|  |  |
| --- | --- |
|  | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI** |
| CÙ THANH LONG | **---------------------------------------**  ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC NGÀNH CNKT ĐIỆN TỬ VIỄN THÔNG  **SỬ DỤNG CẢM BIẾN QUÉT MÃ QR, BĂNG TẢI SERVO ĐỂ ĐIỀU KHIỂN VÀ PHÂN LOẠI** |
| **CBHD: TS. Phạm Xuân Thành**  **Sinh viên: Cù Thanh Long**  **Mã số sinh viên: 2018605608** |
| NGÀNH CNKT ĐIỆN TVIỄTHÔNG THÔNG | Hà Nội - 2022 |

# 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 TS. Phạm Xuân Thành, 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

Cù Thanh Long

**MỤC LỤC**

[LỜI CẢM ƠN i](#_Toc104240426)

[DANH MỤC HÌNH VẼ VÀ BẢNG BIỂU v](#_Toc104240427)

[DANH MỤC TỪ VIẾT TẮT vii](#_Toc104240428)

[LỜI MỞ ĐẦU 1](#_Toc104240429)

[CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG GIÁM SÁT VÀ THỐNG KÊ DÒNG ĐIỆN, ĐIỆN ÁP DC 2](#_Toc104240430)

[1.1 Đặt vấn đề 2](#_Toc104240431)

[1.2 Tổng quan về đề tài 2](#_Toc104240432)

[1.3 Một số dụng cụ đo dòng điện và điện áp một chiều phổ biến 3](#_Toc104240433)

[1.4 Kết luận chương 1 5](#_Toc104240434)

[CHƯƠNG 2: Các Công nghệ SỬ DỤNG TRONG HỆ THỐNG GIÁM SÁT VÀ THỐNG KÊ DÒNG ĐIỆN ĐIỆN ÁP DC 6](#_Toc104240435)

[2.1 Các chuẩn truyền thông được sử dụng 6](#_Toc104240436)

[2.1.1 Modbus 6](#_Toc104240437)

[2.1.2 Giao thức I2C 8](#_Toc104240438)

[2.1.3 Giao thức HTTP 14](#_Toc104240439)

[2.1.4 Giao thức MQTT 15](#_Toc104240440)

[2.2 Phần cứng của hệ thống 17](#_Toc104240441)

[2.2.1 Vi điều khiển ESP32 17](#_Toc104240442)

[2.2.2 Cảm biến PZEM – 017T 19](#_Toc104240443)

[2.2.3 Màn hình LCD 16x2 21](#_Toc104240444)

[2.2.4 Module TTL-RS485 23](#_Toc104240445)

[2.2.5 IC ổn áp LM7805 24](#_Toc104240446)

[2.2.6 Relay 5V 25](#_Toc104240447)

[2.2.7 Transistor C1815 25](#_Toc104240448)

[2.3 Phần mềm thiết kế và lập trình 26](#_Toc104240449)

[2.3.1 Phần mềm thiết kế Altium Designer 26](#_Toc104240450)

[2.3.2 Phần mềm Visual Studio Code 27](#_Toc104240451)

[2.3.3 Platform IO 28](#_Toc104240452)

[2.3.4 Phần mềm Node-Red 29](#_Toc104240453)

[2.4 Kết luận chương 2 30](#_Toc104240454)

[CHƯƠNG 3: THIẾT KẾ HỆ THỐNG HỆ THỐNG GIÁM SÁT VÀ THỐNG KÊ DÒNG ĐIỆN, ĐIỆN ÁP DC 31](#_Toc104240455)

[3.1 Sơ đồ khối của hệ thống 31](#_Toc104240456)

[3.2 Sơ đồ nguyên lý toàn mạch 32](#_Toc104240457)

[3.2.1 Khối nguồn và ổn áp nguồn 33](#_Toc104240458)

[3.2.2 Khối xử lý trung tâm 35](#_Toc104240459)

[3.2.3 Khối TTL-RS485 36](#_Toc104240460)

[3.2.4 Khối Relay 37](#_Toc104240461)

[3.2.5 Khối hiển thị 38](#_Toc104240462)

[3.2.6 Khối còi báo 38](#_Toc104240463)

[3.3 Sơ đồ mạch in 39](#_Toc104240464)

[3.4 Thiết kế phần mềm 40](#_Toc104240465)

[3.4.1 Lưu đồ thuật toán 40](#_Toc104240466)

[3.4.2 Phần mềm giám sát trên Web 41](#_Toc104240467)

[3.5 Mô hình thực tế 42](#_Toc104240468)

[KẾT LUẬN 43](#_Toc104240469)

[TÀI LIỆU THAM KHẢO 45](#_Toc104240470)

[PHỤ LỤC 46](#_Toc104240471)

# DANH MỤC HÌNH VẼ VÀ BẢNG BIỂU

[Hình 1.1: Đồng hồ vạn năng 3](#_Toc104240311)

[Hình 1.2: Đồng hồ Selec MA12-DC 4](#_Toc104240312)

[Hình 2.1: Giao thức Modbus 6](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240313)

[Hình 2.2: Mô hình giao tiếp I2C 8](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240314)

[Hình 2.3: Cách truyền dữ liệu I2C 9](#_Toc104240315)

[Hình 2.4: Mô hình một Master với nhiều Slave 12](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240316)

[Hình 2.5: Mô hình nhiều Master với nhiều Slave 12](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240317)

[Hình 2.6: Mô hình OSI và TCP/IP 14](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240318)

[Hình 2.7: Mô hình Client -Sever 14](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240319)

[Hình 2.8: Mô hình tổng quát của MQTT 16](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240320)

[Hình 2.9: Các ngoại vi trên ESP32 17](#_Toc104240321)

[Hình 2.10. Cảm biến PZEM – 017T 19](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240322)

[Hình 2.11: Sơ đồ đầu nối cảm biến PZEM – 017T 20](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240323)

[Hình 2.12: Màn hình hiển thị LCD 16x2 21](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240324)

[Hình 2.13: Module TTL-RS485 23](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240325)

[Hình 2.14: IC LM7805 24](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240326)

[Hình 2.15: Relay 5V 25](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240327)

[Hình 2.16: Một cửa sổ của Altium Designer 26](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240328)

[Hình 2.17: Visual Studio Code 27](#_Toc104240329)

[Hình 2.18: Platform IO 28](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240330)

[Hình 2.19. Node-red 29](#_Toc104240331)

[Hình 3.1: Sơ đồ nguyên lý của hệ thống 32](#_Toc104240332)

[Hình 3.2: Adapter nguồn 12V 33](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240333)

[Hình 3.3: Khối ổn áp nguồn 34](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240334)

[Hình 3.4. Khối xử lý trung tâm 35](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240335)

[Hình 3.5: Khối TTL – RS485 36](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240336)

[Hình 3.6: RS485 trên cảm biến 36](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240337)

[Hình 3.7: Khối relay 37](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240338)

[Hình 3.8: Nhiễu gây ra trên Relay 37](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240339)

[Hình 3.9: Khối hiển thị 38](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240340)

[Hình 3.10. Khối còi báo 38](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240341)

[Hình 3.11: Sơ đồ mạch in của hệ thống 39](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240342)

[Hình 3.12. Lưu đồ thuật toán 40](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240343)

[Hình 3.13: Quá trình nhận dữ liệu từ ESP32 41](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240344)

[Hình 3.14. Quá trình gửi dữ liệu từ Webserver xuống ESP32 41](#_Toc104240345)

[Hình 3.15: Mô hình thực tế của hệ thống 42](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240346)

[Hình 3.16: Giao diện giám sát trên Webserver 42](file:///D:\THINH\DO_AN\Project_HaUI_2_2022\Cu_Thanh_Long_2018605608\quyen_bao_cao\CTLong_bao_cao.docx#_Toc104240347)

# DANH MỤC TỪ VIẾT TẮT

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Từ viết tắt | Tiếng anh | Tiếng việt |
| 1 | HTTP | Hyper Text Transfer Protocol | Giao thức truyền tải siêu văn bản |
| 2 | IC | Intergrated Circuit | Mạch tích hợp |
| 4 | DC | Direct Current | Dòng điện một chiều |
| 5 | LCD | Liquid Crystal Display | Màn hình tinh thể lỏng |
| 6 | IDE | Integrated Development Environment | Môi trường phát triển tích hợp |
| 7 | MCU | Multipoint Control Unit | Thiết bị điều khiển đa điểm |

# LỜI MỞ ĐẦU

Lịch sử nhân loại đã trải qua nhiều cuộc cách mạng khoa học kĩ thuật, tinh thần tìm tòi sáng tạo giúp con người ngày càng có nhiều phát minh, sáng kiến, tìm ra những công cụ, phương tiện mới, con đường mới đế chinh phục tự nhiên. Ngày nay, chúng ta đang phấn đấu cho mục tiêu công nghiệp hóa - hiện đại hóa, rượt đuổi làn sóng các cuộc cách mạng khoa học, con người đã và đang tiếp cận với vi xử lý.

Trong những năm gần đây, công nghệ vi điện tử phát triển rất mạnh mẽ. Sự ra đời của các vi mạch cỡ lớn, cực lớn với giá thành giảm nhanh, khả năng lập trình ngày càng cao đã mang lại những thay đổi sâu sắc trong ngành kĩ thuật điện tử. Nền công nghiệp thế giới đã đạt được những thành tựu to lớn nhờ ứng dụng những tiến bộ của khoa học kĩ thuật và công nghệ, máy móc đã thay thế con người trong nhiều hoạt động lao động sản xuất.

Để thực hiện công giám sát hiệu suất về dòng và áp của các máy hoạt động trong công nghiệp một cách khoa học và chính xác. Hệ thống đo và giám sát dòng điện và điện áp cho hệ thống là một trong các giải pháp nhà sản xuất đã và đang áp dụng vào dây truyền sản xuất nhằm đáp ứng được nhu cầu tự động hóa, giảm sức lao động cho người lao động và tiết kiệm được chi phí sản xuất.

Việc giám sát dòng điện và điện áp một chiều với vi điều khiển là phần việc không thể thiếu khi chế tạo những máy móc tự động. Trong một cỗ máy tự động, nếu phần cơ khí tạo nên hình dạng và một cơ cấu hoạt động linh hoạt thì phần lập trình và mạch điện tử như một bộ não điều khiển những hoạt động đó.

Qua quá trình học tập và tìm hiểu một số môn học ở trên trường, em đã quyết định chọn đề tài: **“Thiết kế hệ thống giám sát và thống kê dòng điện, điện áp DC ”** là đề tài đồ án tốt nghiệp.

