

**BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN
PHÂN HIỆU TRƯỜNG ĐẠI HỌC THỦY LỢI
BỘ MÔN CÔNG NGHỆ THÔNG TIN**



BÁO CÁO MÔN CHUYÊN ĐỀ CNTT

Tên đề tài: Xây dựng hệ thống tưới nước tự động

Giảng viên hướng dẫn: Ths. Viên Thanh Nhã

Sinh viên thực hiện: Nguyễn Hoài Nguyệt An – 2051067525

Cao Hỷ Khang - 2051067547

Lớp: 62TH

TPHCM, ngày 30 tháng 9 năm 2023

MỤC LỤC

LỜI MỞ ĐẦU	4
CHƯƠNG 1: XÁC ĐỊNH VÀ PHÂN TÍCH YÊU CẦU.....	5
1.1. Tổng quát về nhu cầu tưới tiêu tự động trong nông nghiệp và vườn rau mini tại nhà ở.....	5
1.2. Khái niệm IoT	5
1.3. Mục tiêu và phạm vi nghiên cứu	6
1.4. Trình tự tìm hiểu các phương thức và thực hiện nghiên cứu	6
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	6
2.1. Mạch Aduino ESP32	7
2.1.1. Cấu hình ESP32	7
2.1.2. Các chân và sơ đồ chân của ESP32.....	8
2.1.3. NodeMCU ESP32 Wifi BLE - CP2102- (kit thu phát IoT).....	11
2.1.4. Cài đặt Aduino IDE và ngôn ngữ sử dụng	13
2.1.4.1. Môi trường lập trình (IDE)	13
2.1.4.2. Thư viện lập trình (SDK).....	13
2.1.4.3. Aduino IDE	13
2.1.4.4. Cài đặt Aduino IDE	14
2.1.5. Cài đặt ESP32	17
2.2. Relay Kích Mức Thấp 5V.....	20
2.3. DHT11 - Cảm Biến Nhiệt Độ và Độ Ẩm cho Arduino.....	21
2.4. Cảm biến đo độ ẩm đất.....	21
2.5. Mạch hạ áp ổn áp 5V - 3A.....	22
2.6. Động Cơ Bơm Chìm Mini V2 3V~6V.....	23
2.7. Test board mạch hàn	23
2.8. Mạch chống nhiễu điện từ LC Technology	23
2.9. Khái quát các ngôn ngữ sử dụng lập trình Website điều khiển hệ thống tưới từ xa.....	24
2.9.1. HTML	24
2.9.2. CSS	25
2.9.3. JavaScript.....	27
2.9.4. Bootstrap	28
2.10. Giao thức MQTT	31
2.11. Ứng dụng di động MyMQTT.....	33
CHƯƠNG 3: MÔ HÌNH.....	35
CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC	35
4.1. Hình ảnh hệ thống tưới nước tự động.....	35
4.2. Hình ảnh website điều khiển hoạt động tưới nước từ xa	36
CHƯƠNG 5: DEMO	36
5.1. Lập trình ESP32, kết nối với Wifi và MQTT	36
5.2. Lập trình website.....	47

MỤC LỤC HÌNH ẢNH

CHƯƠNG 1: XÁC ĐỊNH VÀ PHÂN TÍCH YÊU CẦU.....	5
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	6
Hình 2.1.1.1. Sơ đồ khối chức năng của ESP32	7
Hình 2.1.2.1. Sơ đồ chân của Module ESP-WROOM-32.....	8
Hình 2.1.3.1. Kit NodeMCU ESP32 Wifi BLE	12
Hình 2.1.3.2. Module trung tâm: Mạch thu phát Wifi BLE SoC ESP32 ESP-WROOM-32E.....	12
Hình 2.1.3.3. Tích hợp Nút BOOT và ENABLE, đèn LED màu đỏ (nguồn) và xanh lam (GPIO2)	12
Hình 2.1.3.4. Kiểu Anten: PCB.....	13
Hình 2.1.3.5. Tích hợp mạch nạp và giao tiếp UART CH340.....	13
Hình 2.1.4.3.1. Một góc giao diện Aduino IDE.....	13
Hình 2.1.4.4.1. Giao diện Aduino IDE	17
Hình 2.2.1. Module 1 Relay Kích Mức Thấp 5V.....	20
Hình 2.3.1. DHT11 - Cảm Biến Nhiệt Độ và Độ Ẩm.....	21
Hình 2.4.1. Cảm biến đo độ ẩm đất.....	21
Hình 2.5.1. Mạch hạ áp ổn áp 5V- 3A	22
Hình 2.6.1. Động Cơ Bơm Chìm Mini V2 3V~6V	23
Hình 2.7.1. Test board mạch hàn	23
Hình 2.8.1. Mạch chống nhiễu điện từ, bảo vệ tiếp điểm LC Technology	23
Hình 2.8.2. Sơ đồ đấu nối của Mạch chống nhiễu điện từ, bảo vệ tiếp điểm LC Technology	24
Hình 2.9.1.1. Bộ cục HTML5	25
Hình 2.9.2.1. Bộ cục của CSS	26
Hình 2.9.2.2. Minh họa CSS dạng Inline	26
Hình 2.9.2.3. Minh họa CSS dạng Internal	26
Hình 2.9.2.4. Minh họa CSS dạng External	27
Hình 2.9.3.1. Minh họa CSS dạng Internal	27
Hình 2.9.3.2. Minh họa CSS dạng External	28
Hình 2.11.1. Giao diện MyMQTT sau khi đã tải về máy	34
CHƯƠNG 3: MÔ HÌNH (Hình vẽ mô phỏng sơ đồ kết nối mạch)	35
Hình 3.1. Hình vẽ mô phỏng sơ đồ kết nối mạch.....	35
CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC	35
Hình 4.1.1. Hệ thống tưới nước tự động	35
Hình 4.2.1. Website điều khiển hoạt động tưới nước từ xa	36
CHƯƠNG 5: DEMO.....	36

LỜI MỞ ĐẦU

Việc tận dụng các công nghệ phát triển, cụ thể là IoT vào đời sống đang dần trở nên phổ biến. Với mục đích giải quyết đơn giản hóa các công việc tay chân nặng nhọc thường ngày, các nghiên cứu ứng dụng được sáng tạo trong quá trình tích hợp giữa các dữ liệu, linh kiện điện tử và hệ thống máy tính ra đời nhanh chóng.

Các thành phẩm này hỗ trợ cải tiến hiệu suất công việc, đảm bảo độ bảo mật và tin cậy thông qua các tính năng nhận dạng, cũng như cải thiện tình hình kinh tế đất nước theo xu hướng tiện lợi bằng cách dự đoán hiệu quả dựa vào sự kết nối và giao tiếp giữa các thiết bị.

Hiện nay, Việt Nam xác định nông nghiệp là trụ đỡ của nền kinh tế, góp phần nâng cao đời sống nhân dân, giữ vững ổn định chính trị, trật tự an toàn xã hội. Do đó, thông qua các nhu cầu học tập và làm việc hiện có, chúng tôi đã bắt đầu tìm hiểu sâu hơn và quyết định ứng dụng lĩnh vực IoT trong nông nghiệp một cách thiết thực nhất với đề tài là “Xây dựng hệ thống tưới nước tự động”.

Trong quá trình thực hiện, chúng tôi đã từng bước nghiên cứu, phân tích và ứng dụng mô phỏng hệ thống tưới nước tự động trong thực tế. Song, kinh nghiệm còn non trẻ và kiến thức còn hạn chế nên vẫn không tránh khỏi những thiếu sót. Vì vậy, chúng tôi rất mong nhận được sự đóng góp ý kiến của giảng viên để có thể xây dựng đề tài ngày càng hoàn thiện hơn.

Chúng tôi xin chân thành cảm ơn!

CHƯƠNG 1: XÁC ĐỊNH VÀ PHÂN TÍCH YÊU CẦU

1.1. Tổng quát về nhu cầu tưới tiêu tự động trong nông nghiệp và vườn rau mini tại nhà ở

Hệ thống tưới tự động là một phương pháp hiện đại và tiện lợi để cung cấp nước và dưỡng chất cho cây trồng một cách tự động, mà không cần can thiệp trực tiếp từ người lao động. Điều này mang lại nhiều lợi ích quan trọng, không chỉ trong việc tối ưu hóa sử dụng lao động mà còn trong việc cải thiện hiệu suất và bảo vệ môi trường.

Hệ thống này có sự đa dạng trong cách tưới nước, bao gồm tưới nhỏ giọt, tưới phun sương, tưới phun mưa và nhiều phương pháp khác. Đặc biệt, hệ thống này được điều khiển bởi các bộ điều khiển điện tử kết hợp với các cảm biến, cho phép chúng hoạt động theo lịch trình được thiết lập trước đó một cách chính xác.

Không chỉ giúp giảm sự tốn kém về lao động không cần thiết, hệ thống tưới tự động còn giúp hạn chế sử dụng các loại thuốc trừ sâu và phân bón hóa học, những thứ có thể gây hại cho môi trường và sức khỏe của con người. Điều này đồng nghĩa với việc thúc đẩy phát triển nông nghiệp bền vững và thân thiện với môi trường.

Hệ thống này cũng rất linh hoạt trong việc điều chỉnh thời gian và tần suất tưới nước dựa trên môi trường cụ thể và nhu cầu của cây trồng. Điều này không chỉ giúp tránh lãng phí tài nguyên nước mà còn giúp cải thiện tính thẩm mỹ và hiệu suất của cảnh quan nông nghiệp.

1.2. Khái niệm IoT

Internet of Things (IoT), được dịch sang tiếng Việt là "Internet vạn vật" hoặc "vạn vật kết nối", đại diện cho một sự tiến bộ quan trọng trong cuộc cách mạng công nghệ hiện đại. Theo định nghĩa bởi Wikipedia, IoT là một khái niệm trong đó mọi đối tượng, từ các thiết bị điện tử thông minh cho đến con người, đều được trang bị một mã định danh riêng biệt. Tất cả những đối tượng này được kết nối và có khả năng truyền tải thông tin và dữ liệu thông qua một hệ thống mạng toàn cầu, mà không cần sự can thiệp trực tiếp của con người hoặc máy tính.

IoT ra đời là kết quả của sự tiến bộ đáng kể trong công nghệ không dây, công nghệ vi mạch và Internet. Các yếu tố này đã hội tụ để tạo ra một hệ thống phức tạp, bao gồm một loạt các thiết bị thông minh, mỗi thiết bị đều có khả năng kết nối với nhau, từ Internet tới thế giới xung quanh, để thực hiện một loạt nhiệm vụ và chức năng đa dạng.

IoT đã mở ra một thế giới đầy tiềm năng, đem lại sự xuất hiện của hàng loạt ứng dụng và dịch vụ mới mà trước đây chúng ta chỉ có thể mơ ước. Từ máy giặt thông minh cho đến đèn chiếu sáng có khả năng tự động điều chỉnh độ sáng, từ xe hơi tự lái đến thiết bị y tế giám sát sức khỏe, tất cả đều có thể trở thành một phần của mạng lưới IoT, chia sẻ thông tin và tương tác với môi trường xung quanh một cách thông minh. IoT mở ra nhiều cơ hội đảm bảo cho vấn đề cải thiện hiệu suất, tiết kiệm năng lượng, tạo ra môi trường sống thông minh và tiện nghi hơn cho chúng ta. Điều này mang đến tư duy mới thay đổi cách chúng ta tương tác với thế giới xung quanh và cải thiện chất lượng cuộc sống.

1.3. Mục tiêu và phạm vi nghiên cứu

Mục tiêu của chúng tôi trong việc thực hiện đồ án "Xây dựng hệ thống tưới nước tự động" không chỉ đơn thuần là tạo ra một sản phẩm tưới cây tự động nhỏ với nguyên tắc hoạt động dựa trên cảm biến độ ẩm của đất. Mục tiêu lớn hơn là phát triển một hệ thống hoàn chỉnh, có khả năng tương tác thông qua các giao diện điều khiển từ xa. Chúng tôi hướng đến việc cung cấp một giải pháp tiện lợi và hiệu quả cho việc tưới cây, giúp tiết kiệm thời gian và công sức của người trồng cây.

Để đạt được điều này, chúng tôi đang tập trung vào việc nghiên cứu và phát triển các tính năng bổ sung như theo dõi điều kiện thời tiết, hiển thị lịch trình tưới cây tối ưu dựa trên dữ liệu độ ẩm và điều kiện thời tiết. Bằng cách kết hợp công nghệ và thiết kế mạch đơn giản, chúng tôi hy vọng rằng sản phẩm của mình sẽ trở thành một giải pháp phổ biến và hữu ích trong việc quản lý vườn cây cá nhân và các khu vườn nhỏ khác.

