|  |  |
| --- | --- |
|  |  |
| MAI XUÂN KIÊN | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
|  |
| ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC CNKT ĐIỆN TỬ - VIỄN THÔNG |
|  |
| **THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT CÁC THIẾT BỊ TỪ XA BẰNG WIFI SỬ DỤNG ESP32** |
|  |
|  |
| **CBHD: ThS. Trần Xuân Phương** |
| CNKT ĐIỆN TỬ - VIỄN THÔNG | **Sinh viên: Mai Xuân Kiên** |
| **Mã số sinh viên: 2018605403** |
|  |
|  |
|  |
|  |
| Hà Nội – Năm 2022 |

# MỤC LỤC

[MỤC LỤC I](#_Toc104674826)

[DANH MỤC TỪ VIẾT TẮT V](#_Toc104674827)

[DANH MỤC BẢNG BIỂU VII](#_Toc104674828)

[DANH MỤC HÌNH ẢNH VIII](#_Toc104674829)

[LỜI CẢM ƠN X](#_Toc104674830)

[LỜI CAM ĐOAN XI](#_Toc104674831)

[LỜI MỞ ĐẦU XII](#_Toc104674832)

[Lý do chọn đề tài: XIII](#_Toc104674833)

[Mục tiêu đề tài: XIV](#_Toc104674834)

[Đối tượng nghiên cứu: XIV](#_Toc104674835)

[Phạm vi nghiên cứu: XIV](#_Toc104674836)

[Vai trò và ý nghĩa khoa học, thực tiễn của đề tài: XIV](#_Toc104674837)

[CHƯƠNG 1: TỔNG QUAN VỀ Đề tài Thiết kế HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT CÁC THIẾT BỊ TỪ XA BẰNG WIFI SỬ DỤNG ESP32 1](#_Toc104674838)

[1.1 Giới thiệu chung về đề tài 1](#_Toc104674839)

[1.2 Công nghệ sử dụng để điều khiển và giám sát từ xa hiện nay 1](#_Toc104674840)

[1.2.1 Wifi và Internet 1](#_Toc104674841)

[1.2.2 Bluetooth và Bluetooth Low Engine (BLE) 1](#_Toc104674842)

[1.2.3 Tiêu chuẩn khu vực mạng lưới cá nhân Zigbee 2](#_Toc104674843)

[1.2.4 Công nghệ LoRa 2](#_Toc104674844)

[1.3 Các hệ thống điều khiển và giám sát ngoài thị trường 2](#_Toc104674845)

[1.3.1 Hệ thống điều khiển giám sát điện năng 3](#_Toc104674846)

[1.3.2 Hệ thống điều khiển và giám sát nhà thông minh 4](#_Toc104674847)

[1.3.3 Hệ thống điều khiển và giám sát nhà máy nước 5](#_Toc104674848)

[1.4 Kết luận chương 1 6](#_Toc104674849)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 7](#_Toc104674850)

[2.1 Các giao thức chính được sử dụng 7](#_Toc104674851)

[2.1.1 Giao thức WiFi 7](#_Toc104674852)

[2.1.2 Giao thức HTTP 9](#_Toc104674853)

[2.1.3 Giao thức MQTT 10](#_Toc104674854)

[2.2 Linh kiện và phần mềm thiết kế mạch 10](#_Toc104674855)

[2.2.1 Vi điều khiển ESP32 10](#_Toc104674856)

[2.2.2 IC cách ly nguồn B5050S 14](#_Toc104674857)

[2.2.3 Mạch Buck LM2596 15](#_Toc104674858)

[2.2.4 Opto quang PC817 15](#_Toc104674859)

[2.2.5 Rơ le 5v 16](#_Toc104674860)

[2.3 Phần mềm thiết kế Altium Designer 16](#_Toc104674861)

[2.4 Phần mềm của hệ thống 19](#_Toc104674862)

[2.4.1 HTML (HyperText Markup Language) 19](#_Toc104674863)

[2.4.2 CSS (Cascading Style Sheets) 19](#_Toc104674864)

[2.4.3 JavaScript 20](#_Toc104674865)

[2.4.4 Phần mềm Visual Studio code 20](#_Toc104674866)

[2.4.5 Phần mềm Arduino IDE 21](#_Toc104674867)

[2.4.6 Hệ điều hành FreeRTOS 22](#_Toc104674868)

[2.5 Kết luận chương 2 23](#_Toc104674869)

[CHƯƠNG 3: THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT 24](#_Toc104674870)

[3.1 Thiết kế sơ đồ khối hệ thống điều khiển và giám sát 24](#_Toc104674871)

[3.1.1 Phân tích yêu cầu công nghệ 24](#_Toc104674872)

[3.1.2 Mô tả hoạt động 24](#_Toc104674873)

[3.1.3 Sơ đồ khối 25](#_Toc104674874)

[3.2 Thiết kế sơ đồ nguyên lý của mạch điện 27](#_Toc104674875)

[3.2.1 Khối xử lý trung tâm 29](#_Toc104674876)

[3.2.2 Khối nguồn và ổn áp nguồn 30](#_Toc104674877)

[3.2.3 Khối cách ly nguồn 30](#_Toc104674878)

[3.2.4 Khối nút nhấn 31](#_Toc104674879)

[3.2.5 Khối Relay 31](#_Toc104674880)

[3.2.6 Các header mở rộng 32](#_Toc104674881)

[3.3 Thiết kế mạch in (PCB) 33](#_Toc104674882)

[3.4 Lập trình vi xử lý ESP32 34](#_Toc104674883)

[3.4.1 Các thư viện được sử dụng 34](#_Toc104674884)

[3.4.2 Xử lý sự kiện nhấn nút 34](#_Toc104674885)

[3.4.3 Xử lý sự kiện nhận về từ MQTT Broker 35](#_Toc104674886)

[3.4.4 Xây dựng giao diện Web 36](#_Toc104674887)

[3.5 Kết quả đạt được 37](#_Toc104674888)

[3.5.1 Mạch sau khi lắp ráp 37](#_Toc104674889)

[3.5.2 Mô hình sản phẩm 38](#_Toc104674890)

[3.6 Giao diện điều khiển trên Webserver 38](#_Toc104674891)

[3.7 Kết quả thực nghiệm 39](#_Toc104674892)

[3.7.1 Thực nghiệm 1 39](#_Toc104674893)

[3.7.2 Thực nghiệm 2 41](#_Toc104674894)

[3.8 Nhận xét, đánh giá 44](#_Toc104674895)

[3.8.1 Phân tích, giải thích kết quả thực nghiệm 44](#_Toc104674896)

[3.8.2 Tính năng và hiệu quả sử dụng của sản phẩm thiết kế 44](#_Toc104674897)

[3.8.3 Tính ứng dụng, mức độ an toàn và tác động của sản phẩm thiết kế tới môi trường, kinh tế và xã hội 44](#_Toc104674898)

[3.9 Kết luận chương 3 45](#_Toc104674899)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 46](#_Toc104674900)

[TÀI LIỆU THAM KHẢO 48](#_Toc104674901)

[PHỤ LỤC 49](#_Toc104674902)

[HƯỚNG DẪN SỬ DỤNG 49](#_Toc104674903)

[CODE CHƯƠNG TRÌNH 55](#_Toc104674904)

# DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **Từ viết tắt** | **Từ Tiếng Anh** | **Từ Tiếng Việt** |
| MQTT | Message Queuing Telemetry Transport | Giao thức truyền thông điệp |
| HTTP | Hyper Text Transfer Protocol | Giao thức truyền tải siêu văn bản |
| MCU | Multipoint Control Unit | Thiết bị điều khiển đa điểm |
| IoT | Internet of Things | Internet vạn vật |
| HTML | HyperText Markup Language | Ngôn ngữ Đánh dấu Siêu văn bản |
| CSS | Cascading Style Sheets | Tập tin định dạng theo tầng |
| JS | JavaScripts | Ngôn ngữ lập trình Java Script |
| IDE | Integrated Development Environment | Môi trường phát triển tích hợp |
| IEEE | Institute of Electrical and Electronics Engineers | Hội Kỹ sư Điện và Điện tử |
| M2M | Machine – to - Machine | Tương tác giữa máy với máy |
| TCP | Transmission Control Protocol | Giao thức điều khiển truyền dẫn |
| IP | Internet Protocol | Giao thức mạng |
| PWM | Pulse Width Modulation | Điều chế độ rộng xung |
| UART | Universal Asynchronous Receiver / Transmitte | Bộ thu / phát không đồng bộ đa năng |
| SPI | Serial Peripheral Interface | Giao diện ngoại vi nối tiếp |
| IDE | Integrated Development Environment | Môi trường phát triển tích hợp |
| GPIO | General Purpose Input Output | Bộ mở rộng cổng |
| VCC | Common colector | Điểm nối nguồn |
| GND | Ground | Điểm nối đất |

# DANH MỤC BẢNG BIỂU

[Bảng 3.1: Một số thẻ HTML cơ bản 36](#_Toc104586704)

[Bảng 3.2: Bật / tắt thiết bị bằng nút nhấn (thực nghiệm 1) 40](#_Toc104586710)

[Bảng 3.3: Bật / tắt thiết bị bằng Webserver (Thực nghiệm 1) 40](#_Toc104586712)

[Bảng 3.4: Bật / tắt thiết bị bằng nút nhấn (Thực nghiẹm 2) 42](#_Toc104586715)

[Bảng 3.5: Một số thẻ HTML cơ bản 43](#_Toc104586717)

# DANH MỤC HÌNH ẢNH

[Hình 1.1: Hệ thống điều khiển và giám sát điện năng 3](#_Toc104674905)

[Hình 1.2: Hệ thống điều khiển và giám sát nhà thông minh 4](#_Toc104674906)

[Hình 1.3: Hệ thống điều khiển và giám sát nhà máy nước 5](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674907)

[Hình 2.1: Modem, Router của Wifi 7](#_Toc104674908)

[Hình 2.2: Các giao thức và dịch vụ 9](#_Toc104674910)

[Hình 2.3: Giao thức HTTP 9](#_Toc104674911)

[Hình 2.4: Mô hình MQTT Broker 10](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674912)

[Hình 2.5: ESP32 DevKit V1 11](#_Toc104674913)

[Hình 2.6: Sơ đồ chân và ngoại vi của ESP32 Dev kit 14](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674914)

[Hình 2.7: Rơ le 5v 16](#_Toc104674917)

[Hình 2.8: Altium Designer 2022 17](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674919)

[Hình 2.9: Một cửa sổ của Altium Designer 18](#_Toc104674920)

[Hình 2.10: Visual Studio Code 20](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674921)

[Hình 2.11: RTOS 22](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674922)

[Hình 2.12: Rtos Kernel 23](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674923)

[Hình 3.1: Quá trình bật tắt nút nhấn với thiết bị 1 25](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674924)

[Hình 3.2: Quá trình bật tắt bằng Webserver với thiết bị 1 25](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674925)

[Hình 3.3: Sơ đồ gửi nhận dữ liệu giữa mạch ESP32 và điện thoại/máy tính 26](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674926)

[Hình 3.4: Sơ đồ khối của mạch ESP32 26](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674927)

[Hình 3.5: Sơ đồ nguyên lý mạch ESP32 28](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674928)

[Hình 3.6: Giao diện phần mềm Altium Designer 29](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674929)

[Hình 3.7: Khối xử lý trung tâm 30](#_Toc104674930)

[Hình 3.8: Khối nguồn và ổn áp nguồn 31](#_Toc104674931)

[Hình 3.9: Khối cách ly nguồn 31](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674932)

[Hình 3.10:Khối nút nhấn 32](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674933)

[Hình 3.11: Khối Relay 32](#_Toc104674934)

[Hình 3.12: Khối mở rộng 33](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674935)

[Hình 3.13: Sơ đồ mạch in 2D 34](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674936)

[Hình 3.14: Sơ đồ mạch in 3D 34](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674937)

[Hình 3.15: Lưu đồ thuật toán xử lý sự kiện nút nhấn 35](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674938)

[Hình 3.16: Lưu đồ thuật toán xử lý sự kiện nhận về từ MQTT Broker 36](file:///C:\Users\Mai%20Xuan%20Kien\Downloads\Đồ%20Án\MXKien_quyen_bao_cao_sua%20t6_2605.docx#_Toc104674939)

[Hình 3.17: Mạch sau khi lắp ráp linh kiện 38](#_Toc104674941)

[Hình 3.18: Đề tài sau khi hoàn thành 39](#_Toc104674942)

[Hình 3.19: Giao diện điều khiển trên Webserver 39](#_Toc104674943)

[Hình 3.20: Nút nhấn điều khiển chế độ trực tiếp 40](#_Toc104674944)

[Hình 3.21: Đèn phòng 3 bật khi nhấn nút lần 1 40](#_Toc104674945)

[Hình 3.22: Giao diện khi điều khiển từ xa qua Webserver 41](#_Toc104674947)

[Hình 3.23: Mô hình hoạt động theo chế độ từ xa qua Webserver. 42](#_Toc104674949)

[Hình 3.24: Nút nhấn điều khiển. 42](#_Toc104674950)

[Hình 3.25: Đèn phòng 5 hoạt động trong chế độ 1. 43](#_Toc104674952)

[Hình 3.26: Mô hình hoạt động theo chế độ 2 44](#_Toc104674954)

# LỜI CẢM ƠN

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

Em xin gửi lời cảm ơn chân thành đến ThS. Trần Xuân Phương, người đã tận tình hướng dẫn, chỉ dạy, giúp đỡ và cung cấp những tài liệu cũng như kinh nghiệm quý báu giúp em hoàn thành các nhiệm vụ được giao trong quá trình thực hiện.

Em cũng xin cảm ơn các thầy cô trong trường Đại học Công nghiệp Hà Nội nói chung, các thầy cô trong khoa Điện tử nói riêng đã chỉ dạy những kiến thức cần thiết, giúp em có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ em trong suốt quá trình học tập.

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

Em xin chân thành cảm ơn!

Hà Nội, Ngày… tháng… năm 2022

Sinh viên thực hiện

Mai Xuân Kiên

# LỜI CAM ĐOAN

Em xin cam đoan rằng đồ án tốt nghiệp với đề tài: “**Thiết kế hệ thống điều khiển và giám sát các thiết bị từ xa bằng Wifi sử dụng ESP32**” là một sản phẩm nghiên cứu của chính bản thân em. Những phần có sử dụng tài liệu tham khảo đều đã được trích dẫn nguồn trong danh mục tài liệu tham khảo. Những số liệu và kết quả được sử dụng đều mang tính chất trung thực, không đạo văn, sao chép từ những công trình nghiên cứu khác. Nếu sai em xin chịu hoàn toàn trách nhiệm.