# TỔNG QUAN VỀ HỆ THỐNG GIÁM SÁT VÀ THỐNG KÊ DÒNG ĐIỆN, ĐIỆN ÁP DC

## Đặt vấn đề

Trên thế giới hiện nay việc giám sát và thống kê dòng điện, điện áp của các thiết bị điện qua mạng internet không còn mới mẻ và xa lạ nữa. Xu hướng hiện nay là có thể giám sát và thống kê dòng điện , điện áp thiết bị thông qua phần mềm với giao diện được lập trình và được cài trên máy tính mà không cần đo đạc, thống kê một cách thủ công.

Ở phạm vi trong nước, vấn đề giám sát và thống kê dòng điện , điện áp các thiết bị điện từ xa luôn là tâm điểm của khoa học hiện nay. Với mong muốn ứng dụng khoa học thế giới vào cuộc sống, họ muốn cuộc sống trở nên công nghệ hơn, hiện đại hơn. Nhưng thực tế chưa có sản phẩm nào đáp ứng được nhiều vấn đề mà nhu cầu thực tế đặt ra và thường giá thành tới tay người tiêu dung để áp dụng và thực tế thường khá cao nên ít được mọi người chú ý tới.

Tóm lại, việc thiết kế sử dụng hệ thống giám sát và thống kê dòng điện, điện áp các thiết bị hiện nay ở Việt Nam đang còn mới mẻ và chưa đi vào thực tiễn ứng dụng nhiều. Điều này khiến em quyết định làm và đưa ra sản phẩm mang tính chất thực tế hơn và áp dụng được vào cuộc sống thực tiễn

## Tổng quan về đề tài

Trong đề tài này, em sẽ thực hiện thiết kế một mô hình sản phẩm hoàn thiện đảm bảo các chức năng giám sát và thống kê dòng điện, điện áp cho các thiết bị điện một chiều (DC). Hiện thị thông số các giá trị dòng điện và điện áp lên màn hình LCD, tương tác với Webserver và gửi dữ liệu lên Webserver.

Ngoài ra mô hình còn có các chức năng khác như: Báo quá áp hoặc quá dòng so với một ngưỡng đã đặt qua còi, có thể đặt ngưỡng để cắt hệ thống qua một relay với các ngưỡng về còi, về relay hoàn toàn có thể cài đặt từ xa thông qua webserver.

## Một số dụng cụ đo dòng điện và điện áp một chiều phổ biến

*Các loại đồng hồ vạn năng*

Đồng hồ vạn năng là một công cụ điện tử thường được sử dụng để đo điện áp và dòng điện trên các mạch. Ngoài ra tùy vào mục đích sử dụng, đồng hồ vạn năng còn có thể đo được các thông số khác như: Điện trở, điện dung, tần số …



Hình 1.1: Đồng hồ vạn năng

Đồng hồ vạn năng là một thiết bị đo với tốc độ nhanh và chính xác cao, vì vậy đồng hồ vạn năng thường được sử dụng như một dụng cụ không thể thiếu để hiệu chỉnh các thiết bị khác.

Trên thị trường hiện nay có 2 loại đồng hồ vạn năng chủ yếu là đồng hồ kim và đồng hồ dạng số điện tử.  Nếu nhu cầu sửa chữa mạch điện như kiểm tra diode, transistor, đo thông mạch, kiểm tra sự phóng nạp, dò, chập của tụ điện ... thì chỉ cần sử dụng loại đồng hồ vạn năng kim vì giá thành rẻ hơn. Nếu cần một chiếc đồng hồ đo điện đa năng thì nên chọn đồng hồ vạn năng số. Ưu điểm hiển thị nhiều kết quả đo đồng thời nhanh chóng và chính xác qua màn hình điện tử

*Đồng hồ đo dòng Selec MA12-DC-20mA 96x48mm*



Hình 1.2: Đồng hồ Selec MA12-DC

Các thông số kỹ thuật của đồng hồ:

* Hiển thị 4 số, dạng LED 7 đoạn
* Cài đặt hệ số RShunt hoặc kết nối trực tiếp
* Dải hiển thị: 0 – 4960A
* Mạng kết nối: 1 pha 2 dây
* Nguồn cấp: AC: 240V AC ±20%, (50 / 60Hz) 110V AC ±20%, (50 / 60Hz)
* Bảo vệ mặt trước: IP65
* Chiều cao của LED hiển thị: 14.2 mm
* Kích thước máy đo 48x96mm

*Đồng hồ đo điện áp DC Autonics MT4W-DV-1N 96x48mm*



*Các thông số kỹ thuật:*

* Phương pháp hiển thị : LED 7 đoạn 4 số
* Chiều cao kí tự : 14.2mm
* Dải hiển thị tối đa : -1999 to 9999
* Đo : Điện áp DC
* Loại ngõ vào : 0-500V, 0-100V, 0-50V, 0-10V, 0-5V, 0-1V, 0-250mV, 0-50mV
* Nguồn cấp : 12-24VDC

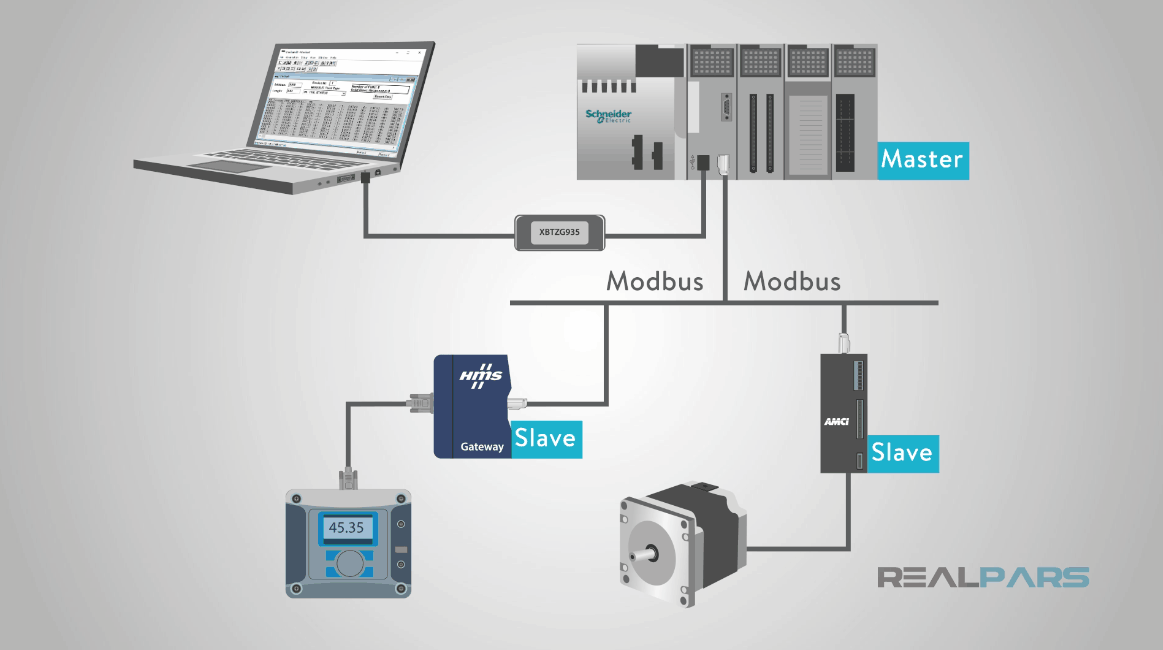
## Kết luận chương 1

# Các Công nghệ SỬ DỤNG TRONG HỆ THỐNG GIÁM SÁT VÀ THỐNG KÊ DÒNG ĐIỆN ĐIỆN ÁP DC

## Các chuẩn truyền thông được sử dụng

### Modbus

Modbus là một giao thức truyền thông dữ liệu ban đầu được phát triển bởi Modicon (nay là Schneider Electric) vào năm 1979 để sử dụng với PLC (Programable Logic Controller). Modbus đã trở thành một giao thức truyền thông tiêu chuẩn thực tế và hiện là một phương tiện thường có sẵn để kết nối các thiết bị điện tử công nghiệp.

Modbus là phổ biến trong môi trường công nghiệp vì nó được công khai và miễn phí bản quyền. Nó được phát triển cho các ứng dụng công nghiệp vì tương đối dễ triển khai và duy trì so với các tiêu chuẩn khác và đặt ra một số hạn chế đối với định dạng của dữ liệu sẽ được truyền đi.

Hình 2.1: Giao thức Modbus

Việc phát triển và cập nhật các giao thức Modbus đã được quản lý bởi Tổ chức Modbus. Tổ chức Modbus là một hiệp hội của người dùng và nhà cung cấp các thiết bị tuân thủ Modbus mà ủng hộ việc tiếp tục sử dụng công nghệ.

Giao thức Modbus sử dụng giao tiếp nối tiếp, Ethernet hoặc giao thức Internet làm lớp vận chuyển (Transport Layer). Modbus hỗ trợ giao tiếp đến và từ nhiều thiết bị được kết nối với cùng một mạng cáp hoặc mạng Ethernet như: Có 2 thiết bị, một thiết bị đo nhiệt độ và một thiết bị khác để đo độ ẩm được kết nối với cùng một đường truyền, cả hai đều truyền dữ liệu của các phép đo đến cùng một máy tính thông qua Modbus.

Modbus thường được sử dụng để kết nối máy tính giám sát nhà máy/hệ thống với thiết bị đầu cuối từ xa (RTU) trong các hệ thống giám sát và thu thập dữ liệu (SCADA).

*Các phiên bản hay được sử dụng của Modbus:*

Modbus RTU

Được sử dụng trong giao tiếp nối tiếp và là phiên bản phổ biến nhất của Modbus. Modbus RTU sử dụng một biểu diễn nhị phân, nhỏ gọn của dữ liệu cho giao tiếp giao thức. Định dạng RTU tuân theo các lệnh/dữ liệu với kiểm tra dự phòng theo chu kỳ làm cơ chế kiểm tra lỗi để đảm bảo độ tin cậy của dữ liệu. Một thông điệp Modbus RTU phải được truyền liên tục mà không bị ngắt quãng giữa các ký tự.