1.4. Trình tự tìm hiểu các phương thức và thực hiện nghiên cứu

Trước hết, chúng tôi tiến hành khám phá chi tiết về mạch ESP32 để hiểu rõ hơn về cấu trúc và tính năng của nó. Chúng tôi tập trung vào việc tìm hiểu về các linh kiện cần thiết cho hệ thống tưới nước tự động, bao gồm relay để điều khiển máy bơm, cảm biến DHT11 để đo độ ẩm, mạch hạ áp DC để cung cấp năng lượng cho mạch, máy bơm để tưới nước, board mạch để kết nối và điều khiển các linh kiện này.

Tiếp theo, chúng tôi tìm hiểu về cách cài đặt môi trường phát triển tích hợp (IDE) cho ESP32 và làm quen với các công cụ lập trình. Chúng tôi xem xét giao thức MQTT, một phương thức truyền dữ liệu quan trọng trong dự án và tìm hiểu ngôn ngữ lập trình cần sử dụng để viết mã cho hệ thống. Chúng tôi cũng nghiên cứu cách viết mã cho trang web điều khiển, một phần quan trọng của dự án để điều khiển hệ thống từ xa.

Khi chúng tôi đã đủ kiến thức căn bản, chúng tôi bắt đầu quá trình lắp ráp mạch thực tế, kết hợp các linh kiện đã nêu trên và viết mã trên Arduino IDE để kiểm tra cách hoạt động của từng linh kiện riêng lẻ. Chúng tôi cũng tiến hành việc hàn các kết nối và linh kiện lên bảng mạch, đảm bảo rằng chúng hoạt động một cách ổn định và đáng tin cậy.

Cuối cùng, chúng tôi bắt tay vào việc viết mã cho trang web điều khiển hệ thống từ xa, tạo giao diện web đẹp và thân thiện để người dùng có thể dễ dàng theo dõi và điều khiển hệ thống tưới nước tự động từ bất kỳ đâu trên thế giới.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

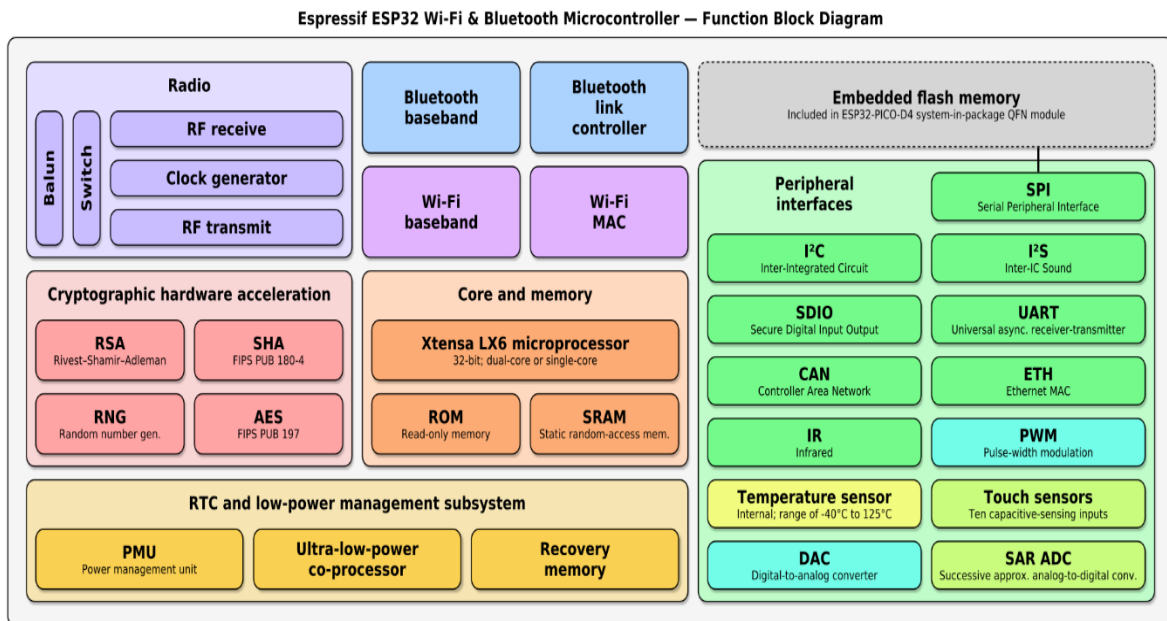
2.1. Mạch Arduino ESP32

ESP32 là một vi điều khiển với giá cả phải chăng và tiêu tốn năng lượng thấp, được trang bị tính năng hỗ trợ kết nối WiFi và dual-mode Bluetooth (hay còn được gọi là Bluetooth chế độ kép).

Ngoài việc hỗ trợ kết nối không dây, ESP32 còn sở hữu khả năng xử lý đa nhiệm với nhiều chức năng khác nhau như cảm biến, động cơ, màn hình và các giao tiếp khác. Điều này làm cho nó trở thành một lựa chọn phổ biến không chỉ trong các dự án IoT mà còn trong các ứng dụng nhúng đa dạng.

Không chỉ đơn thuần là một bản cập nhật, ESP32 được coi là sự tiếp nối hoàn hảo từ dòng vi điều khiển trước đó là ESP8266.

2.1.1. Cấu hình ESP32



Hình 2.1.1.1. Sơ đồ khối chức năng của ESP32.

- **Bộ xử lý:**

CPU: Bộ vi xử lý Xtensa lõi kép (hoặc lõi đơn) 32-bit LX6 600DMIPS, tốc độ xử lý 160MHz → 240 MHz, tốc độ xung nhịp đọc flash chip 40MHz → 80MHz.

Bộ đồng xử lý (co-processor) công suất cực thấp (Ultra low power, viết tắt: ULP) hỗ trợ việc đọc ADC và các ngoại vi khi bộ xử lý chính (main processor) vào chế độ deep sleep.

- **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)

- **Điều kiện hoạt động**

Điện áp hoạt động: Từ 2.2V – 3.6V

Nhiệt độ hoạt động: -40 → 85 độ C

Số cổng GPIOs: 34 cổng với các ngoại vi

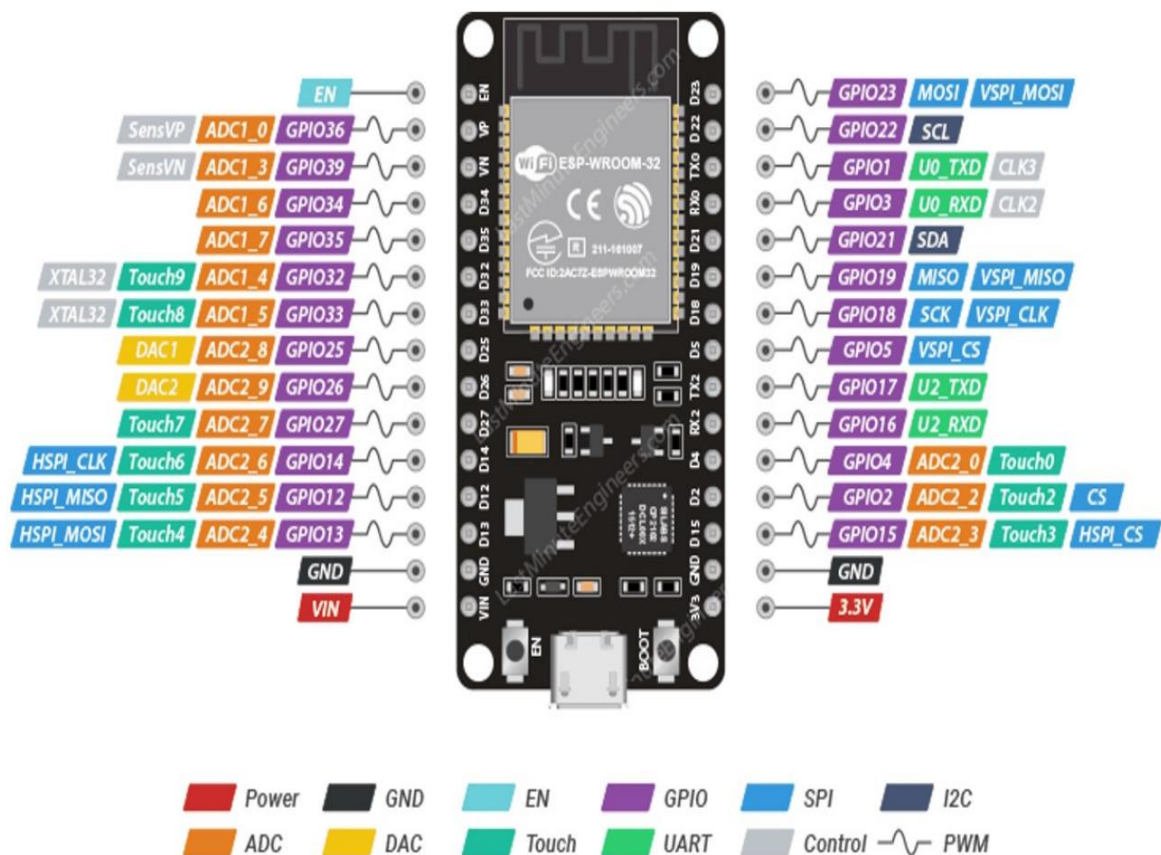
➤ ADC SAR 12 bit, 18 kênh

➤ 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. 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. Các GPIO đều có thể được dùng để triển khai HSPI và VSPI.
- 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
- 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 IR (TX/RX)
- 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)

2.1.2. Các chân và sơ đồ chân của ESP32



Hình 2.1.2.1. Sơ đồ chân của Module ESP-WROOM-32

Trong đó:

- Các chân được đánh dấu màu xanh lá cây là OK để sử dụng.
- Các chân được đánh dấu màu vàng thì có thể sử dụng, nhưng cần chú ý vì chúng có thể có hoạt động không như mong muốn khi khởi động.
- Các chân được đánh dấu màu đỏ không được khuyến khích sử dụng làm đầu vào hoặc đầu ra.

GPIO Pin	Input	Output	Chức năng đặc biệt
0	pulled up	OK	xuất tín hiệu PWM khi khởi động
1	TX pin	OK	debug đầu ra khi khởi động
2	OK	OK	được kết nối với đèn LED trên bo mạch
3	OK	RX pin	HIGH khi khởi động
4	OK	OK	
5	OK	OK	xuất tín hiệu PWM khi khởi động
6	X	X	được kết nối với đèn flash SPI tích hợp
7	X	X	được kết nối với đèn flash SPI tích hợp
8	X	X	được kết nối với đèn flash SPI tích hợp
9	X	X	được kết nối với đèn flash SPI tích hợp
10	X	X	được kết nối với đèn flash SPI tích hợp
11	X	X	được kết nối với đèn flash SPI tích hợp
12	OK	OK	khởi động thất bại nếu kéo lên cao
13	OK	OK	
14	OK	OK	xuất tín hiệu PWM khi khởi động
15	OK	OK	xuất tín hiệu PWM khi khởi động
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	

26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		input only
35	OK		input only
36	OK		input only
39	OK		input only

- **Các chân INPUT ONLY: GPIO từ 34 đến 39** là GPI – chân chỉ đầu vào. Các chân này không có điện trở kéo lên hoặc kéo xuống bên trong. Chúng không thể được sử dụng làm đầu ra, vì vậy chỉ sử dụng các chân này làm đầu vào:

➤ GPIO 34, GPIO 35, GPIO 36, GPIO 39, Chân tích hợp Flash trên ESP32

- **GPIO 6 đến GPIO 11** dùng để kết nối Flash SPI, không khuyến khích sử dụng trong các ứng dụng khác

➤ GPIO 6 (SCK/CLK), GPIO 7 (SDO/SD0), GPIO 8 (SDI/SD1)

➤ GPIO 9 (SHD/SD2), GPIO 10 (SWP/SD3), GPIO 11 (CSC/CMD)

- **Chân cảm biến điện dung**

Các chân ESP32 này có chức năng như 1 nút nhấn cảm ứng, có thể phát hiện sự thay đổi về điện áp cảm ứng trên chân.

Các cảm biến cảm ứng bên trong đó được kết nối với các GPIO sau:

➤ T0 (GPIO 4), T1 (GPIO 0), T2 (GPIO 2), T3 (GPIO 15), T4 (GPIO 13)

➤ T5 (GPIO 12), T6 (GPIO 14), T7 (GPIO 27), T8 (GPIO 33), T9 (GPIO 32)

- **Analog to Digital Converter (ADC)**

ESP32 có các kênh đầu vào ADC 18 x 12 bit (trong khi ESP8266 chỉ có ADC 1x 10 bit).