Hà Nội, Ngày… tháng… năm 2022

Sinh viên thực hiện

Mai Xuân Kiên

# 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 những 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 bảng, … 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 sát, 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: Điện thoại thông minh, máy tính, đồng hồ thông minh, …

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.

* Lý do chọn đề tài:

Cuộc cách mạng công nghiệp 4.0 đang bùng nổ và tập trung vào công nghệ kỹ thuật số từ những thập kỷ gần đây lên một cấp độ hoàn toàn mới. Với sự trợ giúp của kết nối thông qua Internet vạn vật, truy cập dữ liệu thời gian thực và giới thiệu các hệ thống vật lý không gian mạng. Nó cung cấp một cách tiếp cận liên kết và toàn diện hơn cho sản xuất, đồng thời giúp kết nối vật lý với kỹ thuật số và cho phép cộng tác và truy cập tốt hơn giữa các bộ phận, đối tác, nhà cung cấp, sản phẩm và con người. Điều này cho phép các nhà máy thông minh, sản phẩm thông minh và chuỗi cung ứng cũng thông minh, làm cho các hệ thống sản xuất và dịch vụ trở nên linh hoạt và đáp ứng khách hàng hơn.

Với mục đích có thể tạo ra một hệ thống ứng dụng IoT trong việc điều khiển và giám sát thiết bị, thân thiện với người dùng, khả năng tùy biến cao và giá thành phải chăng, em xin giới thiệu đề tài: **“Thiết kế hệ thống điều khiển và giám sát các thiết bị từ xa bằng WiFi 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 các thiết bị từ xa.

* Mục tiêu đề tài:

Với mục đích có thể tạo ra một hệ thống ứng dụng IoT trong việc:

- Điều khiển và giám sát thiết bị.

- Sử dụng giao diện thân thiện với người dùng.

- Khả năng tùy biến cao và giá thành phải chăng.

* Đối tượng nghiên cứu:

Trong đề tài này, em thực hiện thiết kế một mô hình điều khiển, giám sát từ xa bằng WiFi sử dụng vi điều khiển ESP32 và tìm hiểu về Websever.

* Phạm vi nghiên cứu:

Mô hình gồm 2 phần chính là mạch điện tử và phần mềm điều khiển. Đầu vào của mạch điện tử có 7 nút nhấn để điều khiển đầu ra gồm 7 rơ le. Phần mềm sẽ đồng bộ với nút nhấn trên phần cứng để hiển thị trạng thái bật/ tắt của thiết bị, đồng thời chúng ta có thể bật tắt thiết bị ngay trên phần mềm.

* Vai trò và ý nghĩa khoa học, thực tiễn của đề tài:

*- Vai trò của đề tài:*

Đề tài tập trung vào thiết kế mô hình giám sát trạng thái bật tắt các thiết bị qua đầu ra rơ le thông qua Wifi và Internet.

Đề tài giúp người dùng và người vận hành có thể giám sát và điều khiển dễ dàng các thiết bị thông qua điện thoại hoặc máy tính có kết nối Internet.

*- Ý nghĩa khoa học, thực tiễn của đề tài:*

Hệ thống điều khiển và giám sát các thiết bị từ xa bằng Wifi giúp việc tìm hiểu và phát triển về các hệ thống điều khiển từ xa trở nên dễ dàng hơn.

Với việc sử dụng một vi xử lý có tích hợp rất nhiều chức năng điều khiển từ xa và IoT như ESP32, đề tài thúc đẩy việc nghiên cứu về IoT, về điều khiển và tự động hóa các quá trình, các công nghệ không dây đang được phổ biến hiện nay.

Đề tài tập trung vào việc khai thác và lập trình các ngoại vi trên vi điều khiển ESP32, giúp việc tạo ra các sản phẩm về IoT linh hoạt và ổn định hơn.

Về ứng dụng thực tiễn của đề tài, đề tài có rất nhiều ứng dụng thực tiễn trong việc điều khiển và giám sát không chỉ là các phụ tải nhỏ mà còn kết hợp được với các thiết bị công nghiệp để đóng cắt phụ tải lớn hơn.

Đề tài có ứng dụng rất nhiều trong công nghiệp về việc giám sát, vận hành và lên lịch vận hành cho các thiết bị. Ngoài trong công nghiệp, đề tài còn ứng dụng trong nhà thông minh, trong các khu đô thị thông minh, …

Đề tài còn có thể được phát triển theo nhiều hướng khác nhau, theo nhiều yêu cầu đặt ra để điều khiển và giám sát không chỉ là bật tắt mà còn có thể sử dụng cho nhiều các tín hiệu khác nhau tùy từng loại máy móc hoặc yêu cầu cụ thể.

# 

# TỔNG QUAN VỀ Đề tài Thiết kế HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT các thiết bị từ xa bằng wifi sử dụng esp32

* 1. Giới thiệu chung về đề tài

Trong đề tài này, thực hiện thiết kế mô hình điều khiển, giám sát từ xa bằng WiFi sử dụng ESP32. Mô hình gồm 2 phần chính là mạch điện tử và phần mềm điều khiển. Đầu vào của mạch điện tử có 7 nút nhấn để điều khiển đầu ra gồm 7 rơ le. Phần mềm đồng bộ với nút nhấn trên phần cứng để hiển thị trạng thái bật tắt của thiết bị, chúng ta bật tắt thiết bị với nút nhấn vật lý hoặc đồng thời điều khiển ngay trên phần mềm.

* 1. Công nghệ sử dụng để điều khiển và giám sát từ xa hiện nay

Hệ thống giám sát từ xa được điều khiển thông qua các thiết bị có kết nối Internet điển hình như: Điện thoại, máy tính bảng, laptop, ...

Công nghệ được sử dụng phổ biến trong việc điều khiển và giám sát từ xa chủ yếu là các chuẩn truyền thông không dây như: Wifi và Internet, Bluetooth và Bluetooth Low Engine, Zigbee, LoRa, ...

* + 1. Wifi và Internet

Wifi là viết tắt của Wireless Fidelity là hệ thống truy cập internet không dây. Loại sóng vô tuyến này tương tự như sóng điện thoại, truyền hình và radio. 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 được sử dụng để truyền nhận dữ liệu lớn, đây cũng là công nghệ cốt lõi của công nghệ 4.0 với IoT đang được phát triển mạnh mẽ.

* + 1. Bluetooth và Bluetooth Low Engine (BLE)

Bluetooth là chuẩn kết nối không dây tầm ngắn, thiết kế cho các kết nối thiết bị cá nhân, trong phạm vi băng tần 2,4Ghz. Công nghệ này có thể được dùng để kết nối hoặc chuyển dữ liệu giữa các thiết bị không dây, chuyển cuộc gọi và dữ liệu như danh bạ, lịch giữa các desktop, notebook. Bluetooth Low Engine được dựa trên Bluetooth nhưng hoạt động với sự tiêu hao năng lượng thấp. Với lợi thế trên Bluetooth Low Engine được sử dụng rất nhiều trong các nhà thông minh, các thiết bị sử dụng pin và yêu cầu thời gian hoạt động cao.

* + 1. Tiêu chuẩn khu vực mạng lưới cá nhân Zigbee

Zigbee là tiêu chuẩn khu vực mạng lưới cá nhân 802.15.4 của IEEE, đã tồn tại hơn một thập kỷ. Nó được xem là một giải pháp thay thế cho Wi-Fi và Bluetooth của một số ứng dụng bao gồm các thiết bị sử dụng năng lượng thấp mà không cần nhiều băng thông - như các hệ thống cảm biến trong nhà thông minh. Zigbee hỗ trợ kết nối mạng lưới, nên mọi thao tác khi tương tác với các thiết bị sẽ được ổn định hơn, ngay cả khi có một trong những nút phát tín hiệu bị lỗi.

* + 1. Công nghệ LoRa

LoRa là một công nghệ không dây được phát triển để cho phép truyền tốc độ dữ liệu thấp trên một khoảng cách lớn bởi các cảm biến và bộ truyền động cho M2M và IoT cũng như các ứng dụng IoT. LoRa hướng tới các kết nối M2M ở khoảng cách lớn. Nó có thể hỗ trợ liên lạc ở khoảng cách lên tới 15 – 20 km, với hàng triệu node mạng. Nó có thể hoạt động trên băng tần không phải cấp phép, với tốc độ thấp từ 0,3kbps đến khoảng 30kbps. Với đặc tính này, mạng LoRa phù hợp với các thiết bị thông minh trao đổi dữ liệu ở mức thấp nhưng duy trì trong một thời gian dài. Thực tế các thiết bị LoRa có thể duy trì kết nối và chia sẻ dữ liệu trong thời gian lên đến 10 năm chỉ với năng lượng pin. [6]

* 1. Các hệ thống điều khiển và giám sát ngoài thị trường

Một số mô hình về hệ thống giám sát và điều khiển trên thị trường:

* + 1. Hệ thống điều khiển giám sát điện năng [12]



Hình 1.1: Hệ thống điều khiển và giám sát điện năng

Hệ thống giám sát điện năng (Power monitoring) là hệ thống bao gồm các thiết bị đo lường đa chức năng có tốc độ lấy mẫu cao, phân tích nhiều thông số, cho phép kết nối thành hệ thống và kết nối truyền thông với hệ máy tính để lưu trữ dữ liệu. Phần mềm chuyên dụng sẽ phân tích dữ liệu phục vụ việc hiển thị đo lường việc tiêu thụ năng lượng, giám sát hoạt động của hệ thống điện theo thời gian thực, điều khiển hệ thống điện từ xa bằng máy tính, lưu trữ, phân tích dữ liệu sử dụng và lập báo cáo tiêu thụ năng lượng của hệ thống điện.

* **Các chức năng chính của hệ thống:**

- Lập sơ đồ mô phỏng hệ thống điện.

- Hiển thị các thông số điện, năng lượng theo thời gian thực.

- Lưu trữ dữ liệu về điện nhằm phục vụ thống kê và phân tích năng lượng.

- Vẽ biểu đồ phụ tải theo thời gian thực.

- Phân tích chất lượng điện năng.

- Cảnh báo sớm sự cố trong hệ thống điện.

- Điều khiển đóng, cắt thiết bị điện từ xa.

- Giám sát tình trạng hoạt động thiết bị tiêu thụ điện.

- Lập báo cáo về năng lượng.

* + 1. Hệ thống điều khiển và giám sát nhà thông minh [13]



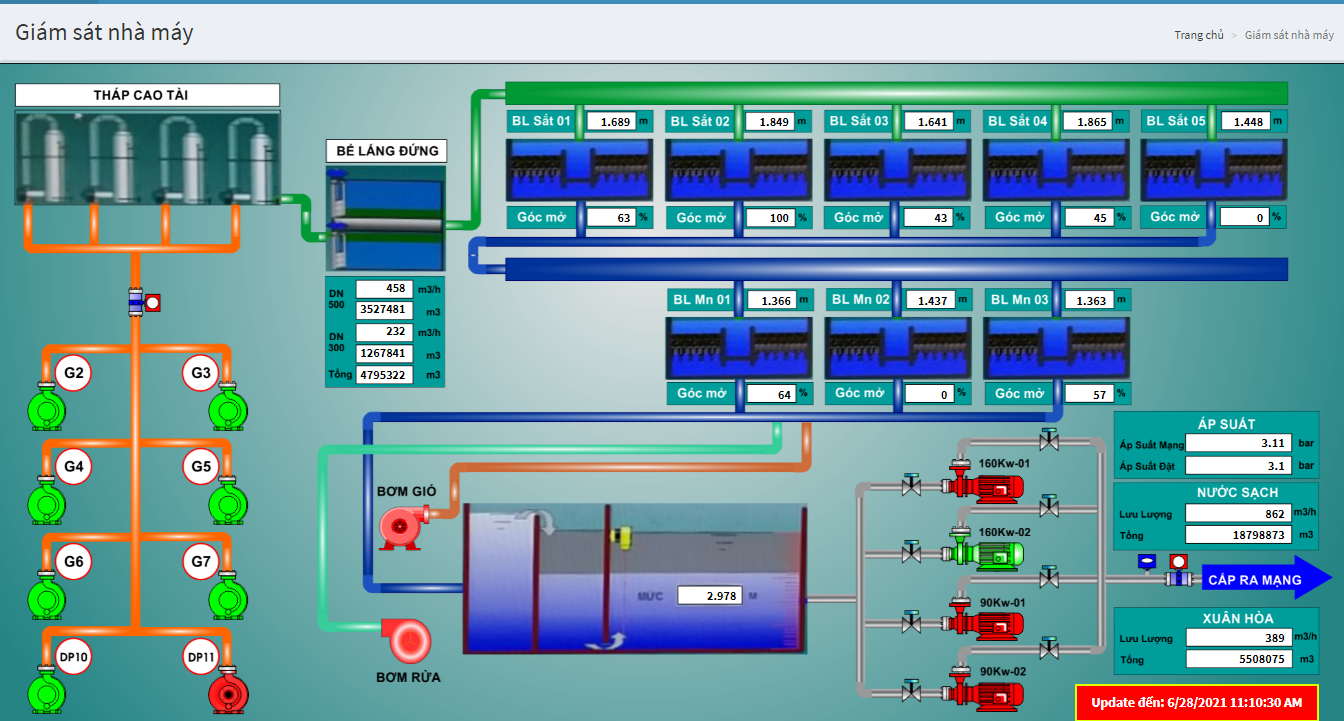
Hình 1.2: Hệ thống điều khiển và giám sát nhà thông minh

Hệ thống điều khiển và giám sát nhà thông minh có nhiều chức năng và lợi ích khác nhau như:

- Hệ thống điều khiển đèn tự động tắt/mở: Điều khiển bằng giọng nói, kiểm soát ánh sáng, tiết kiệm điện.

- Điều khiển dễ dàng với điện thoại thông minh: Tất cả hệ thống ánh sáng, điện, tivi, điều hoà, camera đều được quản lý trong lòng bàn tay với chiếc điện thoại có kết nối Internet.

