mục đích nghiên cứu 14](#_Toc104024046)

[1.4 Mục tiêu đề tài. 15](#_Toc104024047)

[1.5 Phạm vi nghiên cứu 15](#_Toc104024048)

[1.6 Nội dung nghiên cứu. 15](#_Toc104024049)

[1.7 Vai trò và ý nghĩa của đề tài 16](#_Toc104024050)

[1.8 Kết luận chương 1. 17](#_Toc104024051)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 18](#_Toc104024052)

[2.1 Các giao thức chính được sử dụng 18](#_Toc104024053)

[2.1.1 Giao thức WiFi 18](#_Toc104024054)

[2.1.2 Giao thức HTTP 20](#_Toc104024055)

[2.1.3 Giao thức MQTT 21](#_Toc104024056)

[2.2 Linh kiện và phần mềm thiết kế mạch 21](#_Toc104024057)

[2.2.1 Vi điều khiển ESP32 21](#_Toc104024058)

[2.2.2 IC cách ly nguồn B5050S 24](#_Toc104024059)

[2.2.3 Mạch Buck LM2596 25](#_Toc104024060)

[2.2.4 Transistor C1815 26](#_Toc104024061)

[2.2.5 Diode 1N4148 26](#_Toc104024062)

[2.2.6 Relay 5v 27](#_Toc104024063)

[2.2.7 Các linh kiện thụ động 28](#_Toc104024064)

[2.2.8 Phần mềm thiết kế Altium Designer 28](#_Toc104024065)

[2.3 Phần mềm của hệ thống 31](#_Toc104024066)

[2.3.1 HTML (HyperText Markup Language). 31](#_Toc104024067)

[2.3.2 CSS (Cascading Style Sheets) 31](#_Toc104024068)

[2.3.3 JavaScript 32](#_Toc104024069)

[2.3.4 Phần mềm Visual Studio code 32](#_Toc104024070)

[2.3.5 Phần mềm Arduino IDE 33](#_Toc104024071)

[2.3.6 Hệ điều hành FreeRTOS 36](#_Toc104024072)

[2.4 Kết luận chương 2 37](#_Toc104024073)

[CHƯƠNG 3: THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT 38](#_Toc104024074)

[3.1 Thiết kế sơ đồ khối hệ thống điều khiển và giám sát 38](#_Toc104024075)

[3.1.1 Yêu cầu của hệ thống. 38](#_Toc104024076)

[3.1.2 Sơ đồ khối 38](#_Toc104024077)

[3.2 Sơ đồ nguyên lý 40](#_Toc104024078)

[3.2.1 Khối xử lý trung tâm 41](#_Toc104024079)

[3.2.2 Khối nguồn và ổn áp nguồn 42](#_Toc104024080)

[3.2.3 Khối nút nhấn 42](#_Toc104024081)

[3.2.4 Khối Relay 42](#_Toc104024082)

[3.2.5 Khối mở rộng 43](#_Toc104024083)

[3.3 Sơ đồ mạch in (PCB) 44](#_Toc104024084)

[3.4 Lập trình vi xử lý ESP32 44](#_Toc104024085)

[3.4.1 Các thư viện được sử dụng 44](#_Toc104024086)

[3.4.2 Xử lý sự kiện nhấn nút: 45](#_Toc104024087)

[3.4.3 Xử lý sự kiện nhận về từ MQTT Broker 46](#_Toc104024088)

[3.5 Lập trình Webserver 46](#_Toc104024089)

[3.6 Mô hình thực tế của hệ thống. 48](#_Toc104024090)

[3.7 Giao diện điều khiển hệ thống trên Web 48](#_Toc104024091)

[3.8 Kết luận chương 3. 49](#_Toc104024092)

[KẾT LUẬN 50](#_Toc104024093)

[TÀI LIỆU THAM KHẢO. 52](#_Toc104024094)

[PHỤ LỤC 53](#_Toc104024095)

# DANH MỤC HÌNH ẢNH, BẢNG BIỂU

[Hình 1.1. Rèm cửa tự động 9](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024096)

[Hình 1.2. Rèm cửa thông minh điều khiển bằng điện thoại 9](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024097)

[Hình 1.3. Điều khiển không dây 10](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024098)

[Hình 1.4. Rèm cuốn tự động. 12](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024099)

[Hình 1.5. Rèm sáo dọc tự động 13](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024100)

[Hình 1.6. Rèm sáo gỗ tự động. 13](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024101)

[Hình 2.1. Giao thức WiFi 18](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024102)

[Hình 2.2. Các chuẩn Wifi hiện nay 19](#_Toc104024103)

[Hình 2.3. Giao thức HTTP. 20](#_Toc104024104)

[Hình 2.4. Cách thức hoạt động của giao thức HTTP 20](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024105)

[Hình 2.5. Giao thức MQTT. 21](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024106)

[Hình 2.6: Module giảm áp LM2596 25](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024107)

[Hình 2.7. Diode 1N4148. 27](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024108)

[Hình 2.8: Altium Designer 2022 28](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024109)

[Hình 2.9. Giao diện phần mềm Altium Designer 22 30](#_Toc104024110)

[Hình 2.10: Visual Studio Code 32](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024111)

[Hình 2.11. Arduino IDE 33](#_Toc104024112)

[Hình 2.12. Giao diện phần mềm Arduino IDE 34](#_Toc104024113)

[Hình 2.13. Hệ điều hành FreeRTOS 36](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024114)

[Hình 2.14. Chức năng của hệ điều hành FreeRTOS 36](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024115)

[Hình 3.1. Sơ đồ gửi nhận dữ liệu. 38](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024116)

[Hình 3.2. Sơ đồ khối của hệ thống. 39](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024117)

[Hình 3.3. Mạch nguyên lý của hệ thống 40](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024118)

[Hình 3.4: Khối xử lý trung tâm 41](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024119)

[Hình 3.5: Khối nút nhấn 42](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024120)

[Hình 3.6. Khối Relay 43](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024121)

[Hình 3.7: Khối mở rộng 43](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024122)

[Hình 3.8. Sơ đồ mạch in 2D 44](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024123)

[Hình 3.9. Quá trình sử lý nút nhấn. 45](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024124)

[Hình 3.10. Quá trình xử lý sự kiện nhận về MQTT Broker, 46](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Nguyen_Van_Vinh_2018605465\quyen_bao_cao\NVVinh_quyen_bao_cao_v2.docx#_Toc104024125)

[Hình 3.11. Hình ảnh thực tế của hệ thống. 48](#_Toc104024126)

[Hình 3.12: Giao diện điều khiển trên Web 48](#_Toc104024127)

[Bảng 2.1. Thông số ngõ vào ra của B5050S 24](#_Toc104024132)

[Bảng 2.2: Các thông số của transistor C1815 26](#_Toc104024133)

[Bảng 2.3. Các thông số của Relay 5V 27](#_Toc104024134)

[Bảng 2.4. Chức năng các phím trên giao diện Arduino IDE 34](#_Toc104024135)

# LỜI MỞ ĐẦU

Ngày nay, xã hội ngày càng phát triển, công nghiệp hóa, hiện đại hóa ngày càng được nâng cao để phát triển đất nước và cải thiện cuộc sống của người dân. Vì vậy việc ứng dụng khoa học kỹ thuật ngày càng rộng rãi, phổ biến và mang lại hiệu quả cao trong hầu hết các lĩnh vực.

Với sự bùng nổ của Cuộc cách mạng công nghiệp lần thứ tư đang diễn ra từ những năm 2000 gọi là cuộc cách mạng số, thông qua các công nghệ như Internet vạn vật (IoT), trí tuệ nhân tạo (AI), thực tế ảo (VR), tương tác thực tại ảo (AR), mạng xã hội, điện toán đám mây, di động, phân tích dữ liệu lớn (SMAC)... để chuyển hóa toàn bộ thế giới thực thành thế giới số.

Trong các công nghệ của cách mạng 4.0. Với sự phát triển của Internet, công nghệ Internet vạn vật (IoT) đang được phát triển một cách nhanh chóng. Hệ sinh thái IOT cho phép các tổ chức có thể kết nối, kiểm soát và sử dụng các thiết bị IOT. Trong hệ sinh thái này, một tổ chức có thể sử dụng các thiết bị như điện thoại thông minh, máy tính bản, … để gửi đi các hiệu lệnh, hoặc truy cập thông tin từ một mạng lưới các thiết bị IOT khác. Trong trường hợp hiệu lệnh, thiết bị nhận lệnh sẽ thực hiện các công việc được thiết kế, thu thập dữ liệu để được truy cập và phân tích nhanh chóng.

Hệ thống giám sát điều khiển từ xa là một giải pháp vô cùng thông minh. Đây là thành quả của nền công nghiệp 4.0. Thay vì sử dụng phương thức điều khiển và giám sát điều khiển thủ công kém hiệu quả, chính xác. Giờ đây với hệ thống giám, người vận hành có thể giám sát từ xa mọi lúc, mọi nơi thông qua các thiết bị thông minh có hỗ trợ kết nối internet: smartphone, table, laptop,…

Hệ thống đóng vai trò vô cùng quan trọng. Với chức năng thực hiện thu thập và tích hợp dữ liệu đồng thời phân tích để hoàn thành nhiệm vụ đảm bảo cho một hệ thống hoạt động ổn định mang hiệu quả cao. Đặc biệt trong công nghiệp có nhiều các hệ thống nhà máy lớn, việc sử dụng hệ thống giám sát từ xa càng quan trọng. Việc giám sát và điều khiển toàn bộ hệ thống được đảm bảo. Hệ thống còn hỗ trợ việc giám sát – điều khiển tập trung các nhà máy phân tán ở bất kỳ vị trí nào.

Để tìm hiểu về việc điều khiển và giám sát từ xa, cũng như việc ứng dụng các kiến thức đã học về vi điều khiển, vi xử lý, các môn học về điều khiển và giám sát không dây, em đã quyết định chọn đề tài: **“Thiết kế hệ thống điều khiển và giám sát các thiết bị từ xa bằng Wifi sử dụng ESP32”** là đề tài đồ án tốt nghiệp.

Nội dung quyển báo cáo này bao gồm 3 chương chính:

Chương 1 - Tổng quan về hệ thống điều khiển và giám sát

Chương 2 - Cơ sở lý thuyết

Chương 3 - Thiết kế hệ thống điều khiển và giám sát

# TỔNG QUAN VỀ MÔ HÌNH HỆ THỐNG ĐÈN VÀ RÈM CỬA THÔNG MINH SỬ DỤNG ESP32

Tìm hiểu lịch sử phát triển, tình hình nghiên cứu của hệ thống đèn và rèm cửa thông minh tự động của thế giới nói chung và Việt Nam nói riêng. Từ đó nêu ra những vấn đề cần phải nghiên cứu nhằm nắm bắt và hiểu hơn về ưu nhược điểm của những hệ thống đã được áp dụng vào thực thế. quả cao hơn.

* 1. Tổng quan về vấn đề nghiên cứu
     1. Tình hình nghiên cứu ngoài nước.

Rèm cửa là một thiết kế không thể thiếu trong các công trình kiến trúc hiện nay, giúp linh động trong việc che chắn ánh nắng và tùy lúc có thể lấy ánh sáng mặt trời khi cần thiết. Ngoài ra, rèm cũng giúp giữ gìn sự riêng tư, kín đáo trong một số phòng ốc. Trong thời buổi hiện đại số lượng cửa nhiều đồng thời với đó là số lượng rèm cửa cũng nhiều tương ứng. Nếu dùng cách thủ công để kéo lại hoặc mở ra sẽ khá là bất tiện và mất thời gian không cần thiết. Chính vì thế IoT Tương Lai đã đem đến hệ thống rèm cửa thông minh như một giải pháp cho thời buổi hiện đại.