Đây là các GPIO có thể được sử dụng làm ADC và các kênh tương ứng:

➤ ADC1_CH0 (GPIO 36), ADC1_CH1 (GPIO 37), ADC1_CH2 (GPIO 38)

➤ ADC1_CH3 (GPIO 39), ADC1_CH4 (GPIO 32), ADC1_CH5 (GPIO 33)

➤ ADC1_CH6 (GPIO 34), ADC1_CH7 (GPIO 35), ADC2_CH0 (GPIO 4)

➤ ADC2_CH1 (GPIO 0), ADC2_CH2 (GPIO 2), ADC2_CH3 (GPIO 15)

➤ ADC2_CH4 (GPIO 13), ADC2_CH5 (GPIO 12), ADC2_CH6 (GPIO 14)

➤ ADC2_CH7 (GPIO 27), ADC2_CH8 (GPIO 25), ADC2_CH9 (GPIO 26)

- **Digital to Analog Converter (DAC)**

Có các kênh DAC 2 x 8 bit trên ESP32 để chuyển đổi tín hiệu kỹ thuật số thành đầu ra tín hiệu điện áp tương tự. Các kênh này chỉ có độ phân giải 8 bit, nghĩa là có giá trị từ 0 – 255 tương ứng với 0 – 3.3V

Đây là các kênh DAC:

- DAC1 (GPIO25)
- DAC2 (GPIO26)

- **Các chân thời gian thực RTC**

Các chân này có tác dụng đánh thức ESP32 khi đang trong chế độ Low Power Mode. Sử dụng như 1 chân ngắt ngoài.

Các chân RTC:

- RTC_GPIO0 (GPIO36), RTC_GPIO3 (GPIO39), RTC_GPIO4 (GPIO34)
- RTC_GPIO5 (GPIO35), RTC_GPIO6 (GPIO25), RTC_GPIO7 (GPIO26)
- RTC_GPIO8 (GPIO33), RTC_GPIO9 (GPIO32), RTC_GPIO10 (GPIO4)
- RTC_GPIO11 (GPIO0), RTC_GPIO12 (GPIO2), RTC_GPIO13 (GPIO15)
- RTC_GPIO14 (GPIO13), RTC_GPIO15 (GPIO12), RTC_GPIO16 (GPIO14)
- RTC_GPIO17 (GPIO27)

- **Chân PWM**

ESP32 LED PWM có 16 kênh độc lập có thể được định cấu hình để tạo tín hiệu PWM với các thuộc tính khác nhau. Tất cả các chân có thể hoạt động như đầu ra đều có thể được sử dụng làm chân PWM (**GPIO từ 34 đến 39 không thể tạo PWM**).

Để xuất PWM, cần xác định các thông số này trong code:

- Frequency – tần số, Duty cycle, Kênh PWM
- Chân GPIO nơi bạn muốn xuất tín hiệu

- **Chân I2C**

ESP32 có hai kênh I2C và bất kỳ chân nào cũng có thể được đặt làm SDA hoặc SCL. Khi sử dụng ESP32 với Arduino IDE, các chân I2C mặc định là:

- GPIO 21 (SDA)
- GPIO 22 (SCL)

Nếu muốn sử dụng chân khác cho việc điều khiển I2C có thể sử dụng code:

- `Wire.begin(SDA, SCL);`

- **Chân ngắt ngoài**

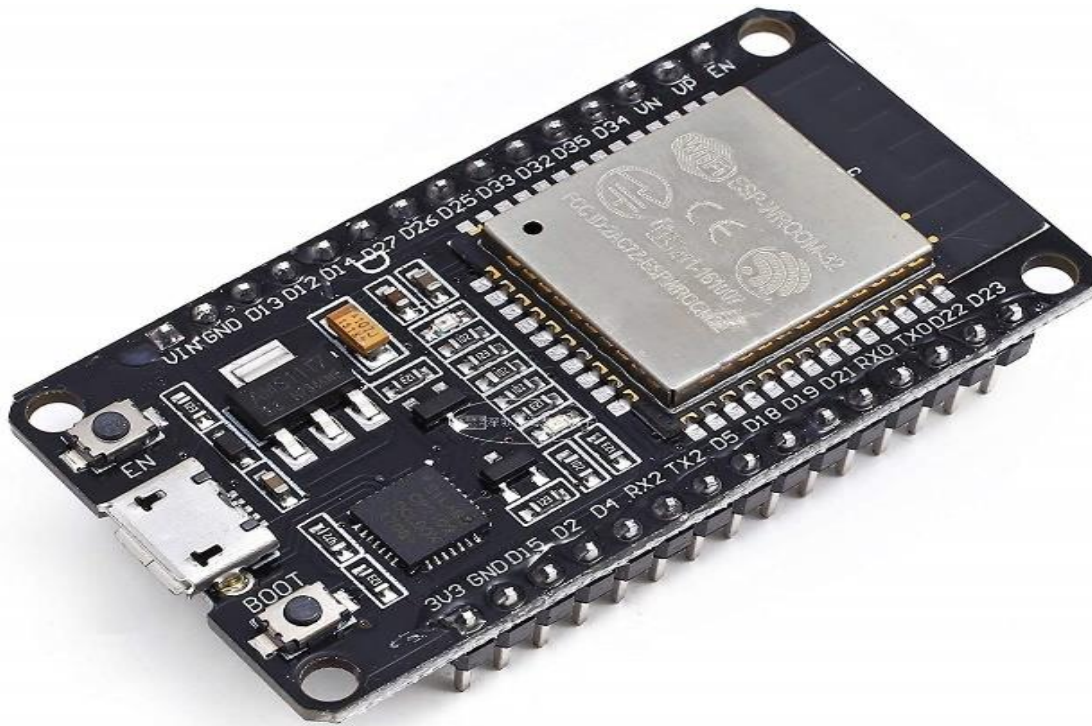
Tất cả các chân ESP32 đều có thể sử dụng ngắt ngoài.

❖ LƯU Ý

- Khi sử dụng cần lưu ý tránh cấp nguồn vượt thông số cho phép trực tiếp vào các chân **Vin, 5V(38pin)** vì có thể sẽ làm hỏng ESP

2.1.3. NodeMCU ESP32 Wifi BLE – CH430 (kit thu phát IoT)

Hệ thống tưới nước tự động sử dụng NodeMCU ESP32 chip nạp CH340 - Cổng USB typeC



Hình 2.1.3.1. Kit NodeMCU ESP32 Wifi BLE

Thông số kỹ thuật:

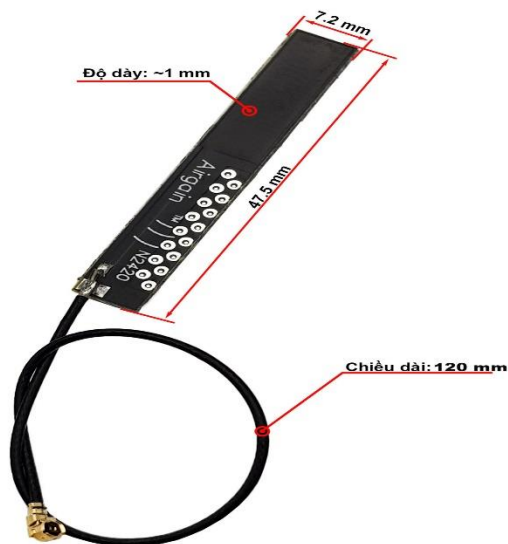
- Số chân (Pin): 30
- Kích thước bảng mạch: 28.33 x 51.45 mm (độ rộng x độ dài)
- Nguồn vào: 5VDC từ cổng micro USB.



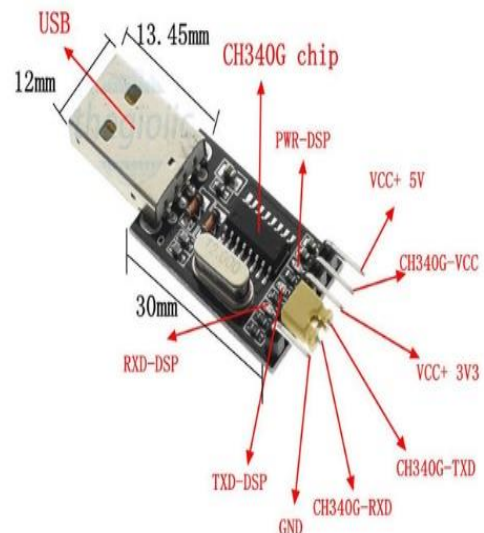
Hình 2.1.3.2. Module trung tâm: Mạch thu phát Wifi BLE SoC ESP32 ESP-WROOM-32E



Hình 2.1.3.3. Tích hợp Nút BOOT và ENABLE, đèn LED màu đỏ (nguồn) và xanh lam (GPIO2)



Hình 2.1.3.4. Kiểu Anten: PCB



Hình 2.1.3.5. Tích hợp mạch nạp và giao tiếp UART CH340.

2.1.4. Cài đặt Adruino IDE và ngôn ngữ sử dụng

2.1.4.1. Môi trường lập trình (IDE)

ArduinoIDE

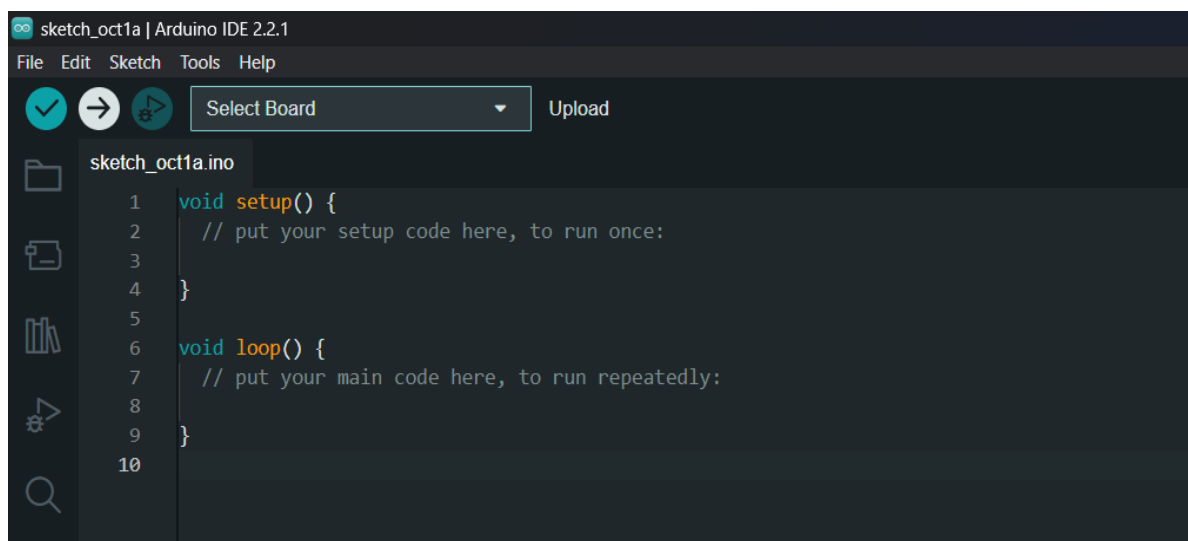
2.1.4.2. Thư viện lập trình (SDK)

Arduino: Được viết dựa trên ESP- IDF (quen thuộc)

ESP-IDF: gói thư viện do hãng phát hành








2.1.4.3. Adruino IDE

Arduino IDE được viết tắt (Arduino Integrated Development Environment) là một trình soạn thảo văn bản, dùng để viết code để nạp vào bo mạch Arduino.



Hình 2.1.4.3.1. Một góc giao diện Adruino IDE

Chú thích các hình ảnh trong giao diện:

Hình ảnh	Tên	Ý nghĩa
	Verify	Kiểm tra lỗi và biên dịch code
	Upload	Dịch và upload code vào bo mạch đã được cài đặt sẵn
	Debug	Tìm kiếm ra lỗi hay nguyên nhân gây ra lỗi (bug ở đâu)
	Sketchbook	Hiển thị các sketch hiện tại mà bạn đã sử dụng cho project của mình
	Board Manager	Tìm các board cần dùng
	Library Manager	Tìm các thư viện cần dùng
	Search	Tìm kiếm

❖ Chi tiết giao diện Arduino IDE

• Sketch

Một chương trình hiển thị trên cửa sổ giao diện được gọi là sketch.

Sketch được tạo từ hai hàm cơ bản là `setup()` và `loop()`:

- **setup()**: Hàm được gọi khi một sketch khởi động, được sử dụng để khởi tạo biến, đặt các chế độ chân (nhận hay xuất tín hiệu), khởi động một thư viện ... Hàm `setup()` chỉ chạy một lần, sau khi cấp nguồn hoặc reset mạch.
- **loop()**: Sau khi khởi tạo hàm `setup()`, hàm `loop()` sẽ được khởi tạo và thiết lập các giá trị ban đầu.
Hàm `loop` tạo các vòng lặp liên tục, có cho phép sự thay đổi và đáp ứng.
Chức năng tương tự như vòng lặp `while()` trong C, hàm `loop()` sẽ điều khiển toàn bộ mạch.