- Điều hoà không khí/ máy nước nóng: Bật tắt điều hoà, máy nước nóng trước, khi về nhà đã có sẵn nước nóng, nhiệt độ thoải mái để nghỉ ngơi.

* + 1. Hệ thống điều khiển và giám sát nhà máy nước [14]

Hình 1.3: Hệ thống điều khiển và giám sát nhà máy nước

Trước đây, toàn bộ hệ thống xử lý nước sạch của nhà máy được quản lý theo phương thức cũ (mô hình SCADA Offline), tất cả được đưa về phòng điều khiển trung tâm và người quản lý chỉ có thể theo dõi được tình hình hệ thống tại đó. Các điểm đo phải thi công mất rất nhiều công đi dây, đưa dây tín hiệu về phòng điều khiển trung tâm. Tuy nhiên với giải pháp điều khiển và giám sát từ xa đã khắc phục những khó khăn tồn đọng của nhà máy. Hoàn toàn có thể theo dõi điều khiển giám sát hệ thống từ xa mà không cần phải đển phòng vận hành. Công việc lắp đặt các trạm bơm, giếng bơm, trạm đo lưu lượng không còn khó khăn như trước nữa, các dữ liệu tại các giếng, các điểm đo được đưa về phòng điều khiển trung tâm thông qua mạng truyền thông không dây.

Các trạng thái hoạt động của nhà máy đều được thể hiện rõ ràng trực quan, trạng thái các thiết bị, … Các thông số bao gồm: Trạng thái đóng mở các bơm, lưu lượng nước, áp suất đặt và áp suất toàn hệ thống, lưu lượng nước ở bể lắng đứng, các thông số của bể lọc sắt, mangan, mức nước của hồ chứa nước.

* 1. Kết luận chương 1

Chương 1 mô tả tổng quan về hệ thống điều khiển và giám sát, một số công nghệ sử dụng để điều khiển và giám sát. Đồng thời chương 1 cũng nêu ra được một số vai trò, ý nghĩa của đề tài.

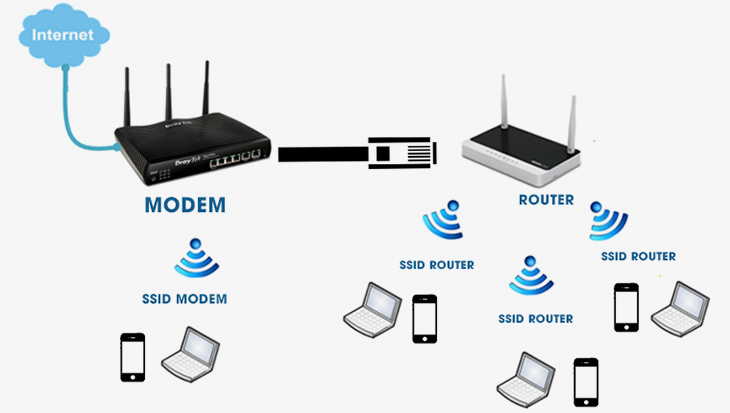
Chương 1 nêu ra được tổng quan về các hệ thống IoT trong thực tế, ứng dụng của các hệ thống IoT trong các ngành khác nhau để nâng cao được hiệu suất sử dụng của máy móc và sự tiện lợi khi điều khiển và giám sát từ xa.

# CƠ SỞ LÝ THUYẾT

* 1. Các giao thức chính được sử dụng
     1. Giao thức WiFi [8]

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.

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 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.



Hình 2.1: Modem, Router của Wifi

* **Một số chuẩn WiFi hiện nay:**

Có rất nhiều chuẩn WiFi phổ biến hiện nay (bảng 2.1). 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.

Bảng 2.1: Các chuẩn wifi hiện nay

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CÁC CHUẨN WIFI HIỆN NAY | | | | | | |
| CHUẨN IEEE | 802.11a | 802.11b | 802.11g | 802.11n | 802.11ac | 802.11ax |
| Năm phát hành | 1999 | 1999 | 2003 | 2009 | 2014 | 2019 |
| Tần số | 5 GHZ | 2.4 GHZ | 2.4 GHZ | 2.4/5 GHZ | 5 GHZ | 6 GHZ |
| Tốc độ tối đa | 54 Mbps | 11 Mbps | 54 Mbps | 600 Mbps | 1 Gbps | 10 Gbps |
| Phạm vi trong nhà | 30m | 30m | 38m | 68m | 73m | Chưa công bố |
| Phạm vi ngoài trời | 30m | 31m | 137m | 250m | 300m | Chưa công bố |

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 [11]

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

**Tầng ứng dụng**

**Tầng giao tiếp mạng**

**Tầng phiên**

**Tầng liên kết**

**Tầng trình bày**

**Tầng ứng dụng**

**Tầng vật lý**

**Tầng vận chuyển**

**Tầng mạng**

**Tầng vận chuyển**

**Tầng mạng**

**Mô hình TCP/IP**

**Mô hình OSI**

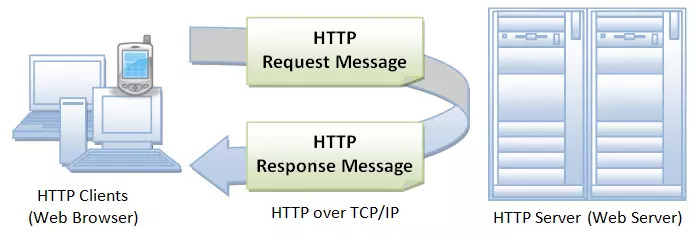
**Các giao thức và dịch vụ**

**HTTP, FTTP, Telnet, NTP, DHCPP, PING**

**TCP, UDP**

**IP, ARP, ICMP, IGMP**

**Enthernet**

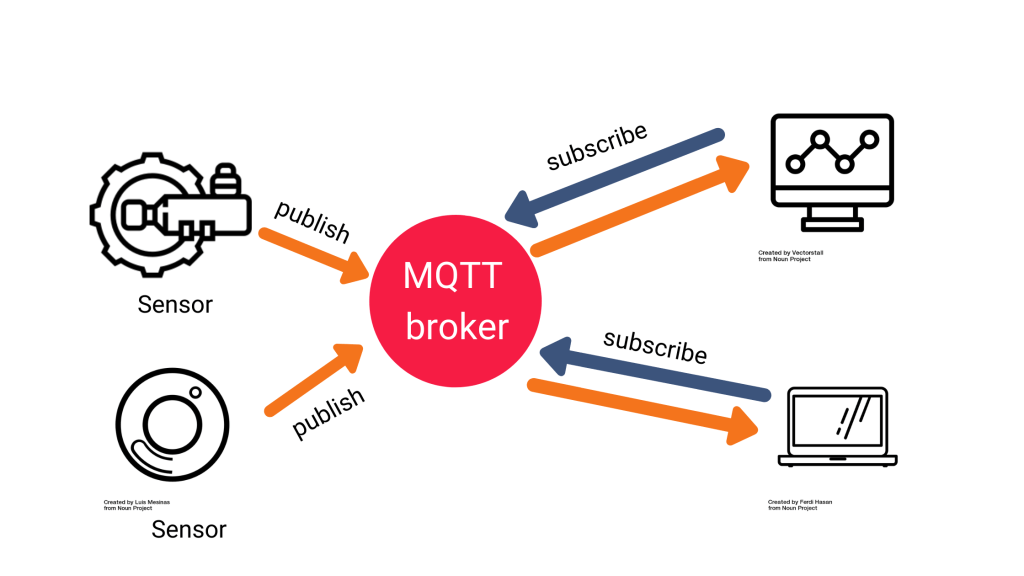
Hình 2.2: Các giao thức và dịch vụ

Hình 2.3: Giao thức HTTP

Giao thức HTTP hoạt động dựa trên mô hình Client – Server. Thông thường khi các truy cập trang wed, 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.

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.

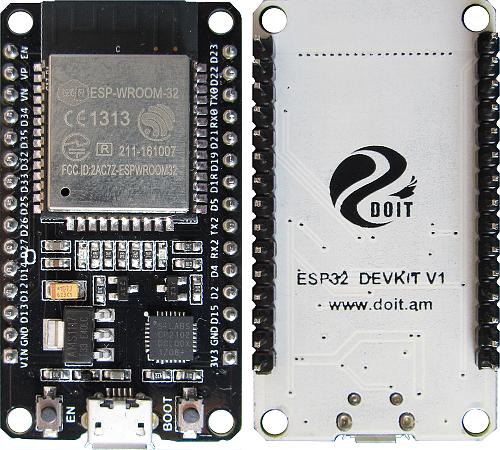
* + 1. Giao thức MQTT

**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. [10]

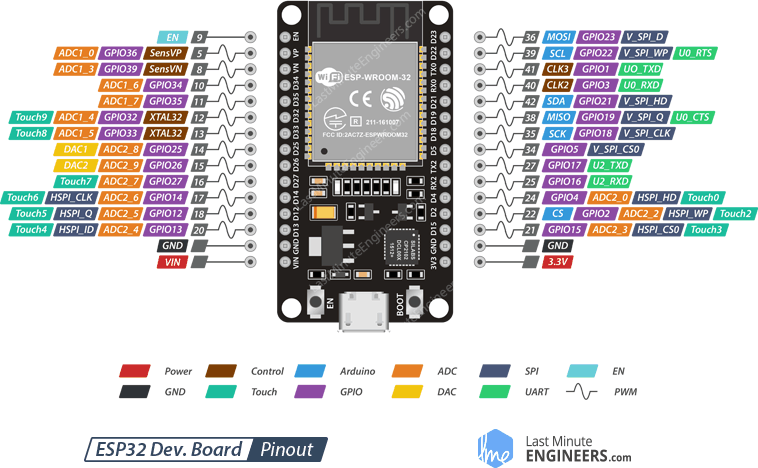
Hình 2.4: Mô hình MQTT Broker

* 1. Linh kiện và phần mềm thiết kế mạch
     1. Vi điều khiển ESP32 [5]

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 (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.



Hình 2.5: ESP32 DevKit V1

* ***Nguồn sử dụng:***
* 3.3V đối với chip ESP32.
* 5V đối với kit ESP32 Dev Kit.
* ***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 (bộ nhớ chỉ đọc) cho việc khởi động và các tính năng lõi.
* 520 KB bộ nhớ RAM (bộ nhớ truy xuất ngẫu nhiên) 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 (khởi động an toàn).
* Mã hóa flash.
* Tăng tốc mã hóa phần cứng: AES, SHA - 2, RSA, elliptic curve cryptography (ECC, 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.

Hình 2.6: Sơ đồ chân và ngoại vi của ESP32 Dev kit

* + 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.

* **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 2.2: Các thông số chính của B0505s

|  |  |  |
| --- | --- | --- |
| 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, cấp cho các thiết bị cần thiết như camera, robot, ...

* **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.
  + 1. Opto quang PC817

Bảng 2.3: Các thông số của Opto quang

|  |  |  |
| --- | --- | --- |
| Thông số | Ký hiệu | Giá trị |
| Điện áp ngược đầu vào lớn nhất | VR | 6V |
| Điện áp đầu ra C - E lớn nhất | VCEO | 35V |
| Dòng thuận lớn nhất | IF | 50 |
| Dòng điện Collector lớn nhất | IC | 50 |

* + 1. Rơ le 5v



Hình 2.7: Rơ le 5v

Bảng 2.4: Các thông số của Rơ le 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. Phần mềm thiết kế Altium Designer

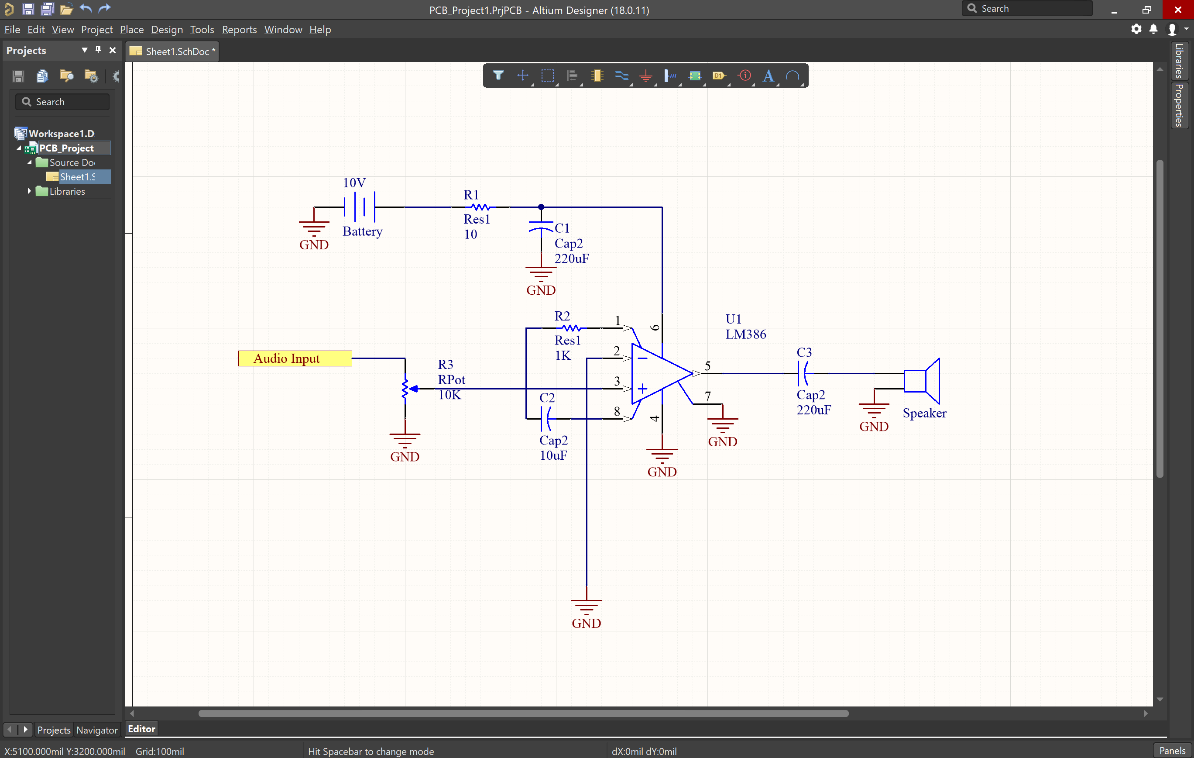
Altium Designer trước kia có tên gọi quen thuộc là Protel DXP, là một trong những công cụ vẽ mạch điện tử mạnh nhất hiện nay. Được phát triển bởi hãng Altium Limited. Altium designer là một phần mềm chuyên nghành được sử dụng trong thiết kế mạch điện tử. Nó là một phần mềm mạnh với nhiều tính năng thú 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.