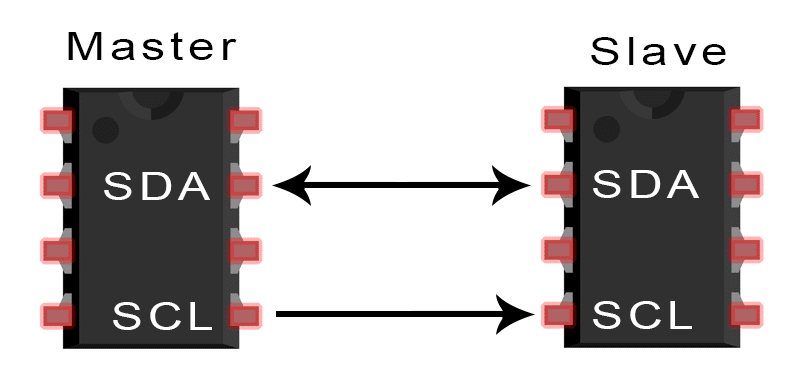
Modbus ASCII

Được sử dụng trong giao tiếp nối tiếp và sử dụng các ký tự ASCII để giao tiếp giao thức. Định dạng ASCII sử dụng kiểm tra kiểm tra dự phòng theo chiều dọc. Các tin nhắn Modbus ASCII được đóng khung bởi một ký tự (":") và theo dõi (CR/LF).

Modbus TCP/IP

Một biến thể Modbus được sử dụng để truyền tin qua mạng TCP/IP, kết nối qua cổng 502. Nó không yêu cầu tính toán tổng kiểm tra, vì các lớp thấp hơn đã cung cấp bảo vệ tổng kiểm tra.

### Giao thức I2C



Hình 2.2: Mô hình giao tiếp I2C

*Giới thiệu về giao tiếp I2C*

Giao tiếp I2C là sự kết hợp các tính năng tốt nhất của UART và SPI, trong giao tiếp I2C, một thiết bị có thể đóng vai trò là Master (chủ) kết nối tới nhiều Slave (tớ), hoặc có thể nhiều thiết bị đóng vai trò là Master kết nối tới một hoặc nhiều thiết bị là Slave.

I2C chỉ sử dụng 2 dây để kết nối các thiết bị với nhau:

* SDA (tín hiệu Data) - Master và Slave gửi và nhận dữ liệu.
* SCL (xung Clock) - Tín hiệu xung Clock

I2C là một giao thức giao tiếp nối tiếp, vì vậy dữ liệu được truyền từng bit theo một dây duy nhất (dây SDA)

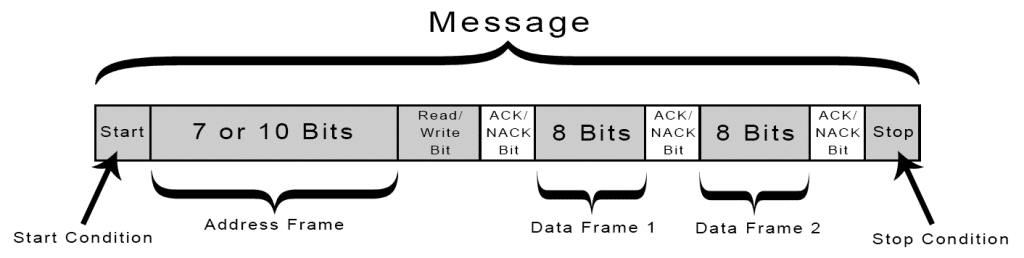
Giống như SPI, I2C sử dụng cơ chế đồng bộ do đó, đầu ra của bit được đồng bộ hóa với việc lấy mẫu các bit bằng tín hiệu Clock được chia sẻ giữa Master và Slave. Tín hiệu Clock luôn được kiểm soát bởi Master.

Bảng 2.1: Thông số chính của giao tiépe I2C

|  |  |
| --- | --- |
| Số lượng dây sử dụng | 2 |
| Tốc độ truyền tối đa | Chế độ tiêu chuẩn = 100 kbps  Chế độ nhanh = 400 kbps  Chế độ tốc độ cao = 3.4 Mbps  Chế độ tốc độ rất cao = 5 Mbps |
| Loại chế độ truyền | Đồng bộ |
| Phương thức truyền | Nối tiếp |
| Số lượng Master tối đa | Không giới hạn về số lượng |
| Số lượng Slave tối đa | 1008 |

*Cách hoạt động của I2C*

Với I2C, dữ liệu được truyền trong các tin nhắn. Tin nhắn được chia thành các khung dữ liệu. Mỗi tin nhắn có một khung địa chỉ chứa địa chỉ nhị phân của địa chỉ slave và một hoặc nhiều khung dữ liệu chứa dữ liệu đang được truyền. Thông điệp cũng bao gồm điều kiện khởi động và điều kiện dừng, các bit đọc / ghi và các bit ACK / NACK giữa mỗi khung dữ liệu:



Hình 2.3: Cách truyền dữ liệu I2C

* Điều kiện khởi động: Đường SDA chuyển từ mức điện áp cao xuống mức điện áp thấp trước khi đường SCL chuyển từ mức cao xuống mức thấp.
* Điều kiện dừng: Đường SDA chuyển từ mức điện áp thấp sang mức điện áp cao sau khi đường SCL chuyển từ mức thấp lên mức cao.
* Khung địa chỉ: Một chuỗi 7 hoặc 10 bit duy nhất cho mỗi slave để xác định slave khi master muốn giao tiếp với nó.
* Bit đọc / ghi: Một bit duy nhất chỉ định master đang gửi dữ liệu đến slave (mức điện áp thấp) hay yêu cầu dữ liệu từ nó (mức điện áp cao).
* Bit ACK / NACK:Mỗi khung trong một tin nhắn được theo sau bởi một bit xác nhận / không xác nhận. Nếu một khung địa chỉ hoặc khung dữ liệu được nhận thành công, một bit ACK sẽ được trả lại cho thiết bị gửi từ thiết bị nhận.

*Địa chỉ*

I2C không có các đường chọn Slave như SPI, vì vậy cần một cách khác để cho slave biết rằng dữ liệu đang được gửi đến slave này chứ không phải slave khác. Nó thực hiện điều này bằng cách định địa chỉ. Khung địa chỉ luôn là khung đầu tiên sau bit khởi động trong một tin nhắn mới.

Master gửi địa chỉ của slave mà nó muốn giao tiếp với mọi slave được kết nối với nó. Sau đó, mỗi slave sẽ so sánh địa chỉ được gửi từ master với địa chỉ của chính nó. Nếu địa chỉ phù hợp, nó sẽ gửi lại một bit ACK điện áp thấp cho master. Nếu địa chỉ không khớp, slave không làm gì cả và đường SDA vẫn ở mức cao.

*Bit đọc / ghi*

Khung địa chỉ bao gồm một bit duy nhất ở cuối tin nhắn cho slave biết master muốn ghi dữ liệu vào nó hay nhận dữ liệu từ nó. Nếu master muốn gửi dữ liệu đến slave, bit đọc / ghi ở mức điện áp thấp. Nếu master đang yêu cầu dữ liệu từ slave, thì bit ở mức điện áp cao.

*Khung dữ liệu*

Sau khi Master phát hiện bit ACK từ Slave, khung dữ liệu đầu tiên đã sẵn sàng được gửi.

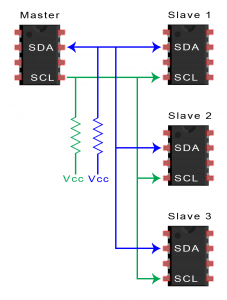
Khung dữ liệu luôn có độ dài 8 bit và được gửi với bit quan trọng nhất trước. Mỗi khung dữ liệu ngay sau đó là một bit ACK / NACK để xác minh rằng khung đã được nhận thành công. Bit ACK phải được nhận bởi Master hoặc Slave (tùy thuộc vào cái nào đang gửi dữ liệu) trước khi khung dữ liệu tiếp theo có thể được gửi.

Sau khi tất cả các khung dữ liệu đã được gửi, Master có thể gửi một điều kiện dừng cho slave để tạm dừng quá trình truyền. Điều kiện dừng là sự chuyển đổi điện áp từ thấp lên cao trên đường SDA sau khi chuyển tiếp từ thấp lên cao trên đường SCL , với đường SCL vẫn ở mức cao.

*Các bước truyền dữ liệu I2C*

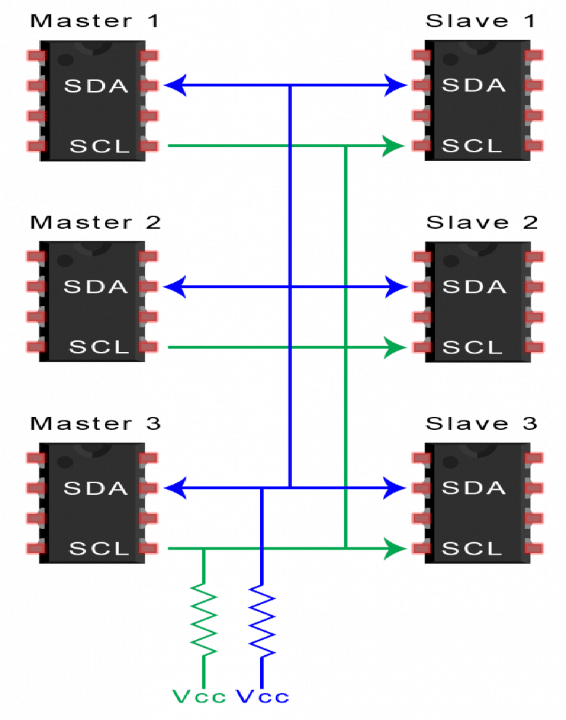
1. Master gửi điều kiện khởi động đến mọi slave được kết nối bằng cách chuyển đường SDA từ mức điện áp cao sang mức điện áp thấp trước khi chuyển đường SCL từ mức cao xuống mức thấp.
2. Master gửi cho mỗi slave địa chỉ 7 hoặc 10 bit của slave mà nó muốn giao tiếp, cùng với bit đọc / ghi.
3. Mỗi slave sẽ so sánh địa chỉ được gửi từ master với địa chỉ của chính nó. Nếu địa chỉ trùng khớp, slave sẽ trả về một bit ACK bằng cách kéo dòng SDA xuống thấp cho một bit. Nếu địa chỉ từ master không khớp với địa chỉ của slave, slave rời khỏi đường SDA cao.
4. Master gửi hoặc nhận khung dữ liệu.
5. Sau khi mỗi khung dữ liệu được chuyển, thiết bị nhận trả về một bit ACK khác cho thiết bị gửi để xác nhận đã nhận thành công khung.
6. Để dừng truyền dữ liệu, master gửi điều kiện dừng đến slave bằng cách chuyển đổi mức cao SCL trước khi chuyển mức cao SDA.