Rèm cửa thông minh được thiết kế phù hợp với từ loại rèm như kéo mở về một bên, từ hai bên kéo vào giữa và từ giữa mở ra, từ dưới kéo lên hoặc từ trên đóng xuống...tùy theo nhu cầu khách hàng

Với các motor giúp cho việc kéo đẩy được cơ khí hóa kèm với các mạch điện tử để nhận và truyền lệnh từ người sử dụng tương ứng giúp cho quá trình kéo mở được tự động hóa. Ngoài ra chúng tôi có phần mềm lập trình trên thiết bị thông minh có thể cài đặt dễ dàng. Người sử dụng chỉ cần dùng smartphone hoặc tablet và vào app vào có thể đóng mở bất cứ vị trí rèm cửa nào mong muốn theo lập trình sẵn.

**Rèm cửa tự động**

Hình 1.1. Rèm cửa tự động

Với hệ thống rèm cửa thông minh kiểm soát hoàn toàn tự động, gia chủ không cần phải sử dụng bất kỳ một thiết bị diều khiển nào, tạo cảm giác thoải mái và tiết kiệm năng lượng

Tự động Đóng/Mở rèm dựa theo ánh sáng ban ngày

Tự động Đóng/Mở rèm theo lịch trình đã thiết lập

**Kiểm soát rèm cửa thông minh bằng iOS / Android**

Hình 1.2. Rèm cửa thông minh điều khiển bằng điện thoại

Hệ thống rèm cửa thông minh tích hợp nhiều loại giải pháp hiện đai như:

* Đóng/Mở rèm cửa chỉ với một lần chạm trên điện thoại thông minh hoặc máy tính bảng
* Kiểm soát rèm cửa mọi lúc mọi nơi chỉ với 1 thiết bị có kết nối internet

**Kiểm soát rèm cửa bằng giọng nói**

Tích hợp giải pháp rèm cửa thông minh với các trợ lý giọng nói Google Assitant, Alexa Echo, Apple Siri

**Điều khiển từ xa không dây Pico**

Hình 1.3. Điều khiển không dây

Đôi khi điều khiển nhà thông minh bằng giọng nói là một cách thuận tiện nhất, nhưng nếu Quý vị đang xem phim hay nghe nhạc thì khả năng điều khiển giọng nói là khó khả thi. Với điều khiển từ xa không dây Pico để bàn, đầu giường, luôn mang đến sự tiện lợi

* + 1. Tình hình nghiên cứu trong nước

Không chỉ ngoài nước mà rèm tự động ở Việt Nam đang dần trở nên phổ biến trong trang trí nội thất hiện đại. Thực chất, rèm tự động là rèm có gắn kèm động cơ rèm (motor rèm), điều khiển bởi thiết bị điều khiển từ xa (Remote). Nhờ được gắn động cơ nên những bộ rèm to nặng, có độ cao lớn đã không còn là trở ngại. Đây được xem là một “tiêu điểm” trong thiết kế không gian hiện đại, là sự phát triển mang tính đột phá trong nghệ thuật rèm trang trí và giải quyết được nhiều nhược điểm của mành rèm truyền thống. Bên cạnh tác dụng che chắn nắng gió như rèm truyền thống, rèm tự động còn có nhiều lợi ích và ý nghĩa hơn thế:

Với không gian: rèm tự động là một điểm nhấn, làm đẹp không gian, thể hiện sự sang trọng, hiện đại, tinh tế và đẳng cấp. Đây là một yếu tố không thể thiếu trong thiết kế nội thất của ngôi nhà thông minh hiện đại. Nhờ thiết bị điều khiển từ xa, bạn có thể bật – tắt, đóng – mở rèm theo ý muốn và tiết kiệm thời gian, công sức. Đặc biệt, bạn cũng có thể tích hợp điều khiển rèm qua các thiết bị thông minh, hiện đại như: điện thoại, máy tính bảng, kết nối với hệ thống điều khiển nhà thông minh…. Chỉ cần một nút bấm, rèm cửa sẽ di chuyển theo ý muốn của bạn. Đồng thời thay đổi được diện tích không gian, không gây cản trở những hoạt động khác của con người.

Với người sử dụng: Rèm tự động là tiện ích vừa thể hiện gu thẩm mỹ vừa đem đến sự an toàn. Khác với mành rèm truyền thống, trẻ em có thể gặp tai nạn bất cứ lúc nào, nhưng với rèm tự động, bạn có thể để chúng chơi đùa thỏa thích gần rèm cửa, đóng – mở, chỉnh ánh sáng gian phòng theo ý muốn mà không cần mất công mệt mỏi. Nếu rèm được làm bằng chất liệu cao cấp, cách nhiệt, chống tia cực tím sẽ giúp bảo vệ con người và các thiết bị nội thất khác khỏi tác hại của thời tiết thất thường.

Hiện tại, IBC Việt Nam có giải pháp rèm tự động cho tất cả các loại rèm, với nhiều kiểu dáng, màu sắc khác nhau:

*Rèm vải tự động*: tại Việt Nam, rèm vải tự động được sử dụng rộng rãi, chiếm đến 80% trong các ngôi nhà, biệt thự, chung cư cao cấp. Vì sử dụng rèm vải là thói quen từ lâu của người Việt. Để đối phó với khí hậu nóng nực mùa hè, khô lạnh mùa đông, rèm vải là giải pháp tối ưu. Sự mềm mại, màu sắc đa dạng, phong phú của rèm vải cũng khiến cho không gian ngôi nhà trở nên sinh động, cuốn hút hơn. Những bộ rèm cao kịch trần được làm từ vải Bỉ Acacia, vải Anh, Thổ, sử dụng động cơ Forest, A-OK, Famax… đang được nhiều người tiêu dùng lựa chọn. Chất liệu vải bền, mềm mại, màu sắc bắt mắt kết hợp với động cơ rèm vận hành êm ái là điểm cộng của loại rèm này.

*Rèm cuốn tự động:* Với sự gọn nhẹ, tinh tế, tiện lợi, rèm cuốn tư động được sử dụng nhiều trong không gian công sở, trường học hoặc ngôi nhà thông minh. Nếu bạn cần một không gian phục vụ cho sự sáng tạo, hiện đại, mang phong cách Hàn Quốc, Châu Âu thì rèm cuốn tự động là một sự lựa chọn hoàn hảo và là điểm nhấn ấn tượng không thể bỏ qua.

Hình 1.4. Rèm cuốn tự động.

*Rèm sáo gỗ tự động:*Gỗ thường nặng hơn các vật liệu làm rèm khác. Tuy nhiên không vì thế mà chúng ta có thể bỏ qua một loại rèm độc đáo này. Được thiết kế dạng mành, rèm sáo gỗ tự động phù hợp với cả không gian hiện đại và cổ kính, đem lại sự sang trọng, đẳng cấp cho không gian. Đặc trưng của rèm sáo gỗ yêu cầu về sự tỉ mỉ trong thiết kế, chính xác trong lắp đặt, vì vậy đòi hỏi thợ thi công phải có kinh nghiệm lâu năm trong nghề.

*Rèm sáo dọc tự động:*IBC Việt Nam là một trong những thương hiệu đầu tiên đưa rèm lá dọc tự động Dooya, Bintronic… vào thị trường Việt Nam. Ưu điểm của rèm lá dọc tự động là vẻ thanh lịch, nhẹ nhàng, sang trọng… phù hợp với các doanh nhân thành đạt, phòng giám đốc, phòng tiếp khách.

Hình 1.5. Rèm sáo dọc tự động

Hình 1.6. Rèm sáo gỗ tự động.

* 1. Đặt vấn đề

Rèm cửa đóng vai trò quan trọng trong mỗi gia đình nó giúp cản sáng, điều chỉnh nhiệt độ và trang trí ngôi nhà của bạn. Tuy nhiên sẽ thật bất tiện khi mà hàng ngày cứ phải dùng tay đi kéo rèm làm rất mất thời gian. Ngày nay với công nghệ 4.0 hiện đại, các thiết bị nội thất được áp dụng công nghệ mới, rèm thông minh ra đời đã giúp cho người sử dụng thao tác dễ dàng hơn

Song song với hệ thống rèm cửa, hệ thống chiếu sáng cũng dần thay đổi theo để đáp ứng được nhu cầu thiết yếu của người dùng. Chiếu sáng thông minh đã dần dần trở nên quên thuộc trong những ngôi nhà hiện đại sang trọng.

Với hệ thống rèm cửa và chiếu sáng thông thường ở trong nhà thì người dùng hay bật tắt bóng đèn bằng công tắc cơ quen thuộc hoặc bằng các điều khiển hồng ngoại. Tuy nhiên, đối với chiều sáng thông minh, người dùng hoàn toàn có thể điều khiển từ xa thông qua điện thoại hoặc máy tính bảng, có thể đóng mở các thiết bị ở bất kỳ nơi nào chỉ với điện thoại hoặc máy tính có kết nối Internet

Với mục đích có thể tạo ra một hệ thống đèn và rèm cửa thông minh, thân thiện với người dùng, khả năng tùy biến cao và giá thành phải chăng, em xin giới thiệu đề tài: “**Thiết kế mô hình hệ thống đèn và rèm cửa thông minh sử dụng ESP32**”. Với mô hình này, chúng ta có thể phát triển thêm thành một hệ thống hoàn chỉnh, liên kết được các thiết bị lại với nhau, có thể giúp con người dễ dàng sử dụng, vận hành cũng như điều khiển từ xa ở mọi lúc mọi nơi chỉ cần có một thiết bị có thể kết nối được với Internet

* 1. Mục đích nghiên cứu

Nghiên cứu và sử dụng các tài nguyên của vi điều khiển ESP32 và các linh kiện sử dụng trong hệ thống. Thiết kế thành công mô hình đèn và rèm của thông minh sử dụng vi điều khiển ESP32.

* 1. Mục tiêu đề tài.

Trong đề tài này, em sẽ thực hiện thiết kế một mô hình hệ thống đèn và rèm cửa thông minh sử dụng ESP32 . Mô hình gồm 2 phần chính là mạch điện tử và phần mềm điều khiển. Đầu vào của mạch điện tử có 6 nút nhấn để điều khiển 5 đèn và 1 rèm cửa Phần mềm sẽ đồng bộ với nút nhấn trên phần cứng để hiển thị trạng thái bật tắt của thiết bị, đồng thời chúng ta có thể bật tắt thiết bị ngay trên phần mềm.

* 1. Phạm vi nghiên cứu
* Nghiên cứu về hệ thống rèm cửa và chiếu sáng thông minh
* Tạo ra một mô hình đơn giản bao gồm cả rèm cửa và đèn chiếu sáng để mô tả hoạt động của chúng
* Tạo một giao diện web để điều khiển và giám sát đèn và rèm cửa từ xa
* Nghiên cứu và ứng dụng vi điều khiển ESP32 vào các ứng dụng điều khiển từ xa, các ứng dụng về IoT (Internet of Things)
  1. Nội dung nghiên cứu.
* Tìm hiểu về các sản phẩm rèm cửa thông minh trong và ngoài nước.
* Xây dựng giải pháp tích hợp để điều khiển đèn và rèm cửa
* Thiết kế mô hình tổng quát của đề tài.
* Thiết kế sơ đồ tổng thể của đề tài.
* Thiết kế, lắp ráp và kiểm tra từng hệ thống riêng lẻ.
* Xây dựng giao diện điều khiển và giám sát từ xa
* Ứng dụng IoT vào theo dõi và giám sát trên các thiết bị có kết nối internet
* Lập chương trình để điều khiển hoạt động của các thiết bị điện.
* Hoàn thiện, kiểm tra và chạy thử nghiệm.
* Đưa ra những ưu điểm và nhược điểm của sản phẩm, hướng phát triển của sản phẩm
  1. Vai trò và ý nghĩa của đề tài

*Vai trò của đề tài*

Đề tài tập trung vào thiết mô hình hệ thống đèn và rèm cửa thông minh sử dụng ESP32.

