# 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 bộ môn Điện Tử Viễn Thông 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 Th.S 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 quý báu, 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.

Hà Nội, Ngày… tháng… năm 2022

Sinh viên thực hiện

Mai Xuân Kiên

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc103331467)

[MỤC LỤC 2](#_Toc103331468)

[DANH MỤC HÌNH ẢNH 5](#_Toc103331469)

[DANH MỤC BẢNG BIỂU 7](#_Toc103331470)

[LỜI MỞ ĐẦU 8](#_Toc103331471)

[CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT từ xa 10](#_Toc103331472)

[1.1 Lý do lựa chọn đề tài 10](#_Toc103331473)

[1.1.1 Đặt vấn đề 10](#_Toc103331474)

[1.1.2 Giới thiệu chung về đề tài 10](#_Toc103331475)

[1.2 Vai trò và ý nghĩa của đề tài 11](#_Toc103331476)

[1.3 Công nghệ sử dụng để điều khiển và giám sát từ xa hiện nay 11](#_Toc103331477)

[1.3.1 Wifi và Internet 11](#_Toc103331478)

[1.3.2 Bluetooth và Bluetooth Low Engine (BLE) 12](#_Toc103331479)

[1.3.3 Zigbee 12](#_Toc103331480)

[1.3.4 LoRa 12](#_Toc103331481)

[1.4 Các hệ thống điều khiển và giám sát ngoài thị trường 13](#_Toc103331482)

[1.4.1 Hệ thống điều khiển giám sát điện năng sử dụng PLC 13](#_Toc103331483)

[1.4.2 Hệ thống điều khiển và giám sát nhà thông minh 13](#_Toc103331484)

[1.4.3 Hệ thống điều khiển và giám sát nhà máy nước 14](#_Toc103331485)

[1.5 Kết luận chương 1 14](#_Toc103331486)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 15](#_Toc103331487)

[2.1 Các giao thức chính được sử dụng 15](#_Toc103331488)

[2.1.1 Giao thức WiFi 15](#_Toc103331489)

[2.1.2 Giao thức HTTP 17](#_Toc103331490)

[2.1.3 Giao thức MQTT 18](#_Toc103331491)

[2.2 Linh kiện và phần mềm thiết kế mạch 18](#_Toc103331492)

[2.2.1 Vi điều khiển ESP32 18](#_Toc103331493)

[2.2.2 IC cách ly nguồn B5050S 22](#_Toc103331494)

[2.2.3 Mạch Buck LM2596 23](#_Toc103331495)

[2.2.4 Transistor C1815 23](#_Toc103331496)

[2.2.5 Opto quang PC817 24](#_Toc103331497)

[2.2.6 Diode 1N4148 24](#_Toc103331498)

[2.3 Relay 5v 25](#_Toc103331499)

[2.3.1 Phần mềm thiết kế Altium Designer 26](#_Toc103331500)

[2.4 Phần mềm của hệ thống 28](#_Toc103331501)

[2.4.1 HTML (HyperText Markup Language) 28](#_Toc103331502)

[2.4.2 CSS (Cascading Style Sheets) 28](#_Toc103331503)

[2.4.3 JavaScript 29](#_Toc103331504)

[2.4.4 Phần mềm Visual Studio code 29](#_Toc103331505)

[2.4.5 Phần mềm Arduino IDE 30](#_Toc103331506)

[2.4.6 Hệ điều hành FreeRTOS 31](#_Toc103331507)

[2.5 Kết luận chương 2 32](#_Toc103331508)

[CHƯƠNG 3: THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT 34](#_Toc103331509)

[3.1 Thiết kế sơ đồ khối hệ thống điều khiển và giám sát 34](#_Toc103331510)

[3.1.1 Yêu cầu 34](#_Toc103331511)

[3.1.2 Mô tả hoạt động 34](#_Toc103331512)

[3.1.3 Sơ đồ khối 35](#_Toc103331513)

[3.2 Thiết kế sơ đồ nguyên lý của mạch điện 37](#_Toc103331514)

[3.2.1 Khối xử lý trung tâm 39](#_Toc103331515)

[3.2.2 Khối nguồn và ổn áp nguồn 40](#_Toc103331516)

[3.2.3 Khối cách ly nguồn 40](#_Toc103331517)

[3.2.4 Khối nút nhấn 41](#_Toc103331518)

[3.2.5 Khối Relay 41](#_Toc103331519)

[3.2.6 Các header mở rộng 42](#_Toc103331520)

[3.3 Thiết kế mạch in (PCB) 43](#_Toc103331521)

[3.3.1 Mạch in sau khi thiết kế 43](#_Toc103331522)

[3.4 Lập trình vi xử lý ESP32 43](#_Toc103331523)

[3.4.1 Các thư viện được sử dụng 43](#_Toc103331524)

[3.4.2 Xử lý sự kiện nhấn nút 44](#_Toc103331525)

[3.4.3 Xử lý sự kiện nhận về từ MQTT Broker 45](#_Toc103331526)

[3.4.4 Xây dựng giao diện Web 46](#_Toc103331527)

[3.5 Kết luận chương 3 47](#_Toc103331528)

[CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC 48](#_Toc103331529)