**Một Master với nhiều Slave**

Vì I2C sử dụng định địa chỉ nên nhiều Slave có thể được điều khiển từ một master duy nhất. Với địa chỉ 7 bit sẽ có 128 (2 mũ 7) địa chỉ duy nhất. Việc sử dụng địa chỉ 10 bit không phổ biến, nhưng nó cung cấp 1.024 (2 mũ 10) địa chỉ duy nhất. Để kết nối nhiều slave đến một master duy nhất, bạn có thể đấu dây như thế này, với điện trở kéo lên 4,7K Ohm kết nối đường SDA và SCL với Vcc:

Hình 2.4: Mô hình một Master với nhiều Slave

*Nhiều master với nhiều slave*

Nhiều master có thể được kết nối với một slave hoặc nhiều slave. Sự cố với nhiều master trong cùng một hệ thống xảy ra khi hai master cố gắng gửi hoặc nhận dữ liệu cùng một lúc qua đường SDA. Để giải quyết vấn đề này, mỗi master cần phải phát hiện xem đường SDA thấp hay cao trước khi truyền tin nhắn.

Hình 2.5: Mô hình nhiều Master với nhiều Slave

Nếu đường SDA thấp, điều này có nghĩa là một master khác có quyền điều khiển bus và master đó phải đợi để gửi tin nhắn. Nếu đường SDA cao thì có thể truyền tin nhắn an toàn. Để kết nối nhiều master với nhiều slave, hãy sử dụng sơ đồ sau, với các điện trở kéo lên 4,7K Ohm kết nối các đường SDA và SCL với Vcc

*Ưu điểm*

* Chỉ sử dụng hai dây
* Hỗ trợ nhiều Master và nhiều Slave
* Bit ACK / NACK xác nhận mỗi khung được chuyển thành công
* Phần cứng ít phức tạp hơn so với UART
* Giao thức nổi tiếng và được sử dụng rộng rãi

*Nhược điểm*

* Tốc độ truyền dữ liệu chậm hơn SPI
* Kích thước của khung dữ liệu bị giới hạn ở 8 bit
* Cần phần cứng phức tạp hơn để triển khai so với SPI

### Giao thức HTTP

Hình 2.6: Mô hình OSI và TCP/IP

**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**

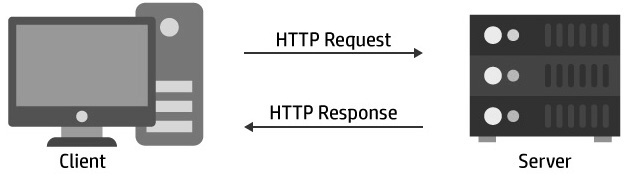
**IP, ARP, ICMP, IGMP**

**Enthernet**

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.

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.

Hình 2.7: Mô hình Client -Sever



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.

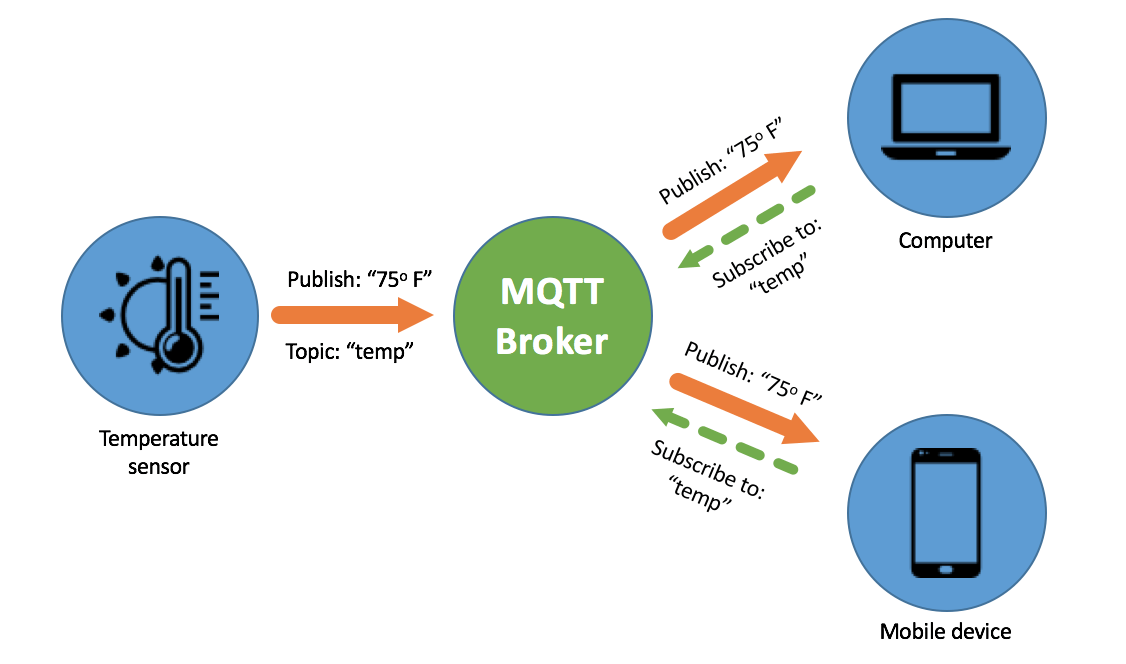
### Giao thức MQTT

MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông điệp (message) theo mô hình publish/subscribe (cung cấp / thuê bao), được sử dụng cho các thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Nó dựa trên một Broker (tạm dịch là “Máy chủ môi giới”) “nhẹ” (khá ít xử lý) và được thiết kế có tính mở (tức là không đặc trưng cho ứng dụng cụ thể nào), đơn giản và dễ cài đặt.

MQTT là lựa chọn lý tưởng trong các môi trường như:

* Những nơi mà giá mạng viễn thông đắt đỏ hoặc băng thông thấp hay thiếu tin cậy.
* Khi chạy trên thiết bị nhúng bị giới hạn về tài nguyên tốc độ và bộ nhớ.
* Bởi vì giao thức này sử dụng băng thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng M2M (Machine to Machine).
* MQTT cũng là giao thức được sử dụng trong Facebook Messenger

Một số ưu điểm nổi bật của MQTT như: băng thông thấp, độ tin cậy cao và có thể sử dụng ngay cả khi hệ thống mạng không ổn định, tốn rất ít byte cho việc kết nối với server và connection có thể giữ trạng thái open xuyên suốt, có thể kết nối nhiều thiết bị (MQTT client) thông qua một MQTT server (broker). Bởi vì giao thức này sử dụng băng thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng IoT.

* Dạng truyền thông điệp theo mô hình Pub/Sub cung cấp việc truyền tin phân tán một chiều, tách biệt với phần ứng dụng.
* Việc truyền thông điệp là ngay lập tức, không quan tâm đến nội dung được truyền.
* Sử dụng TCP/IP là giao thức nền.
* Tồn tại ba mức độ tin cậy cho việc truyền dữ liệu (QoS: Quality of service)
* Phần bao bọc dữ liệu truyền nhỏ và được giảm đến mức tối thiểu để giảm tải cho đường truyền.

Hình 2.8: Mô hình tổng quát của MQTT

Các thành phần của MQTT bao gồm:

Client: Publisher - Nơi gửi thông điệp và Subscriber - Nơi nhận thông điệp

Broker - Máy chủ môi giới:

Trong đó Broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ Client (Publisher/Subscriber). Nhiệm vụ chính của Broker là nhận thông điệp (message) từ Publisher, xếp vào hàng đợi rồi chuyển đến một địa điểm cụ thể. Nhiệm vụ phụ của Broker là nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật message, lưu trữ message, logs, ....

Client thì được chia thành hai nhóm là Publisher và Subscriber. Client chỉ làm ít nhất một trong 2 việc là publish các thông điệp (message) lên một/nhiều topic cụ thể hoặc subscribe một/nhiều topic nào đó để nhận message từ topic này.

## Phần cứng của hệ thống

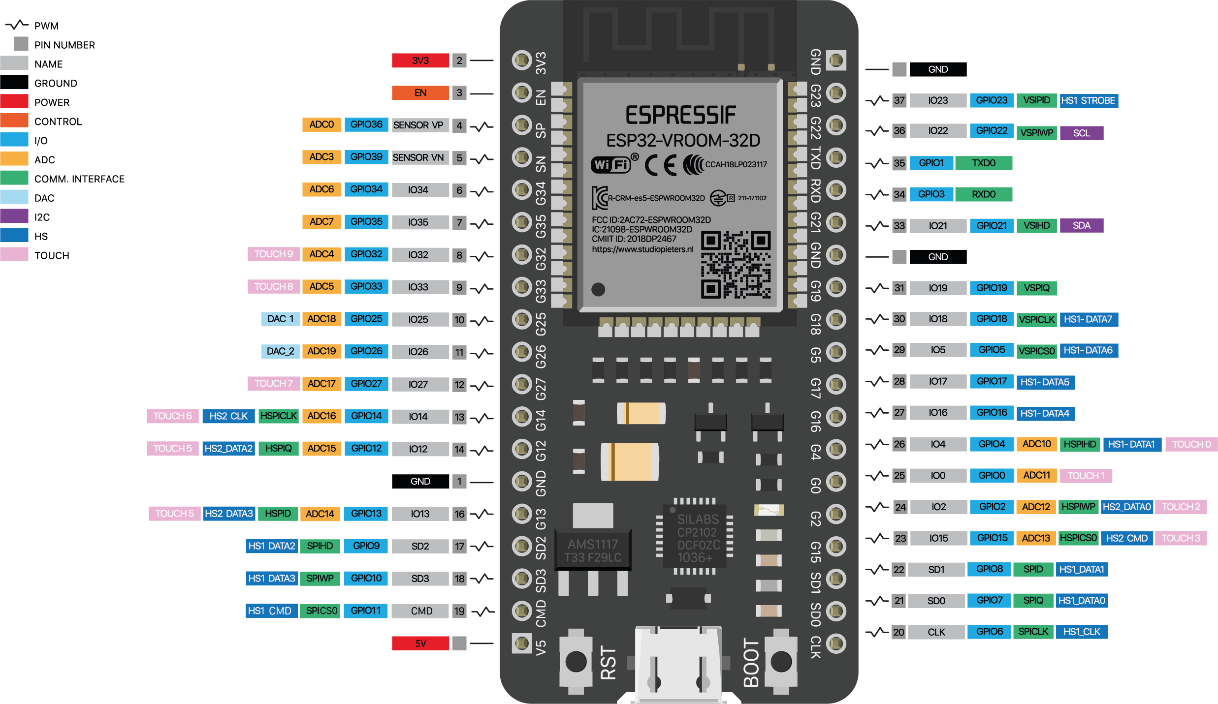
### Vi điều khiển ESP32

Cấu hình ESP32

CPU

* CPU: Xtensa Dual-Core LX6 microprocessor.
* Chạy hệ 32 bit
* Tốc độ xử lý 160MHZ up to 240 MHz
* Tốc độ xung nhịp đọc flash chip 40mhz --> 80mhz (tùy chỉnh khi lập trình)
* RAM: 520 KByte SRAM
* 520 KB SRAM liền chip –(trong đó 8 KB RAM RTC tốc độ cao – 8 KB RAM RTC tốc độ thấp (dùng ở chế độ DeepSleep).