• Cổng Com

Cổng nối tiếp (Serial port - Cổng COM, communication) là một cổng thông dụng trong các máy tính truyền thống dùng kết nối các thiết bị ngoại vi với máy tính như: bàn phím, chuột điều khiển, modem, máy quét...

Ngày nay, do tốc độ truyền dữ liệu chậm hơn so với các cổng mới ra đời nên các cổng nối tiếp đang dần bị loại bỏ trong các chuẩn máy tính hiện nay, chúng được thay thế bằng các cổng có tốc độ nhanh hơn như: USB, FireWire

• Serial Monitor

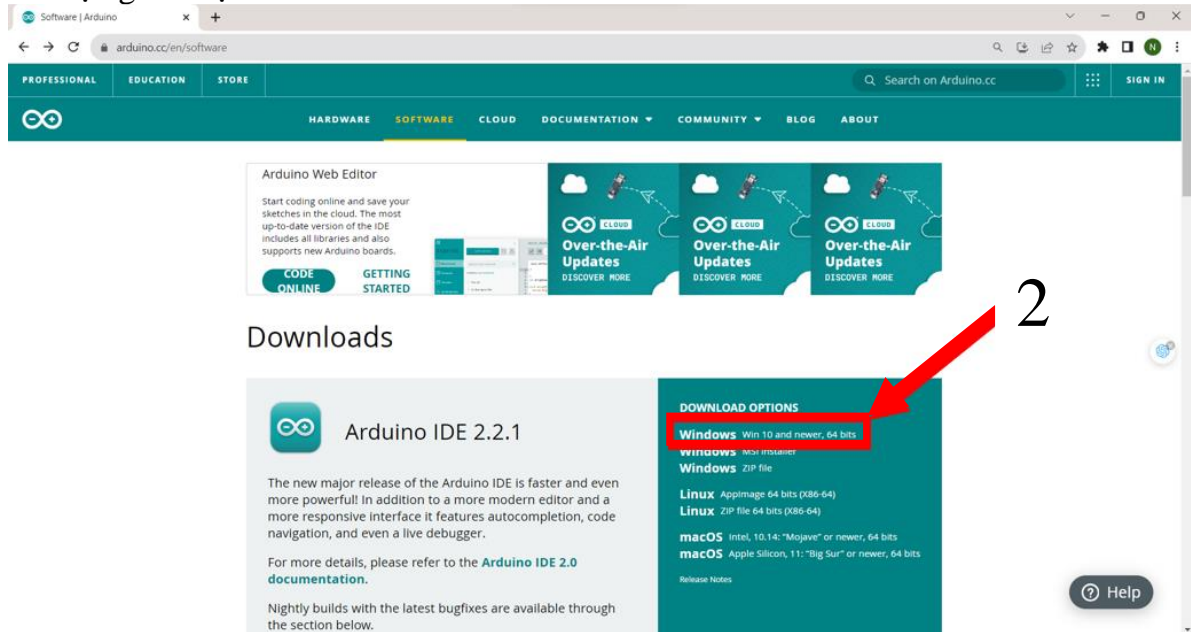
Serial Monitor là thành phần của **Arduino IDE**, giúp bo mạch và máy tính có thể gửi và nhận dữ liệu với nhau qua giao tiếp USB.

2.1.4.4. Cài đặt Arduino IDE

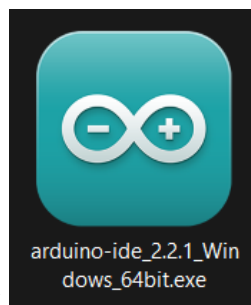
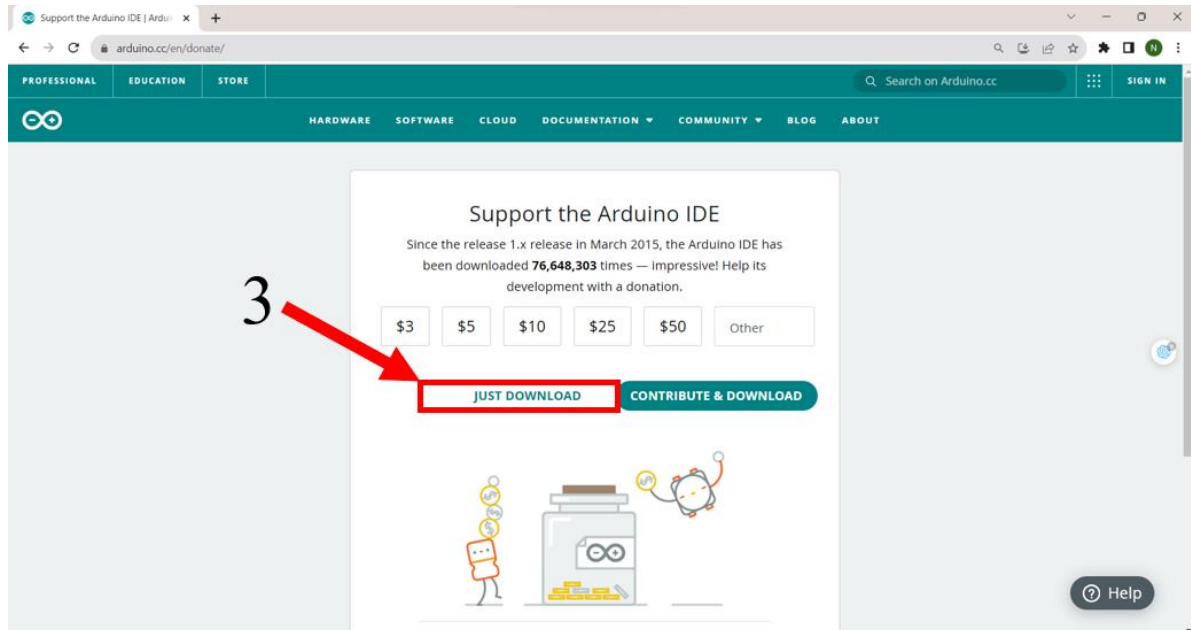
Link website tải Arduino IDE: [Arduino IDE Download](#)

1. Click vào link phía trên

2. Hiện giao diện website. Click vào chỉ dẫn số 2.

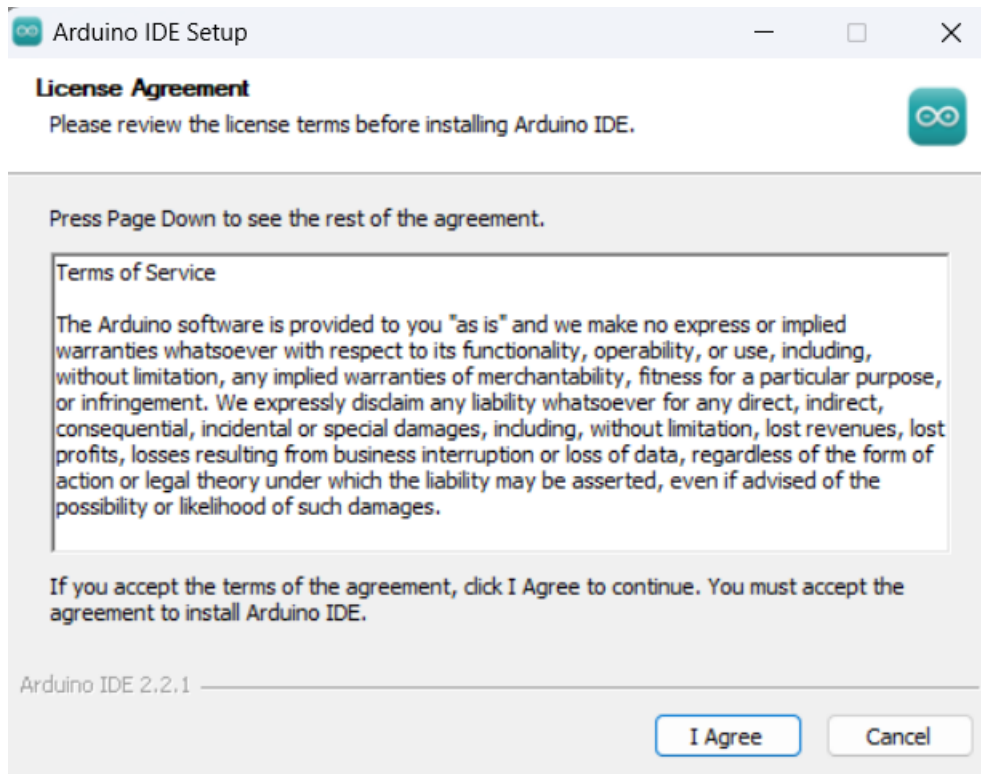


3. Click vào chỉ dẫn số 3

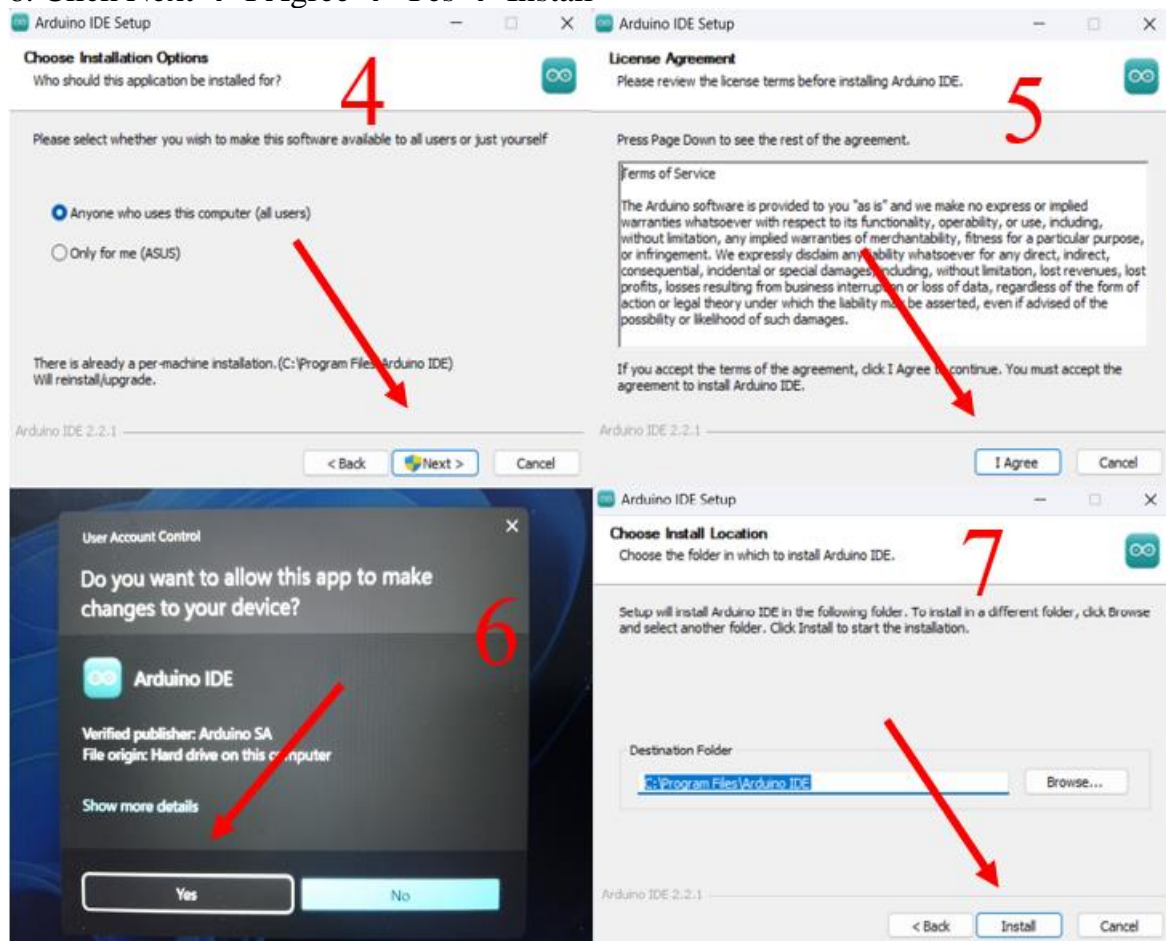


4. File được tải về máy. Click vào file.

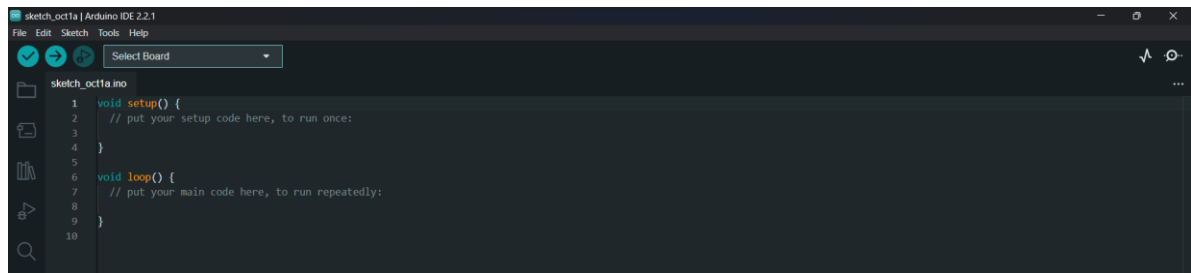
5. Click I Agree



6. Click Next → I Agree → Yes → Install



Giao diện Aduino IDE sau khi cài đặt:

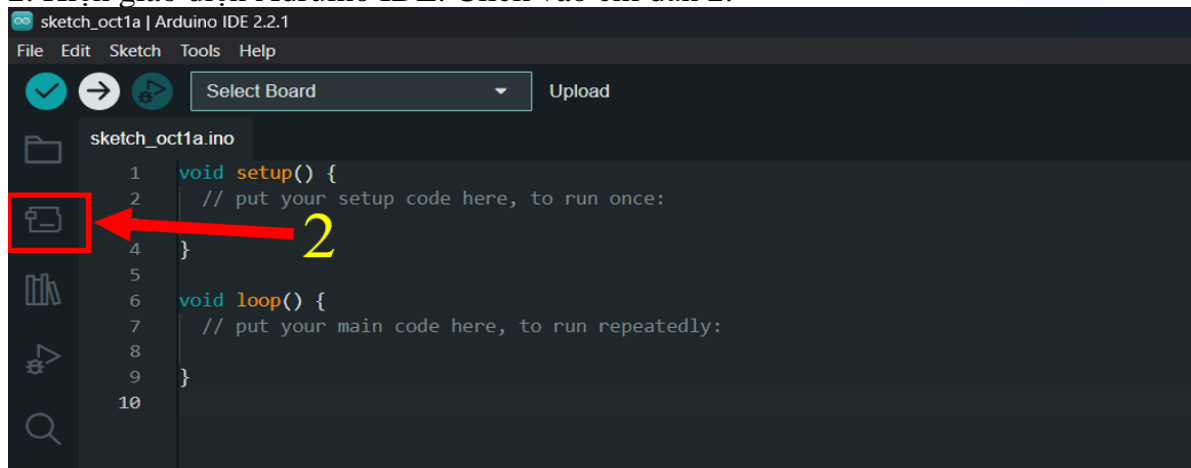


Hình 2.1.4.4.1. Giao diện Arduino IDE

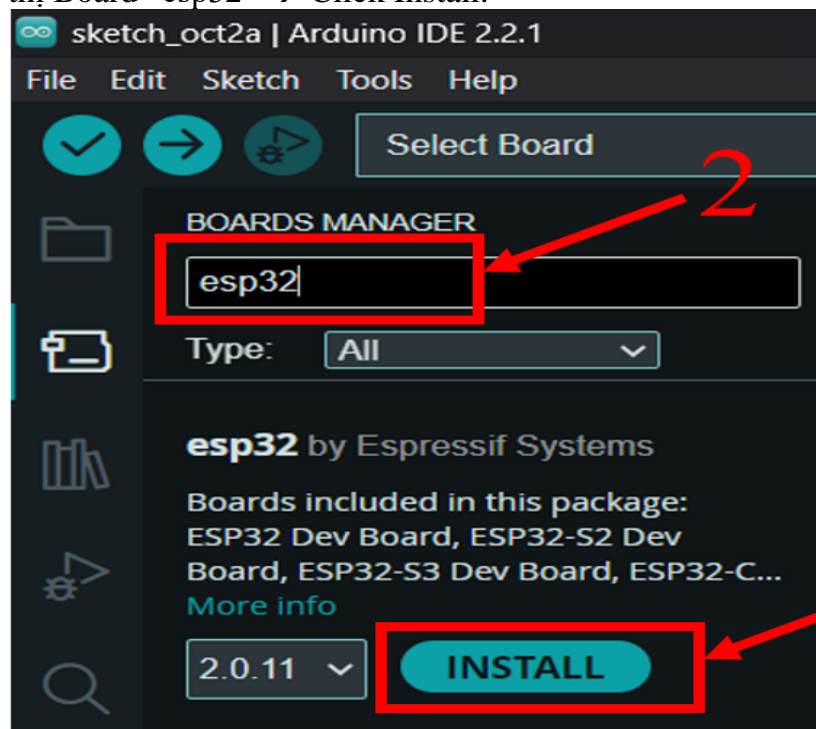
2.1.5. Cài đặt ESP32 và các thư viện cho từng linh kiện

1. Mở Arduino IDE

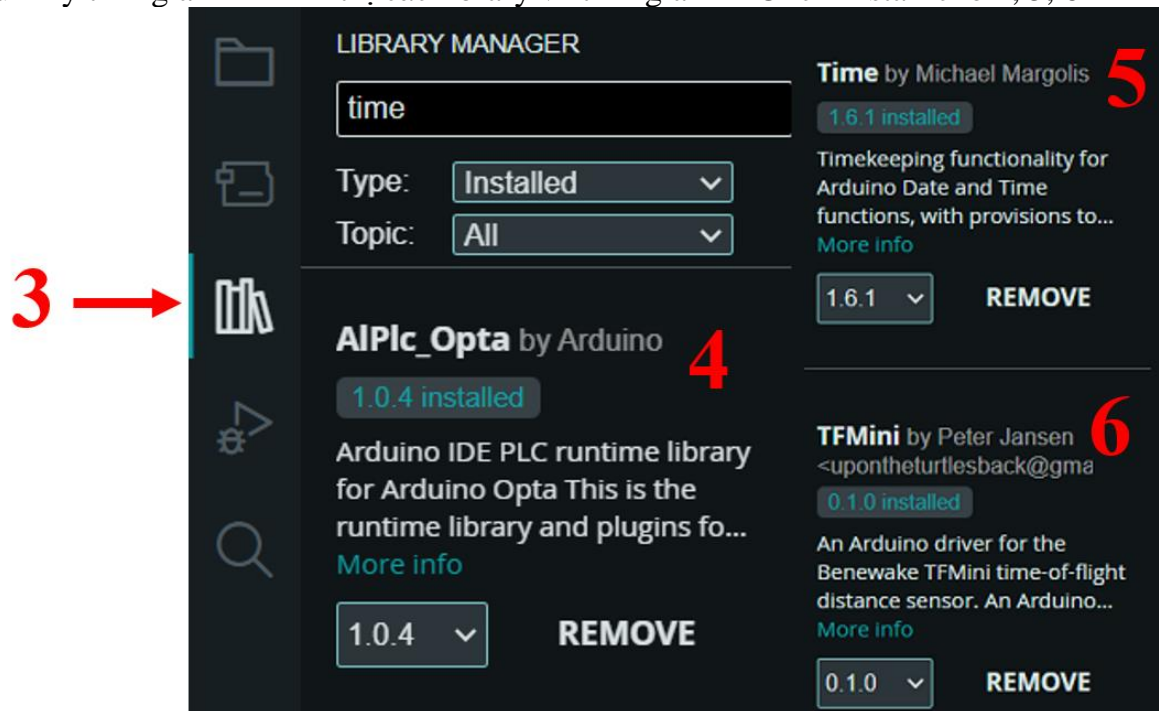
2. Hiện giao diện Arduino IDE. Click vào chỉ dẫn 2.



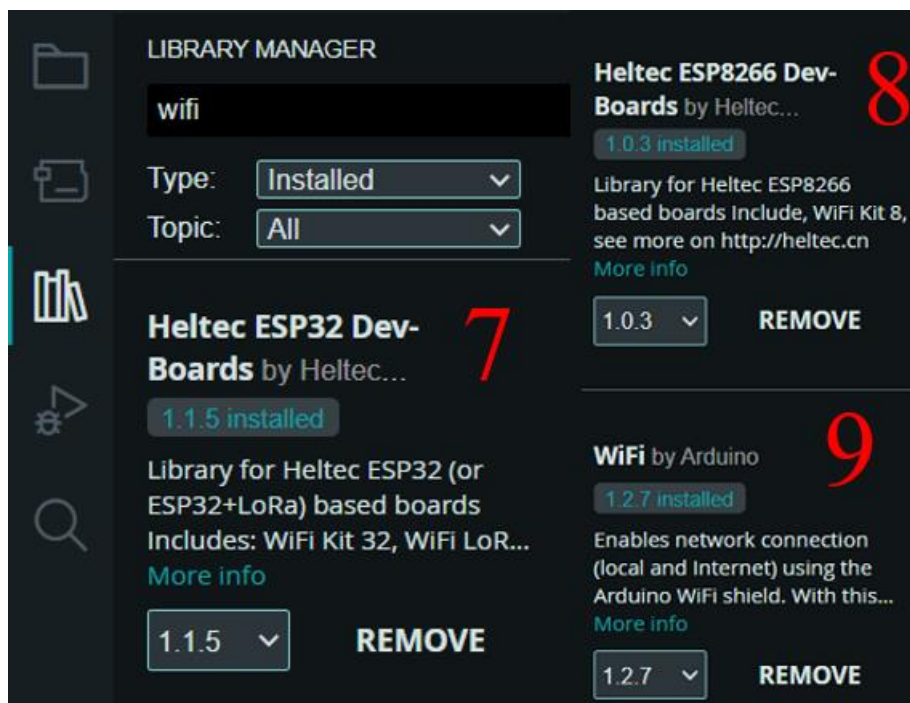
3. Hiện thị Board Manager → Nhập “esp32” tìm thư viện cho mạch ESP32 → Hiện thị Board “esp32” → Click Install.



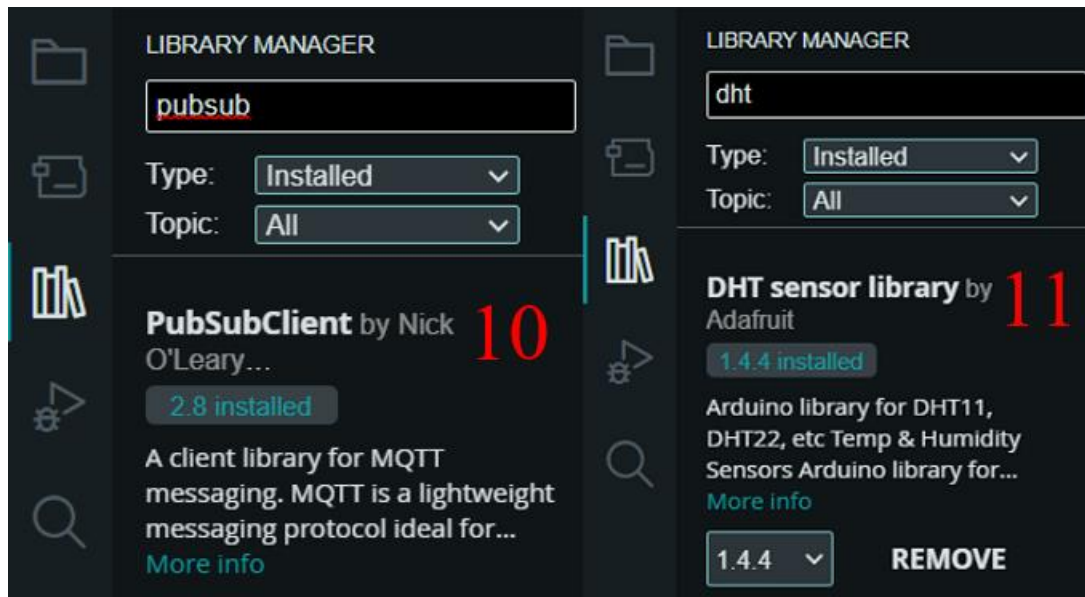
4. Click vào chỉ dẫn 3 → Hiển thị Library Manager → Nhập “time” tìm thư viện để quản lý thời gian → Hiển thị các library về thời gian → Click Install cho 4, 5, 6