[4.1 Một số hình ảnh thực tế của hệ thống 48](#_Toc103331530)

[4.2 Giao diện điều khiển trên Webserver 49](#_Toc103331531)

[4.3 Hướng dẫn sử dụng 49](#_Toc103331532)

[4.4 Hướng phát triển của đề tài 51](#_Toc103331533)

[4.5 Kết luận chương 4 51](#_Toc103331534)

[KẾT LUẬN 52](#_Toc103331535)

[TÀI LIỆU THAM KHẢO 54](#_Toc103331536)

[PHỤ LỤC 55](#_Toc103331537)

# DANH MỤC HÌNH ẢNH

[Hình 1‑1: Hệ thống điều khiển và giám sát điện năng 13](#_Toc103331431)

[Hình 1‑2: Hệ thống điều khiển và giám sát nhà thông minh 13](#_Toc103331432)

[Hình 1‑3: Hệ thống điều khiển và giám sát nhà máy nước 14](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331433)

[Hình 2‑1: Modem, Router của Wifi 15](#_Toc103331434)

[Hình 2‑2: Giao thức HTTP 17](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331435)

[Hình 2‑3: Các giao thức và dịch vụ 17](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331436)

[Hình 2‑4: Mô hình MQTT Broker 18](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331437)

[Hình 2‑5: ESP32 DevKit V1 19](#_Toc103331438)

[Hình 2‑6: Sơ đồ chân và ngoại vi của ESP32 Dev kit 22](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331439)

[Hình 2‑7:Diode 1n4148 25](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331440)

[Hình 2‑8: Altium Designer 2022 26](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331441)

[Hình 2‑9: Một cửa sổ của Altium Designer 27](#_Toc103331442)

[Hình 2‑10: Visual Studio Code 29](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331443)

[Hình 2‑11: RTOS 31](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331444)

[Hình 2‑12: Rtos Kernel 32](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331445)

[Hình 3‑1: Quá trình bật tắt nút nhấn với thiết bị 1 34](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331446)

[Hình 3‑2: Quá trình bật tắt bằng Webserver với thiết bị 1 34](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331447)

[Hình 3‑3: Sơ đồ khối của mạch ESP32 35](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331448)

[Hình 3‑4: Sơ đồ nguyên lý mạch ESP32 37](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331449)

[Hình 3‑5: Giao diện phần mềm Altium Designer 38](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331450)

[Hình 3‑6: Khối MCU 39](#_Toc103331451)

[Hình 3‑7: Khối nguồn và ổn áp nguồn 40](#_Toc103331452)

[Hình 3‑8: Khối cách ly nguồn 40](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331453)

[Hình 3‑9:Khối nút nhấn 41](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331454)

[Hình 3‑10: Khối Relay 41](#_Toc103331455)

[Hình 3‑11: Khối mở rộng 42](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331456)

[Hình 3‑12: Sơ đồ mạch in 43](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331457)

[Hình 3‑13: Lưu đồ thuật toán sử lý sự kiện nút nhấn 44](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331458)

[Hình 3‑14: Xử lý sự kiện nhận về từ MQTT Broker 45](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331459)

[Hình 4‑1: Đề tài sau khi hoàn thành (mặt nghiêng) 48](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331460)

[Hình 4‑2: Đề tài sau khi hoàn thành (mặt bằng) 48](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331461)

[Hình 4‑3: Giao diện phát wifi 49](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331462)

[Hình 4‑4: Vị trí cấp nguồn 24v 50](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331463)

[Hình 4‑5: Loại nút nhấn và vị trí nút nhấn trên mô hình 50](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Mai_Xuan_Kien_2018605403\Quyen_Bao_cao\MXKien_quyen_bao_cao.docx#_Toc103331464)

[Hình 4‑6: File index.html 50](#_Toc103331465)

[Hình 4‑7: Phòng 1 trên giao diện 51](#_Toc103331466)

# DANH MỤC BẢNG BIỂU

[Bảng 2‑1: Các chuẩn wifi hiện nay 16](#_Toc103273695)

[Bảng 2‑2: Các thông số chính của B0505s 22](#_Toc103273696)

[Bảng 3‑1: Một số thẻ HTML cơ bản 45](#_Toc103273697)

# 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 HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT từ xa

* 1. Lý do lựa chọn đề tài
     1. Đặt vấn đề

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

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 cách thiết bị từ xa ở mọi lúc mọi nơi chỉ cần có một thiết bị có thể kết nối được với Internet

* + 1. Giới thiệu chung về đề tài

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

* 1. Vai trò và ý nghĩa 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 của đề tài*

* Ứng dụng IoT vào trong các mạch điện tử giúp trở nên thông minh hơn
* Giảm thiểu được chi phí quản lý và giám sát
* Dần đưa các thiết bị thông minh vào nhà máy và nhà thông minh
  1. Công nghệ sử dụng để điều khiển và giám sát từ xa hiện nay

Với sự phát triển của cuộc cách mạng công nghệ 4.0 hiện nay, hệ thống điều khiển và giám sát từ xa đang được phát triển và ứng dụng mạnh mẽ cả trong công nghiệp cũng như trong dân dụng, ngay cả trong nhà cũng có các hệ thống nhà thông minh.