Đề tài được đưa ra nhằm tạo ra hệ thống đèn và rèm cửa thông minh ứng dụng nhiều trong các ngôi nhà cần yêu cầu một hệ thống tự động và điều khiển từ xa

Ngoài ra, đối với sinh viên, đề tài còn là chủ đề để tìm hiểu về công nghệ điều khiển và giám sát từ xa, công nghệ IoT…

*Ý nghĩa của đề tài*

Đề tài “Thiết kế hệ thống đèn và rèm cửa thông minh sử dụng ESP32” tự động bật đèn khi trời tối, điều khiển rèm không dây. Khi sử dụng hệ thống người dùng có thể quản lí các thiết bị điện một cách dễ dàng và chủ động, hệ thống còn giúp tự động của cảm biến ánh .

Về mặt lý luận, toàn bộ chương trình và bản thiết kế của để tài sẽ trở thành tài liệu nghiên cứu hữu ích cho các bạn sinh viên. Từ đây, các bạn sinh viên có thể dựa trên lý thuyết và mô hình để đi sâu tìm hiểu, phát triển hoàn thiện hơn hệ thống nhà thông minh và đưa vào áp dụng trong cuộc sống.

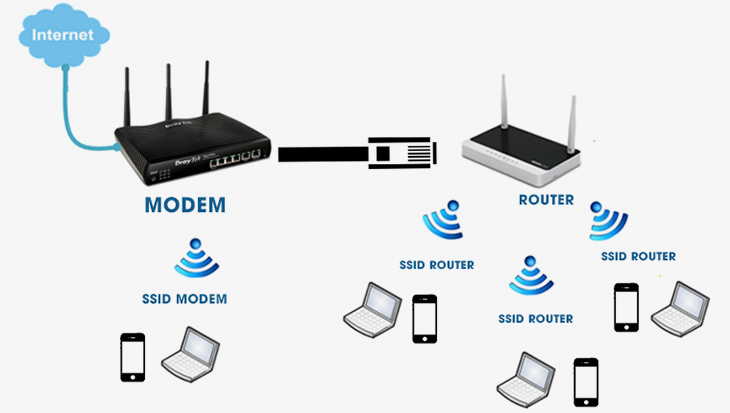
Về mặt khoa học, sự thành công của đề tài sẽ góp phần giúp cho các bạn sinh viên mới nói chung và các bạn sinh viên khoa Điện Tử - Viễn Thông nói riêng thấy rõ được ý nghĩa thực tế và thêm yêu thích chuyên ngành mình đã chọn. Xây dựng cơ sở khoa học về áp dụng tự động hóa vào cuộc sống thường ngày.

* Tìm hiểu về các hệ thống đèn và rèm cửa thông minh
* Dần đưa các thiết bị thông minh vào trong cuộc sống
  1. Kết luận chương 1.

Sau khi tìm hiểu được sự phát triển, quy mô của hệ thống đèn và rèm cửa thông minh. Tiếp thu và nắm bắt những ưu điểm đã được phát triển trước đó. Cố gắng nắm bắt được những điểm mạnh điểm yếu của hệ thống, đưa ra giải pháp giải quyết vấn đề, đồng thời phát triển ý tưởng, đưa những sáng kiến có thể thực hiện, áp dụng vào thực tiễn. Tiếp theo, từ những kiến thức đã tìm hiểu được, đưa ra bài toán vận hành cho hệ thống, xây dựng sơ đồ khối đồng thời đưa ra chức năng của từng khối. Từ đó lựa tính toán lựa chọn linh kiện phù hợp với yêu cầu đã đặt ra.

# CƠ SỞ LÝ THUYẾT

* 1. Các giao thức chính được sử dụng
     1. Giao thức WiFi

Wi-Fi là một họ các giao thức mạng không dây, dựa trên các tiêu chuẩn của họ IEEE 802.11, được sử dụng rộng rãi trong cho việc kết nối không dây của thiết bị trong mạng nội bộ và việc kết nối Internet. WiFi cho phép các thiết bị điện tử trong phạm vi ngắn chia sẻ dữ liệu thông qua sóng vô tuyến. Ngày nay, WiFi được sử dụng phổ biến trong các hệ thống mạng máy tính trên thế giới, như trong các hộ gia đình, văn phòng làm việc cho việc kết nối các máy tính bàn, laptop, tablet, điện thoại thông minh, máy in,... mà không cần đến cáp mạng, cũng như việc kết nối Internet cho các thiết bị này. Nhiều địa điểm công cộng cũng được bố trí WiFi để phục vụ nhu cầu kết nối Internet cho các thiết bị di động [2].

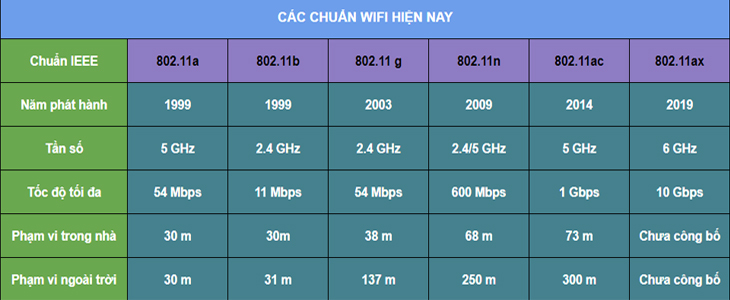
Hình 2.1. Giao thức WiFi

Để tạo được kết nối WiFi nhất thiết phải có Router (bộ thu phát), Router này lấy thông tin từ mạng Internet qua kết nối hữu tuyến rồi chuyển nó sang tín hiệu vô tuyến và gửi đi, bộ chuyển tín hiệu không dây (adapter) trên các thiết bị di động thu nhận tín hiệu này rồi giải mã nó sang những dữ liệu cần thiết. Quá trình này có thể thực hiện ngược lại, Router nhận tín hiệu vô tuyến từ Adapter và giải mã chúng rồi gửi qua Internet.

Một số chuẩn WiFi hiện nay:

Có rất nhiều chuẩn WiFi phổ biến hiện nay (hình …) . Tất cả các chuẩn WiFi trên Việt Nam đều có sử dụng. Tuy nhiên, hai chuẩn phổ biến nhất hiện nay là 802.11g và 802.11n và được sử dụng nhiều nhất vẫn là 802.11n, hoạt động ở 2 dải tần 2.4GHz và 5GHz.

Ngày nay một số thiết bị mới được sản xuất ở Việt Nam đã sử dụng các chuẩn 802.11ac, tuy nhiên số lượng này chưa nhiều (mặc dù ở các nước phát triển đã sử dụng rất phổ biến), một phần do chưa phù hợp với hạ tầng mạng còn hạn chế ở nước ta hiện nay.

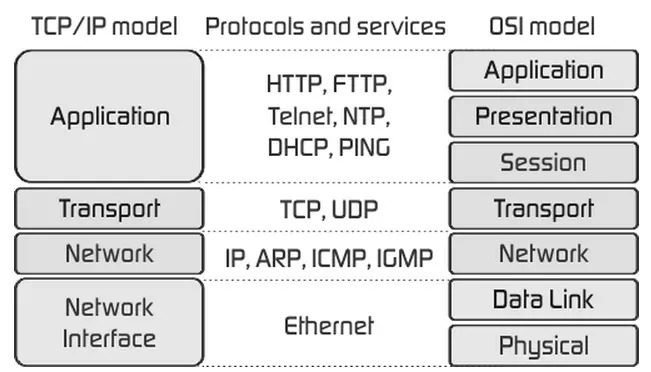


Hình 2.2. Các chuẩn Wifi hiện nay

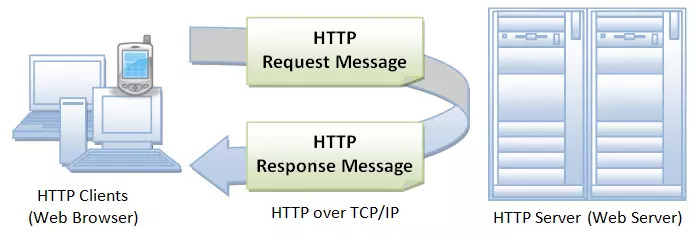
WiFi 6 là tiêu chuẩn mới nhất trong WiFi. Phiên bản mới nhất của tiêu chuẩn WiFi là 802.11ax (WiFi-6) và là bản nâng cấp so với tiêu chuẩn trước đó, là 802.11ac (WiFi-5). Tiêu chuẩn nâng cấp này, WiFi 6, chủ yếu dành cho các thiết bị tương thích (như bộ định tuyến) để truyền tín hiệu WiFi hiệu quả hơn. WiFi 6 được xây dựng để đáp ứng với số lượng thiết bị ngày càng tăng trên thế giới và để cải thiện hiệu suất trong mật độ mạng cao như căn hộ có nhiều bộ định tuyến hoặc sân vận động ngoài trời. Thuật ngữ WiFi 6 được liên minh WiFi đặt ra là một chỉ định của ngành và được coi là một tên thân thiện với người tiêu dùng đối với tên tiêu chuẩn công nghiệp của nó là 802.11ax.

* + 1. Giao thức HTTP

HTTP (HyperText Transfer Protocol – Giao thức truyền tải siêu văn bản) là một trong các giao thức chuẩn về mạng Internet, được dùng để liên hệ thông tin giữa Máy cung cấp dịch vụ (Web server) và Máy sử dụng dịch vụ (Web client), là giao thức Client/Server dùng cho World Wide Web – WWW[1]



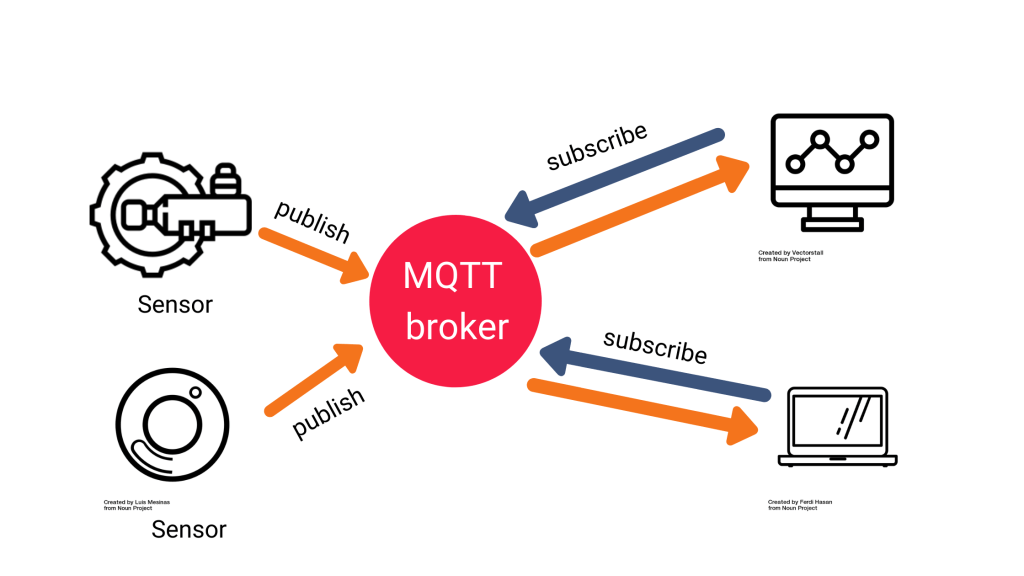
Hình 2.3. Giao thức HTTP.