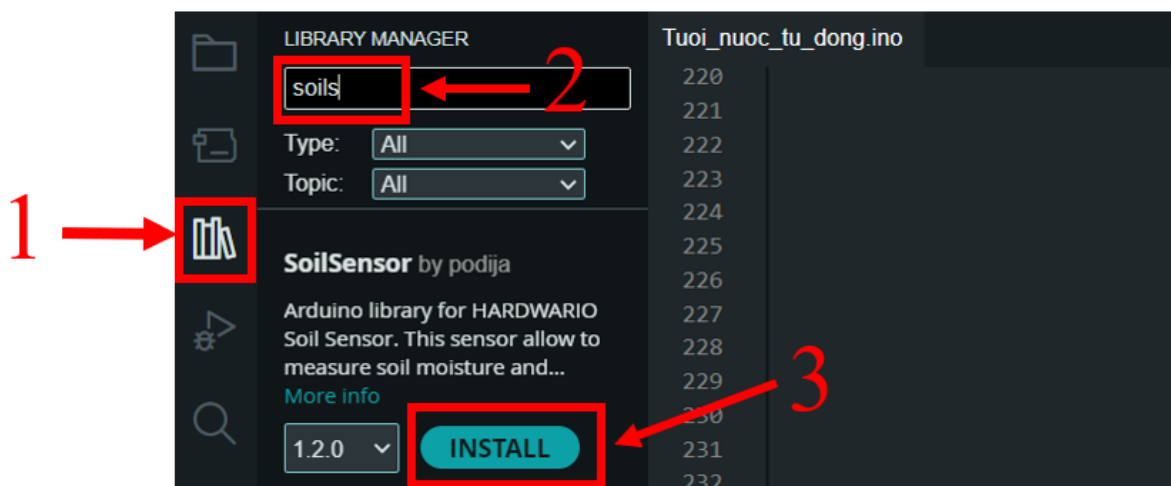
5. Thực hiện tương tự đối với “wifi” → Hiển thị các library về wifi → Click Install cho 7, 8, 9



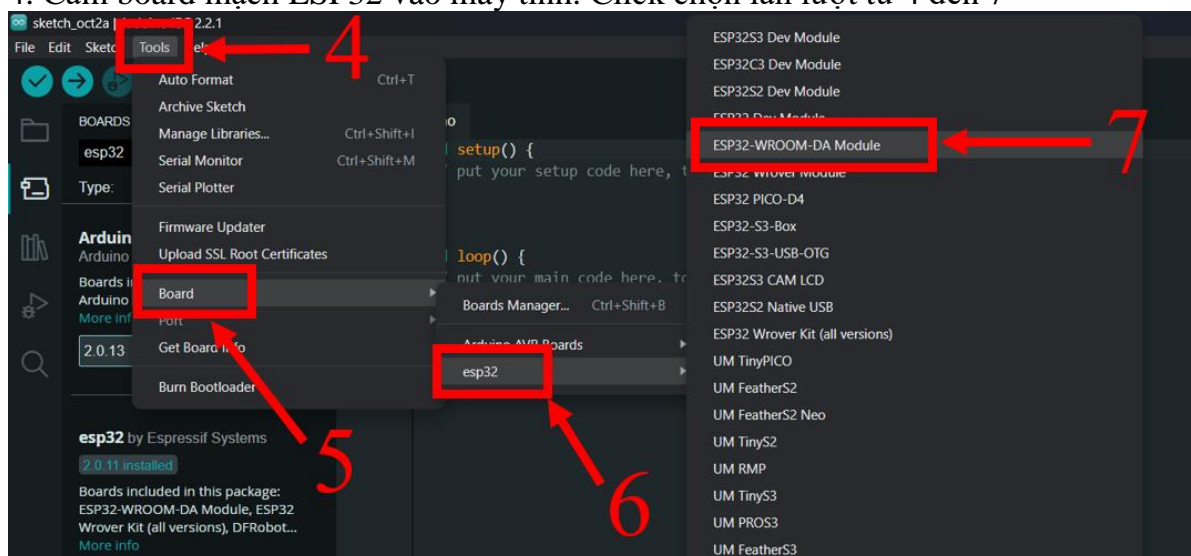
6. Cuối cùng nhập “pubsub” và “dht” → Hiển thị các library về PubSubClient và DHT → Click Install cho 10, 11



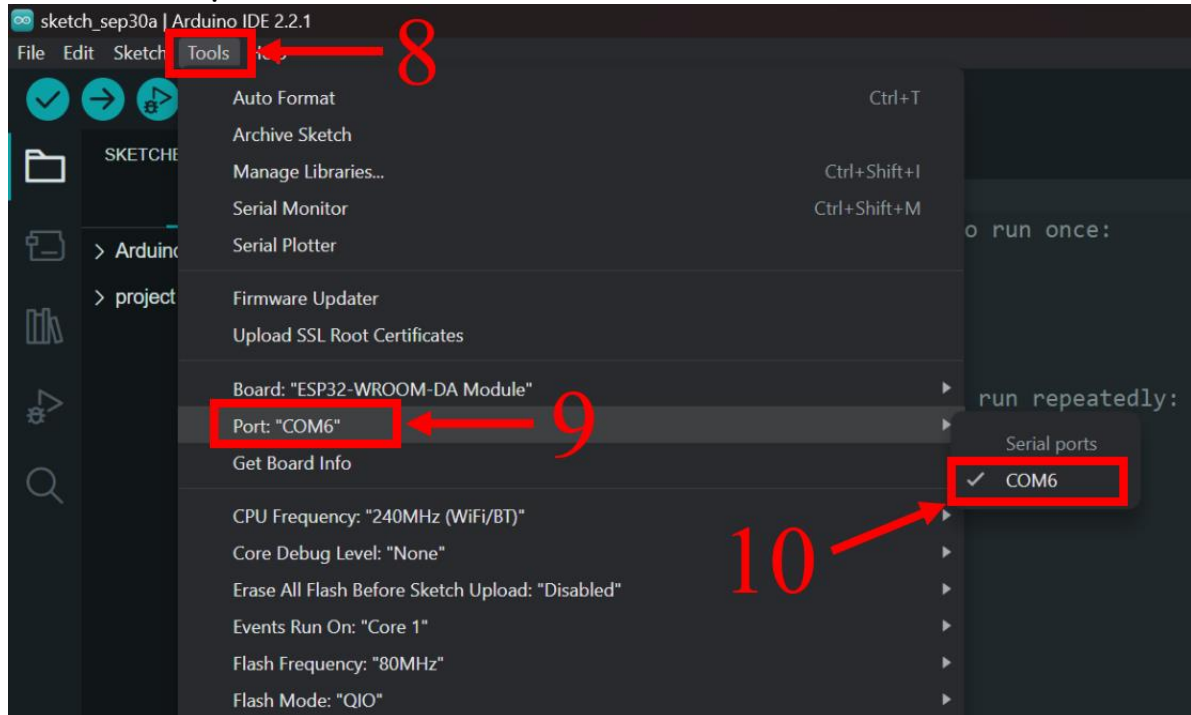
7. Cuối cùng, nhập “soils” tìm thư viện để đọc dữ liệu từ DHT11 và chọn cài đặt



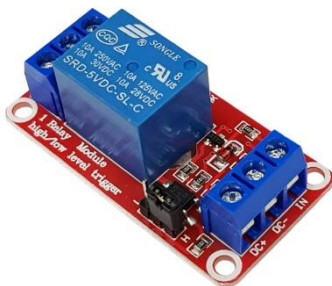
4. Cắm board mạch ESP32 vào máy tính. Click chọn lần lượt từ 4 đến 7



5. Click lần lượt từ 8 đến 10.



2.2. Relay Kích Mức Thấp 5V



Hình 2.2.1. Module 1 Relay Kích Mức Thấp 5V

- Relay 5V là một thiết bị điện tử sử dụng để điều khiển các thiết bị điện (như đèn, quạt, máy bơm, vv.) bằng cách mở hoặc đóng mạch điện.
- Điều khiển đóng ngắt điện DC hoặc AC, có thể điều khiển tải AC 220V-10A

Đầu vào:

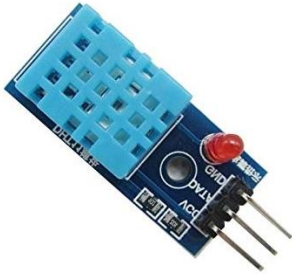
- Điện áp nuôi: 5VDC
- Tín hiệu vào điều khiển: 0V
 - + Tín hiệu là 0: Relay đóng
 - + Tín hiệu là 1: Relay mở

Đầu ra:

- + Tiếp điểm relay 220V-10A (Lưu ý tiếp điểm, không phải điện áp ra)
- + NC: Thường đóng (khi kích tiếp điểm mở ra)
- + NO: Thường mở (khi kích tiếp điểm đóng lại)
- + COM: Chân chung

Ký hiệu nguồn:

- + VCC, GND là nguồn nuôi Relay
- + In là chân tín hiệu điều khiển

2.3. DHT11 - Cảm Biến Nhiệt Độ và Độ Ẩm cho Arduino**Hình 2.3.1. DHT11 - Cảm Biến Nhiệt Độ và Độ Ẩm**

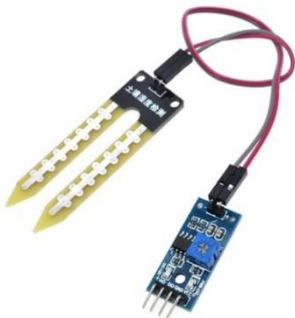
DHT11 được tích hợp trong một mạch duy nhất gồm 3 ngõ ra: GND, VCC, và dây tín hiệu (SIG). Thuận tiện cho việc xuất dữ liệu thông qua giao tiếp 1 - wire (giao tiếp digital 1 - wire truyền dữ liệu duy nhất)

Thông số kỹ thuật:

- Điện áp hoạt động: 3-5.5V DC
- Ngưỡng độ ẩm: 20 – 90%
- Sai số độ ẩm: $\pm 5\%RH$
- Ngưỡng nhiệt độ: 0 – 55°C
- Sai số nhiệt độ: $\pm 2^{\circ}C$
- Tần số lấy mẫu tối đa: 1Hz
- Khoảng cách truyền tối đa: 20m

Kết nối cảm biến nhiệt độ và độ ẩm DHT11 với Node WiFi 32:

DHT11	Node WiFi32
GND	GND
VCC	VCC
Dây tín hiệu (SIG)	Bất kỳ chân digital GPIO nào

2.4. Cảm biến đo độ ẩm đất**Hình 2.4.1. Cảm biến đo độ ẩm đất**

Độ nhạy của cảm biến độ ẩm đất có thể tùy chỉnh được (bằng cách điều chỉnh chiết áp màu xanh trên board mạch).

Phần đầu DO được cắm vào đất để phát hiện độ ẩm của đất, khi độ ẩm của đất đạt ngưỡng thiết lập, đầu ra DO sẽ chuyển trạng thái từ mức thấp (0V) lên mức cao khi đất thiếu nước (5V).

Thông số kỹ thuật:

Kích thước: 3 x 1.6cm.

Điện áp hoạt động: 3.3V ~ 5V

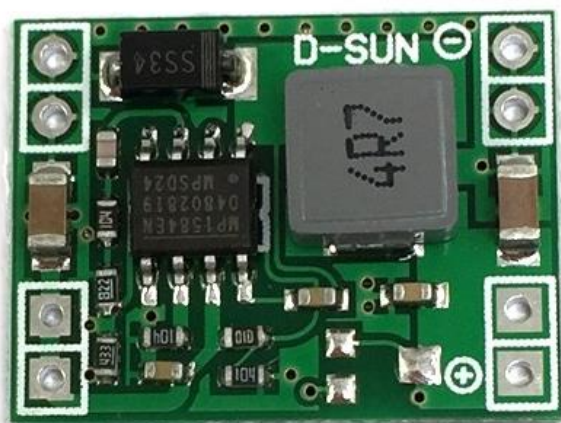
Tín hiệu đầu ra:

- Analog: theo điện áp cấp nguồn tương ứng.
- Digital: High hoặc Low, có thể điều chỉnh độ ẩm mong muốn bằng biến trở thông qua mạch so sánh LM393 tích hợp.