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 chuyền thông không dây như:

* + 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. 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 tháo tác của bạn 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.

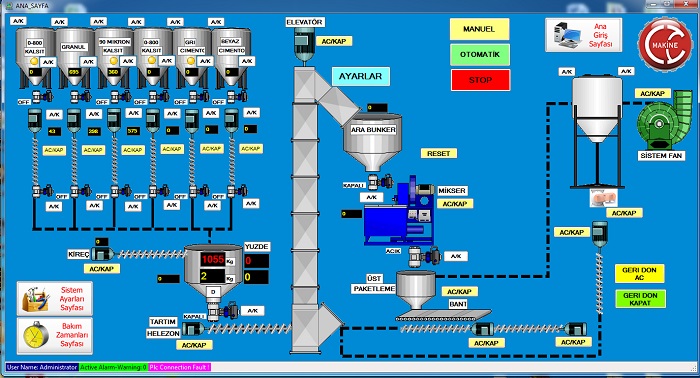
* + 1. LoRa

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.

* 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 sử dụng PLC

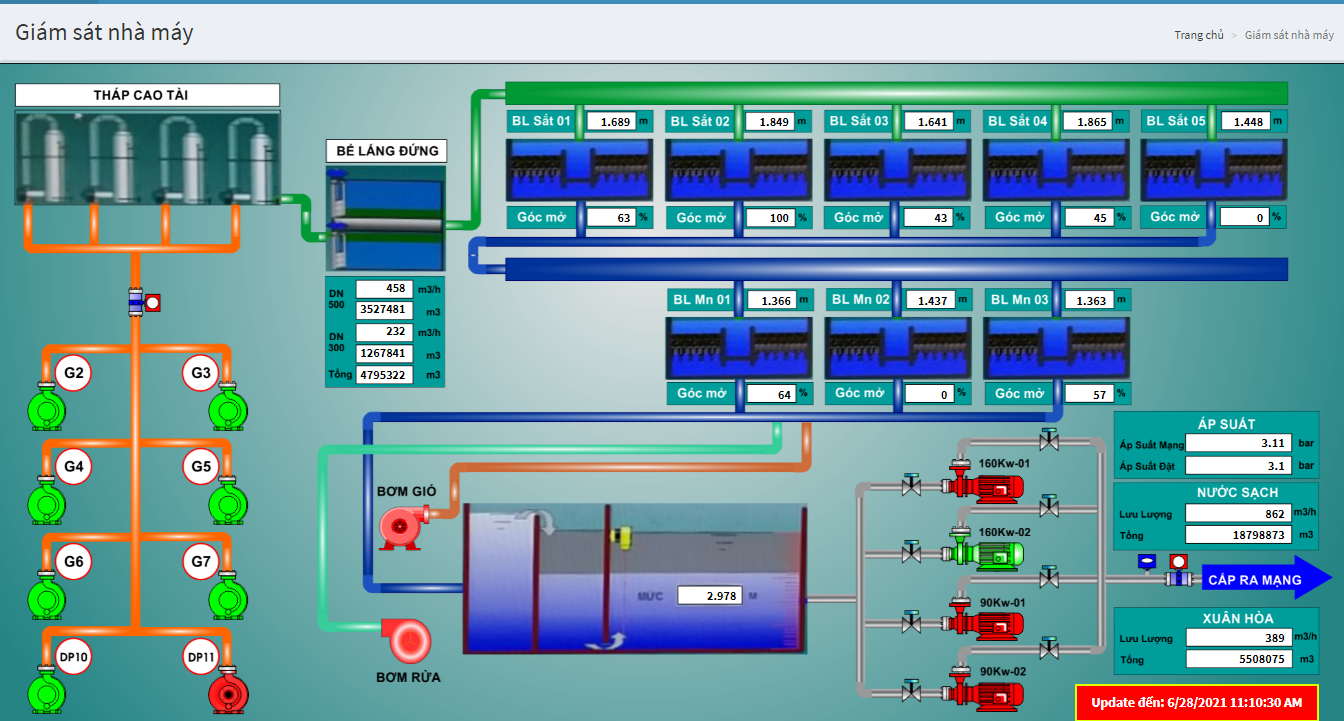


Hình 1‑1: Hệ thống điều khiển và giám sát điện năng

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



Hình 1‑2: Hệ thống điều khiển và giám sát nhà thông minh

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

Hình 1‑3: Hệ thống điều khiển và giám sát nhà máy 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.

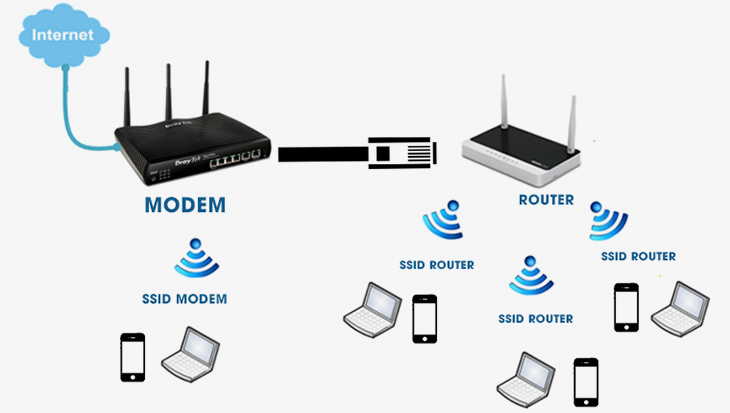
Chương 1 nêu ra được tổng quan về các hệ thống IoT trong thực tế, ứng dụng của các hệ thống IoT trong các ngành khác nhau để nâng cao được hiệu suất sử dụng của máy móc và sự tiện lợi khi điều khiển và giám sát từ xa

# CƠ SỞ LÝ THUYẾT

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

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. [8]

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.