Hình 2.8: Altium Designer 2022

* **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: Một cửa sổ của Altium Designer

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 (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 (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.

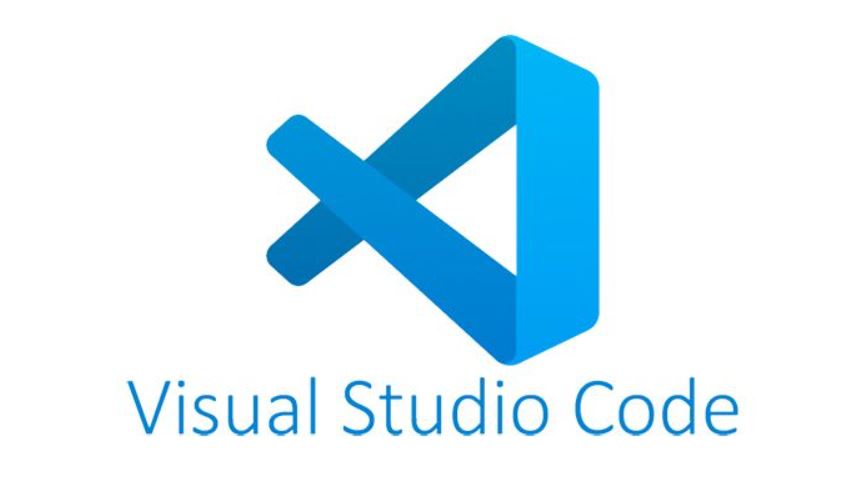
* + 1. JavaScript

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 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

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 phép người dùng thay đổi theme, phím tắt, và các tùy chọn khác.

Hình 2.10: Visual Studio Code

* ***Một số tính năng của phần mềm:***

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 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 [4]

Arduino IDE là một phần mềm mã nguồn mở chủ yếu được sử dụng để viết và biên dịch mã vào module Arduino. Đây là một phần mềm Arduino chính thức, giúp cho việc biên dịch mã trở nên dễ dàng mà ngay cả một người bình thường không có kiến thức kỹ thuật cũng có thể làm được.

Nó có các phiên bản cho các hệ điều hành như MAC, Windows, Linux và chạy trên nền tảng Java đi kèm với các chức năng và lệnh có sẵn đóng vai trò quan trọng để gỡ lỗi, chỉnh sửa và biên dịch mã trong môi trường. Có rất nhiều các module Arduino như Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro và nhiều module khác.

Mỗi module chứa một bộ vi điều khiển trên bo mạch được lập trình và chấp nhận thông tin dưới dạng mã. Mã chính, còn được gọi là sketch, được tạo trên nền tảng IDE sẽ tạo ra một file Hex, sau đó được chuyển và tải lên trong bộ điều khiển.

Môi trường IDE chủ yếu chứa hai phần cơ bản: Trình chỉnh sửa và Trình biên dịch, phần đầu sử dụng để viết mã được yêu cầu và phần sau được sử dụng để biên dịch và tải mã lên module Arduino. Môi trường này hỗ trợ cả ngôn ngữ C và C ++.

* + 1. Hệ điều hành FreeRTOS

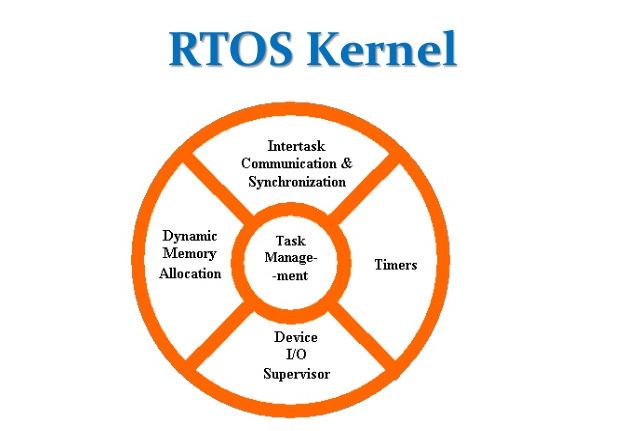
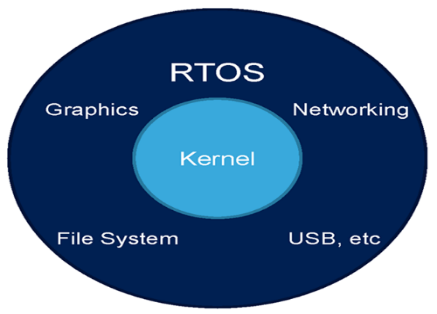
FreeRTOS là một hệ điều hành nhúng thời gian thực (Real Time Operating System) mã nguồn mở được phát triển bởi Real Time Engineers Ltd, sáng lập và sở hữu bởi Richard Barry. [3]

Hình 2.11: RTOS

Một số tính năng của FreeRTOS:

* Kích thước bộ nhớ nhỏ, chi phí thấp và thực thi nhanh chóng.
* Tùy chọn ít đánh dấu cho các ứng dụng năng lượng thấp.
* Dành cho cả những người có sở thích và các nhà phát triển chuyên nghiệp làm việc trên các sản phẩm thương mại.
* Bộ lập lịch có thể được cấu hình cho cả đa nhiệm với các tác vụ.

Hình 2.12: Rtos Kernel



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.

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. Phân tích yêu cầu công nghệ

Hệ thống hoàn thiện đảm bảo các chức năng bật tắt bằng 7 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. Mô tả hoạt động

Quá trình bật bằng nút nhấn với thiết bị 1:

**Nút nhấn**

**Tín hiệu mức thấp vào vi điều khiển**

**Vi điều khiển**

**MQTT Broker**

**Gửi chuỗi “11” lên MQTT Broker**

**Chuyển tiếp chuỗi “11” lên giao diện Web**

**Giao diện web đổi trạng thái**

**Bật Rơ Le 1**

Hình 3.1: Quá trình bật tắt nút nhấn với thiết bị 1

Quá trình bật bằng Webserver với thiết bị 1:

**Vi điều khiển**

**MQTT Broker**

**Gửi chuỗi “11” về vi điều khiển**

**Chuyển tiếp chuỗi “11” lên MQTT Broker**

**Giao diện web đổi trạng thái**

**Bật Rơ Le 1**

**Gửi chuỗi “11” lại về MQTT Broker thông báo đã bật**

Hình 3.2: Quá trình bật tắt bằng Webserver với thiết bị 1

* + 1. Sơ đồ khối

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

Hình 3.3: Sơ đồ gửi nhận dữ liệu giữa mạch ESP32 và điện thoại/máy tính

**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

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 3.3

* ***Sơ đồ khối của mạch ESP32******:***

Khối nút nhấn

Khối nguồn và ổn áp 5v

Khối xử lý trung tâm (ESP32)

Khối cách ly nguồn

Khối Relay

Khối thiết bị (đèn, quạt)

Cách ly quang

Khuếch đại

Hình 3.4: Sơ đồ khối của mạch ESP32

* **Mạch ESP32 điều khiển các thiết bị bao gồm 8 khối:**
* Khối nút nhấn:

Bao gồm 7 nút nhấn để bật tắt 7 thiết bị tương ứng.

* Khối xử lý trung tâm:

Chứa vi điều khiển ESP32, xử lý các tín hiệu gửi đi, nhận về.

* Khối nguồn và ổn áp 5V:

Mạch được cấp nguồn từ 5 tới 12V, được ổn áp qua mạch ổn áp.

* Khối cách ly nguồn:

Cách ly độc lập thành 2 nguồn riêng biệt không chung GND, tránh tất cả mọi nhiễu từ nguồn điều khiển phía relay.

* Khối cách ly quang:

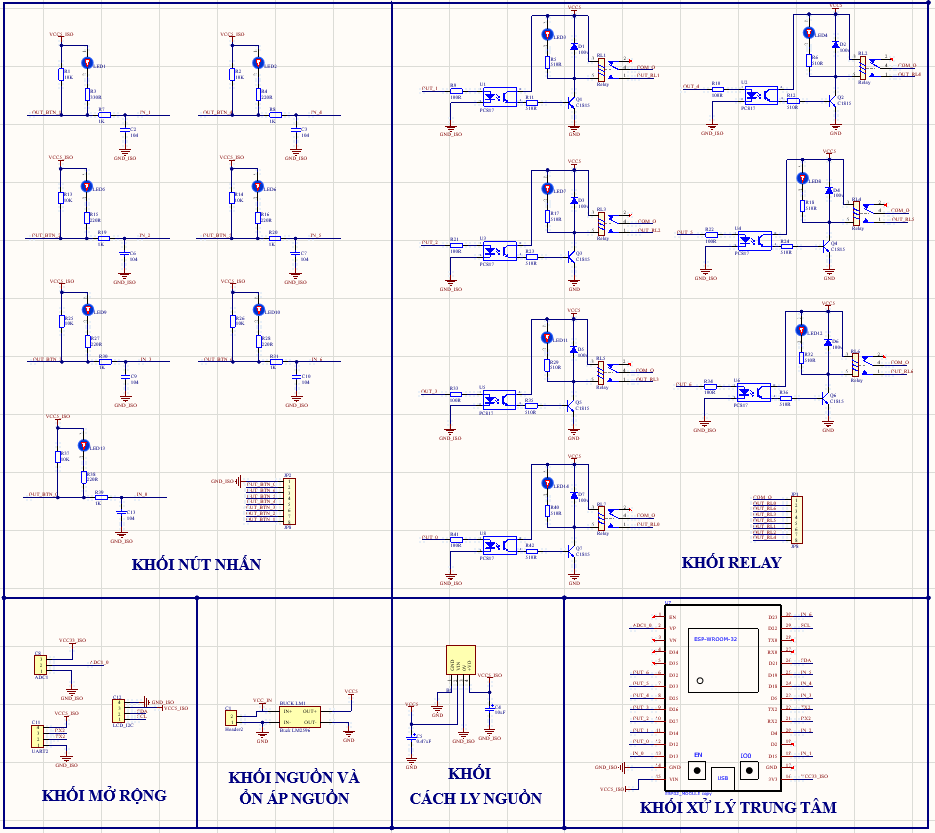
Cách ly tín hiệu của mạch điều khiển với relay, tránh bị nhiễu từ relay.

* Khối relay:

Điều khiển thiết bị 1 chiều hoặc xoay chiều thông qua tiếp điểm.

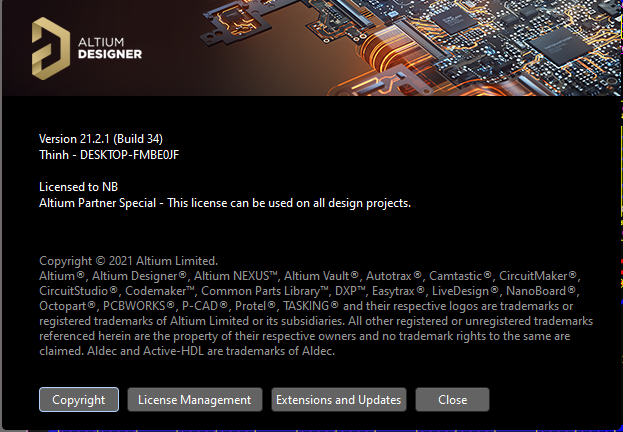
* Khối thiết bị:

Nhận tín hiệu từ relay và điều khiển thiết bị.

* 1. Thiết kế sơ đồ nguyên lý của mạch điện 

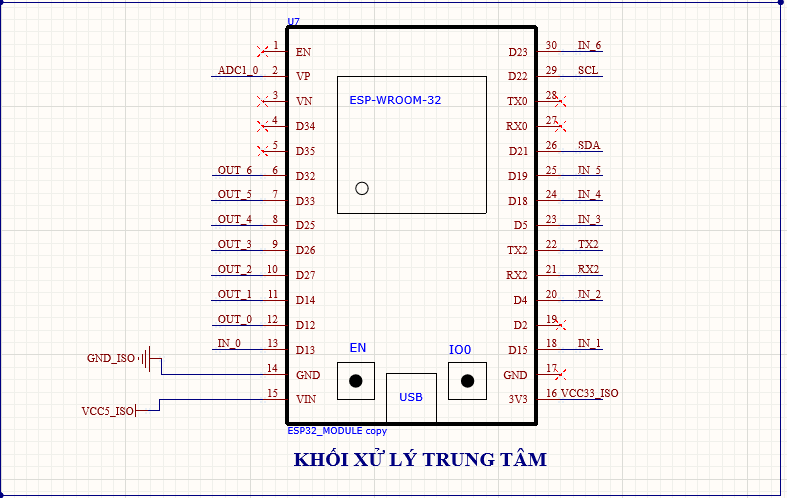
Hình 3.5: Sơ đồ nguyên lý mạch ESP32

Sử dụng phần mềm Altium Designer 21.2.1 để thiết kế mạch nguyên lý:

Theo sơ đồ nguyên lý hình 3.5: Nguồn điện sử dụng trong mạch là 12v. Nguồn 12V được lấy từ bộ chuyển đổi 220V về 12V cấp cho mạch hạ áp LM2596. Module hạ áp LM2596 sẽ biến đổi 12V về 5V. Sau khi biến đổi, nguồn 5V sẽ được cung cấp cho các tiếp điểm của Relay thông qua mạch khuếch đại dùng C1815. Ngoài ra 5V sẽ đi qua khối cách ly nguồn với IC cách ly nguồn B0505S. Đầu ra của khối cách ly nguồn được nuôi cho KIT ESP32. Người dùng truy cập vào trang Web và gửi các yêu cầu lên MQTT Broker. MQTT Broker gửi về cho ESP32 dữ liệu. ESP32 nhận dữ liệu, xử lý dữ liệu và xuất các tín hiệu điều khiển qua các chân. Khi chân vi điều khiển xuất ra mức cao, Opto quang sẽ dẫn và đưa tín hiệu vào mạch khuếch đại. Mạch khuếch đại sẽ đưa tín hiệu vào cuộn hút của Relay. Cuộn hút của Relay có điện hút tiếp điểm, thiết bị sẽ bật. Ngược lại với trường hợp chân điều khiển xuất ra mức thấp, thiết bị sẽ tắt.