Hỗ trợ 2 giao tiếp không dây

* Wi-Fi: 802.11 b/g/n/e/i
* Bluetooth: v4.2 BR/EDR và BLE

Hình 2.9: Các ngoại vi trên ESP32

Hỗ trợ tất cả các loại giao tiếp

* 8-bit DACs( Digital To Analog) – 2 cổng
* Analog(ADC): 12-bit 16 cổng.
* I²C – 2 cổng
* UART – 3 cổng
* SPI – 3 cổng (1 cổng cho chip FLASH )
* I²S – 2 cổng
* Thẻ SD /SDIO/MMC
* Slave (SDIO/SPI)
* Ethernet MAC
* CAN bus 2.0
* IR (TX/RX)
* Băm xung PWM (tất cả các chân )
* Chế độ tiết kiệm năng lượng

Cảm biến tích hợp trên chip ESP32

* 1 cảm biến Hall (cảm biến từ trường)
* 1 cảm biến đo nhiệt độ
* Cảm biến chạm (điện dung) với 10 đầu vào khác nhau

Bảo mật

* IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
* Bảo mật khi Boot
* Mã hóa flash

Nguồn điện hoạt động

* Nhiệt độ hoạt động -40 + 85C
* Điện áp hoạt động: 2.2-3.6V
* Số cổng GPIOs : 34

### PZEM-017 DC Energy Meter with Arduino – A blog about DIY solar and arduino projectsCảm biến PZEM – 017T

Hình 2.10. Cảm biến PZEM – 017T

Mạch đo điện DC đa năng 200A giao tiếp RS485 Modbus PZEM-017 được sử dụng để đo và theo dõi gần như hoàn toàn các thông số về điện năng DC của hệ thống như điện áp hoạt động, dòng tiêu thụ, công suất và năng lượng tiêu thụ, mạch sử dụng giao tiếp RS485 Modbus chuẩn công nghiệp dễ dàng kết nối truyền dữ liệu tới PLC, Vi điều kiển hoặc máy tính, thích hợp cho các ứng dụng theo dõi năng lượng, IoT,...

Thông số kỹ thuật:

* Model: PZEM-017 kèm trở Shunt 200A
* Điện áp đo và hoạt động: 0.05~300VDC, sai số 0.01
* Dòng điện đo và hoạt động: 0.02 ~ 200A, sai số 0.01
* Công suất đo và hoạt động: 0.2 ~ 90kW, sai số 0.1
* Năng lượng đo và hoạt động: 0~9999kWh
* Giao tiếp RS485 Modbus baudrate mặc định 9600, 8, 1.
* Có opto cách ly an toàn giữa mạch đo và mạch nhận tín hiệu RS485 Modbus.
* Lưu giữ thông số năng lượng tiêu thụ trong bộ nhớ.
* Kích thước: 90 x 60.5 mm

Mạch đo điện DC đa năng 200A giao tiếp RS485 Modbus PZEM-017 nhỏ gọn, dễ lắp đặt, đi kèm trở Shunt lên đến 100A, mạch có chất lượng gia công và linh kiện tốt, độ bền cao.

Sơ đồ đấu nối của cảm biến:

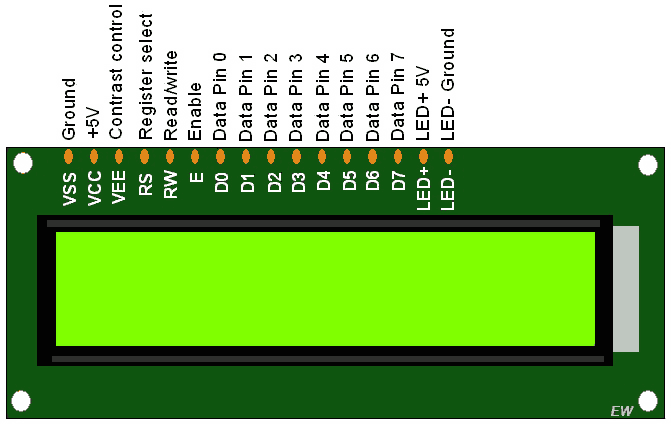
Hình 2.11: Sơ đồ đầu nối cảm biến PZEM – 017T



Nguyên tắc đo dòng điện dựa trên Rshunt:

Một điện trở shunt là một thành phần điện tử được sử dụng để đo và phát hiện dòng điện. Các giá trị điện trở nằm trong khoảng từ 100μΩ đến vài 100mΩ, với giá trị điện trở được sử dụng phổ biến nhất là vài MΩ đến vài trăm MΩ. Lý tưởng nhất là sử dụng điện trở shunt với điện trở thấp nhất có thể, nhưng trong thực tế, bạn nên chọn nó theo hệ số khuếch đại của bộ khuếch đại hoạt động và giá trị hiện tại được phát hiện. Đặc biệt với điện trở thấp, lượng điện áp giảm là nhỏ và bộ vi điều khiển rất khó phát hiện điện áp, do đó, sử dụng bộ khuếch đại vận hành có độ chính xác cao với điện áp bù đầu vào nhỏ để phát hiện dòng điện.

### Màn hình LCD 16x2

*Giới thiệu*

Hình 2.12: Màn hình hiển thị LCD 16x2

LCD có rất nhiều ưu điểm so với các dạng hiển thị khác: Nó có khả năng hiển thị kí tự đa dạng, trực quan dễ dàng đưa vào mạch ứng dụng theo nhiều giao thức giao tiếp khác nhau, tốn ít tài nguyên hệ thống và giá thành rẻ.[5]

*Cấu tạo*

LCD 16x2 có 2 hàng, mỗi hàng 16 ký tự, trong 16 chân của LCD được chia làm 3 dạng tín hiệu như sau:

* Các chân cấp nguồn: Chân số 1 nối mass (0V), chân số 2 là VDD nối với nguồn 5V, chân số 3 dùng để chỉnh contrast thường nối với biến trở.
* Các chân điều khiển: Chân số 4 là chân RS dùng để điều khiển lựa chọn thanh ghi. Chân R/W dùng để điều khiển quá trình đọc và ghi. Chân E là chân cho phép dạng xung chốt.
* Các chân dữ liệu DB0 - DB7: Là chân từ số 7 đến 14 dùng để trao đổi dữ liệu giữa thiết bị điều khiển và LCD.
* Chân 15 nối nguồn +5V hoặc 4.2V nối với led, chân 16 nối GND.

*Ứng dụng*

LCD thường được sử dụng trong các mạch điện tử, hiển thị thời gian thực, giá trị, kết quả, hiệu ứng.

*Module chuyển I2C LCD 16x2*

Module I2C LCD ra đời và giải quyết vấn đề LCD có quá nhiều nhiều chân gây khó khăn trong quá trình đấu nối và chiếm dụng nhiều chân trên vi điều khiển. [6]

Thay vì phải mất 6 chân vi điều khiển để kết nối với LCD 16x2 (RS, EN, D7, D6, D5 và D4) thì module IC2 bạn chỉ cần tốn 2 chân (SCL, SDA) để kết nối

Module I2C hỗ trợ các loại LCD sử dụng driver HD44780(LCD 16x2, LCD 20x4, ...) và tương thích với hầu hết các vi điều khiển hiện nay.

Ưu điểm của nó là tiết kiệm chân cho vi điều khiển, dễ dàng kết nối với LCD.

Một số thông số kỹ thuật :

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

### UART TTL to RS485 Two-way ConverterModule TTL-RS485

Hình 2.13: Module TTL-RS485

Tính năng:

* Module này chuyển từ TTL sang RS485 giúp khoản cách truyền xa lên đến 800m và chống nhiễu tốt.
* Module được thiết kế nhỏ gọn nhưng vẫn đảm bảo được khả năng truyền tín hiệu đi xa khi sử dụng với cặp dây cáp xoắn với nhau.
* IC MAX485 được tích hợp sẵn trên board. Truyền nhận với năng lượng thấp theo chuẩn RS-485.
* Các chân của IC đã được đưa ra thích hợp cho việc sử dụng với vi điều khiển.

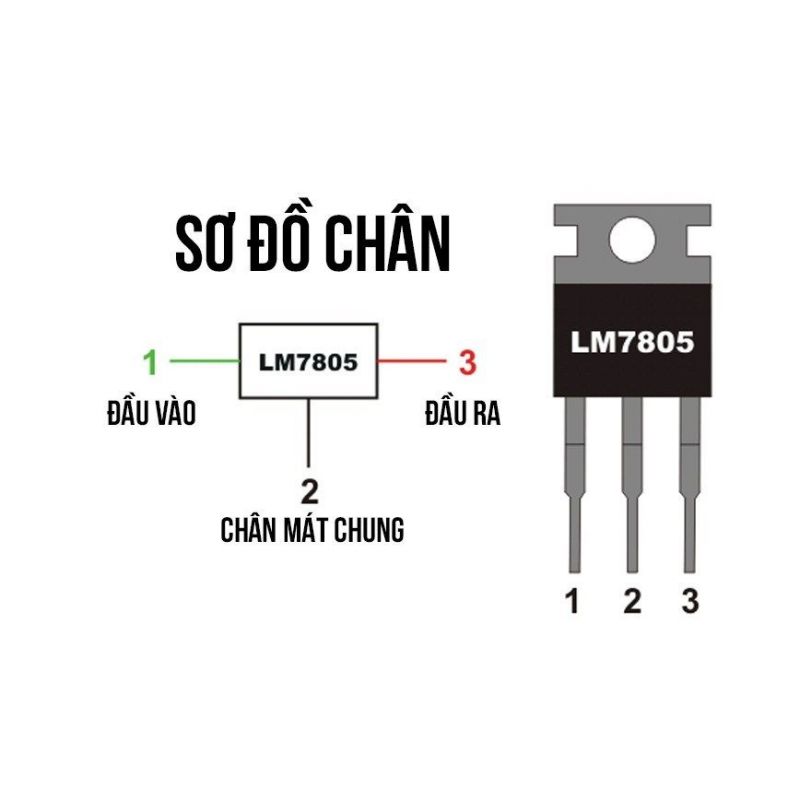
Thông số kỹ thuật:

* Max485 là bộ truyền nhận dữ liệu dùng cho chuẩn RS485.
* Tốc độ truyền tối đa: 2,5Mbps
* Điện áp hoạt động : 5V DC
* Dòng điện hoạt động: 300uA
* Bus Max485 truyền dữ liệu vi sai bằng 2 dây A, B nên khoảng cách truyền lớn, khả năng chống nhiễu tốt.
* Kết nối tối đa 128 thiết bị phù hợp với sự mở rộng hệ thống.