Giao thức HTTP hoạt động dựa trên mô hình Client – Server. Thông thường khi các bạn lướt web, các máy tính của người dùng sẽ đóng vai trò làm máy khách (Client). Sau một thao tác nào đó của người dùng, các máy khách sẽ gửi yêu cầu đến máy chủ (Server) và chờ đợi câu trả lời từ những máy chủ này.

Hình 2.4. Cách thức hoạt động của giao thức HTTP

Ngoài ra, khi các hệ thống trao đổi dữ liệu với nhau, chúng cũng sử dụng giao thức này nhưng 2 bên đều là server.

* + 1. Giao thức MQTT

**MQTT là một giao thức truyền tải dữ liệu, sử dụng mô hình mạng Publish – Subscribe nhằm mục đính truyền dữ liệu giữa các thiết bị. Giao thức thường chạy qua TCP / IP. Tuy nhiên, bất kỳ giao thức mạng nào cung cấp các kết nối theo thứ tự, không mất dữ liệu, hai chiều đều có thể hỗ trợ MQTT. Nó được thiết kế cho các kết nối với các vị trí ở xa hoặc băng thông mạng bị hạn chế[3].

Hình 2.5. Giao thức MQTT.

* 1. Linh kiện và phần mềm thiết kế mạch
     1. Vi điều khiển ESP32

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

*Bộ xử lý:*

* CPU: Bộ vi xử lý Xtensa lõi kép (hoặc lõi đơn) 32-bit LX6, hoạt động ở tần số 240 MHz (160 MHz cho ESP32-S0WD và ESP32-U4WDH) và hoạt động ở tối đa 600  MIPS (200 MIPS với ESP32-S0WD/ESP32-U4WDH)
* Bộ đồng xử lý (co-processor) công suất cực thấp (Ultra low power, viết tắt: ULP)

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

*Bộ nhớ nội:*

* 448 KB bộ nhớ ROM cho việc booting và các tính năng lõi
* 520 KB bộ nhớ SRAM trên chip cho dữ liệu và tập lệnh

*Kết nối không dây:*

* Wi-Fi: 802.11 b/g/n
* Bluetooth: v4.2 BR/EDR và BLE (chia sẻ sóng vô tuyến với Wi-Fi)

*34 GPIO vật lý với các ngoại vi*

* ADC SAR 12 bit, 18 kênh
* DAC 2 × 8-bit
* 10 cảm biến cảm ứng (touch sensor) (GPIO cảm ứng điện dung)
* 3 SPI (SPI, HSPI và VSPI) hoạt động ở cả 2 chế độ master/slave.[8] Module ESP32 hỗ trợ 4 ngoại vi SPI với SPI0 và SPI1 kết nối đến bộ nhớ flash của ESP32 còn SPI2 và SPI3 tương ứng với HSPI và VSPI.[9]
* 2 I²S
* 2 I²C, hoạt động được ở cả chế độ master và slave, với chế độ Standard mode (100 Kbit/s) và Fast mode (400 Kbit/s). Hỗ trợ 2 chế độ định địa chỉ là 7-bit và 10-bit.Các GPIO đều có thể được dùng để triển khai I²C.
* 3 UART (UART0, UART1, UART2) với tốc độ lên đến 5 Mbps[10]
* SD/SDIO/CE-ATA/MMC/eMMC host controller
* SDIO/SPI slave controller
* Ethernet MAC interface cho DMA và IEEE 1588 Precision Time Protocol (tạm dịch: Giao thức thời gian chính xác IEEE 1588)
* CAN bus 2.0
* Bộ điều khiển hồng ngoại từ xa (TX/RX, lên đến 8 kênh)
* PWM cho điều khiển động cơ
* LED PWM (lên đến 16 kênh)
* Cảm biến hiệu ứng Hall
* Bộ tiền khuếch đại analog công suất cực thấp (Ultra low power analog pre-amplifier)

*Bảo mật:*

* Hỗ trợ tất cả các tính năng bảo mật chuẩn IEEE 802.11, bao gồm WFA, WPA/WPA2 và WAPI.
* Secure boot (tạm dịch: khởi động an toàn)
* Mã hóa flash
* 1024-bit OTP, lên đến 768-bit cho khách hàng
* Tăng tốc mã hóa phần cứng: AES, SHA-2, RSA, elliptic curve cryptography (ECC, tạm dịch: mật mã đường cong ellip), trình tạo số ngẫu nhiên (random number generator)

*Quản lý năng lượng*

* Bộ ổn áp nội với điện áp rơi thấp (internal low-dropout regulator)
* Miền nguồn riêng (individual power domain) cho RTC
* Dòng 5 μA cho chế độ deep sleep
* Trở lại hoạt động từ ngắt GPIO, timer, đo ADC, ngắt với cảm ứng điện dung[4].
  + 1. IC cách ly nguồn B5050S

IC cách ly nguồn B0505S được thiết kế cho ứng dụng yêu cầu đầu ra cách ly khỏi hệ thống điện, giúp giảm tối đa nhiễu từ các nguồn điện khác như dòng ngược, điện áp ngược gây ra[5].

Một số đặc điểm chính của B0505S:

* Hiệu quả lên tới 80%
* Sử dụng chuẩn đóng gói SIP/DIP
* Nhiệt độ hoạt động: -40°C ~ +85°C

Bảng 2.1. Thông số ngõ vào ra của B5050S

|  |  |  |
| --- | --- | --- |
| Đầu vào | Đầu ra | |
| Điện áp đầu vào  4.5V – 5V | Điện áp đầu ra | 5.0 ±0.2V |
| Dòng điện | 20mV –200mV |

* + 1. Mạch Buck LM2596

Mạch giảm áp DC-DC Buck LM2596 3A có kích thước nhỏ gọn có khả năng giảm áp từ 30VDC xuống 1.5VDC mà vẫn đạt hiệu suất cao (92%), thích hợp cho các ứng dụng chia nguồn, hạ áp…

Thông số kỹ thuật:

* Điện áp đầu vào: Từ 3V đến 30V.
* Điện áp đầu ra: Điều chỉnh được trong khoảng 1.5V đến 30V.
* Dòng đáp ứng tối đa là 3A.
* Hiệu suất : 92%
* Công suất : 15W
* Kích thước: 45 (dài) \* 20 (rộng) \* 14 (cao) mm

Hình 2.6: Module giảm áp LM2596

Khi dòng điện đầu ra giữ lớn hơn 2,5A (hoặc công suất đầu ra lớn hơn 10W), cần phải gán tản nhiệt để mạch chạy ổn định

Do bộ chuyển đổi LM2596 là nguồn điện chế độ chuyển đổi, hiệu quả của nó cao hơn đáng kể so với các bộ điều chỉnh tuyến tính ba đầu phổ biến, đặc biệt là với điện áp đầu vào cao hơn. LM2596 hoạt động ở tần số chuyển mạch là 150 kHz

* + 1. Transistor C1815

Transistor C1815 là loại Transistor ngược (NPN) được sử dụng cho mục đích khuếch đại hoặc là đóng cắt các tải. Một số thông số chính của Transistor được nêu lên ở bảng 2.2

Bảng 2.2: Các thông số của transistor C1815

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ký hiệu | Thông số | Điều kiện thử nghiệm | Min | Max |
| ICBO | Dòng cắt Collector | VCB=60V, IE=0 |  | 0.1µA |
| IEBO | Dòng cắt Emitter | VEB=5V, IC=0 |  | 0.1µA |
| hFE1 hFE2 | Hệ số khuyếch đại DC | VCE=6V, IC=2mA VCE=6V, IC=150mA | 70  25 | 700 |
| VCE (sat) | Điện áp bão hòa Collector-Emitter | IC=100mA, IB=10mA |  | 0.25V |
| VBE (sat) | Điện áp bão hòa Base-Emitter | IC=100mA, IB=10mA |  | 1V |

* + 1. Diode 1N4148

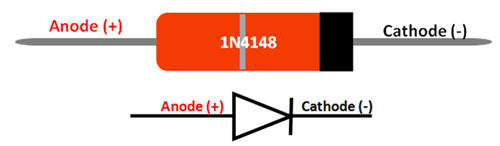
1N4148 là một diode chuyển mạch (switching diode). Đây là một trong những diode chuyển mạch phổ biến và tồn tại lâu dài nhất vì các thông số kỹ thuật đáng tin cậy và chi phí thấp. 1N4148 rất hữu ích trong việc chuyển đổi các ứng dụng lên tới khoảng 100 MHz với thời gian phục hồi ngược không quá 4 ns.

Là diode chuyển mạch được sản xuất khối phổ biến nhất, 1N4148 đã thay thế 1N914 cũ hơn. Chúng khác nhau chủ yếu trong đặc điểm kỹ thuật .Tuy nhiên, ngày nay hầu hết các nhà sản xuất đều liệt kê các thông số kỹ thuật phổ biến.

Một số thông số của 1N4148

* Điện áp ngược VRRM = 100 V
* Dòng chỉnh lưu trung bình IO = 200 mA
* Dòng điện thuận IF = 300 mA
* dòng chuyển tiếp đỉnh định kỳ If = 400 mA
* Điện áp chuyển tiếp tối đa VF = 1 V ở 10 mA
* Điện áp phân hủy tối thiểu và dòng rò ngược VR = 75 V ở 5 μA; 100 V ở 100 μA
* Thời gian phục hồi ngược tối đa Trr = 4 ns

Hình 2.7. Diode 1N4148.



* + 1. Relay 5v

Bảng 2.3. Các thông số của Relay 5V

|  |  |  |
| --- | --- | --- |
| Cuộn hút | Điện áp cuộn hút | 5V |
| Dòng điện qua cuộn hút | 89.3 mA |
| Điện áp Pull-In | 75% Max. |
| Điện áp Drop - Out | 10% Min. |
| Tiếp điểm | Dòng điện định mức | 10A |
| Điện áp định mức | 250V |

* + 1. Các linh kiện thụ động
* Tụ điện: Tụ điện là một loại linh kiện điện tử thụ động, là một hệ hai vật dẫn và ngăn cách nhau bởi một lớp cách điện. 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.
* 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.
* Led 3mm: Chiều dài Pin: 16mm, đường kính: 3mm, dòng tiêu thụ: 5mA - 18mA, điện áp tham chiếu: Đỏ, Vàng: 1.8V - 2.2V, X.Lá – X.Dương: 2.1V – 2.6V
  + 1. Phần mềm thiết kế Altium Designer