Hình 3.6: Giao diện phần mềm Altium Designer

* + 1. Khối xử lý trung tâm



Hình 3.7: Khối xử lý trung tâm

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

ESP32 xử lý 7 tín hiệu đầu vào từ nút nhấn, 7 tín hiệu đầu ra rơ le. Ngoài ra, ESP32 trên mạch còn có thể xử lý các tín hiệu nếu muốn ghép nối như: ADC, I2C, USART …

Ngoài 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.

* + 1. Khối nguồn và ổn áp nguồn



Hình 3.8: Khối nguồn và ổn áp nguồn

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

* + 1. Khối cách ly nguồn

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

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

Hình 3.9: Khối cách ly nguồn

* + 1. 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.

Hình 3.10:Khối nút nhấn

* + 1. Khối Relay



Hình 3.11: Khối Relay

Khối Relay được điều khiển bởi các chân GPIO của vi điều khiển ESP32. Mạch sử dụng 7 relay để điều khiển các phụ tải.

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

Mục đích dùng Opto quang kết hợp với dùng cách ly nguồn nhằm đảm bảo chống nhiễu tối đa và an toàn cho mạch điều khiển khi có sự cố từ thiết bị mà Relay điều khiển.

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

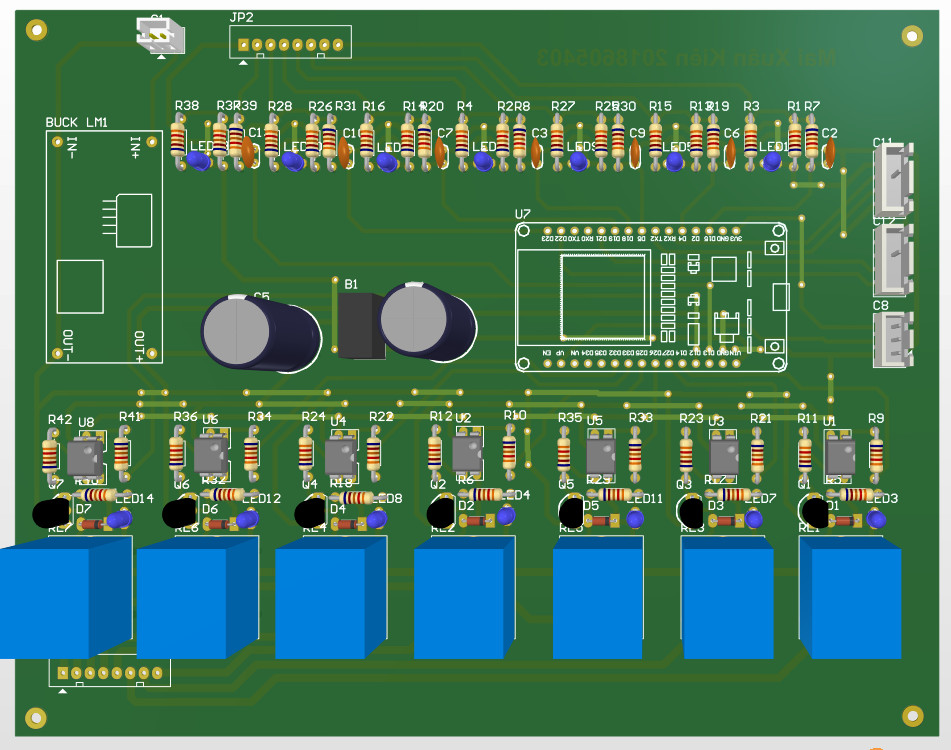
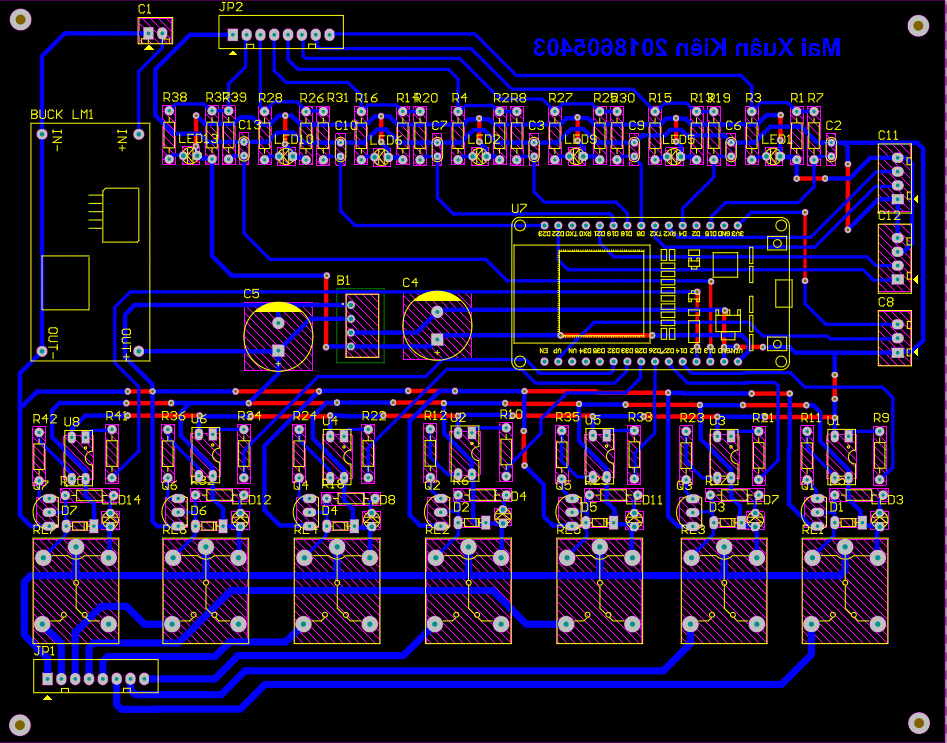
Do tín hiệu lấy ra từ vi điều khiển và lấy ra từ Opto quang không đủ, mạch sử dụng thêm các mạch khuếch đại với transistor C1815 mắc phân cực cố định nhằm tăng dòng điều khiển cho cuộn hút của Relay.

* + 1. Các header mở rộng

Bao gồm 3 header để mở rộng thêm cho mạch:

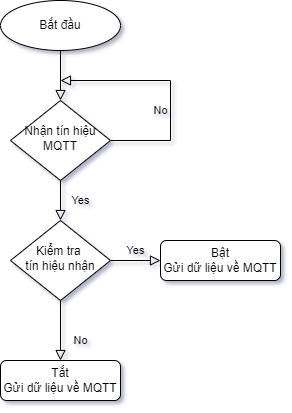
Hình 3.12: Khối mở rộng

* ADC: Mở rộng ngõ vào ADC giúp mạch đọc các cảm biến, các ngoại vi có trang bị ADC.
* UART: Nhằm giao tiếp giữa các mạch hoặc giữa mạch với các thiết bị ngoại vi khác có UART.
* I2C: Chủ yếu để giao tiếp với màn hình LCD I2C.
  1. Thiết kế mạch in (PCB)

Sử dụng phần mềm Altium Designer 21.2.1 để thiết kế mạch in (PCB) cho đề tài.

Hình 3.14: Sơ đồ mạch in 3D

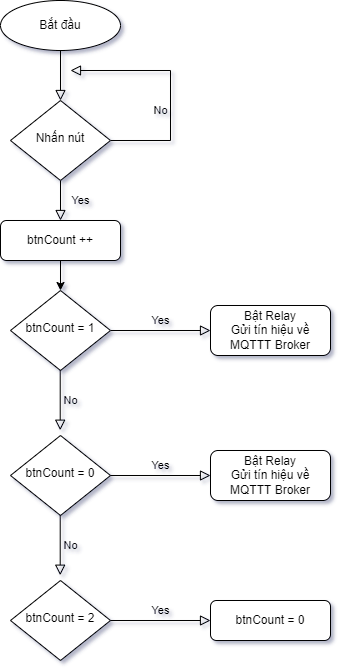
Hình 3.13: Sơ đồ mạch in 2D

* 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> [5].
* 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
* Khi nhấn nút:

Hình 3.15: Lưu đồ thuật toán xử lý sự kiện nút nhấn

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

Quá trình được mô tả như hình 3.13:

* 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

Hình 3.16: Lưu đồ thuật toán xử lý sự kiện nhận về từ MQTT Broker

Tín hiệu nhận được sẽ được xử lý thông qua một hàm callback() [12].

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.

* + 1. Xây dựng giao diện Web

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

*Sử dụng HTML dựng khung giao diện:*

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

<!DOCTYPE html>

<html>

<head>

<title>Page Title</title>

</head>

<body>

<h1>This is a Heading</h1>

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

</body>

</html>

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

Bảng 3.1: Một số thẻ HTML cơ bản

|  |  |
| --- | --- |
| 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;

}

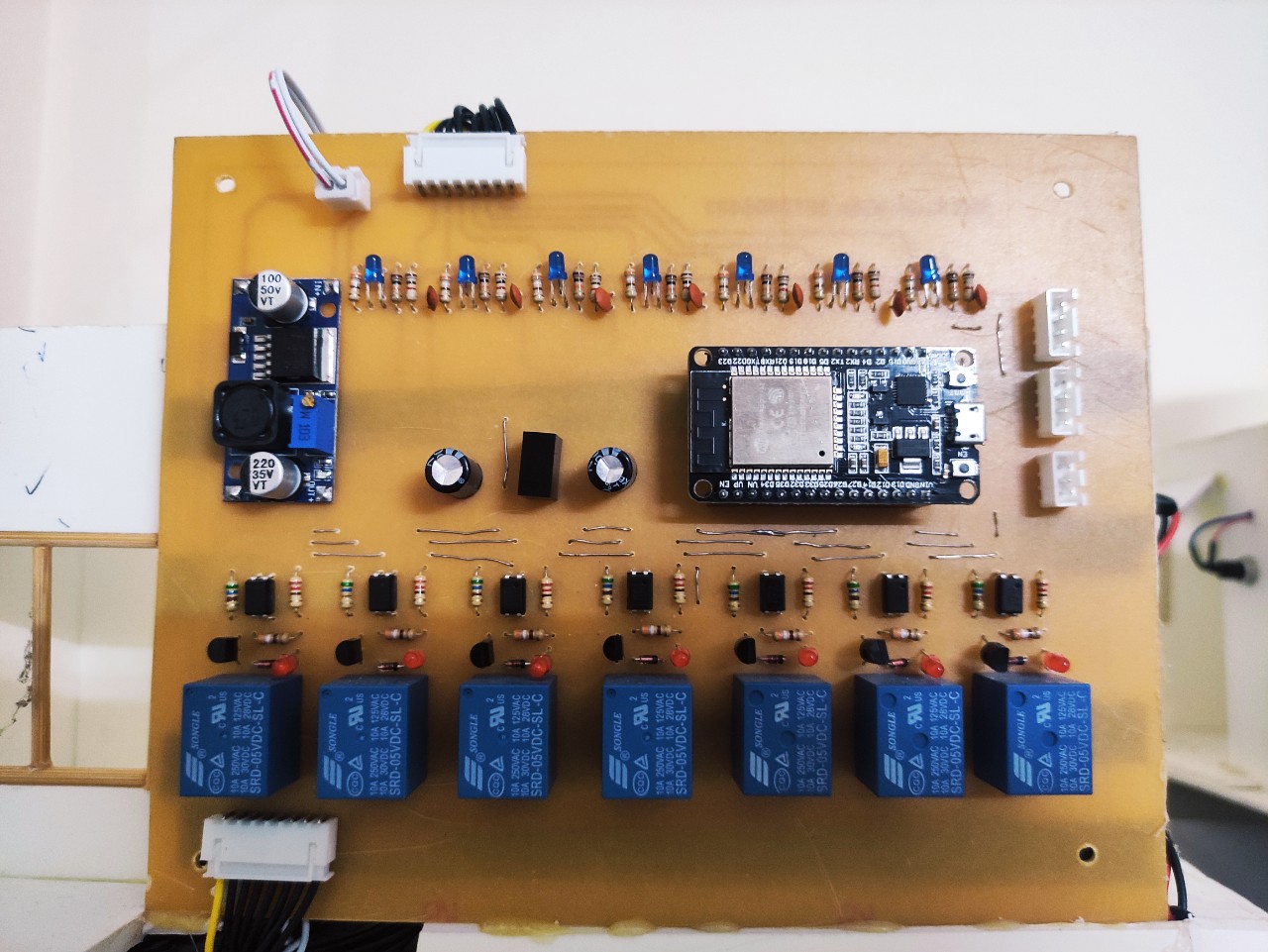
Thay đổi nền thẻ <body>

body {

background-color: lightblue;