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.

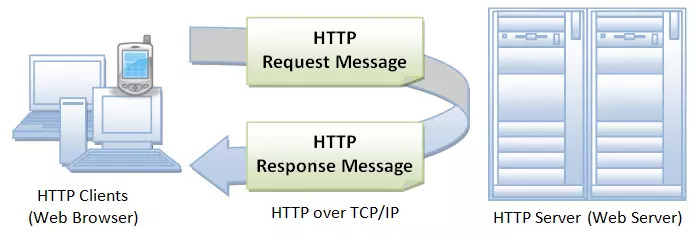
Ngày nay một số thiết bị mới được sản xuất ở Việt Nam đã sử dụng các chuẩn 802.11ac, tuy nhiên số lượng này chưa nhiều (mặc dù ở các nước phát triển đã sử dụng rất phổ biến), một phần do chưa phù hợp với hạ tầng mạng còn hạn chế ở nước ta hiện nay.

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

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[9]

Hình ‑: Giao thức HTTP

Hình ‑: Các giao thức và dịch vụ

**Tầng ứng dụng**

**Tầng vận chuyển**

**Tầng mạng**

**Tầng giao tiếp mạng**

**Tầng ứng dụng**

**Tầng trình bày**

**Tầng liên kết**

**Tầng vận chuyển**

**Tầng vật lý**

**Tầng mạng**

**Tầng phiên**

**Mô hình TCP/IP**

**Mô hình OSI**

**Các giao thức và dịch vụ**

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

**TCP, UDP**

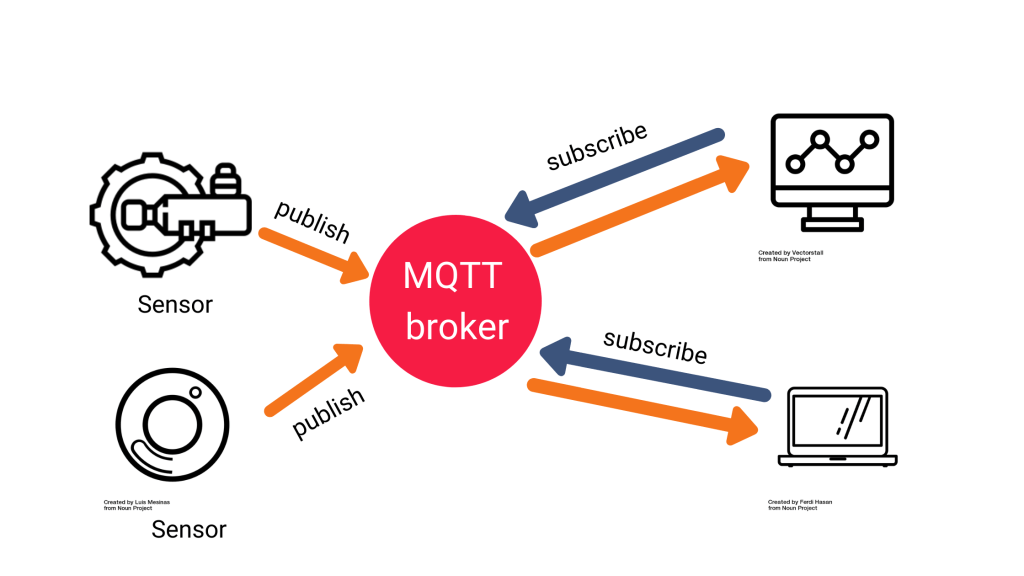
**IP, ARP, ICMP, IGMP**

**Enthernet**

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.

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.

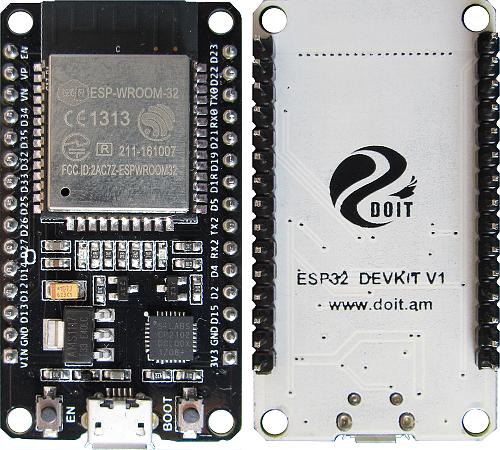
* + 1. Giao thức MQTT

**[12]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.

Hình ‑: Mô hình MQTT Broker

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

[6]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:



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 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)