Sơ đồ chân cảm biến độ ẩm đất:

Tên	Ý nghĩa
VCC	3.3V ~ 5V
GND	GND của nguồn ngoài
DO	Đầu ra tín hiệu số (mức cao hoặc mức thấp)
AO	Đầu ra tín hiệu tương tự (Analog)

2.5. Mạch hạ áp ổn áp 5V- 3A



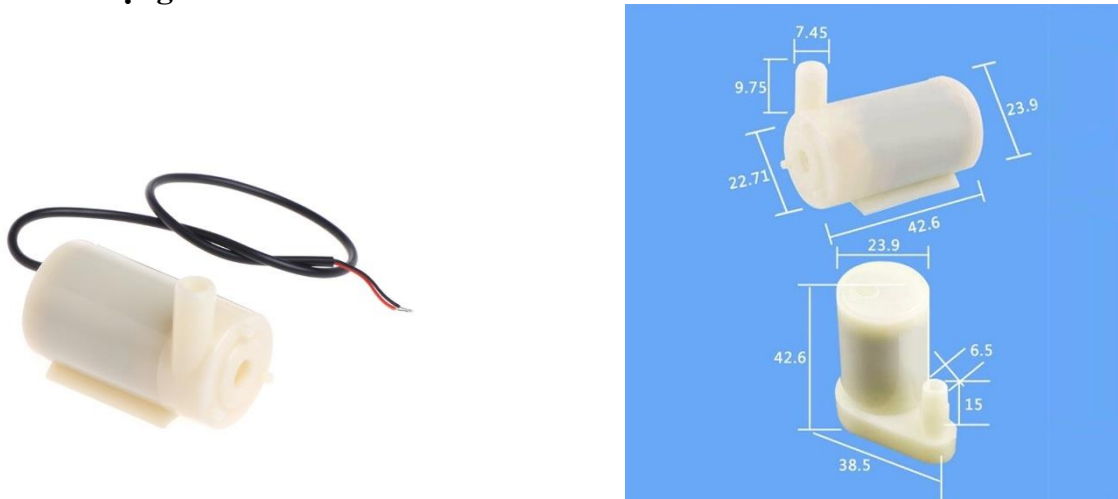
Hình 2.5.1. Mạch hạ áp ổn áp 5V- 3A

Mạch hạ áp ổn áp 5V - 3A có chức năng hạ điện áp DC và cố định điện áp ở mức 5VDC với dòng tải ngõ ra tối đa 3A.

Thông số kỹ thuật:

- Điện áp ngõ vào: 7 - 28VDC
- Điện áp ngõ ra: 5VDC
- Dòng tải tối đa: 3A
- Dòng tải thường: 1.5A
- Hiệu suất chuyển đổi: 96%
- Gợn ngõ ra: <30mV
- Nhiệt độ hoạt động: -45°C ~ 85°C
- Kích thước: 22 x 17 x 4mm

2.6. Động Cơ Bơm Chìm Mini V2 3V~6V



Hình 2.6.1. Động Cơ Bơm Chìm Mini V2 3V~6V

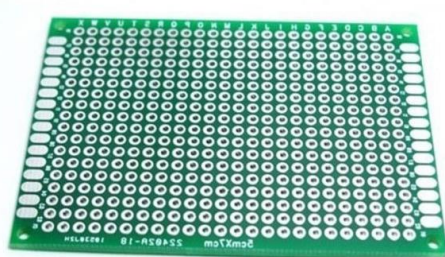
Toàn bộ động cơ có thể bỏ chìm trong nước hoàn toàn.

Sử dụng điện áp 5VDC thông dụng cho tốc độ dòng nước từ 1,2 - 1,6L/phút.

Thông số kỹ thuật:

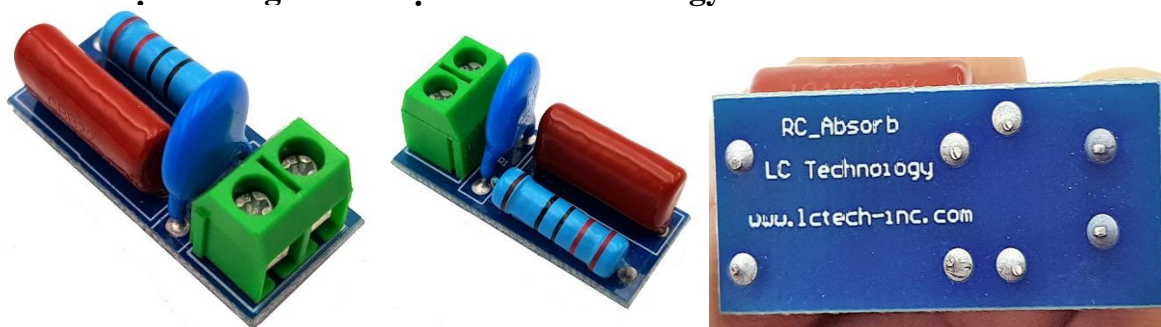
- Điện áp hoạt động: DC 3V - 6V
- Dòng tải: $\geq 100\text{-}200\text{mA}$

2.7. Test board mạch hàn



Hình 2.7.1. Test board mạch hàn

2.8. Mạch chống nhiễu điện từ LC Technology



Hình 2.8.1. Mạch chống nhiễu điện từ, bảo vệ tiếp điểm LC Technology

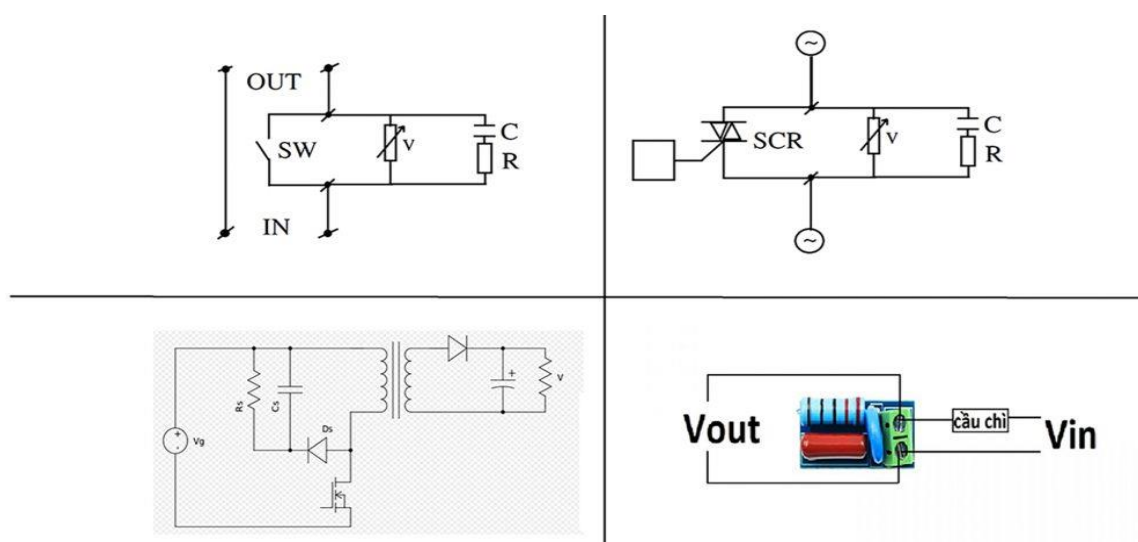
Mạch chống nhiễu điện từ, bảo vệ tiếp điểm LC Technology là sản phẩm của LC Technology. Mạch chống nhiễu điện từ, bảo vệ tiếp điểm này có thể gọi tắt là Mạch

chống nhiễu điện từ, nó là mạch được thiết kế cho việc chống nhiễu của dòng điện. thường được mắc song song với tiếp điểm Relay hoặc Công tắc giúp chống nhiễu cho Vi điều khiển khi Relay hoặc công tắc đóng ngắt tạo ra tia lửa điện làm hệ thống bị treo hoặc Reset, mạch còn có tác dụng bảo vệ Relay khi đóng ngắt các tải cảm.

Mạch chống nhiễu điện từ, bảo vệ tiếp điểm này được thiết kế có một varistor để ngăn điện áp dao động quá cao và dòng điện quá lớn, từ đó bảo vệ cho các tiếp điểm role không dính vào nhau. Mạch sử dụng domino giúp thuận tiện hơn cho việc đi dây.

❖ Thông số kỹ thuật của Mạch chống nhiễu điện từ, bảo vệ tiếp điểm

- Kích thước: 31×13.7mm
- Trọng lượng: 5g
- Phù hợp với tải cảm ứng AC hoặc DC 5-400V (tải dưới 1000W) để bảo vệ các tiếp điểm role hoặc thyristor.
- Mạch hấp thụ RC hấp thụ lực điện động cảm ứng của tải cảm ứng.
- Một số kiểu đấu nối của Mạch chống nhiễu điện từ, bảo vệ tiếp điểm



Hình 2.8.2. Sơ đồ đấu nối của Mạch chống nhiễu điện từ, bảo vệ tiếp điểm LC Technology

2.9. Khái quát các ngôn ngữ sử dụng lập trình Website điều khiển hệ thống tưới từ xa

2.9.1. HTML và HTML5

HTML (HyperText Markup Language - "Ngôn ngữ Đánh dấu Siêu văn bản") tạo ra các tài liệu hiển thị được trên trình duyệt, sử dụng các THẺ (tag) để xác định cấu trúc và nội dung của trang.

- **Trang web tĩnh:** Các trang HTML có thể là các file (.html, .htm) được lưu trữ trên hệ thống file của máy chủ web (webserver), từ đó trình duyệt truy cập đọc được và hiển thị.
- **Trang web động:** Các trang HTML mà trình duyệt lấy về từ webserver được phát sinh bởi một ứng dụng chạy trên server (ứng dụng phát triển bằng các ngôn ngữ lập trình như PHP, C# ...).

- **Siêu văn bản (HyperText):** sự liên kết giữa các trang, một trang HTML có liên kết một trang khác trên cùng một Website hay giữa các website.

HTML5 là phiên bản mới (chuẩn mới) của HTML đã cải tiến form và tích hợp API. Chứa ba thành phần: HTML, CSS, JavaScript.

Một số thuật ngữ thường gặp:

- **Thẻ (Tag)** tạo ra các phần tử nhằm hình thành cấu trúc tài liệu HTML.
 - mở thẻ `<tên_thẻ>`
 - nội dung bên trong phần tử
 - đóng thẻ `</tên_thẻ>`

VD 

- **Element:** chỉ định xác định nội dung, cấu trúc của các đối tượng trong một Website. Tên Element được bao quanh bằng dấu ngoặc `< >`

VD: ``

- **Attribute:** thuộc tính cung cấp thông tin bổ sung về một Element.

VD: Element `<a>` gồm một Attribute href: ` `

Bố cục HTML5

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title></title>
</head>
<body>
</body>
</html>
```

Hình 2.9.1.1. Bố cục HTML5

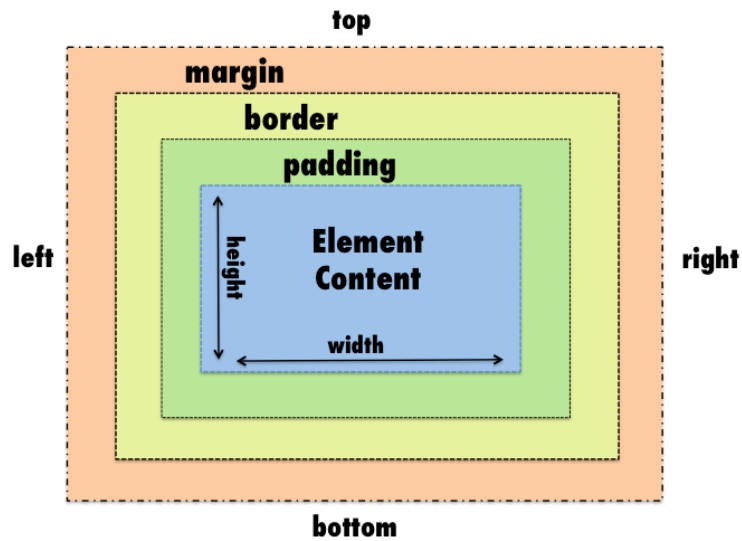
Trong đó:

- `<!DOCTYPE html>`: khai báo kiểu dữ liệu hiển thị
- `<html>` và `</html>`: cặp thẻ bắt buộc, chứa tất cả nội dung của trang HTML
- `<head>` và `</head>`: khai báo các thông tin meta của trang web
- `<title>` và `</title>`: nằm bên trong thẻ `<head>`, dùng để khai báo tiêu đề của trang
- `<body>` và `</body>`: chứa tất cả các nội dung sẽ hiển thị trên trang
- `<meta>`: các thông tin (gọi là metadata - siêu dữ liệu) mô tả cho trang

2.9.2. CSS

CSS (Cascading Style Sheets) là một ngôn ngữ dùng để trình bày hình thức thể hiện của các phần tử HTML. Như định dạng các phần tử văn bản (cỡ chữ, font chữ, màu sắc ...), bố cục, dàn trang ...

Bố cục của CSS



Hình 2.9.2.1. Bố cục của CSS

Nhúng CSS vào HTML

- **Inline CSS:** đặt mã CSS vào thẳng thuộc tính style của phần tử

```
1 <p style="color:white; background-color:red;">
2   Đây là ví dụ về CSS dạng inline
3 </p>
4
```

Kết quả:

Đây là ví dụ về CSS dạng inline

Hình 2.9.2.2. Minh họa CSS dạng Inline

- **Internal CSS:** mã CSS trong chính văn bản HTML, nằm trong khối thẻ <style>