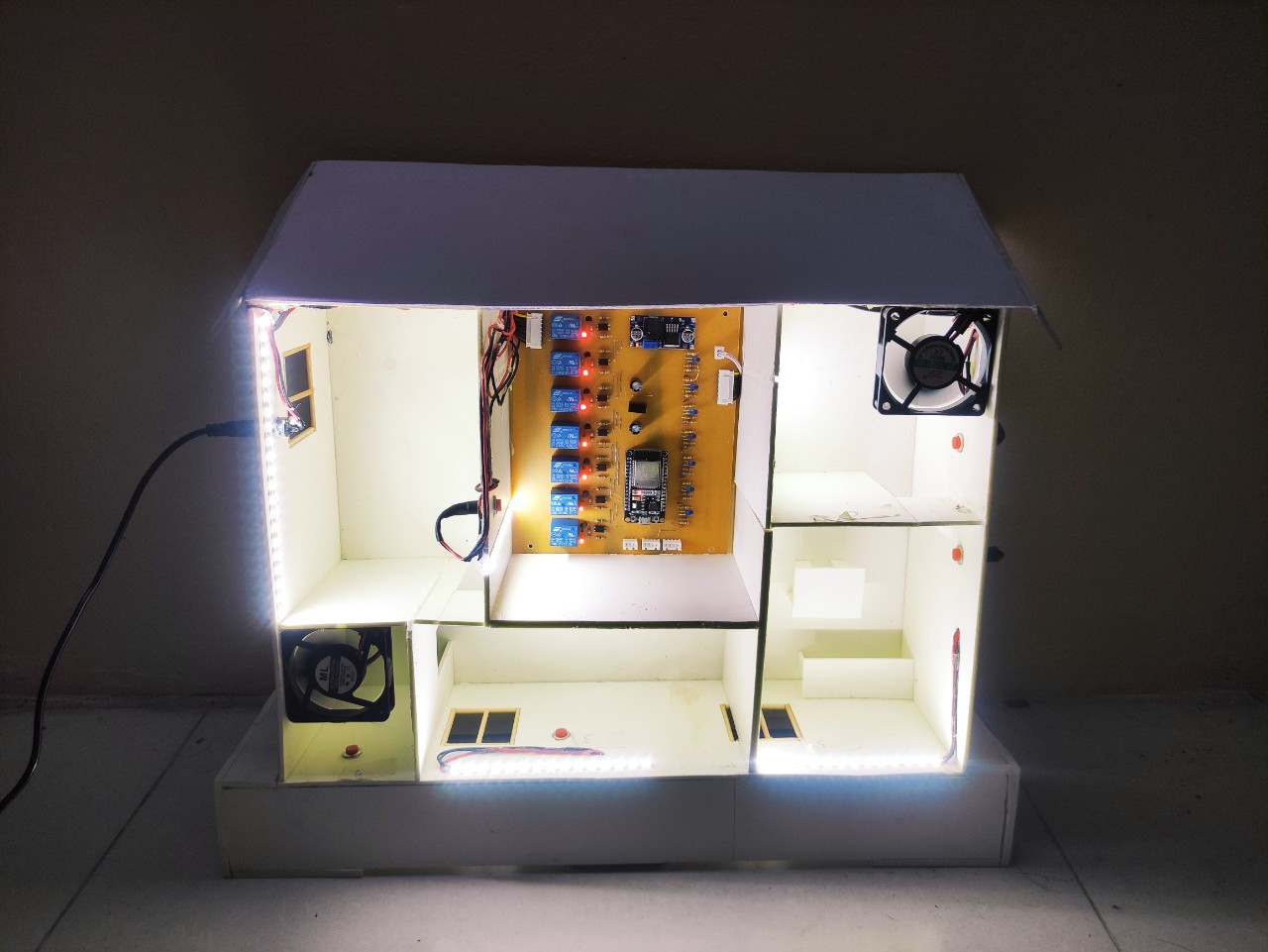
}

* 1. Kết quả đạt được
     1. Mạch sau khi lắp ráp



Hình 3.17: Mạch sau khi lắp ráp linh kiện

* + 1. Mô hình sản phẩm



Hình 3.18: Đề tài sau khi hoàn thành

* 1. Giao diện điều khiển trên Webserver

Hình 3.19: Giao diện điều khiển trên Webserver

Giao diện bao gồm 7 phòng và 1 chế độ khác:

Mỗi phòng có 2 nút nhấn bật và tắt đèn hoặc quạt.

Một chế độ khác bao gồm các nút nhấn: Bật tất cả quạt, tắt tất cả quạt, bật tất cả đèn và tắt tất cả đèn.

* 1. Kết quả thực nghiệm

Mô hình hoạt động theo 2 chế độ: chế độ điều khiển trực tiếp qua nút nhấn, chế độ điều khiển từ xa qua Webserver.

* + 1. Thực nghiệm 1
* Sử dụng chế độ 1: Chế độ điều khiển trực tiếp qua nút nhấn.



Hình 3.20: Nút nhấn điều khiển chế độ trực tiếp



Hình 3.21: Đèn phòng 3 bật khi nhấn nút lần 1

Bảng 3.2: Bật / tắt thiết bị bằng nút nhấn (thực nghiệm 1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Số Phòng** | **Phòng 01** | **Phòng 02** | **Phòng 03** | **Phòng 04** | **Phòng 05** | **Phòng 06** |
| **Thiết bị** | Đèn | Quạt | Đèn | Đèn | Đèn | Đèn, quạt |
| **Nhấn nút lần 1** | Bật | Bật | Bật | Bật | Bật | Bật |
| **Nhấn nút lần 2** | Tắt | Tắt | Tắt | Tắt | Tắt | Tắt |

**Nhận xét:** Khi sử dụng nút nhấn để bật tắt thiết bị, đèn và quạt hoạt động tốt độ trễ xấp xỉ bằng không.

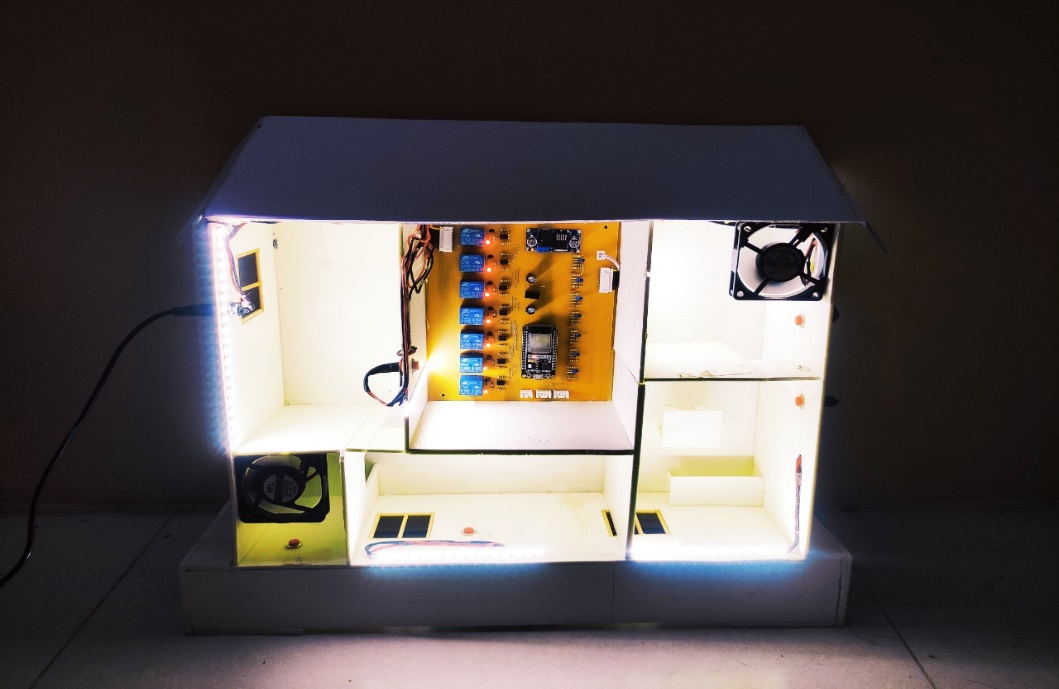
* Sử dụng chế độ 2: Điều khiển từ xa qua Webserver.

Hình 3.22: Giao diện khi điều khiển từ xa qua Webserver

Bảng 3.3: Bật / tắt thiết bị bằng Webserver (Thực nghiệm 1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Số Phòng** | **Phòng 01** | **Phòng 02** | **Phòng 03** | **Phòng 04** | **Phòng 05** | **Phòng 06** |
| **Thiết bị** | Đèn | Quạt | Đèn | Đèn | Đèn | Đèn, quạt |
| **Nhấn nút bật** | Bật | Bật | Bật | Bật | Bật | Bật |
| **Nhấn nút tắt** | Tắt | Tắt | Tắt | Tắt | Tắt | Tắt |

**Nhận xét:** Khi sử dụng Webserver để bật tắt các thiết bị điện, đèn và quạt hoạt động tốt độ trễ khoảng 1- 1,5s.



Hình 3.23: Mô hình hoạt động theo chế độ từ xa qua Webserver.

* + 1. Thực nghiệm 2

Sử dụng chế độ 1: Chế độ điều khiển trực tiếp qua nút nhấn.

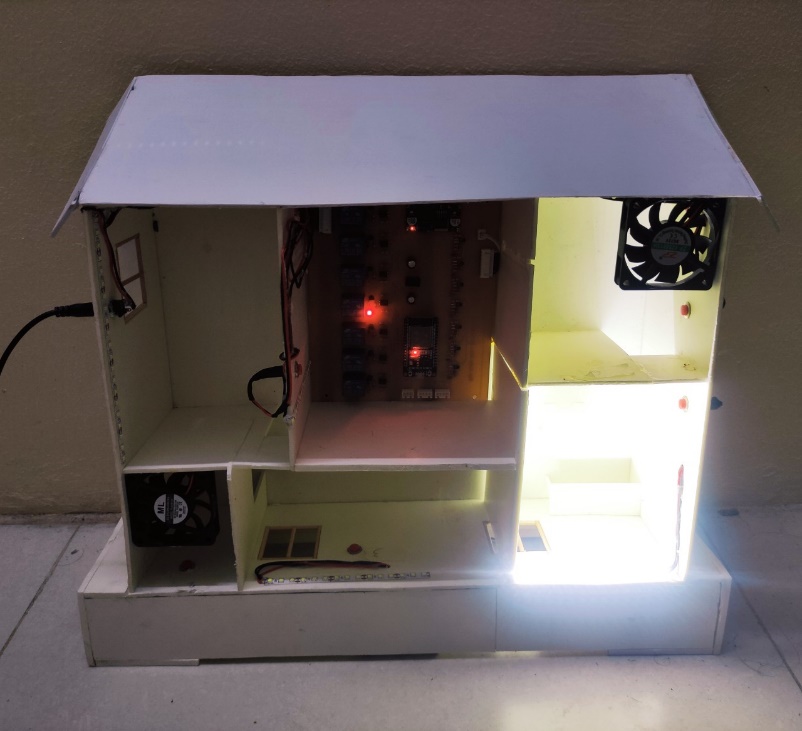


Hình 3.24: Nút nhấn điều khiển.

**Nhận xét:** Khi sử dụng nút nhấn để bật tắt thiết bị, đèn và quạt hoạt động tốt độ trễ xấp xỉ bằng không.

Bảng 3.4: Bật / tắt thiết bị bằng nút nhấn (Thực nghiẹm 2)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Số Phòng** | **Phòng 01** | **Phòng 02** | **Phòng 03** | **Phòng 04** | **Phòng 05** | **Phòng 06** |
| **Thiết bị** | Đèn | Quạt | Đèn | Đèn | Đèn | Đèn, quạt |
| **Nhấn nút lần 1** | Bật | Bật | Bật | Bật | Bật | Bật |
| **Nhấn nút lần 2** | Tắt | Tắt | Tắt | Tắt | Tắt | Tắt |

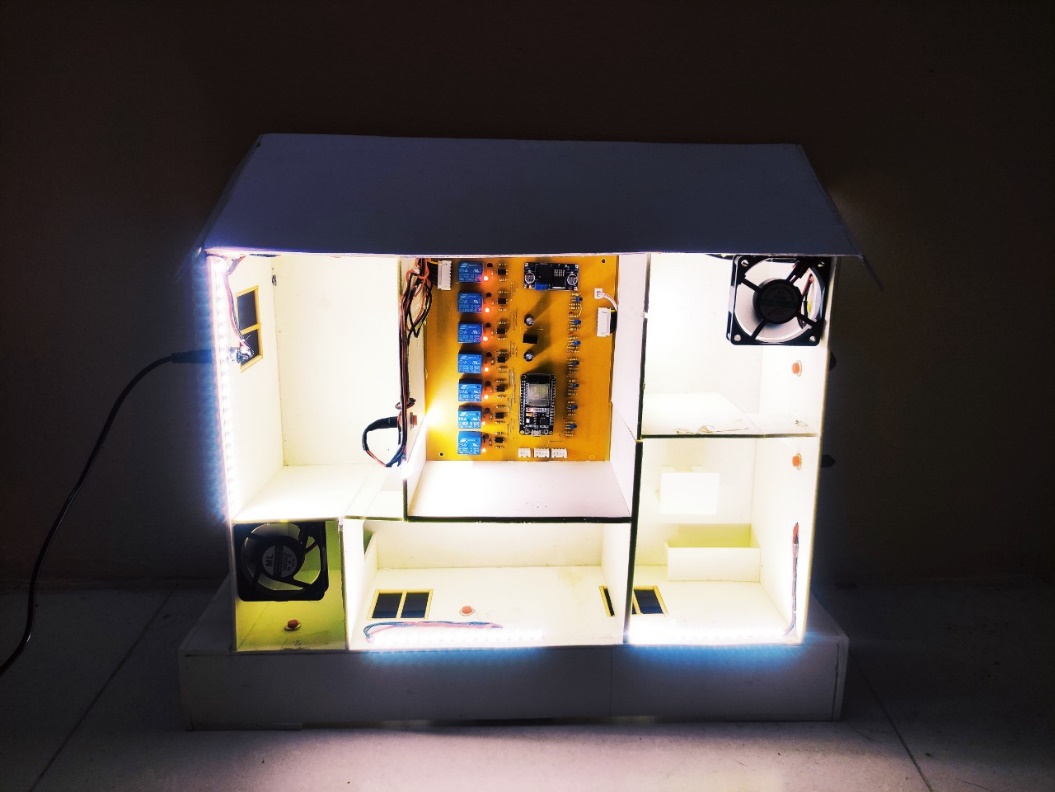


Hình 3.25: Đèn phòng 5 hoạt động trong chế độ 1.

Bảng 3.5: Một số thẻ HTML cơ bản

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Số Phòng** | **Phòng 01** | **Phòng 02** | **Phòng 03** | **Phòng 04** | **Phòng 05** | **Phòng 06** |
| **Thiết bị** | Đèn | Quạt | Đèn | Đèn | Đèn | Đèn, quạt |
| **Nhấn nút bật** | Bật | Bật | Bật | Bật | Bật | Bật |
| **Nhấn nút tắt** | Tắt | Tắt | Tắt | Tắt | Tắt | Tắt |

**Nhận xét:** Sử dụng trực tiếp nút nhấn, thiết bị được bật nhanh chóng, ổn định, độ trễ thấp.

Sử dụng chế độ 2: Điều khiển từ xa qua Webserver.

Hình 3.26: Mô hình hoạt động theo chế độ 2

**Nhận xét:** Sử dụng điều khiển từ xa qua Webserver hoạt động độ trễ bật tắt thiết bị với độ trễ thấp do đã thay đổi mạng khi kết nối trực tiếp với module wifi.