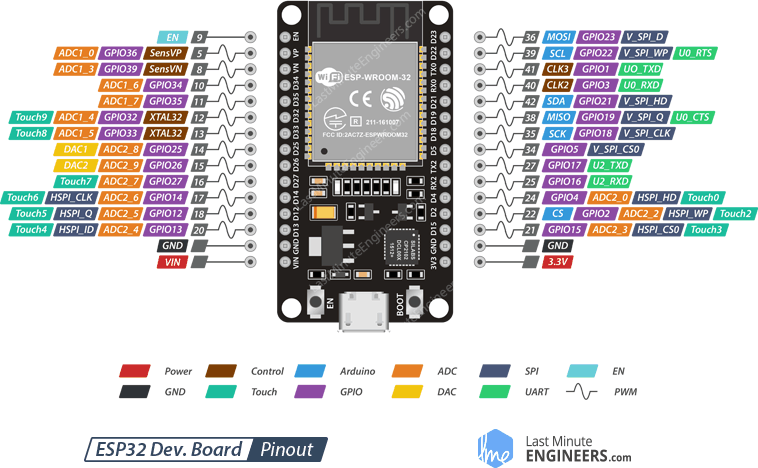
*34 GPIO vật lý với các ngoại vi*

* 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

Hình ‑: 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 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. 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 khuếch 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 |

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 …

* + 1. Opto quang PC817

|  |  |  |
| --- | --- | --- |
| 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. Diode 1N4148

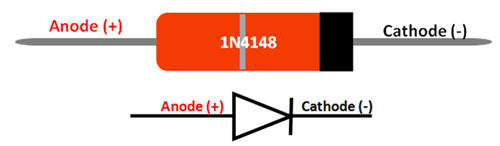
1N4148 là một diode chuyển mạch (switching diode). Đây là một trong những diode chuyển mạch phổ biến và tồn tại lâu dài nhất vì các thông số kỹ thuật đáng tin cậy và chi phí thấp. 1N4148 rất hữu ích trong việc chuyển đổi các ứng dụng lên tới khoảng 100 MHz với thời gian phục hồi ngược không quá 4 ns.

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 ‑:Diode 1n4148



* 1. 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. Phần mềm thiết kế Altium Designer

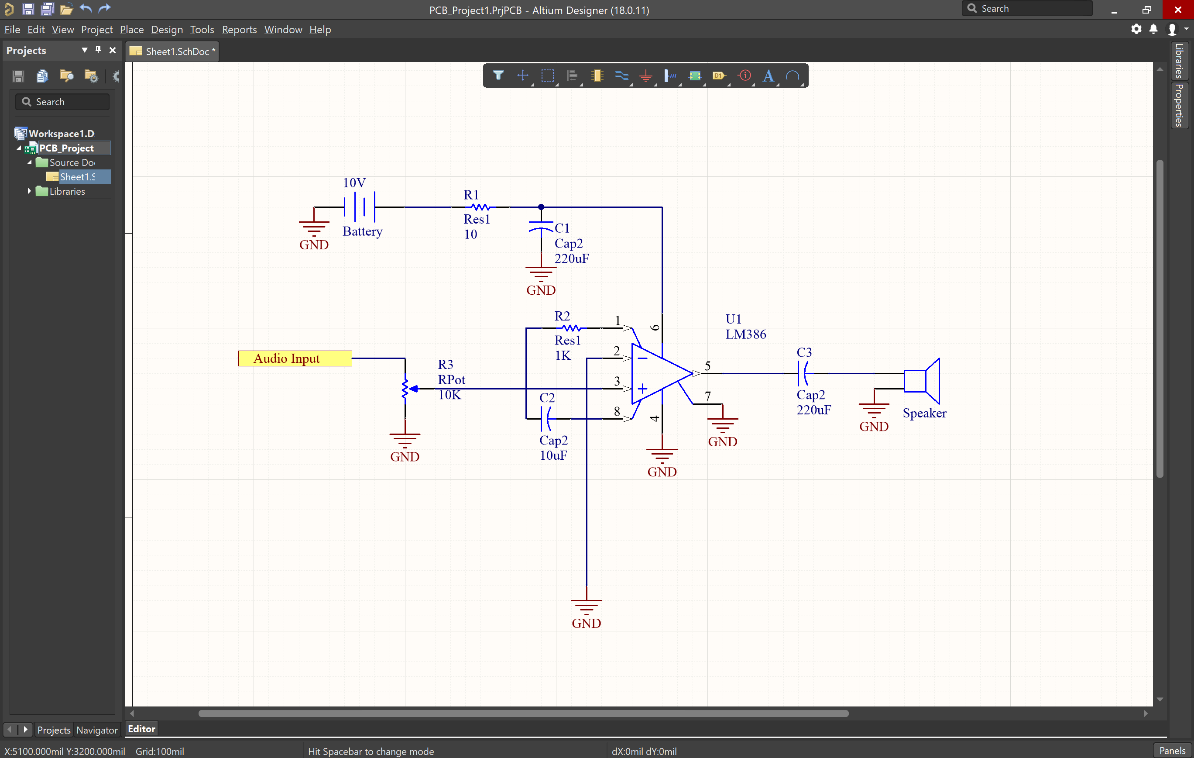
Altium Designer trước kia có tên gọi quen thuộc là Protel DXP, là một trong những công cụ vẽ mạch điện tử mạnh nhất hiện nay. Được phát triển bởi hãng Altium Limited. Altium designer là một phần mềm chuyên nghành được sử dụng trong thiết kế mạch điện tử. Nó là một phần mềm mạnh với nhiều tính năng thú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