### IC ổn áp LM7805

IC ổn áp LM7805 giúp nguồn điện được ổn định tránh các trường hợp nguồn điện đầu vào lên cao hoặc xuống thấp hơn mức mong muốn. ICLM7805 điều chỉnh điện áp duy trì điện áp đầu ra ở giá trị không đổi là 5V. IC ổn áp 7805 là một thành viên của họp 78xx được sử dụng để ổn áp với nguyên lý tương tự, XX là chỉ số điện áp đầu ra được ổn áp.

Hình 2.14: IC LM7805



Các thông số của IC7805:

* Điện áp đầu vào: 7V- 35V
* Dòng điện đầu ra: 1A
* Điện áp đầu ra: VMax=5.2V ,VMin=4.8V

Bảng 2.2: Các chân và chức năng của LM7805

|  |  |
| --- | --- |
| Chân | Chức năng |
| Đầu vào | Đầu vào (7V-35V) |
| GND | Ground (0V) |
| Đầu ra | 5V (4.8V-5.2V) |

### Relay 5v - 5 chân 10A (2 chiếc) | Shopee Việt NamRelay 5V

Hình 2.15: Relay 5V

Bảng 2.3. Các thông số của Relay 5V

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

### Transistor C1815

Transistor C1815 là loại Transistor ngược (NPN) được sử dụng cho mục đích khuếch đại hoặc là đóng cắt các tải.

Bảng 2.4: Các thông số chính của transisor C1815

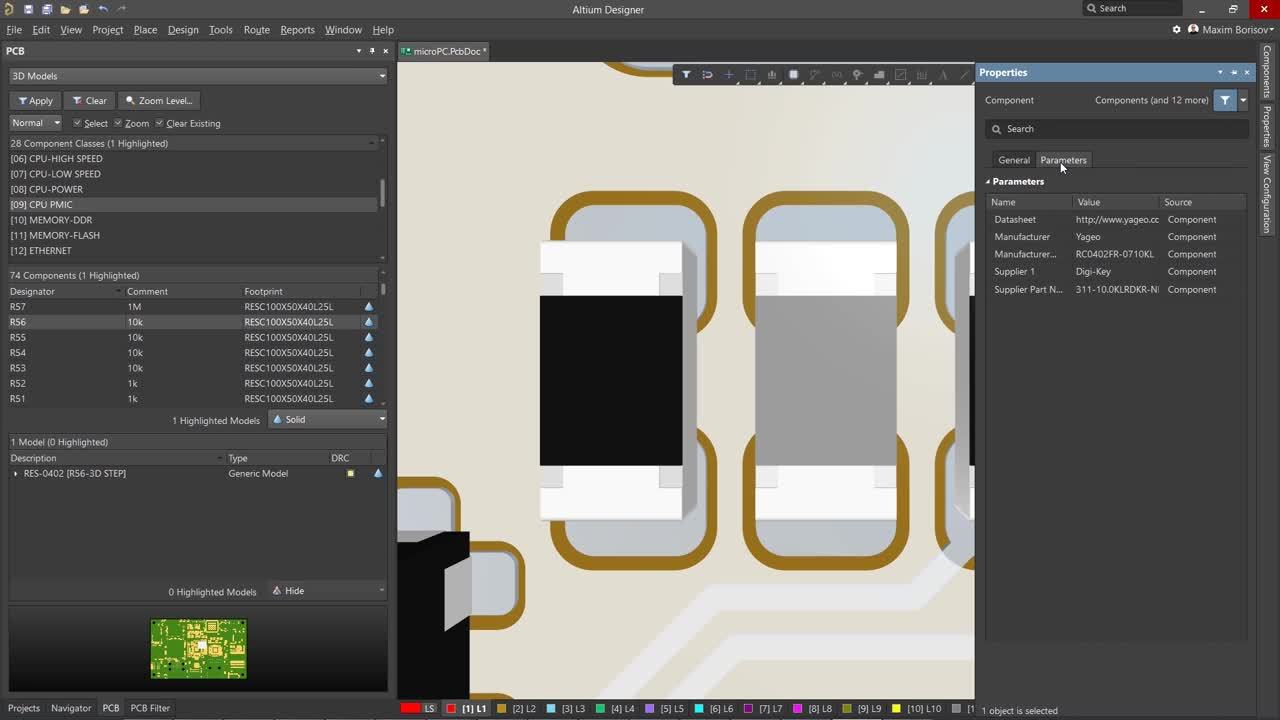
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ký hiệu | Thông số | Điều kiện thử nghiệm | Min | Max |
| ICBO | Dòng cắt Collector | VCB=60V, IE=0 |  | 0.1µA |
| IEBO | Dòng cắt Emitter | VEB=5V, IC=0 |  | 0.1µA |
| VCE (sat) | Điện áp bão hòa CE | IC=100mA, IB=10mA |  | 0.25V |
| VBE (sat) | Điện áp bão hòa BE | IC=100mA, IB=10mA |  | 1V |

## Phần mềm thiết kế và lập trình

### Phần mềm thiết kế Altium Designer

Altium Designer cung cấp một môi trường thiết kế hợp nhất, giúp cho các kỹ sư có một cái nhìn duy nhất về mọi khía cạnh của quy trình thiết kế PCB từ sơ đồ, đến bố cục PCB, để thiết kế. Bằng cách truy cập tất cả các công cụ thiết kế ở một nơi, các kỹ sư có thể hoàn thành toàn bộ quy trình thiết kế của họ trong cùng một môi trường trực quan và cung cấp các sản phẩm chất lượng cao một cách nhanh chóng.

Altium Designer dễ dàng tạo ra các thiết kế PCB với giao diện trực quan kết nối hoàn toàn bạn với mọi chi tiết của quy trình thiết kế điện tử từ ý tưởng đến sản xuất.

Biên dịch các dự án để tạo ra một mô hình duy nhất, gắn kết có thể đặt trung tâm cho quy trình thiết kế. Dễ dàng truy cập và thao tác dữ liệu chi tiết trong mô hình (sơ đồ, bố cục, mô phỏng) từ mô hình trung tâm này thay vì lưu trữ dữ liệu riêng cho từng phần tử thiết kế.

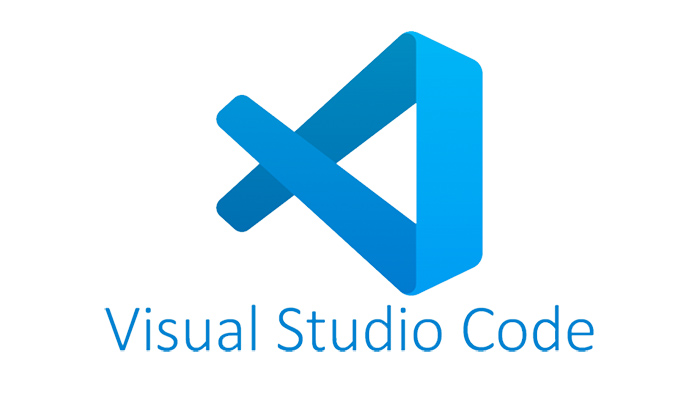
Hình 2.16: Một cửa sổ của Altium Designer

Ngoài ra, các phiên bản mới hiện nay hỗ trợ Altium 365 - là nền tảng thiết kế sản phẩm điện tử dựa trên đám mây, hợp nhất thiết kế PCB, MCAD, quản lý dữ liệu và làm việc cùng nhau

### Phần mềm Visual Studio Code

Visual Studio Code là một trình soạn thảo, biên tập code hoàn toàn miễn phí dành được Microsoft phát triển cho các lập trình viên và có mặt trên hầu hết các hệ điều hành phổ biến như: Windows, Linux và macOS. Có thể nói rằng, Visual Studio Code là một sự kết hợp độc đáo – đỉnh cao giữa IDE và Code Editor.

Không chỉ là soạn thảo, chỉnh sửa code, Visual Studio Code còn có thể hỗ trợ làm được rất nhiều việc như: đổi theme, hàng loạt phím tắt tiện dụng, có chức năng debug đi kèm, hỗ trợ Git, syntax highlighting hỗ trợ quá trình gõ code, phần gợi ý code thông minh,…



Hình 2.17: Visual Studio Code

Visual Studio Code tích hợp rất nhiều tính năng giúp người dùng có thể lập trình, thao tác với các thư mục của bạn một cách chuyên nghiệp nhất mà không cần phải thay đổi giữa các màn hình hay phải trở về thư mục gốc.

Visual Studio Code cũng có thể mở cùng lúc nhiều tệp tin, thư mục mà không cần thiết phải liên quan đến nhau.

Visual Studio Code là một phần mềm miễn phí và được sử dụng hầu hết với các lập trình viên.

### Platform IO

PlatformIO là một hệ sinh thái mã nguồn mở được viết trên Python để phát triển IoT và là một IDE đa nền tảng với trình gỡ lỗi hợp nhất chạy trên Windows, Mac và Linux.

Hình 2.18: Platform IO

PlatformIO đi kèm với trình quản lý thư viện cho các nền tảng như Arduino hay MBED cùng với kiểm thử phần mềm và cập nhật firmware.

PlatformIO hỗ trợ một số nền tảng như Arduino, ESP32, ESP8266 và đi kèm với một số ví dụ và thư viện. Nó độc lập với nền tảng mà nó đang chạy và chỉ yêu cầu Python được cài đặt trên máy tính.