* 1. Nhận xét, đánh giá
     1. Phân tích, giải thích kết quả thực nghiệm

Mạch chạy ổn định với 5 nút nhấn điều khiển 5 đèn led và 2 nút nhấn điều khiển quạt. Khi nhấn nút nhấn vật lý thì trạng thái ở trên webserver cũng sẽ thay đổi theo bật tắt của bóng đèn hoặc quạt. Có thể nhấn liên tục nhiều nút nhấn một lúc để bật tắt nhiều đèn một lúc mà không ảnh hưởng tới chức năng của sản phẩm. Độ trễ khi nhấn nút trên web và nút nhấn vật lý là không đáng kể, hầu như là ngay lập tức. Thử nghiệm với việc bật tắt liên tục thì trạng thái không bị sai, không bị đổi lệch ở trên web.

* + 1. Tính năng và hiệu quả sử dụng của sản phẩm thiết kế
* Đề tài có tính năng điều khiển bật tắt và giám sát các thiết bị từ xa chỉ cần 1 thiết bị có kết nối internet.
* Mạch điện có 7 đầu vào và 7 đầu ra cho phép điều khiển 7 thiết bị với 7 nút nhấn khác nhau.
* Giao diện web điều khiển dễ dàng cài đặt và và thân thiện với người dùng.
* Đề tài có tính ứng dụng cao ngay cả trong nhà máy và trong các khu đô thị thông minh, các ngôi nhà thông minh.
  + 1. Tính ứng dụng, mức độ an toàn và tác động của sản phẩm thiết kế tới môi trường, kinh tế và xã hội
* Mạch điều khiển có cách ly, chống nhiễu nhiều tầng, vì vậy, mạch hạn chế tối đa việc cháy nổ hoặc chạm chập từ các thiết bị bên ngoài ảnh hưởng tới mạch điều khiển.
* Hệ sử dụng để điều khiển từ xa chỉ với một thiết bị có Internet, vì vậy hệ thống giúp giảm thiểu việc đi lại và chi phí giám sát ở nhiều công đoạn khác nhau với những khoảng cách về địa lý khác nhau.
* Đối tượng nghiên cứu: nghiên cứu về các ngoại vi của ESP32 và lập trình ESP32 sử dụng arduino IDE.
* Nghiên cứu về IoT và ứng dụng của IoT trong công nghiệp cũng như trong dân dụng.
* Nghiên cứu về giao thức MQTT để truyền nhận dữ liệu từ web tới vi điều khiển ESP32.
  1. Kết luận chương 3

Chương 3 nói về thiết kế hệ thống điều khiển và giám sát:

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

* Tìm hiểu các bước thực hiện và giải quyết một bài toán ứng dụng cụ thể.
* Thiết kế mạch và lập trình ứng dụng vào mạch thực tế.
* Khắc phục lỗi, sự cố của sản phẩm.
* Thử nghiệm và đánh giá kết quả sản phẩm.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Qua quá trình thực hiện làm đồ án, em đã trình bày các cơ sở lý thuyết liên quan và chạy thành công hệ thống **“Thiết kế hệ thống điều khiển và giám sát các thiết bị từ xa bằng WiFi sử dụng ESP32”.** Tuy thời gian làm đồ án thực sự không quá dài nhưng được sự giúp đỡ tận tình của ThS. Trần Xuân Phương 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.
* 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 vào việc điều khiển, truyền nhận tín hiệu và giám sát.
* Nắm được các hệ thống điều khiển và giám sát từ xa.
* Xây dựng được giao diện điều khiển và giám sát.
* Thiết kế và vận hành thành công mạch giám sát và điều khiển thiết bị qua WiFi sử dụng vi điều khiển ESP32.
* **Ưu điểm:**

**-** Hệ thống hoạt động ổn định, giao diện điều khiển dễ sử dụng.

- Chi phí thấp.

- Sử dụng đồng thời 2 chế độ vận hành: nút nhấn vật lý và Webserver.

* **Nhược điểm:**

- Mô hình với kích thước chưa tối ưu.

- Điều khiển còn được ít thiết bị, trạng thái điều khiển còn đơn giản chỉ bật và tắt.

- Điều khiển thông qua Webserver đôi khi chưa ổn định do tốc độ truyền phụ thuộc vào mạng.

* **Hướng phát triển đề tài:**
* 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: đề tài không chỉ giới hạn ở Web mà còn có thể phát triển để phù hợp với điện thoại Android và IOS.
* Phát triển các chức năng cấu hình ngay trên điện thoại.
* Phát triển nhiều giao thức hơn nữa thay vì chỉ sử dụng MQTT để trong tất cả các trường hợp đều có thể bật tắt các thiết bị.
* Phát triển không chỉ điều khiển các thiết bị bật hoặc tắt mà còn có thể điều khiển nhiều tín hiệu khác nữa.
* Điều khiển và giám sát thêm các thiết bị khác, các loại thiết bị có công suất lớn.

# TÀI LIỆU THAM KHẢO

[1] PhạmQuangHuy, LêCảnhTrung(2016), *Lập trình điều khiển với Arduino*, NXB Khoa học & Kỹ thuật.

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

[3] https://www.freertos.org/RTOS.html

[4] https://vn.got-it.ai/blog/phan-mem-arduino-ide-la-gi-chi-tiet-nhat

[5]https://www.espressif.com/sites/default/files/documentation/esp32\_datasheet\_en.pdf

[6] LoRa Alliance. White Paper: A Technical Overview of Lora and Lorawan; The LoRa Alliance: San Ramon, CA, USA, 2015

[7] 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

[8] IoTvietnam.com - Cộng đồng đam mê Internet of Thing.

[9] http://www.steves-internet-guide.com/mqtt-works/

[10] https://viblo.asia/p/mqtt-la-gi-vai-tro-cua-mqtt-trong-iot-V3m5WL3bKO7

[11] <https://developer.mozilla.org/en-US/docs/Web/HTTP>

[12] https://e-tranle.vn/giai-phap-giam-sat-dien-nang-power-monitoring/

[13] https://happysmarthome.vn/goi-lap-dat/

[14] https://truongsontech.vn/cong-trinh/he-thong-scada-nha-may-nuoc-phan-mem-dieu-khien-giam-sat-tu-xa-nha-may-nuoc-tai-vinh-phuc/

# PHỤ LỤC

HƯỚNG DẪN SỬ DỤNG MÔ HÌNH HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT CÁC THIẾT BỊ TỪ XA BẰNG WIFI SỬ DỤNG ESP32

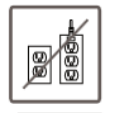
**1.** **Cảnh báo về an toàn**

Bạn phải đọc các cảnh báo này. Việc lơ là các cảnh báo này có thể gây thương vong cho bạn và người thân.

**Nguy cơ điện giật!**



* Không sử dụng ổ cắm nhiều lỗ hoặc dây nối dài



* Không cắm vào ổ cắm bị hỏng hoặc ổ cắm có dây bị hở
* Không cắm hoặc rút phích cắm ra khỏi ổ cắm điện bằng tay ướt để ngăn điện giật
* Tuyệt đối không chạm vào mạch sản phẩm bằng tay hoặc chân ướt hoặc ẩm



* Tuyệt đối không kéo dây khi rút phích cắm, nắm phích cắm khi rút phích cắm



**2. Tổng quan về thiết bị**

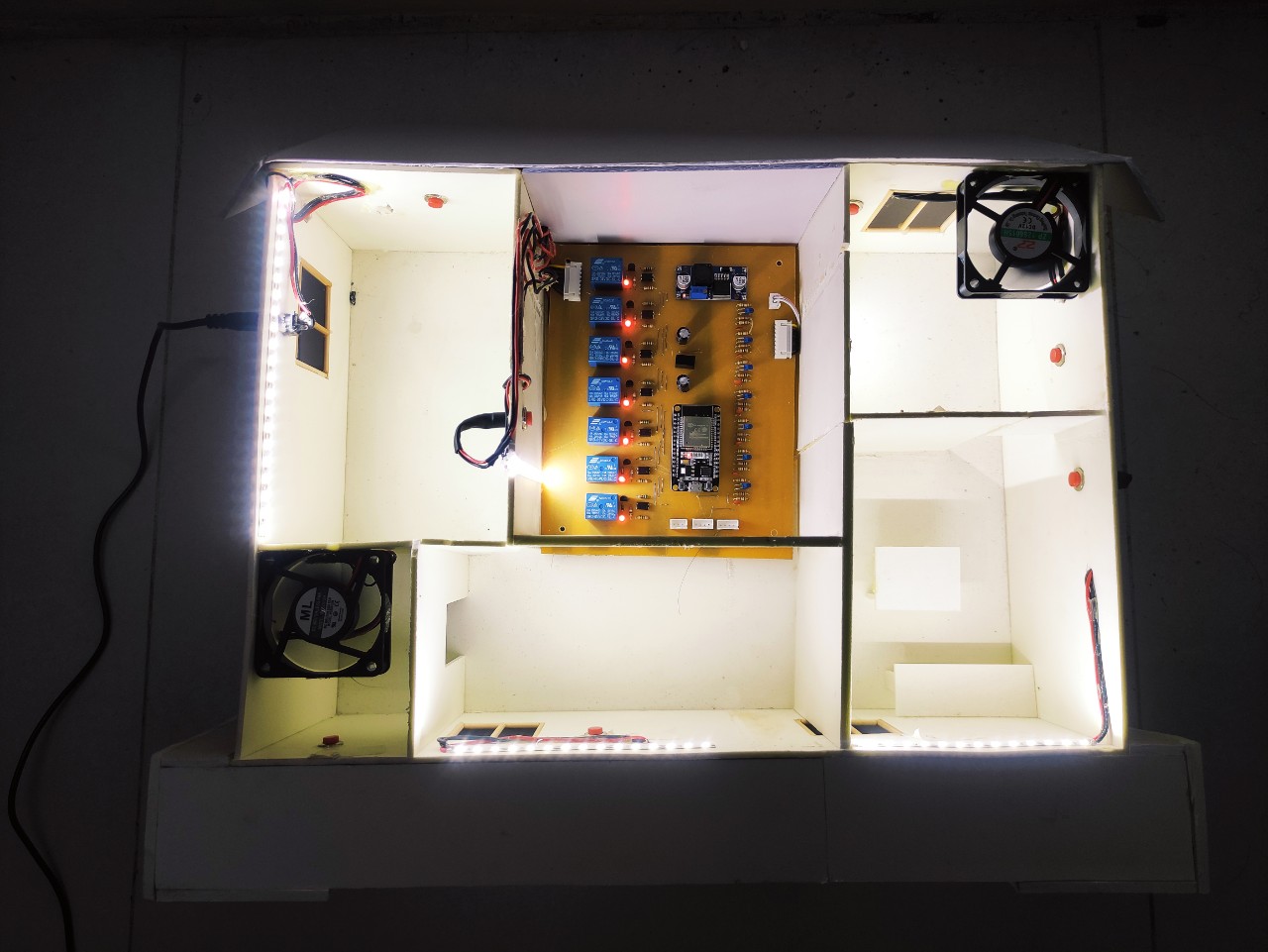
* **Hình thức tổng thể**

Jack cắm nguồn

Nút nhấn phòng 01

Nút nhấn và đèn phòng 06

Nút nhấn và quạt phòng 06



Quạt phòng 02

Nút nhấn quạt phòng 02

Nút nhấn, đèn phòng 04

Nút nhấn và đèn phòng 03

Nút nhấn đèn phòng 05

Đèn phòng 05

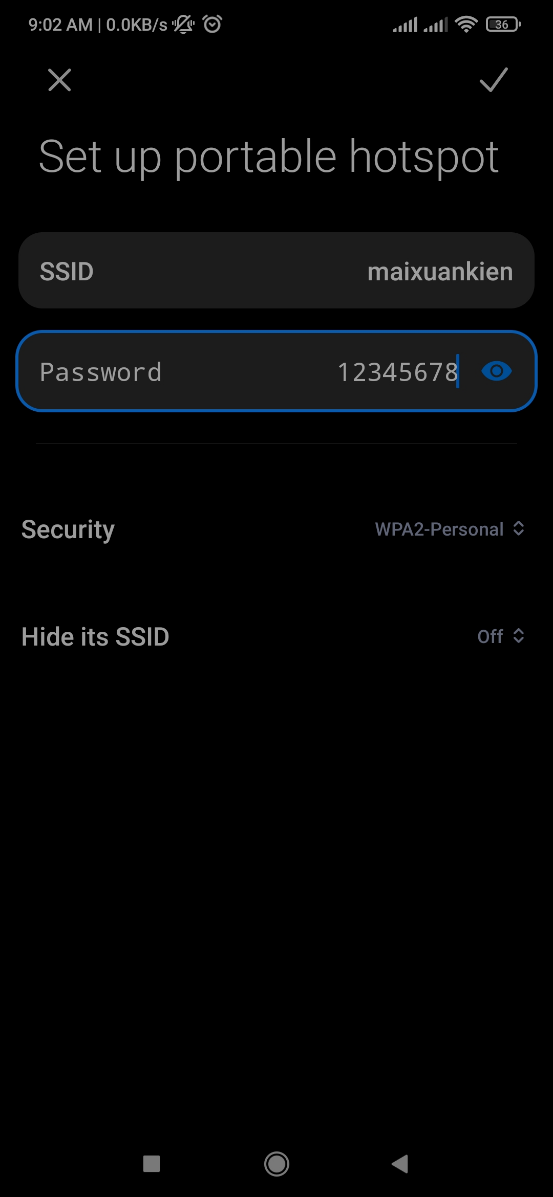
* **Chức năng của mô hình hệ thống:**

Sản phẩm “Thiết kế hệ thống điều khiển và giám sát các thiết bị từ xa bằng WiFi sử dụng ESP32” được thiết kế với một số tính năng như sau:

* Mạch điện có 7 đầu vào và 7 đầu ra cho phép điều khiển bật/ tắt 7 thiết bị với 7 nút nhấn khác nhau.
* Điều khiển và giám sát các thiết bị qua Giao diện web.

**3. Thao tác thực hiện**

Bước 1: Dùng thiết bị phát wifi phát wifi với các thông tin như sau:

* Tên wifi: “maixuankien”.
* Mật khẩu: “12345678”.
* Khi mô hình khởi đọng sẽ tự động kết nối với wifi đã cài đặt như trên.

Hình 1: Giao diện phát wifi

Bước 2: Cấp nguồn 12V qua Jack Dc 5.5 cho hệ thống.