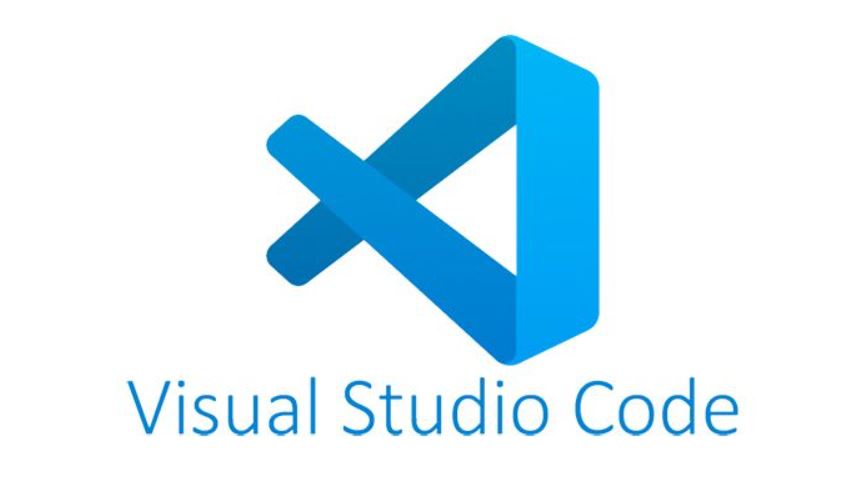
* + 1. JavaScript

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

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.

Hình ‑: Visual Studio Code

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

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 trên bo.

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

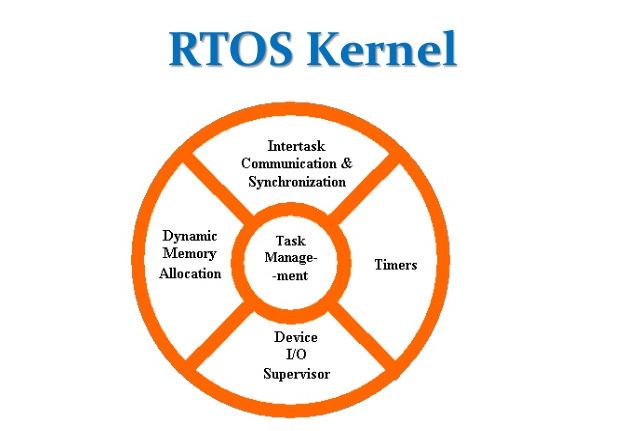
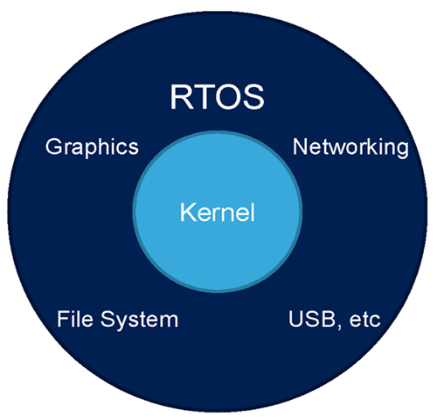
FreeRTOS là một hệ điều hành nhúng thời gian thực (Real Time Operating System) mã nguồn mở được phát triển bởi Real Time Engineers Ltd, sáng lập và sở hữu bởi Richard Barry.[11]

Một số tính năng của FreeRTOS:

Hình ‑: RTOS

* 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.
* Scheduler can be configured for both preemptive or cooperative multitasking.

Hình ‑: 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. Yêu cầu

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. Mô tả hoạt động

Quá trình bật bằng nút nhấn với thiết bị 1

Hình ‑: Quá trình bật tắt 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**

Quá trình bật bằng Webserver với thiết bị 1

Hình ‑: Quá trình bật tắt 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**

* + 1. Sơ đồ khối

*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

khuyếch đại

Hình ‑: Sơ đồ khối của mạch ESP32

Mạch ESP32 điều khiển các thiết bị bao gồm 8 khối chính:

* 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

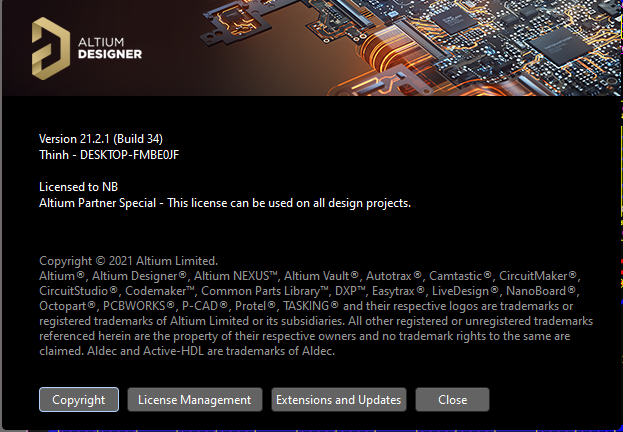
điều khiển thiết bị 1 chiều hoặc xoay chiều thông qua tiếp điểm

* Khối thiết bị

Nhận tín hiệu từ relay và điều khiển thiết bị

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

Hình ‑: Sơ đồ nguyên lý mạch ESP32

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

Hình ‑: Giao diện phần mềm Altium Designer

Theo sơ đồ nguyên lý hình ‑: 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 sẽ gửi về cho ESP32 dữ liệu. ESP32 sẽ 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

* + 1. Khối xử lý trung tâm



Hình 3‑6: Khối MCU

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.