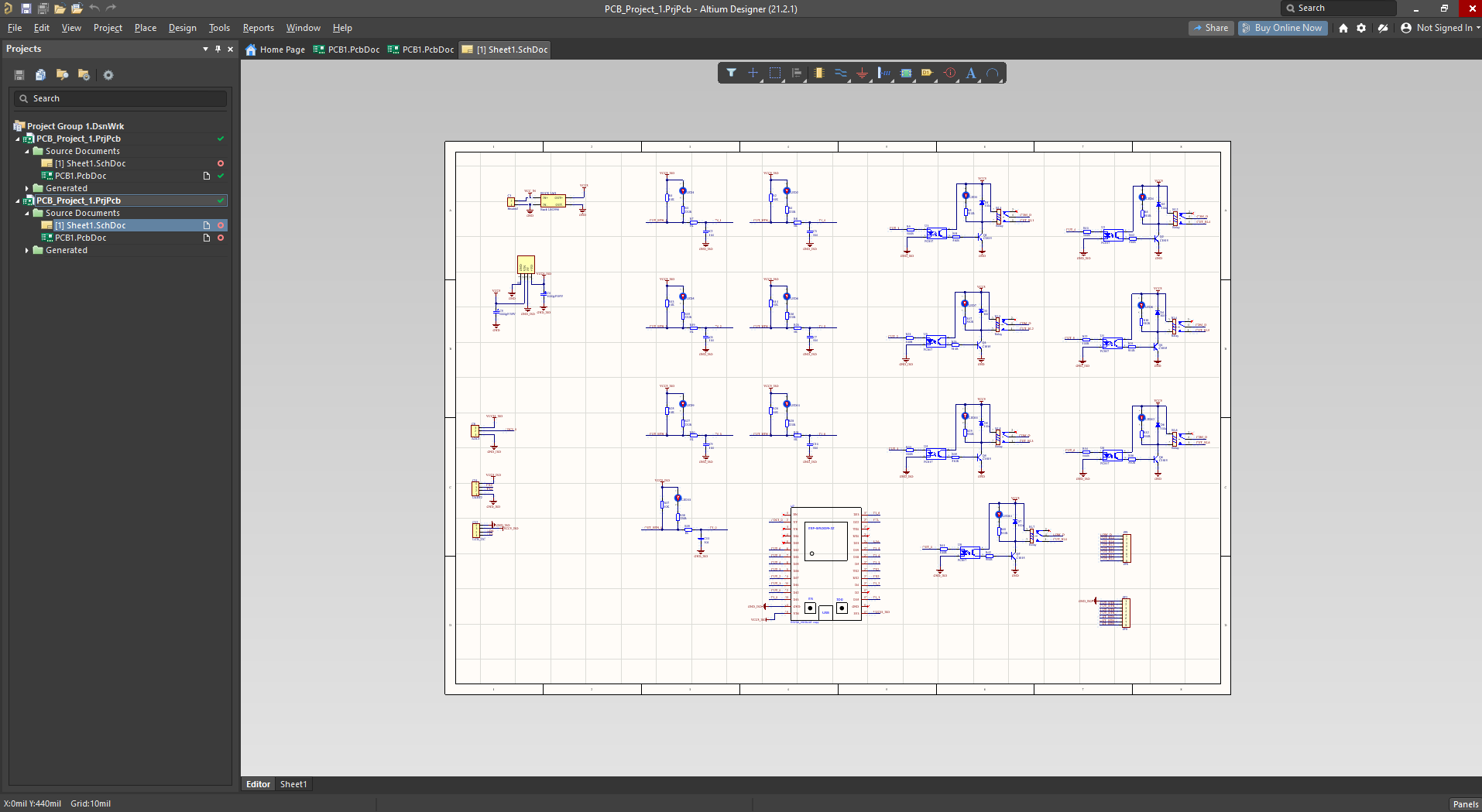
Hình 2.8: Altium Designer 2022

Altium Designer trước kia có tên gọi quen thuộc là Protel DXP, là một trong những công cụ vẽ mạch điện tử mạnh nhất hiện nay. Được phát triển bởi hãng Altium Limited. Altium designer là một phần mềm chuyên nghành được sử dụng trong thiết kế mạch điện tử. Nó là một phần mềm mạnh với nhiều tính năng thú vị, tuy nhiên phần mềm này còn được ít người biết đến so với các phần mềm thiết kế mạch khác như Orcad hay Proteus

Altium Designer có một số đặc trưng sau:

* Giao diện thiết kế, quản lý và chỉnh sửa thân thiện, dễ dàng biên dịch, quản lý file, quản lý phiên bản cho các tài liệu thiết kế.
* Hỗ trợ mạnh mẽ cho việc thiết kế tự động, đi dây tự động theo thuật toán tối ưu, phân tích lắp ráp linh kiện. Hỗ trợ việc tìm các giải pháp thiết kế hoặc chỉnh sửa mạch, linh kiện, netlist có sẵn từ trước theo các tham số mới.
* Mở, xem và in các file thiết kế mạch dễ dàng với đầy đủ các thông tin linh kiện, netlist, dữ liệu bản vẽ, kích thước, số lượng…
* Hệ thống các thư viện linh kiện phong phú, chi tiết và hoàn chỉnh bao gồm tất cả các linh kiện nhúng, số, tương tự…
* Đặt và sửa đối tượng trên các lớp cơ khí, định nghĩa các luật thiết kế, tùy chỉnh các lớp mạch in, chuyển từ schematic sang PCB, đặt vị trí linh kiện trên PCB.
* Mô phỏng mạch PCB 3D, đem lại hình ảnh mạch điện trung thực trong không gian 3 chiều, hỗ trợ MCAD-ECAD, liên kết trực tiếp với mô hình STEP, kiểm tra khoảng cách cách điện, cấu hình cho cả 2D và 3D
* Hỗ trợ thiết kế PCB sang FPGA và ngược lại.

Từ đó, chúng ta thấy Altium Designer có nhiều điểm mạnh so với các phần mềm khác như đặt luật thiết kế, quản lý đề tài mô phỏng dễ dàng, giao diện thân thiện,…



Hình 2.9. Giao diện phần mềm Altium Designer 22

Việc thiết kế mạch điện tử trên phần mềm altium designer có thể được tóm tắt gồm các bước như sau:

* Đặt ra các yêu cầu bài toán.
* Lựa chọn linh kiện.
* Thiết kế mạch nguyên lý.
* Lựa chọn các chân linh kiện để chuyển sang mạch in Update mạch nguyên lý sang mạch in.
* Lựa chọn kích thước mạch in Sắp sếp các vị trí các loại linh kiện như điện trở , tụ điện, IC...
* Đặt kích thước các loại dây nối.
* Đi dây trên mạch.
* Kiểm tra toàn mạch.
  1. Phần mềm của hệ thống
     1. HTML (HyperText Markup Language).

HTML là viết tắt của cụm từ Hypertext Markup Language (tạm dịch là Ngôn ngữ đánh dấu siêu văn bản). HTML được sử dụng để tạo và cấu trúc các thành phần trong trang web hoặc ứng dụng, phân chia các đoạn văn, heading, titles, blockquotes… và HTML không phải là ngôn ngữ lập trình.

Một tài liệu HTML được hình thành bởi các phần tử HTML (HTML Elements) được quy định bằng các cặp thẻ (tag và attributes). Các cặp thẻ này được bao bọc bởi một dấu ngoặc ngọn (ví dụ <html>) và thường là sẽ được khai báo thành một cặp, bao gồm thẻ mở và thẻ đóng. Ví dụ, chúng ta có thể tạo một đoạn văn bằng cách đặt văn bản vào trong cặp tag mở và đóng văn bản <p> và </p> :

* + 1. CSS (Cascading Style Sheets)

CSS là chữ viết tắt của Cascading Style Sheets, nó là một ngôn ngữ được sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi các ngôn ngữ đánh dấu (HTML). Nói ngắn gọn hơn là ngôn ngữ tạo phong cách cho trang web. Bạn có thể hiểu đơn giản rằng, nếu HTML đóng vai trò định dạng các phần tử trên website như việc tạo ra các đoạn văn bản, các tiêu đề, bảng, …thì CSS sẽ giúp chúng ta có thể thêm style vào các phần tử HTML đó như đổi bố cục, màu sắc trang, đổi màu chữ, font chữ, thay đổi cấu trúc…

CSS được phát triển bởi W3C (World Wide Web Consortium) vào năm 1996, vì HTML không được thiết kế để gắn tag để giúp định dạng trang web.

Phương thức hoạt động của CSS là nó sẽ tìm dựa vào các vùng chọn, vùng chọn có thể là tên một thẻ HTML, tên một ID, class hay nhiều kiểu khác. Sau đó là nó sẽ áp dụng các thuộc tính cần thay đổi lên vùng chọn đó.

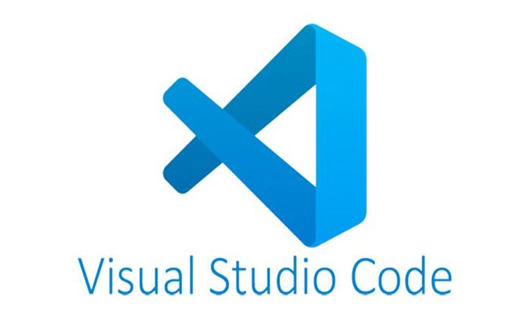
Mối tương quan giữa HTML và CSS rất mật thiết. HTML là ngôn ngữ markup (nền tảng của site) và CSS định hình phong cách (tất cả những gì tạo nên giao diện website), chúng là không thể tách rời.

* + 1. JavaScript

JavaScript là ngôn ngữ lập trình phổ biến nhất trên thế giới trong suốt 20 năm qua. Nó cũng là một trong số 3 ngôn ngữ chính của lập trình web:

* HTML: Giúp bạn thêm nội dung cho trang web.
* CSS: Định dạng thiết kế, bố cục, phong cách, canh lề của trang web.
* JavaScript: Cải thiện cách hoạt động của trang web.

JavaScript có thể học nhanh và dễ dàng áp dụng cho nhiều mục đích khác nhau, từ việc cải thiện tính năng của website đến việc chạy game và tạo phần mềm nền web. Hơn nữa, có hàng ngàn mẫu template JavaScript và ứng dụng ngoài kia, nhờ vào sự cống hiến của cộng đồng, đặc biệt là Github.

* + 1. Phần mềm Visual Studio code

Hình 2.10: Visual Studio Code

Là một trình biên tập lập trình code miễn phí dành cho Windows, Linux và macOS, Visual Studio Code được phát triển bởi Microsoft. Nó được xem là một sự kết hợp hoàn hảo giữa IDE và Code Editor.

Visual Studio Code hỗ trợ chức năng debug, đi kèm với Git, có syntax highlighting, tự hoàn thành mã thông minh, snippets, và cải tiến mã nguồn. Nhờ tính năng tùy chỉnh, Visual Studio Code cũng cho p hép người dùng thay đổi theme, phím tắt, và các tùy chọn khác.

Một số tính năng của phần mềm:

*Hỗ trợ nhiều ngôn ngữ lập trình*

Visual Studio Code hỗ trợ nhiều ngôn ngữ lập trình như C/C++, C#, F#, Visual Basic, HTML, CSS, JavaScript, … Vì vậy, nó dễ dàng phát hiện và đưa ra thông báo nếu chương chương trình có lỗi.

Các trình viết code thông thường chỉ được sử dụng hoặc cho Windows hoặc Linux hoặc Mac Systems. Nhưng Visual Studio Code có thể hoạt động tốt trên cả ba nền tảng trên.

*Cung cấp kho tiện ích mở rộng*

Trong trường hợp lập trình viên muốn sử dụng một ngôn ngữ lập trình không nằm trong số các ngôn ngữ Visual Studio hỗ trợ, họ có thể tải xuống tiện ích mở rộng. Điều này vẫn sẽ không làm giảm hiệu năng của phần mềm, bởi vì phần mở rộng này hoạt động như một chương trình độc lập

* + 1. Phần mềm Arduino IDE

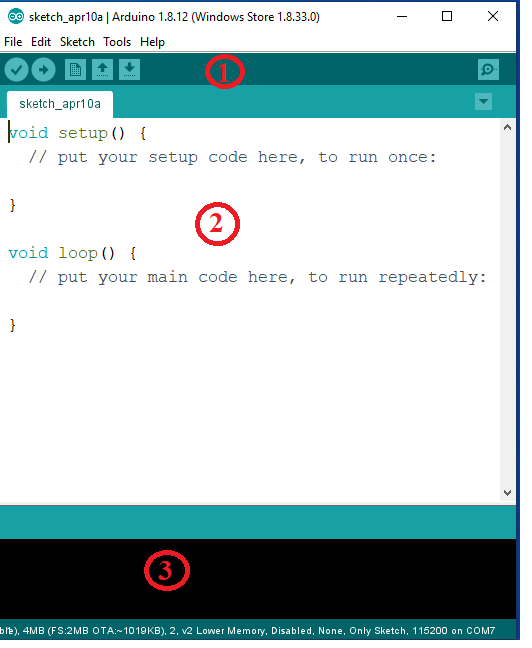
Sử dụng phần mềm Arduino IDE để lập trình cho ESP32



Hình 2.11. Arduino IDE

Arduino IDE là môi trường để lập trình và nạp code cho các dòng Arduino. Arduino IDE được xây dựng trên miền nền tảng Java nên hỗ trợ hầu hết các hệ điều hành hiện nay.