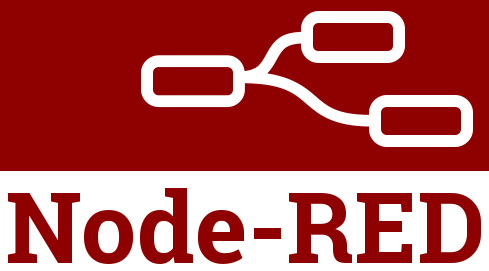
PlatformIO giúp người dùng viết chương trình nhanh chóng với các tính năng như hoàn tất code C/C ++ và công cụ lập trình thông minh để phát triển dự án chuyên nghiệp và nhanh chóng. Các tính năng này vốn không có trong Arduino IDE. Nó cũng đi kèm với gợi ý thông minh và định dạng khi lập trình. Các tính năng cốt lõi bao gồm hệ thống xây dựng đa nền tảng, trình quản lý thư viện, theo dõi cổng nối tiếp …

### Phần mềm Node-Red

Sử dụng Node-red để lập trình web điều khiển và giám sát:

[11] Node-Red là một công cụ lập trình dựa trên dạng luồng (flow), ban đầu được phát triển bởi nhóm Dịch vụ Công nghệ mới nổi của IBM và hiện là một phần của Tổ chức OpenJS.

Node-Red là một công cụ lập trình để kết nối các thiết bị phần cứng, API và dịch vụ trực tuyến theo những cách mới và thú vị. Nó cung cấp một trình soạn thảo dựa trên trình duyệt giúp dễ dàng kết nối các luồng cùng nhau bằng cách sử dụng một loạt các nút trong bảng màu có thể được triển khai để chạy trong một cú nhấp chuột đơn.



Hình 2.19. Node-red

Node-Red cung cấp một trình soạn thảo lưu đồ (flow) dựa trên trình duyệt giúp dễ dàng kết nối các luồng bằng cách sử dụng phạm vi rộng của các nút trong bảng màu. Các hàm viết bằng ngôn ngữ JavaScript có thể được tạo trong trình chỉnh sửa bằng trình chỉnh sửa văn bản phong phú. Thư viện tích hợp cho phép lưu các chức năng mẫu hoặc luồng để sử dụng. Các luồng được tạo có thể dễ dàng được chia sẽ và sử dụng dưới dạng tệp JSON

## Kết luận chương 2

# THIẾT KẾ HỆ THỐNG HỆ THỐNG GIÁM SÁT VÀ THỐNG KÊ DÒNG ĐIỆN, ĐIỆN ÁP DC

## Sơ đồ khối của hệ thống

## Sơ đồ nguyên lý toàn mạch



Hình 3.1: Sơ đồ nguyên lý của hệ thống

### Khối nguồn và ổn áp nguồn

*Khối nguồn*

Hình 3.2: Adapter nguồn 12V

Mạch sử dụng 12V-2A Delta chuyên dụng.

Thông số kỹ thuật:

* Điện áp đầu vào: 100-240V-0.5A
* Tần số hoạt động: 50-60 Hz
* Điện áp đầu ra : 12V-2A
* Model: ADP-24ZB
* Kích thước jack đầu ra: 5.5 x 2.1mm
* Chiều dài dây cắm từ ổ điện tới Adapter: 44 cm
* Chiều dài dây từ Adapter tới jack 12V: 48 cm
* Chiều dài sợi dây nối dài: 1.5m

Độ bảo vệ:

* Bảo vệ quá tải.
* Bảo vệ quá áp.
* Bảo vệ ngắn mạch.

*Khối ổn áp nguồn*

Hình 3.3: Khối ổn áp nguồn

Điện áp đầu vào được cung cấp qua 1 jack DC và điện áp 5V ở ngõ ra sẽ được lấy ở VCC5.

Tụ C3 và C2 để lọc điện áp cấp cho tải tiêu thụ lấy từ chân Vo của IC 7805, tụ C2 có các dụng cung cấp điện áp tạm thời cho tải khi điện áp tải đột ngột bị sụt áp, tụ C3 trở kháng lớn, C2 có tác dụng lọc nhiễu điện áp đầu ra (nhiễu là các điện áp không mong muốn làm cho dạng sóng điện áp ngõ ra có hình răng cưa).

IC 7805 dễ toả nhiệt nên để mạch hoạt động ổn định và lâu dài, chúng ta nên gắn thêm tản nhiệt cho IC

Mạch ổn áp này phù hợp để cấp nguồn cho các mạch điện tử vận hành với điện áp 5V và dòng điện 1A

### Khối xử lý trung tâm

Hình 3.4. Khối xử lý trung tâm

### Khối TTL-RS485

Hình 3.5: Khối TTL – RS485

Chứa module chuyển đổi TTL-RS485, giúp chuyển đổi tín hiệu TTL sang RS485 để giao tiếp với cảm biến đo dòng điện điện áp DC.

Hình 3.6: RS485 trên cảm biến

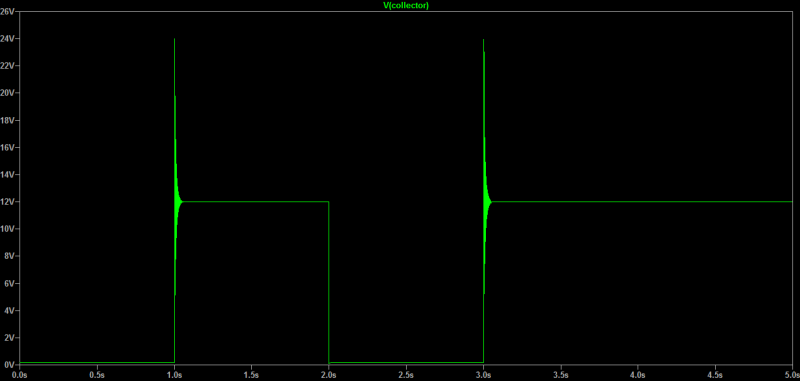


Module được cấp nguồn 5V và sử dụng ngoại vi là USART của ESP32. Chân TX của module nối với chân TX của ESP32 và chân RX của module nối với chân RX của cảm biến. Đầu ra của module là chuẩn RS485 với 2 chân A + và B - và chân GND. Chân GND để nối với mass của thiết bị nhằm giảm nhiễu cho hệ thống.

### Khối Relay

Hình 3.7: Khối relay

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

Khối Relay được điều khiển thông qua Transistor C1815 để điều khiển cuộn hút Relay. Khi chân B của Transistor ở mức 1 thì cuộn hút có điện sẽ nối chân C và E với nhau, khi đó đèn LED1 sẽ sáng và cuộn hút của Relay sẽ có diện, thường mở của relay sẽ đóng lại điều khiển các đèn và các thiết bị.

Hình 3.8: Nhiễu gây ra trên Relay

Ngoài ra, mạch sử dụng thêm các diode (1N4148) mắc song song và mắc ngược với cuộn hút của Relay. Mục đích của việc làm này nhằm ngăn chặn sự tăng đột biến điện áp lớn phát sinh khi nguồn điện bị ngắt (gọi là điện áp ngược) bảo vệ cho Relay và mạch điều khiển (hình 3.8)

### Khối hiển thị

Hình 3.9: Khối hiển thị

Kết nối tới màn hình LCD I2C:

* Chân GND: nối GND
* Chân VCC5: nối với VCC5
* Chân SDA, SCL: 2 chân dữ liệu của giao tiếp I2C

### Khối còi báo

Hình 3.10. Khối còi báo

Khối còi báo sử dụng transistor C1815 và một còi báo 5v. Khối còi báo có chức năng phát ra tiếng kêu khi điện áp hoặc dòng điện lớn hơn điện áp ngưỡng. Transistor có tác dụng khuếch đại dòng điện đi vào còi để dòng diện đi vào còi tăng hơn so với lấy trực tiếp từ vi điều khiển.

## Sơ đồ mạch in

*Đặt luật cho mạch in:*

* Khoảng cách giữa các thành phần (Clearance): 1mm
* Chiều rộng dây tín hiệu: 0.8mm
* Chiều rộng dây GND: 1mm
* Chiều rộng dây VCC: 1mm