Khối xử lý trung tâm – 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 …

Khối xử lý trung tâm 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 3‑7: 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 ‑: 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.

Hình ‑:Khối nút nhấn

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



Hình 3‑10: 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

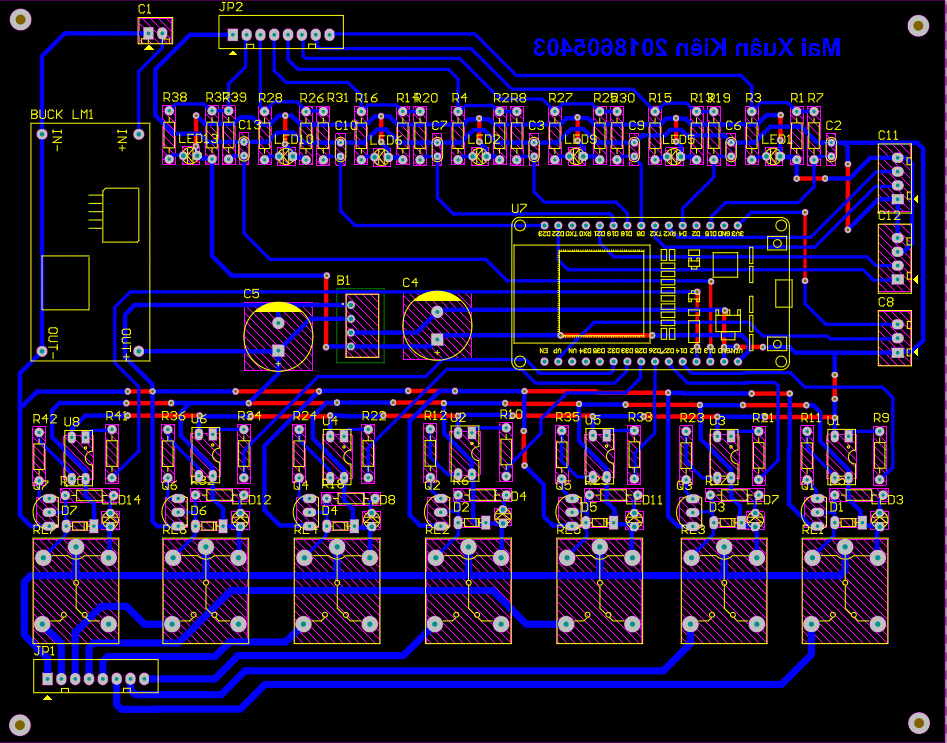
* + 1. Các header mở rộng

Bao gồm 3 header để mở rộng thêm cho mạch:

Hình ‑: Khối mở rộng

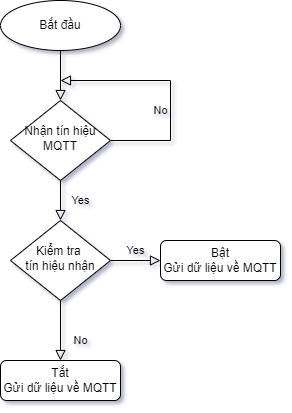
* ADC: mở rộng ngõ vào ADC giúp mạch đọc các cảm biến, các ngoại vi có trang bị ADC
* UART: Nhằm giao tiếp giữa các mạch hoặc giữa mạch với các thiết bị ngoại vi khác có UART
* I2C: chủ yếu để giao tiếp với màn hình LCD I2C
  1. Thiết kế mạch in (PCB)

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

* + 1. Mạch in sau khi thiết kế

Hình ‑: Sơ đồ mạch in

* 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

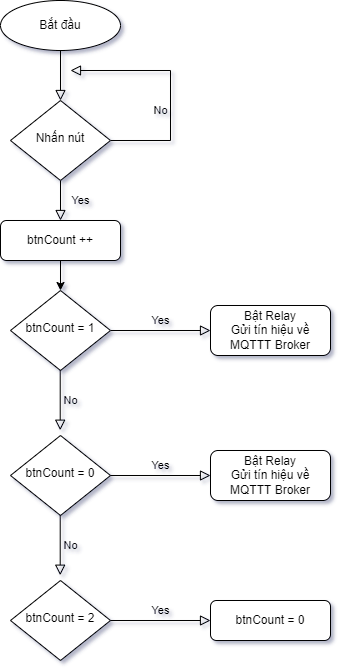
Khi nhấn nút:

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

Hình ‑: Lưu đồ thuật toán sử lý sự kiện nút nhấn

* + 1. Xử lý sự kiện nhận về từ MQTT Broker

[12]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 ‑: Xử lý sự kiện nhận về từ MQTT Broker

* + 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 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 QUẢ ĐẠT ĐƯỢC

* 1. Một số hình ảnh thực tế của hệ thống



Hình ‑: Đề tài sau khi hoàn thành (mặt nghiêng)



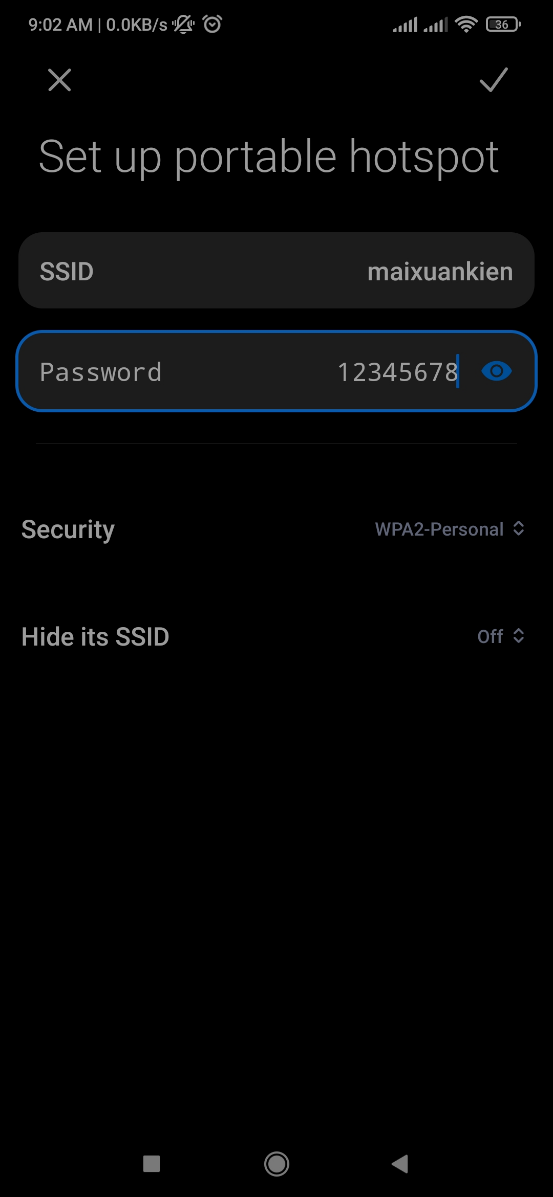
Hình ‑: Đề tài sau khi hoàn thành (mặt bằng)

* 1. 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. Hướng dẫn sử dụng

Hình ‑: Giao diện phát wifi

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

* Tên wifi: “maixuankien”
* Mật khẩu: “12345678”

Bước 2: Cấp nguồn 24V qua Jack Dc 5.5 cho hệ thống

Hình ‑: Vị trí cấp nguồn 24v

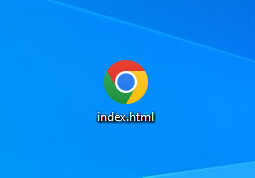


Bước 3: Bật tắt các thiết bị bằng các nút nhấn trên mô hình

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



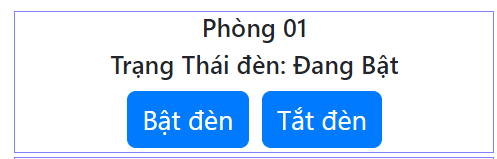
Bước 4: Mở file index.html, giao diện điều khiển sẽ hiện lên bằng các trình duyệt mà máy tính đã cài



Hình 4‑6: 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.

Ví dụ với phòng 1:



Hình 4‑7: Phòng 1 trên giao diện

* 1. Hướng phát triển của đề tài
* 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
  1. Kết luận chương 4

Chương 4 đưa ra một số kết quả đạt được về đề tài, đưa ra được các hình ảnh thực tế của sản phẩm, nêu lên được các hướng phát triển của đề tài. Ngoài ra chương 4 cũng đưa ra được chi tiết hướng dẫn sử dụng của sản phẩm để người sử dụng có thể dễ thao tác hơn.

# 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 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 Th.S 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, ưu điểm và nhược điểm của từng hệ thống
* Nghiên cứu và tìm hiểu về vi điều khiển ESP32, các ngoại vi và lập trình ESP32
* Tìm hiểu về chuẩn truyền thông không dây WiFi, ứng dụng nó vào trong việc điều khiển, truyền nhận tín hiệu và giám sát
* Nắm được các hệ thống điều khiển và giám sát từ xa hoạt động
* Xây dựng được giao diện điều khiển và giám sát
* Thiết kế và vận hành thành công mạch giám sát và điều khiển thiết bị qua WiFi sử dụng vi điều khiển ESP32

Hướng phát triển đề tài:

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

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] 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://openplanning.net/12815/flutter

[4] http://arduino.vn/bai-viet/1219-tap-lenh-voi-esp32

[5] https://en.wikipedia.org/wiki/ESP32

[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] https://en.wikipedia.org/wiki/MQTT

[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

# PHỤ LỤC

#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);

}