Giao diện của Arduino IDE được chia thành 3 vùng chính:



Hình 2.12. Giao diện phần mềm Arduino IDE

Vùng 1: Các phím chức năng được nêu như bảng …:

Bảng 2.4. Chức năng các phím trên giao diện Arduino IDE

|  |  |  |
| --- | --- | --- |
| Icon |  | Chức năng |
|  |  | Biên dịch chương trình soạn thảo để kiểm tra các lỗi. |
|  |  | Biên dịch và upload chương trình đang soạn thảo. |
|  |  | Mở một trang soạn thảo mới. |
|  |  | Mở các chương trình đã lưu. |
|  |  | Lưu chương trình đang soạn thảo. |
|  |  | Mở cửa sổ Serial Monitor để gửi và nhận dữ liệu giữa máy tính và board vi điều khiển. |

Vùng 2: Cửa sổ để viết chương trình

* Chương trình Code sẽ được viết tại đây. Ở đây có hai hàm quan trọng là setup() và loop().
* Hàm setup() được khởi chạy một lần duy nhất. Chức năng của hàm này dùng để khởi tạo các biến, khai báo chức năng các chân, khởi tạo các thông số bán đầu
* Hàm loop() là nơi chương trình được chạy lặp đi lặp lại đến khi ngắt vi điều khiển.
* Các dấu “//” dùng để tạo chú thích, giúp cho việc đọc code được dễ dàng hơn
* Trong lập trình có phân biệt ký tự hoa, thường, tuyệt đối phải đánh chích xác, đồng thời cuối mỗi câu lệnh cần phải có dấu chấm phẩy (;), trừ lệnh khai báo thư viện.

Vùng 3: Hiển thị các thông tin liên quan đến chương trình

Là cửa sổ để hiển thị về việc build chương trình, nạp chương trình thành công xuống vi điều khiển và các cảnh báo khác liên quan đến chương trình và điều khiển của chúng ta. Lưu ý, mọi thông báo và trạng thái của cả quá trình viết chương trình (write code), xây dựng chương trình (build code) và nạp chương trình (program code) đều được hiển thị tại đây. Cửa sổ này được gọi là cửa sổ debug.

* + 1. Hệ điều hành FreeRTOS

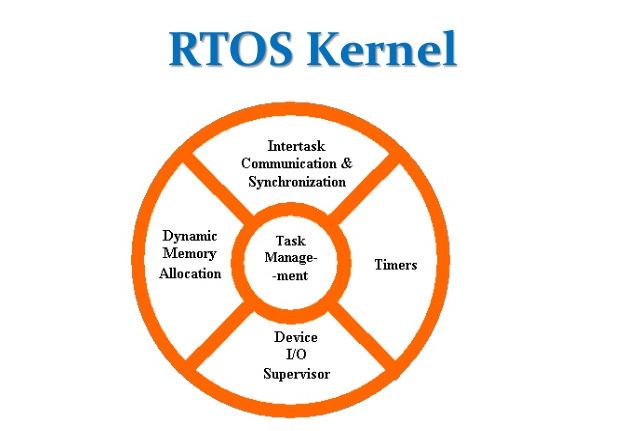
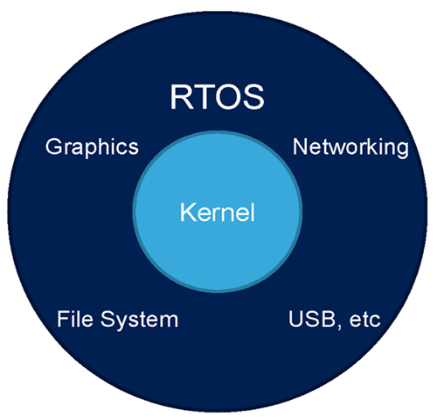
FreeRTOS là một hệ điều hành nhúng thời gian thực (Real Time Operating System) mã nguồn mở được phát triển bởi Real Time Engineers Ltd, sáng lập và sở hữu bởi Richard Barry[6].



Hình 2.13. Hệ điều hành FreeRTOS

FreeRTOS được thiết kế phù hợp cho nhiều hệ nhúng nhỏ gọn vì nó chỉ triển khai rất ít các chức năng như: cơ chế quản lý bộ nhớ và tác vụ cơ bản, các hàm API quan trọng cho cơ chế đồng bộ. Nó không cung cấp sẵn các giao tiếp mạng, drivers, hay hệ thống quản lý tệp (file system) như những hệ điều hành nhúng cao cấp khác.

Hình 2.14. Chức năng của hệ điều hành FreeRTOS



Tuy vậy, FreeRTOS có nhiều ưu điểm, hỗ trợ nhiều kiến trúc vi điều khiển khác nhau, kích thước nhỏ gọn (4.3 Kbytes sau khi biên dịch trên Arduino), được viết bằng ngôn ngữ C và có thể sử dụng, phát triển với nhiều trình biên dịch C khác nhau (GCC, OpenWatcom, Keil, IAR, Eclipse, …), cho phép không giới hạn các tác vụ chạy đồng thời, không hạn chế quyền ưu tiên thực thi, khả năng khai thác phần cứng.

Ngoài ra, nó cũng cho phép triển khai các cơ chế đồng bộ giữa các tiến trình như: Queues, Counting Semaphore, Mutexes.

* 1. Kết luận chương 2

Chương 2 nói về việc sử dụng các thiết bị, các phần mềm và các công cụ đã dùng trong quá trình hoàn thành sản phẩm và các kiến trúc tổng quan phục vụ cho việc giải bài toán được đặt ra. Cùng với đó, chương 2 giới thiệu cũng như cách sử dụng các công cụ để lập trình và debug chương trình vi điều khiển ESP32. Ngoài ra chương 2 cũng giới thiệu về các công cụ lập trình bên phía Webserver như HTML, CSS, JavaScript.

# THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT

* 1. Thiết kế sơ đồ khối hệ thống điều khiển và giám sát
     1. Yêu cầu của hệ thống.

Thiết kế một hệ thống điều khiển đèn và rèm cửa thông minh. Sử dụng cảm biến ánh sáng phát hiện khi ban ngày thì đèn bật, ban đêm thì đèn tắt. Rèm cửa điều khiển bằng giao diện Webserver

Hệ thống hoàn thiện đảm bảo các chức năng bật tắt bằng nút nhấn và bật tắt bằng giao diện Webserver. Chức năng bật nút nhấn phải đồng bộ ngay lập tức với giao diện Webserver.

* + 1. Sơ đồ khối

*Sơ đồ gửi nhận dữ liệu giữa mạch ESP32 và điện thoại/máy tính*

Hình 3.1. Sơ đồ gửi nhận dữ liệu.

**MQTT Broker**

**Mạch ESP32**

**Pub**

**Sub**

**Pub**

**Sub**

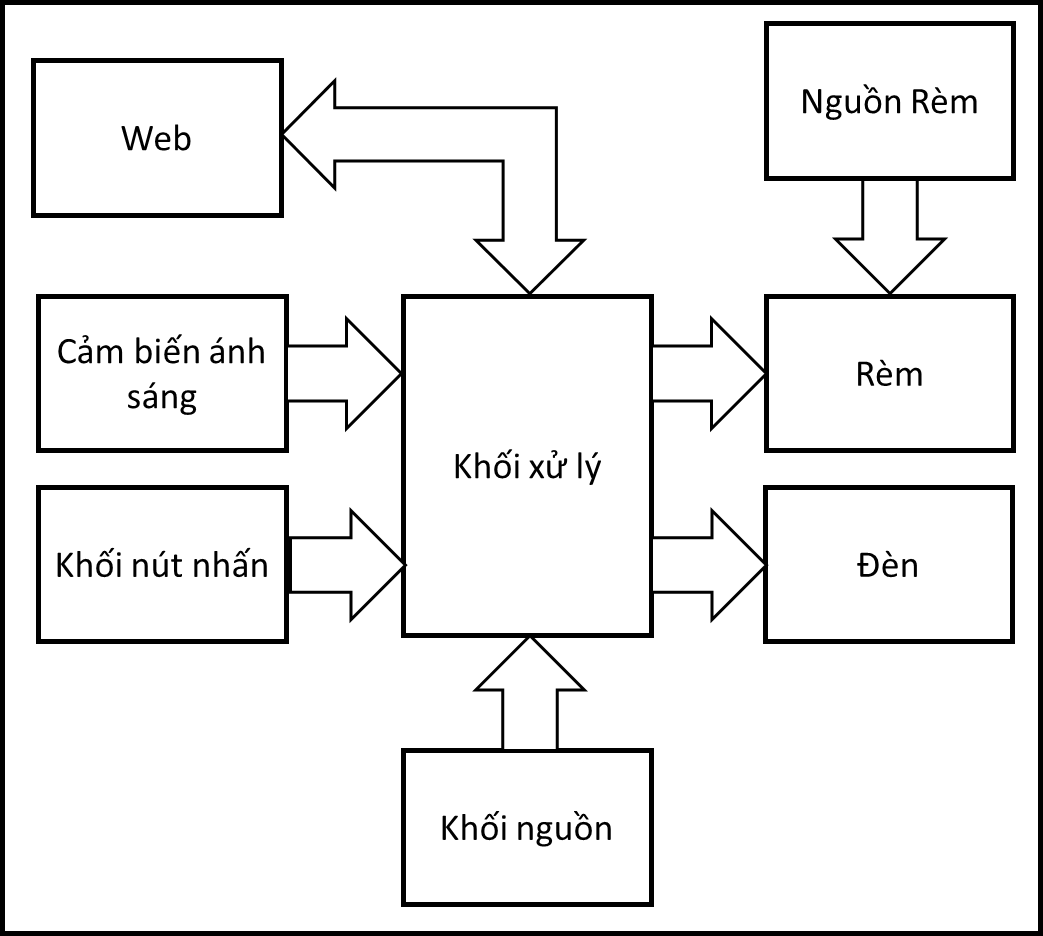
**Máy tính**

**Điện thoại**

MQTT Broker được sử dụng trong đề tài được lấy miễn phí tại trang: <https://www.hivemq.com/public-mqtt-broker/> với:

* Tên broker: Broker: broker.hivemq.com
* TCP Port: 1883
* Websocket Port: 8000
* Không có tên tài khoản và mật khẩu

MQTT broker sẽ làm nhiệm vụ trung gian truyền nhận các tín hiệu nút nhấn tới phần mềm hoặc truyền nhận tín hiệu từ phần mềm tới máy tính với cơ chế Publish và Subscribe như hình …

*Sơ đồ khối của mạch ESP32*

Hình 3.2. Sơ đồ khối của hệ thống.