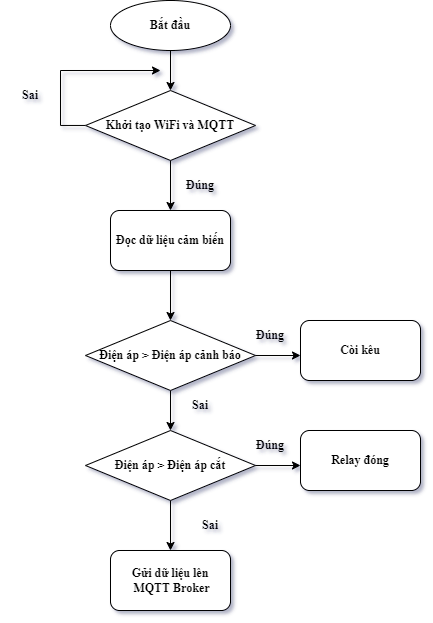
*Sơ đồ mạch in*



Hình 3.11: Sơ đồ mạch in của hệ thống

## Thiết kế phần mềm

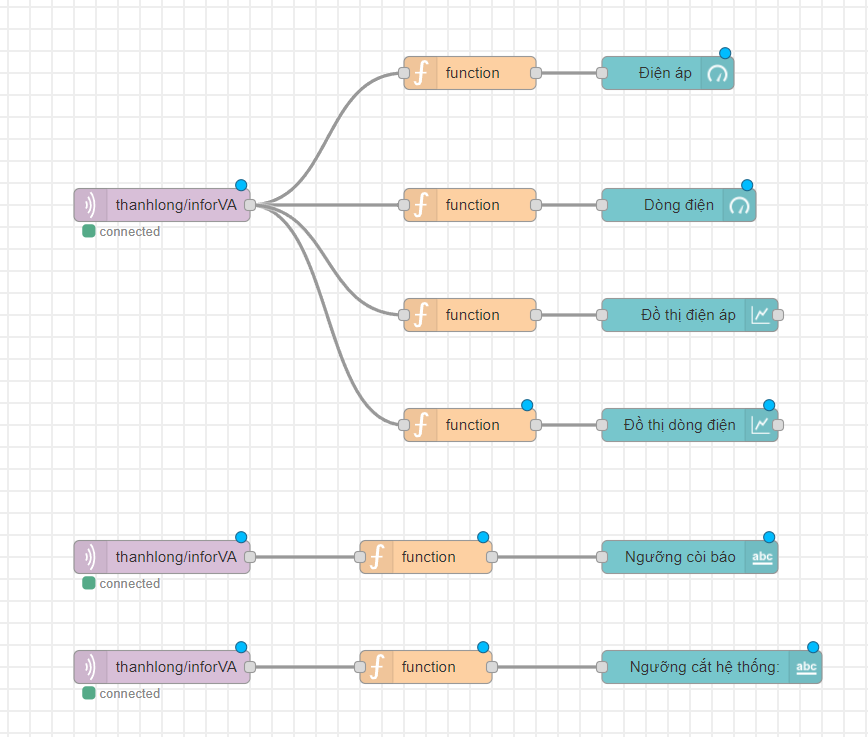
### Lưu đồ thuật toán



Hình 3.12. Lưu đồ thuật toán

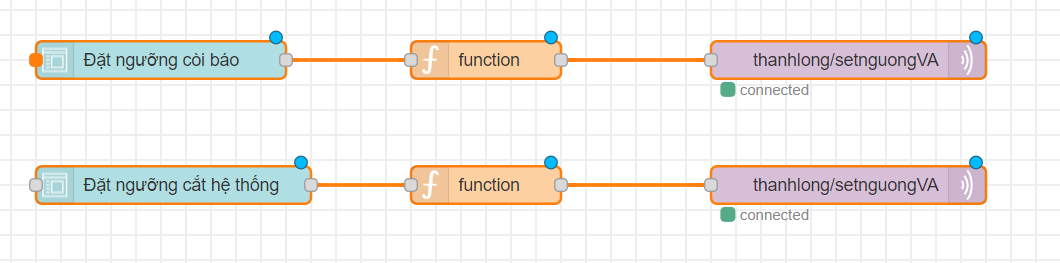
### Phần mềm giám sát trên Web

Xây dựng quá trình gửi và nhận giữa mạch ESP32 và Webserver trên Node-Red:

Quá trình nhận dữ liệu từ ESP32:

Hình 3.13: Quá trình nhận dữ liệu từ ESP32

Quá trình gửi dữ liệu từ Webserver xuống ESP32:



Hình 3.14. Quá trình gửi dữ liệu từ Webserver xuống ESP32

## Mô hình thực tế

Hình 3.15: Mô hình thực tế của hệ thống



Hình 3.16: Giao diện giám sát trên Webserver

# KẾT LUẬN

# TÀI LIỆU THAM KHẢO

https://en.wikipedia.org/wiki/Modbus

<https://www.javatpoint.com/mqtt-protocol>

<https://www.altium.com/altium-designer>

<https://www.electronicsforu.com/technology-trends/learn-electronics/7805-ic-voltage-regulator>

<https://wiki.tino.org/visual-studio-code-la-gi/>

<https://dientuviet.com/lap-trinh-arduino-voi-platformio/>

<https://www.cuidevices.com/blog/rs-485-serial-interface-explained>

[http://arduino.vn/tutorial/1570-gioi-thieu-module-esp32-va-huong-dan-cai-trinh-bi en-dich-tren-arduino-ide](http://arduino.vn/tutorial/1570-gioi-thieu-module-esp32-va-huong-dan-cai-trinh-bi%20en-dich-tren-arduino-ide)

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

# PHỤ LỤC

Code ESP32

#include <ModbusRTU.h

#include <PubSubClient.h>

#include <WiFi.h>

#include <LiquidCrystal\_I2C.h>

#include <ArduinoJson.h>

LiquidCrystal\_I2C lcd(0x27, 16, 2);

#define REG 0

#define REG\_NUM 2

#define SLAVE\_ID1 1

#define MBUS\_HW\_SERIAL Serial2

#define MBUS\_TXD\_PIN 17

#define MBUS\_RXD\_PIN 16

#define ssid "thanhlong"

#define password "thanhlong123"

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

#define mqtt\_user ""

#define mqtt\_pwd ""

const uint16\_t mqtt\_port = 1883;

int pinBuz = 18;

int pinRL = 19;

String mqtt\_topic\_infor\_VA = "thanhlong/inforVA";

String mqtt\_topic\_set\_nguong = "thanhlong/setnguongVA";//publish

WiFiClient espClient;

PubSubClient client(espClient);

ModbusRTU mb;

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

void setup\_wifi();

void reconnect();

void sendDataMQTT();

void pinInit();

void taskInit();

void displayDataTask();

void sendMqttTask();

void displayDataTask();

void readDataSensorTask();

void sendMqtt(void \*parameter);

void readDataSensor( void \* pvParameters);

void displayData(void \* pvParameters);

void readDataSensor( void \* pvParameters );

float nguongDienApCoi = 10;

float nguongDienApCat = 20;

uint16\_t hregs1[REG\_NUM];

xSemaphoreHandle xMutex;

Modbus::ResultCode err;

Modbus::ResultCode readSync(uint8\_t address, uint16\_t start, uint16\_t num, uint16\_t\* buf) {

xSemaphoreTake(xMutex, portMAX\_DELAY);

if (mb.slave()) {

xSemaphoreGive(xMutex);

return Modbus::EX\_GENERAL\_FAILURE;

}

Serial.printf("SlaveID: %d Hreg %d\r\n", address, start);

mb.readIreg(address, start, buf, num, [](Modbus::ResultCode event, uint16\_t, void\*) {

err = event;

return true;

});

while (mb.slave()) {

vTaskDelay(1);

mb.task();

}

Modbus::ResultCode res = err;

xSemaphoreGive(xMutex);

return res;

}

Modbus::ResultCode writeSync(uint8\_t address, uint16\_t start, uint16\_t num, uint16\_t\* buf) {

xSemaphoreTake(xMutex, portMAX\_DELAY);

if (mb.slave()) {

xSemaphoreGive(xMutex);

return Modbus::EX\_GENERAL\_FAILURE;

}

Serial.printf("SlaveID: %d Hreg %d\r\n", address, start);

mb.writeHreg(address, start, buf, num, [](Modbus::ResultCode event, uint16\_t, void\*) {

err = event;

return true;

});

while (mb.slave()) {

vTaskDelay(1);

mb.task();

}

Modbus::ResultCode res = err;

xSemaphoreGive(xMutex);

return res;

}

void setup() {

Serial.begin(115200);

MBUS\_HW\_SERIAL.begin(9600, SERIAL\_8N2, MBUS\_RXD\_PIN, MBUS\_TXD\_PIN);

mb.begin(&MBUS\_HW\_SERIAL);

mb.master();

xMutex = xSemaphoreCreateMutex();

WiFi.begin(ssid, password);

pinInit();

taskInit();

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

client.setCallback(callback);

lcd.init();

lcd.backlight();

}

void loop() {

setup\_wifi();

delay(5000);

}

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

{

String Data = "";

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

{

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

}

Serial.println(Data);

if(payload[length-1] == 65)

{

nguongDienApCoi = Data.toFloat();

}

else if(payload[length-1] == 66)

nguongDienApCat = Data.toFloat();

}

void setup\_wifi() {

delay(10);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

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

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

void reconnect()

{

while (!client.connected())

{

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

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

{

Serial.println("Connected MQTT");

client.subscribe(mqtt\_topic\_set\_nguong.c\_str());

}

else

{

Serial.println("Not Connnected MQTT");

delay(3000);

}

}

}

void sendDataMQTT()

{

DynamicJsonDocument doc(1024);

char inforInverterBuff[256];

doc["dienap"] = hregs1[0];

doc["dongdien"] = hregs1[1];

doc["nguongcoi"] = nguongDienApCoi;

doc["nguongcat"] = nguongDienApCat;

serializeJson(doc, inforInverterBuff);

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

}

void pinInit()

{

pinMode(pinBuz, OUTPUT);

pinMode(pinRL, OUTPUT);

}

void sendMqttTask()

{

xTaskCreatePinnedToCore( // Use xTaskCreate() in vanilla FreeRTOS

sendMqtt, // Function to be called

"sendMqtt", // Name of task

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

NULL, // Parameter to pass to function

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

NULL, // Task handle

1);

}

void sendMqtt(void \*parameter) {

while (1) {

sendDataMQTT();

vTaskDelay(2000 / portTICK\_PERIOD\_MS);

}

}

void readDataSensorTask()

{

xTaskCreatePinnedToCore(

readDataSensor, /\* Task function. \*/

"readDataSensor", /\* name of task. \*/

10000, /\* Stack size of task \*/

NULL, /\* parameter of the task \*/

10, /\* priority of the task \*/

NULL, /\* Task handle to keep track of created task \*/

0); /\* pin task to core 1 \*/

}

void readDataSensor( void \* pvParameters ){

while(true) {

delay(1000);

if (readSync(SLAVE\_ID1, REG, REG\_NUM, hregs1) == Modbus::EX\_SUCCESS)

{

for(int i = 0; i< REG\_NUM; i++)

{

Serial.print(REG+i);

Serial.print("\*\*\*\*\*\*\*");

Serial.println(hregs1[i]);

}

float fDienAp = (float)(hregs1[0]/100.0);

if(fDienAp > nguongDienApCoi)

{

digitalWrite(pinBuz, HIGH);

vTaskDelay(500);

digitalWrite(pinBuz, LOW);

vTaskDelay(500);

}

else

digitalWrite(pinBuz, LOW);

if(fDienAp > nguongDienApCat)

{

digitalWrite(pinRL, HIGH);

}

else

digitalWrite(pinRL, LOW);

}

else

Serial.println("ko doc duoc data");

}

}

void displayDataTask()

{

xTaskCreatePinnedToCore(

displayData, /\* Task function. \*/

"displayData", /\* name of task. \*/

2048, /\* Stack size of task \*/

NULL, /\* parameter of the task \*/

0, /\* priority of the task \*/

NULL, /\* Task handle to keep track of created task \*/

0); /\* pin task to core 1 \*/

}

void displayData(void \* pvParameters )

{

while(true) {

if (!client.connected()) {

reconnect();

}

client.loop();

char cDienAp[20];

char cDongDien[20];

float fDienAp = (float)(hregs1[0]/100.0);

float fDongdien = (float)(hregs1[1]/100.0);

sprintf(cDienAp, "Ap DC: %0.2f V ", fDienAp);

lcd.setCursor(0, 0);

lcd.print(cDienAp);

lcd.setCursor(0, 1);

sprintf(cDongDien, "Dong DC: %0.2f A ", fDongdien);

lcd.print(cDongDien);

}

}

void taskInit()

{

readDataSensorTask();

sendMqttTask();

displayDataTask();

}