Hình 2: Vị trí cấp nguồn 12v

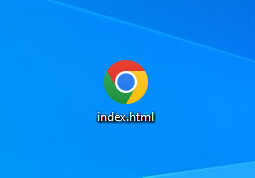
Bước 3: Bật tắt các thiết bị bằng các nút nhấn trên mô hình.

Hình 3: Loại nút nhấn và vị trí nút nhấn trên mô hình



Nhấn và tắt thiết bị được tích hợp trên một nút nhấn. Nhấn nút trên tùng phòng để điều khiển các thiết bị bên trong.

Bước 4: Mở file index.html, giao diện điều khiển sẽ hiện lên bằng các trình duyệt mà máy tính đã cài.



Hình 4: File index.html

Bước 5: Giao diện hiện lên trên màn hình máy tính, thao tác bật tắt các thiết bị bằng cách bật tắt các nút nhấn trên giao diện.

Hình 5: Giao diện Webserver

**4. Các chỉ dẫn tình huống và các bước thực hiện**

**Bảng xử lý sự cố:**

|  |  |  |
| --- | --- | --- |
| **Lỗi** | **Nguyên nhân có thể xảy ra** | **Phương pháp giải quyết** |
| Điều khiển qua wed nhưng mô hình không hoạt động hoặc trễ | * Mạng kết nối không ổn định hoặc mất mạng | * - Kiểm tra modem WiFi   - Thay đổi mạng kết nối |
| Không lên nguồn trong mạch | Lỗi adapter | Thay adapter mới |

**5. Các lỗi khuyến cáo và cảnh báo**

* Cảnh báo cháy nổ nếu cấp nguồn quá lớn cho mô hình.
* Mô hình không có tính năng chống nước, tránh tiếp xúc trực tiếp ánh sáng mặt trời, nơi có độ ẩm cao.

**6. Thông tin liên hệ hỗ trợ**

* Họ và tên: Mai Xuân Kiên
* Email: [maixuankien1234@gmail.com](mailto:maixuankien1234@gmail.com)
* Sđt: 0987394335

Code chương trình

#include <Arduino.h>

#include <PubSubClient.h>

#include <WiFi.h>

//ota update

#include <AsyncTCP.h>

#include <ESPAsyncWebServer.h>

#include <AsyncElegantOTA.h>

#include <ArduinoJson.h>

//define pin to use

//input

#define BTN\_01 15

#define BTN\_02 4

#define BTN\_03 5

#define BTN\_04 18

#define BTN\_05 19

#define BTN\_06 23

#define BTN\_07 13

//output

#define OUT\_01 14

#define OUT\_02 27

#define OUT\_03 26

#define OUT\_04 25

#define OUT\_05 33

#define OUT\_06 12

#define OUT\_07 32

//define wifi

#define ssid "maixuankien"//"WIFi bỊ nhiễm virut"

#define password "123456789"//"minhdien04"

//define mqtt

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

#define mqtt\_user ""

#define mqtt\_pwd ""

const uint16\_t mqtt\_port = 1883;

String mqtt\_topic\_pub = "maixuankien2018605403/iotBtnInfo";

String mqtt\_topic\_sub = "maixuankien2018605403/iotRelayControl";

//user variable

long lastMsg = 0;

char msg[50];

int value = 0;

int dataInt = 0;

String Data = "";

String ChuoiSendWebJson = "";

float nhietdo = 0;

unsigned long last = 0, bien = 0;

char inforInverterBuff[256];

//define

int button1PressCount = 0;

int button2PressCount = 0;

int button3PressCount = 0;

int button4PressCount = 0;

int button5PressCount = 0;

int button6PressCount = 0;

int button7PressCount = 0;

WiFiClient espClient;

PubSubClient client(espClient);

AsyncWebServer server(80);

//array of pin

const int inputPinArr[7] = {BTN\_01, BTN\_02, BTN\_03, BTN\_04, BTN\_05, BTN\_06, BTN\_07};

const int outputPinArr[7] = {OUT\_01, OUT\_02, OUT\_03, OUT\_04, OUT\_05, OUT\_06, OUT\_07};

//function define

void pinInit(int pinNumberInput, int pinNumberOnput);

void buttonInit();

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

void setupWifi();

void reconnect();

void updateOTAInit();

//task init

void taskInit();

//button 1 task

void handleButton1Task();

void handleButton1(void \*parameter);

//button 2 task

void handleButton2Task();

void handleButton2(void \*parameter);

//button 3 task

void handleButton3Task();

void handleButton3(void \*parameter);

//button 4 task

void handleButton4Task();

void handleButton4(void \*parameter);

//button 5 task

void handleButton5Task();

void handleButton5(void \*parameter);

//button 6 task

void handleButton6Task();

void handleButton6(void \*parameter);

//button 7 task

void handleButton7Task();

void handleButton7(void \*parameter);

//mqtt handle

void handleMQTTTask();

void handleMQTT(void \*parameter);

void dataToBuffer();

void setup() {

Serial.begin(115200);

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

client.setCallback(callback);

pinInit(7, 7);

taskInit();

setupWifi();

updateOTAInit();

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

}

void pinInit(int pinNumberInput, int pinNumberOnput)

{

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

{

pinMode(inputPinArr[i], INPUT);

}

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

{

pinMode(outputPinArr[i], OUTPUT);

}

}

void buttonInit()

{

}

void taskInit()

{

handleButton1Task();

handleButton2Task();

handleButton3Task();

handleButton4Task();

handleButton5Task();

handleButton6Task();

handleButton7Task();

handleMQTTTask();

}

void handleButton1Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton1, // Function to be called

"handleButton1", // Name of task

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

NULL, // Parameter to pass to function

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

NULL, // Task handle

1);

}

void handleButton2Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton2, // Function to be called

"handleButton2", // Name of task

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

NULL, // Parameter to pass to function

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

NULL, // Task handle

1);

}

void handleButton3Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton3, // Function to be called

"handleButton3", // Name of task

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

NULL, // Parameter to pass to function

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

NULL, // Task handle

1);

}

void handleButton4Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton4, // Function to be called

"handleButton4", // Name of task

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

NULL, // Parameter to pass to function

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

NULL, // Task handle

1);

}

void handleButton5Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton5, // Function to be called

"handleButton5", // Name of task

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

NULL, // Parameter to pass to function

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

NULL, // Task handle

1);

}

void handleButton6Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton6, // Function to be called

"handleButton6", // Name of task

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

NULL, // Parameter to pass to function

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

NULL, // Task handle

1);

}

void handleButton7Task()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleButton7, // Function to be called

"handleButton7", // Name of task

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

NULL, // Parameter to pass to function

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

NULL, // Task handle

1);

}

void handleMQTTTask()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

handleMQTT, // Function to be called

"handleMQTT", // Name of task

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

NULL, // Parameter to pass to function

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

NULL, // Task handle

1);

}

void handleButton1(void \*parameter) {

while (1) {

while (digitalRead(BTN\_01) == HIGH);

while (digitalRead(BTN\_01) == LOW);

button1PressCount ++;

if (button1PressCount == 2)

{

button1PressCount = 0;

}

if (button1PressCount == 1)

{

digitalWrite(OUT\_01, HIGH);

// dataToBuffer();

}

else if (button1PressCount == 0)

{

digitalWrite(OUT\_01, LOW);

// dataToBuffer();

}

}

}

//handle button 2

void handleButton2(void \*parameter) {

while (1) {

while (digitalRead(BTN\_02) == HIGH);

while (digitalRead(BTN\_02) == LOW);

button2PressCount ++;

if (button2PressCount == 2)

{

button2PressCount = 0;

}

if (button2PressCount == 1)

{

digitalWrite(OUT\_02, HIGH);

// dataToBuffer();

}

else if (button2PressCount == 0)

{

digitalWrite(OUT\_02, LOW);

// dataToBuffer();

}

}

}

//handle button 3

void handleButton3(void \*parameter) {

while (1) {

while (digitalRead(BTN\_03) == HIGH);

while (digitalRead(BTN\_03) == LOW);

button3PressCount ++;

if (button3PressCount == 2)

{

button3PressCount = 0;

}

if (button3PressCount == 1)

{

digitalWrite(OUT\_03, HIGH);

// dataToBuffer();

}

else if (button3PressCount == 0)

{

digitalWrite(OUT\_03, LOW);

// dataToBuffer();

}

}

}

void handleButton4(void \*parameter) {

while (1) {

while (digitalRead(BTN\_04) == HIGH);

while (digitalRead(BTN\_04) == LOW);

button4PressCount ++;

if (button4PressCount == 2)

{

button4PressCount = 0;

}

if (button4PressCount == 1)

{

digitalWrite(OUT\_04, HIGH);

// dataToBuffer();

}

else if (button4PressCount == 0)

{

digitalWrite(OUT\_04, LOW);

// dataToBuffer();

}

}

}

void handleButton5(void \*parameter) {

while (1) {

while (digitalRead(BTN\_05) == HIGH);

while (digitalRead(BTN\_05) == LOW);

button5PressCount ++;

if (button5PressCount == 2)

{

button5PressCount = 0;

}

if (button5PressCount == 1)

{

digitalWrite(OUT\_05, HIGH);

// dataToBuffer();

}

else if (button5PressCount == 0)

{

digitalWrite(OUT\_05, LOW);

// dataToBuffer();

}

}

}

void handleButton6(void \*parameter) {

while (1) {

while (digitalRead(BTN\_06) == HIGH);

while (digitalRead(BTN\_06) == LOW);

button6PressCount ++;

if (button6PressCount == 2)

{

button6PressCount = 0;

}

if (button6PressCount == 1)

{

digitalWrite(OUT\_06, HIGH);

// dataToBuffer();

}

else if (button6PressCount == 0)

{

digitalWrite(OUT\_06, LOW);

// dataToBuffer();

}

}

}

void handleButton7(void \*parameter) {

while (1) {

while (digitalRead(BTN\_07) == HIGH);

while (digitalRead(BTN\_07) == LOW);

button7PressCount ++;

if (button7PressCount == 2)

{

button7PressCount = 0;

}

if (button7PressCount == 1)

{

digitalWrite(OUT\_07, HIGH);

// client.publish(mqtt\_topic\_pub.c\_str(), "71");

}

else if (button7PressCount == 0)

{

digitalWrite(OUT\_07, LOW);

// client.publish(mqtt\_topic\_pub.c\_str(), "70");

}

}

}

void handleMQTT(void \*parameter) {

while (1)

{

dataToBuffer();

vTaskDelay(2000/portTICK\_PERIOD\_MS);

}

}

void reconnect()

{

while (!client.connected())

{

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

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

{

Serial.println("Connected MQTT ngoinhaiot.com");

client.subscribe(mqtt\_topic\_sub.c\_str());

digitalWrite(16, HIGH);

}

else

{

Serial.println("Không thể kết nối MQTT ngoinhaiot.com");

digitalWrite(16, LOW);

delay(5000);

}

}

}

//call back mqtt

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

{

Data = "";

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

{

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

}

dataInt = Data.toInt();

Serial.println(dataInt);

///button 1

if (dataInt == 11)

{

button1PressCount = 1;

digitalWrite(OUT\_01, HIGH);

// client.publish(mqtt\_topic\_pub.c\_str(), "11");

}

else if (dataInt == 10)

{

button1PressCount = 0;

digitalWrite(OUT\_01, LOW);

// client.publish(mqtt\_topic\_pub.c\_str(), "10");

}

///button 2

else if (dataInt == 21)

{

button2PressCount = 1;

digitalWrite(OUT\_02, HIGH);

// client.publish(mqtt\_topic\_pub.c\_str(), "21");

}

else if (dataInt == 20)

{

button2PressCount = 0;

digitalWrite(OUT\_02, LOW);

// client.publish(mqtt\_topic\_pub.c\_str(), "20");

}

///button 3

else if (dataInt == 31)

{

button3PressCount = 1;

digitalWrite(OUT\_03, HIGH);

// client.publish(mqtt\_topic\_pub.c\_str(), "31");

}

else if (dataInt == 30)

{

button3PressCount = 0;

digitalWrite(OUT\_03, LOW);

// client.publish(mqtt\_topic\_pub.c\_str(), "30");

}

///button 4

else if (dataInt == 41)

{

button4PressCount = 1;

digitalWrite(OUT\_04, HIGH);

// client.publish(mqtt\_topic\_pub.c\_str(), "41");

}

else if (dataInt == 40)

{

button4PressCount = 0;

digitalWrite(OUT\_04, LOW);

// client.publish(mqtt\_topic\_pub.c\_str(), "40");

}

///button 5

else if (dataInt == 51)

{

button5PressCount = 1;

digitalWrite(OUT\_05, HIGH);

// client.publish(mqtt\_topic\_pub.c\_str(), "51");

}

else if (dataInt == 50)

{

button5PressCount = 0;

digitalWrite(OUT\_05, LOW);

// client.publish(mqtt\_topic\_pub.c\_str(), "50");

}

///button 6

else if (dataInt == 61)

{

button6PressCount = 1;

digitalWrite(OUT\_06, HIGH);

// client.publish(mqtt\_topic\_pub.c\_str(), "61");

}

else if (dataInt == 60)

{

button6PressCount = 0;

digitalWrite(OUT\_06, LOW);

// client.publish(mqtt\_topic\_pub.c\_str(), "60");

}

///button 7

else if (dataInt == 71)

{

button7PressCount = 1;

digitalWrite(OUT\_07, HIGH);

// client.publish(mqtt\_topic\_pub.c\_str(), "71");

}

else if (dataInt == 70)

{

button7PressCount = 0;

digitalWrite(OUT\_07, LOW);

// client.publish(mqtt\_topic\_pub.c\_str(), "70");

}

}

//set up wifi

void setupWifi() {

delay(10);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

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

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

void updateOTAInit()

{

server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request) {

request->send(200, "text/plain", "Hi! I am ESP32.");

});

AsyncElegantOTA.begin(&server); // Start ElegantOTA

server.begin();

Serial.println("HTTP server started");

}

void dataToBuffer()

{

DynamicJsonDocument doc(1024);

doc["den1"] = button1PressCount;

doc["den2"] = button2PressCount;

doc["den3"] = button3PressCount;

doc["den4"] = button4PressCount;

doc["den5"] = button5PressCount;

doc["den6"] = button6PressCount;

doc["den7"] = button7PressCount;

serializeJson(doc, inforInverterBuff);

client.publish(mqtt\_topic\_pub.c\_str(), inforInverterBuff);

}