Mạch ESP32 điều khiển các thiết bị bao gồm 6 khối chính:

* Khối nguồn 1: Cấp nguồn cho relay, đèn và khối xử lý trung tâm
* Khối nguồn 2: Cấp nguồn cho khối điều khiển rèm và rèm
* Khối nút nhấn: Bao gồm 6 nút nhấn để làm tín hiệu đầu vào, có thể dùng để điều khiển rèm đóng, rèm mở hoặc bật tắt đèn
* Khối xử lý trung tâm: Chứa vi điều khiển ESP32, xử lý các tín hiệu gửi đi, nhận về từ MQTT Broker, nhận các tín hiệu từ nút nhấn, xuất ra các tín hiệu để bật tắt relay, đóng rèm hoặc mở rèm
* Khối relay: điều khiển thiết bị 1 chiều hoặc xoay chiều thông qua tiếp điểm thường mở
* Khối điều khiển rèm: Chứa mạch L298N dùng để đảo chiều động cơ rèm, điều khiển rèm đóng hoặc mở
  1. Sơ đồ nguyên lý

Hình 3.3. Mạch nguyên lý của hệ thống



* + 1. Khối xử lý trung tâm

Khối xử lý trung tâm gồm có 1 vi xử lý chính là ESP32. ESP32 được sử dụng trong mạch ở dạng Kit cắm có tên là ESP32 Devkit V1.

Hình 3.4: Khối xử lý trung tâm

Khối xử lý trung tâm điều khiển mọi hoạt động của mạch bao gồm: Điều khiển cơ cấu cấp hành, nhận các tín hiệu từ nút nhấn vật lý. Ngoài việc xử lý các tín hiệu ở tầng vật lý thì còn gửi và nhận tín hiệu thông qua MQTT Broker dưới dạng Publish và Subscrice từ máy tính hoặc các thiết bị khác điều khiển tới để điều khiển các cơ cấu chấp hành

Khối xử lý trung tâm – ESP32 xử lý 6 tín hiệu đầu vào từ nút nhấn, 5 tín hiệu đầu ra rơ le và 1 PWM để điểm khiển động cơ.

Ngoài ra, ESP32 trên mạch còn có thể xử lý các tín hiệu néu muốn ghép nối như: ADC, I2C, USART …

.

* + 1. Khối nguồn và ổn áp nguồn

Mạch sử dụng nguồn điện 12V cấp vàp Module nguồn LM2569. Module nguồn LM2569 có tác dụng biến đổi nguồn cấp 12V về 5V để cấp cho Module ESP32.

* + 1.  Khối nút nhấn

Hình 3.5: Khối nút nhấn

Khối nút nhấn được gắn một điện trở kéo lên có giá trị 10K, ngoài ra còn có các Led 3mm báo tín hiệu.

Ngoài ra mạch còn được trang bị thêm mạch lọc thông cao RC nhằm tránh việc nút nhấn bị nhiễu khi nhấn

* + 1. Khối Relay

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

Khối Relay được điều khiển thông qua Transistor C1815 để điều khiển cuộn hút Relay. Khi chân B của Transistor ở mức 1 thì cuộn hút có điện sẽ nối chân C và E với nhau, khi đó đèn LED5 sẽ sáng và cuộn hút của Relay sẽ có diện, thường mở của relay sẽ đóng lại điều khiển các đèn và các thiết bị.



Hình 3.6. Khối Relay

Ngoài ra, mạch sử dụng thêm các diode (1N4148) mắc song song và mắc ngược với cuộn hút của Relay. Mục đích của việc làm này nhằm ngăn chặn sự tăng đột biến điện áp lớn phát sinh khi nguồn điện bị ngắt (gọi là điện áp ngược) bảo vệ cho Relay và mạch điều khiển.

Hình 3.7: Khối mở rộng

* + 1. Khối mở rộng

Bao gồm 3 header để mở rộng thêm cho mạch:

VCC5: mở rộng nguồn 5V để cấp cho các thiết bị sử dụng nguồn 5v

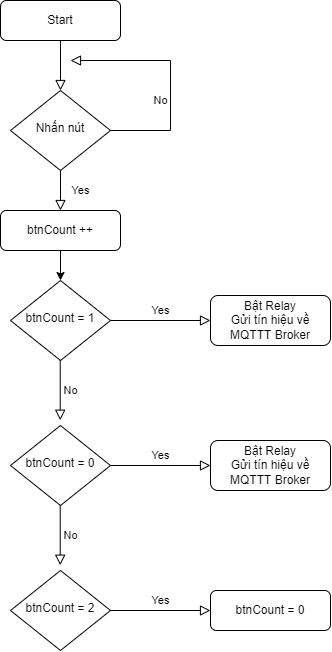
VCC12: mở rộng nguồn 12V để cấp cho các thiết bị sử dụng nguồn 12v

I2C: Mở rộng giao tiếp I2C để chủ yếu để giao tiếp với màn hình LCD I2C

* 1. Sơ đồ mạch in (PCB)



Hình 3.8. Sơ đồ mạch in 2D

* 1. Lập trình vi xử lý ESP32
     1. Các thư viện được sử dụng
* Thư viện Arduino.h cung cấp các hàm dạng Arduino cho vi điều khiển ESP32: #include <Arduino.h>
* Thư viện PubSubClient cung cấp các hàm về MQTT để nhận về và gửi các dữ liệu lên MQTT Broker: #include <PubSubClient.h>
* Thư viện Wifi.h cung cấp các hàm về kết nối WiFi cho ESP32:   
  #include <WiFi.h>
  + 1. Xử lý sự kiện nhấn nút:

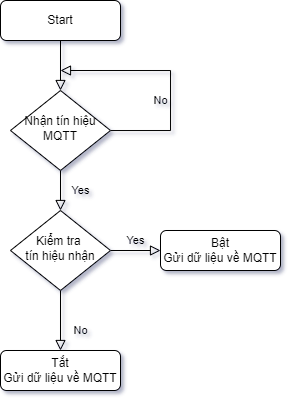
Hình 3.9. Quá trình sử lý nút nhấn.

Khi nhấn nút:

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

Quá trình được mô tả như hình …

* Nếu btnCount = 0 thì tắt rơ le
* Nếu btnCount = 1 thì bật rơ le
* Nếu btnCount = 2 thì reset lại trạng thái của btnCount về 0
  + 1. Xử lý sự kiện nhận về từ MQTT Broker



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

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

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

Hình 3.10. Quá trình xử lý sự kiện nhận về MQTT Broker,

* 1. Lập trình Webserver

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

*Sử dụng HTML dựng khung giao diện:*

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

<!DOCTYPE html>

<html>

<head>

<title>Page Title</title>

</head>

<body>

<h1>This is a Heading</h1>

<p>This is a paragraph.</p>

</body>

</html>

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

|  |  |
| --- | --- |
| Tên thẻ | Chức năng |
| <h1> đến <h6> | Tạo heading 1 |
| <p> | Tạo một đoạn văn bản |
| <button> | Tạo một nút nhấn |
| <div> | Chia các đoạn HTML |

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

Vd: thay đổi màu và định dạng chữ thẻ <p>

p {

color: red;

text-align: center;

}

Thay đổi nền thẻ <body>

body {

background-color: lightblue;

}

* 1. Mô hình thực tế của hệ thống.

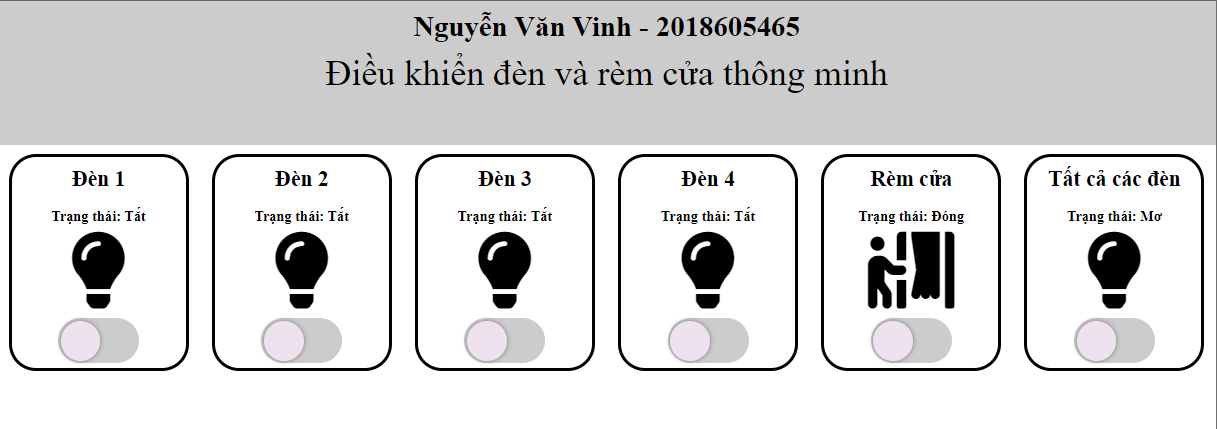


Hình 3.11. Hình ảnh thực tế của hệ thống.

Mô hình đèn và rèm cửa thông minh có 4 phần chính:

* Thứ nhất là phần mạch điều khiển toàn hệ thống được gắn là bên phải của mô hình.
* Phần thứ 2 phần động cơ có chức năng thu kéo rèm cửa.
* Thứ 3 là đèn được gắn trên mô hình, được điều khiển thông qua các nút nhấ trên mạch điều khiển.
* Cuối cùng là rèm cửa được điều khiển bởi động cơ.
  1. Giao diện điều khiển hệ thống trên Web

Giao diện điều khiển trên web bao gồm 6 nút nhấn để bật tắt và điều khiển rèm:



Hình 3.12: Giao diện điều khiển trên Web

* 1. Kết luận chương 3.

Hoàn thành việc thiết kế mạch nguyên ly cho toàn hệ thống. Xuất mạch in PCB và thực hiện bước tiếp theo hàn mạch hoàn thành hộp điều khiển hệ thống. Thiết kế phần mềm điều khiển bằng cách vẽ lưu đồ thuật toàn của các chế độ hoạt động.

Việc thiết kế mạch thực tế phải tính tới các phần cách ly để không bị nhiễu tới các khối khác, đặc biệt là khối vi điều khiển. Mạch chủ yếu lắp trong các khu vực ngoài trời, vì vậy mạch phải được thiết kế để chống chịu trong môi trường khắc nhiệt như: Độ ẩm cao, có nhiều xung gây nhiễu, ảnh hưởng bởi thời tiết.

Việc lập trình vi điều khiển yêu cầu người lập trình phải nắm rõ các kiến thức về ngôn ngữ lập trình C, kiến thức về các ngoại vi cơ bản của vi điều khiển, xử lý dữ liệu để đưa ra tín hiệu điều khiển…Việc lập trình với Sever yêu cầu người lập trình phải nắm rõ các kiến thức về Python, về công cụ mình đang sử dụng. Ngoài ra việc sử dụng thành thạo các công cụ debug, các công cụ kiểm soát và lập trình rất quan trọng để debug trong quá trình sảy ra lỗi. Việc lập trình và đưa trang web lên websever yêu cầu người lập trình phải nắm vững các kiến thức về web cơ bản, xử lý sự kiện trên websever, thao tác với giao diện điều khiển …

Tạo ra một một giao diện Web đơn giản không quá phức tạp, đầy đủ các tính năng giúp người sử dụng có thể hiểu và nắm bắt cách sử dụng một cách nhanh chóng.

# KẾT LUẬN

Qua quá trình thực hiện làm đồ án, em đã trình bày các cơ sở lý thuyết liên quan và chạy thành công hệ thống “**Thiết kế mô hình hệ thống đèn và rèm cửa thông minh sử dụng ESP32”**. Tuy thời gian làm đồ án thực sự không quá dài nhưng được sự giúp đỡ tận tình của Th.S Duơng Thị Hằng cùng với sự nỗ lực và cố gắng của bản thân, sự chỉ bảo của các Thầy Cô trong khoa Điện tử em đã hoàn thành đề tài theo yêu cầu và đúng thời gian quy định với những nội dung sau:

* Nghiên cứu và tìm hiểu về các hệ thống điều khiển và giám sát trên thực tế, ưu điểm và nhược điểm của từng hệ thống
* Nghiên cứu và tìm hiểu về vi điều khiển ESP32, các ngoại vi và lập trình ESP32
* Tìm hiểu về chuẩn truyền thông không dây WiFi, ứng dụng nó vào trong việc điều khiển, truyền nhận tín hiệu và giám sát
* Nắm được các hệ thống điều khiển và giám sát từ xa hoạt động
* Xây dựng được giao diện điều khiển và giám sát
* Thiết kế và vận hành thành công mô hình hệ thống đèn và rèm cửa thông minh sử dụng ESP32

Hướng phát triển đề tài:

* Tạo giao diện phong phú và đa dạng hơn
* Giám sát và điều khiển các thiết bị không chỉ ON/OFF mà cả các tín hiệu nhiều dạng khác nhau
* Phát triển đa nền tảng (Android, IOS…)

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

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

Em xin chân thành cảm ơn!

# TÀI LIỆU THAM KHẢO.

[1] Wikipedia. 2021. “Hypertext Transfer Protocol”. Truy cập ngày 18/4. <https://vi.wikipedia.org/wiki/Hypertext_Transfer_Protocol>. (Wikipedia 2021).

[2] Wikipedia. 2021. “Giao thức Wifi”. Truy cập ngày 20/4. <https://vi.wikipedia.org/wiki/Wi-Fi>. (Wikipedia 2021).

[3] Viblo Asia. 2018. “Giao thức MQTT”. Truy cập ngày 21/4. https://viblo.asia/p/mqtt-la-gi-vai-tro-cua-mqtt-trong-iot-V3m5WL3bKO7. (Viblo Asia 2018).

[4] Wikipedia. 2021. “ESP32”. Truy cập ngày 12/4. <https://vi.wikipedia.org/wiki/ESP32>. (Wikipedia 2021).

[5] Linh Kiện Việt. 2022. “Module nguồn cách ly B5050S”. Truy cập ngày 22/4. http://linhkienviet.vn/module-nguon-cach-ly-b0505s-2w-dc-dc-5v-5v. (Linh Kiện Việt 2022).

[6] Wikipedia. 2021. “Hệ điều hành FreeRTOS”. Truy cập ngày 30/4.

<https://vi.wikipedia.org/wiki/FreeRTOS>. (Wikipedia 2021).

[7] Wikipedia. 2021. “HTML”. Truy cập ngày 29/4. https://vi.wikipedia.org/wiki/HTML. (Wikipedia 2021).

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

# PHỤ LỤC

Code lập trình ESP32

#include <PubSubClient.h>

#include <WiFi.h>

#include <ArduinoJson.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

//ota update

#include <ESPAsyncWebServer.h>

#include <AsyncElegantOTA.h>

#define ssid "Zoro"

#define password "minhdien04"

#define mqtt\_server "broker.hivemq.com"

#define mqtt\_user ""

#define mqtt\_pwd ""

const uint16\_t mqtt\_port = 1883;

String mqtt\_topic\_control\_TB = "nguyenvanvinh/controlTB"; //subscribe

String mqtt\_topic\_infor\_TB\_ALL = "nguyenvanvinh/inforTBALL";//publish

String mqtt\_topic\_infor\_TB\_Relay = "nguyenvanvinh/inforTBRelay"; //publish

WiFiClient espClient;

PubSubClient client(espClient);

AsyncWebServer server(80);

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

void setup\_wifi();

void reconnect();

void taskInit();

void sendMqttTask();

void sendMqtt(void \*parameter);

void sendDataMQTT();

void otaUpdate();

String Data = "";

uint16\_t dataInt = 0;

float dataFloat = 0;

//button variable

int den3Pin = 15;

int den2Pin = 4;

int den1Pin = 16;

int den4Pin = 17;

int denAllPin = 5;

int remPin = 18;

int den4OutPin = 26;

int den1OutPin = 25;

int den5OutPin = 33;

int den3OutPin = 23;

int den2OutPin = 32;

int l298COutPin = 14;

int l298DOutPin = 27;

int button1PressCount = 0;

int button2PressCount = 0;

int button3PressCount = 0;

int button4PressCount = 0;

int button5PressCount = 0;

int button6PressCount = 0;

//pwm channel

int PWMchannel0 = 0;

int PWMchannel1 = 1;

void l298nInit();

void rotateClock();

void pinInit();

///handle button task

void handleButton1Task();

void handleButton2Task();

void handleButton3Task();

void handleButton4Task();

void handleButton5Task();

void handleButton6Task();

void handleButton1(void \*parameter);

void handleButton2(void \*parameter);

void handleButton3(void \*parameter);

void handleButton4(void \*parameter);

void handleButton5(void \*parameter);

void handleButton6(void \*parameter);

void handleAutoTask();

void handleAuto(void \*parameter);

uint16\_t dataLight = 0;

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, password);

pinInit();

l298nInit();

setup\_wifi();

otaUpdate();

client.setServer(mqtt\_server, mqtt\_port);

client.setCallback(callback);

taskInit();

}

void loop()

{

dataLight = analogRead(36);

// Serial.println(dataLight);

if (!client.connected()) {

reconnect();

}

client.loop();

}

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

{

String Data = "";

for (int i = 0; i < length; i++)

{

Data += (char)payload[i]; // abcde

}

dataInt = Data.toInt();

if(strcmp(topic, mqtt\_topic\_control\_TB.c\_str()) == 0)

{

String Data = "";

for (int i = 0; i < length; i++)

{

Data += (char)payload[i]; // abcde

}

dataInt = Data.toInt();

Serial.println(dataInt);

///button 1

if (dataInt == 10)

{

button1PressCount = 0;

digitalWrite(den1OutPin, LOW);

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "10");

}

else if (dataInt == 11)

{

button1PressCount = 1;

digitalWrite(den1OutPin, HIGH);

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "11");

}

///button 2

else if (dataInt == 20)

{

button2PressCount = 0;

digitalWrite(den2OutPin, LOW);

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "20");

}

else if (dataInt == 21)

{

button2PressCount = 1;

digitalWrite(den2OutPin, HIGH);

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "21");

}

///button 3

else if (dataInt == 30)

{

button3PressCount = 0;

digitalWrite(den3OutPin, LOW);

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "30");

}

else if (dataInt == 31)

{

button3PressCount = 1;

digitalWrite(den3OutPin, HIGH);

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "31");

}

///button 4

else if (dataInt == 40)

{

digitalWrite(den4OutPin, LOW);

button4PressCount = 0;

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "40");

}

else if (dataInt == 41)

{

digitalWrite(den4OutPin, HIGH);

button4PressCount = 1;

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "41");

}

///button 5

else if (dataInt == 50)

{

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "50");

button5PressCount = 0;

rotateNotClock();

}

else if (dataInt == 51)

{

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "51");

button5PressCount = 1;

rotateClock();

}

else if (dataInt == 60)

{

digitalWrite(den1OutPin, LOW);

button1PressCount = 0;

digitalWrite(den2OutPin, LOW);

button2PressCount = 0;

digitalWrite(den3OutPin, LOW);

button3PressCount = 0;

digitalWrite(den4OutPin, LOW);

button4PressCount = 0;

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "10");

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "20");

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "30");

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "40");

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "60");

button6PressCount = 0;

}

else if (dataInt == 61)

{

digitalWrite(den1OutPin, HIGH);

button1PressCount = 1;

digitalWrite(den2OutPin, HIGH);

button2PressCount = 1;

digitalWrite(den3OutPin, HIGH);

button3PressCount = 1;

digitalWrite(den4OutPin, HIGH);

button4PressCount = 1;

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "11");

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "21");

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "31");

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "41");

client.publish(mqtt\_topic\_infor\_TB\_Relay.c\_str(), "61");

button6PressCount = 1;

}

}

}

void setup\_wifi() {

delay(10);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

void reconnect()

{

while (!client.connected())

{

String clientId = String(random(0xffff), HEX);

if (client.connect(clientId.c\_str(), mqtt\_user, mqtt\_pwd))

{

Serial.println("Connected MQTT");

client.subscribe(mqtt\_topic\_control\_TB.c\_str());

}

else

{

Serial.println("Not Connnected MQTT");

delay(3000);

}

}

}

void sendDataMQTT()

{

DynamicJsonDocument doc(1024);

char inforInverterBuff[256];

doc["den1"] = button1PressCount;

doc["den2"] = button2PressCount;

doc["den3"] = button3PressCount;

doc["den4"] = button4PressCount;

doc["rem"] = button5PressCount;

doc["denALL"] = button6PressCount;

serializeJson(doc, inforInverterBuff);

client.publish(mqtt\_topic\_infor\_TB\_ALL.c\_str(), inforInverterBuff);

}

//rtos task

void sendMqttTask()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

sendMqtt, // Function to be called

"sendMqtt", // Name of task

2048, // Stack size (bytes in ESP32, words in FreeRTOS)

NULL, // Parameter to pass to function

1, // Task priority (0 to configMAX\_PRIORITIES - 1)

NULL, // Task handle

1);

}

void sendMqtt(void \*parameter) {

while (1) {

sendDataMQTT();

vTaskDelay(2000 / portTICK\_PERIOD\_MS);

}

}

void handleButton1Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton1, // Function to be called

"handleButton1", // Name of task

2048, // Stack size (bytes in ESP32, words in FreeRTOS)

NULL, // Parameter to pass to function

1, // Task priority (0 to configMAX\_PRIORITIES - 1)

NULL, // Task handle

1);

}

void handleButton2Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton2, // Function to be called

"handleButton2", // Name of task

2048, // Stack size (bytes in ESP32, words in FreeRTOS)

NULL, // Parameter to pass to function

1, // Task priority (0 to configMAX\_PRIORITIES - 1)

NULL, // Task handle

1);

}

void handleButton3Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton3, // Function to be called

"handleButton3", // Name of task

2048, // Stack size (bytes in ESP32, words in FreeRTOS)

NULL, // Parameter to pass to function

1, // Task priority (0 to configMAX\_PRIORITIES - 1)

NULL, // Task handle

1);

}

void handleButton4Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton4, // Function to be called

"handleButton4", // Name of task

2048, // Stack size (bytes in ESP32, words in FreeRTOS)

NULL, // Parameter to pass to function

1, // Task priority (0 to configMAX\_PRIORITIES - 1)

NULL, // Task handle

1);

}

void handleButton5Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton5, // Function to be called

"handleButton5", // Name of task

2048, // Stack size (bytes in ESP32, words in FreeRTOS)

NULL, // Parameter to pass to function

1, // Task priority (0 to configMAX\_PRIORITIES - 1)

NULL, // Task handle

1);

}

void handleButton6Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton6, // Function to be called

"handleButton6", // Name of task

2048, // Stack size (bytes in ESP32, words in FreeRTOS)

NULL, // Parameter to pass to function

1, // Task priority (0 to configMAX\_PRIORITIES - 1)

NULL, // Task handle

1);

}

void handleAutoTask()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleAuto, // Function to be called

"handleAuto", // Name of task

2048, // Stack size (bytes in ESP32, words in FreeRTOS)

NULL, // Parameter to pass to function

5, // Task priority (0 to configMAX\_PRIORITIES - 1)

NULL, // Task handle

1);

}