```
1 <html>
2   <head>
3     <style>
4       /*style mở ra một khu vực để viết mã CSS*/
5       p {
6         color:white; background-color:red;
7       }
8     </style>
9   </head>
10  <body>
11    <p>Đoạn văn 1. </p>
12    <p>Đoạn văn 2. </p>
13  </body>
14 </html>
```

Kết quả:

Đoạn văn 1.

Đoạn văn 2.

Hình 2.9.2.3. Minh họa CSS dạng Internal

- Kiểu inline và Internal chèn css vào html trực tiếp trong văn bản HTML.

- **External CSS:** mã CSS ở một file riêng biệt (*.css) sau đó dùng thẻ <link> đặt ở phần <head> để nạp vào HTML (liên kết css với html)

```

1 <html>
2   <head>
3     /*Thuộc tính href trỏ đến địa chỉ URL của file CSS*/
4     <link rel="stylesheet" href="demo.css">
5   </head>
6   <body>
7     <p>Đoạn văn 1. </p>
8     <p>Đoạn văn 2. </p>
9   </body>
10 </html>

```

Trong file `demo.css` bạn viết nội dung CSS (không có thẻ style), ví dụ:

```

p {
  color:white;
  background-color:red;
}

```

Hình 2.9.2.4. Minh họa CSS dạng External

2.9.3. JavaScript

JavaScript là một ngôn ngữ lập trình thông dịch với khả năng hướng đến đối tượng, điều khiển tương tác với trang, thường sẽ được nhúng trực tiếp vào một trang web hoặc được tham chiếu qua file .js riêng.

JavaScript là ngôn ngữ từ phía client nên script sẽ được tải về máy client khi truy cập và được xử lý tại đó. Một số Framework thông dụng:

- Reactjs: Thư viện dùng cho ứng dụng mobile.
- Node.js: Dùng để xây dựng và phát triển ứng dụng realtime từ phía máy chủ.
- Angular: Dùng để xây dựng ứng dụng Single Page....

Tất cả các đoạn mã JavaScript đều được đặt trong cặp thẻ đóng mở <script></script>

```

<script >
|   <!--Chỗ này viết code JavaScript-->
</script>

```

Có 2 cách đặt thẻ script thường được sử dụng:

- **Internal:** viết trong file html hiện tại.

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7     <script>
8       <!--Chỗ này viết code JavaScript-->
9     </script>
10  </head>
11  <body>
12
13  </body>
14 </html>

```

Hình 2.9.3.1. Minh họa CSS dạng Internal

- **External:** viết ra một file js khác rồi import vào bằng thẻ script.

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7   <script src="demo.js"> </script>
8 </head>
9 <body>
10
11 </body>
12 </html>

```

Hình 2.9.3.2. Minh họa CSS dạng External

2.9.4. Bootstrap

Bootstrap là một framework front-end mã nguồn mở phổ biến được sử dụng để phát triển trang web và ứng dụng web. Nó cung cấp một bộ công cụ và tài liệu để giúp thiết kế và xây dựng giao diện người dùng (UI) trên trang web một cách dễ dàng và hiệu quả. Bootstrap phiên bản mới nhất hiện nay là Bootstrap 5.0 với v5.3.2

Bootstrap sử dụng jQuery trong một số phiên bản cũ (như Bootstrap 3), nhưng Bootstrap 5.3 đã loại bỏ sự phụ thuộc vào jQuery và thay thế nó bằng JavaScript thuần, giúp giảm kích thước tệp và cải thiện hiệu suất của trang web. Vì vậy, trong Bootstrap 5.3, không cần phải bao gồm thư viện jQuery nếu không có yêu cầu cụ thể sử dụng nó.

Các thành phần và tính năng cụ thể của Bootstrap:

- **Hệ thống ô lưới (Grid System):** Hệ thống ô lưới của Bootstrap cho phép bạn chia trang thành các hàng (rows) và cột (columns). Bằng cách sử dụng các lớp CSS như `.container`, `.row`, và `.col`, bạn có thể xác định cách cột sẽ tỷ lệ trên nhiều kích thước màn hình. Ví dụ, bạn có thể tạo một cột chiếm 6/12 phần trên màn hình máy tính và 12/12 phần trên điện thoại di động. Điều này giúp tạo ra các bố cục đáp ứng một cách dễ dàng.
- **Thành phần giao diện người dùng (UI Components):** Bootstrap đi kèm với nhiều thành phần giao diện đã được thiết kế sẵn, bao gồm nút, biểu mẫu, thanh điều hướng, hộp thoại, bảng, hình ảnh, tiện ích và nhiều thành phần khác. Các thành phần này giúp bạn tạo ra các phần giao diện chuyên nghiệp mà không cần phải viết mã từ đầu.
- **CSS Tùy chỉnh:** Nếu bạn muốn điều chỉnh giao diện Bootstrap để phù hợp với thiết kế cụ thể của dự án của bạn, bạn có thể tùy chỉnh CSS bằng cách sử dụng biến Sass hoặc chỉnh sửa các tệp CSS trực tiếp. Điều này cho phép bạn thay đổi màu sắc, phông chữ, khoảng cách và các thuộc tính khác theo ý muốn.
- **Plugin JavaScript:** Bootstrap cung cấp nhiều plugin JavaScript như carousel (điều hướng ảnh trượt), modal (hộp thoại popup), collapse (xò xuống/đổ ra),

tooltip (chú thích), popover (thanh thông báo), v.v. Các plugin này giúp bạn thêm tính năng tương tác vào trang web của mình một cách dễ dàng.

- **Hỗ trợ đa trình duyệt:** Bootstrap đã được tối ưu hóa để hoạt động trên nhiều trình duyệt khác nhau, đảm bảo rằng trang web của bạn sẽ hiển thị và hoạt động một cách nhất quán trên mọi nền tảng.
- **Đáp ứng (Responsive Design):** Bootstrap được xây dựng với ý niệm đáp ứng, nghĩa là các trang web sử dụng Bootstrap tự động thích nghi với kích thước màn hình khác nhau, từ máy tính để bàn đến điện thoại di động. Điều này giúp cung cấp trải nghiệm người dùng tốt trên mọi thiết bị.
- **Cộng đồng và Tài liệu phong phú:** Bootstrap có một cộng đồng lớn của nhà phát triển và thiết kế web. Có rất nhiều tài liệu, ví dụ và hỗ trợ trực tuyến giúp bạn học cách sử dụng Bootstrap và giải quyết các vấn đề cụ thể.

❖ Các biến Sass và ghi đè các tệp CSS

➤ Sử dụng biến Sass:

Bootstrap sử dụng Sass (hoặc SCSS) cho quá trình phát triển CSS. Bạn có thể tùy chỉnh giao diện của trang bằng cách sử dụng biến Sass có sẵn. Ví dụ, bạn có thể sử dụng biến để thay đổi màu chữ, màu nền, biên và nhiều thuộc tính khác. Để sử dụng biến Sass, bạn cần tạo một tệp Sass riêng và nhập Bootstrap Sass vào đó. Sau đó, bạn có thể ghi đè các biến Sass theo ý muốn.

```
SCSS

// Tạo biến Sass tùy chỉnh
$primary-color: #ff5733;

// Nhập tệp Bootstrap Sass
@import "bootstrap";

// Ghi đè biến Bootstrap
$body-bg: #f5f5f5;

// Thêm các tùy chỉnh CSS bổ sung
.my-custom-button {
  background-color: $primary-color;
  color: white;
  border: none;
}
```

➤ Tùy chỉnh CSS:

Ngoài việc sử dụng biến Sass, bạn cũng có thể tùy chỉnh CSS trực tiếp. Tạo một tệp CSS tùy chỉnh và liên kết nó với trang HTML của bạn sau khi liên kết tệp CSS của

Bootstrap. Bất kỳ quy tắc CSS nào bạn đặt trong tệp CSS tùy chỉnh sẽ ghi đè các quy tắc tương tự trong Bootstrap.

```
html

<!-- Liên kết tệp CSS Bootstrap -->
<link rel="stylesheet" href="bootstrap.css">

<!-- Liên kết tệp CSS tùy chỉnh -->
<link rel="stylesheet" href="custom.css">
```

Trong tệp custom.css, bạn có thể ghi đè các quy tắc CSS Bootstrap bằng cách định rõ các lựa chọn và thuộc tính tùy chỉnh.

❖ Hỗ trợ CSS Grid

Bootstrap 5 cho phép bạn sử dụng CSS Grid, một phần của CSS nâng cao, để xây dựng giao diện của trang web hoặc ứng dụng web của bạn, cho phép bạn kiểm soát các cột và hàng một cách linh hoạt hơn, giúp tạo ra các bố cục phức tạp và tùy chỉnh dễ dàng hơn.

Một số chi tiết về sự hỗ trợ của Bootstrap 5 cho CSS Grid:

- **Hệ thống ô lưới parallax (Fluid Grid System):** Bootstrap 5 đã cải tiến hệ thống ô lưới của mình để sử dụng CSS Grid. Thay vì dựa vào hệ thống ô lưới dựa trên float và các lớp CSS cụ thể, Bootstrap 5 sử dụng CSS Grid để tạo ra một hệ thống ô lưới linh hoạt hơn. Điều này cho phép bạn xây dựng các bố cục ô lưới một cách dễ dàng bằng cách sử dụng các thuộc tính CSS Grid như grid-template-columns, grid-template-rows, và grid-gap.
- **Custom Layouts:** CSS Grid cho phép bạn tạo các bố cục tùy chỉnh và phức tạp. Bạn có thể kiểm soát từng phần tử trong ô lưới, điều này có nghĩa là bạn có thể thiết kế các bố cục một cách chi tiết và linh hoạt hơn.
- **Responsive Design:** CSS Grid kết hợp tốt với tính năng đáp ứng (responsive design) của Bootstrap. Bạn có thể xác định cách ô lưới thay đổi dựa trên kích thước màn hình, giúp tạo ra trải nghiệm người dùng tốt trên cả máy tính và thiết bị di động.
- **Thay thế cho Flexbox:** Trong trường hợp bạn cần một hệ thống ô lưới mạnh mẽ hơn so với Flexbox, CSS Grid có thể là một lựa chọn tốt. CSS Grid cho phép bạn quản lý cả hàng ngang và hàng dọc, trong khi Flexbox chủ yếu tập trung vào hàng ngang.
- **Thư viện Cơ sở:** Bootstrap 5 không yêu cầu bạn sử dụng CSS Grid, nhưng nó cung cấp khả năng sử dụng nó trong trường hợp bạn muốn tận dụng sức mạnh của CSS Grid.

❖ Liên kết CDN của Bootstrap

Content Delivery Network (CDN) là một dịch vụ được sử dụng để phân phối các tài nguyên web, như các tệp CSS, JavaScript và hình ảnh cho các trang web hay ứng dụng web. Bootstrap là một framework CSS và JavaScript phổ biến được sử dụng để thiết kế giao diện web, nó có thể được tích hợp vào trang web của bạn thông qua các tệp CSS và JavaScript.

CDN của Bootstrap là một dịch vụ, máy chủ hoặc tập hợp các máy chủ được quản lý bởi Bootstrap hoặc các tổ chức liên quan để cung cấp các tệp Bootstrap cho trang web của bạn. Thay vì tải các tệp Bootstrap trực tiếp từ máy chủ của bạn, bạn có thể sử dụng CDN để cung cấp tệp này từ máy chủ CDN, giúp tăng tốc độ tải trang web của bạn và giảm tải cho máy chủ của bạn. Điều này cũng có lợi ích về bảo mật, vì các tệp Bootstrap được cập nhật và duyệt qua kiểm tra bảo mật trước khi được phân phối qua CDN.

Một ví dụ về CDN của Bootstrap là sử dụng Bootstrap CDN của MaxCDN hoặc sử dụng các phiên bản Bootstrap được cung cấp trên các CDN khác. Để tích hợp Bootstrap vào trang web của bạn thông qua CDN, bạn có thể thêm mã HTML như sau vào trang web của bạn:

```
<!-- Dùng Bootstrap CSS từ Bootstrap CDN -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">

<!-- Dùng Bootstrap JavaScript từ Bootstrap CDN -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
```

Lưu ý rằng phiên bản liên kết CDN có thể thay đổi theo thời gian, vì vậy bạn nên kiểm tra tài liệu Bootstrap hoặc trang web của CDN để có thông tin cụ thể về phiên bản và URL CDN hiện tại.

2.10. Giao thức MQTT Explorer

Link trang web tải MQTT Explorer: [MQTT Explorer Download](#)

MQTT (Message Queuing Telemetry Transport) là một giao thức truyền thông nhẹ, hiệu quả và đáng tin cậy được phát triển để truyền tải dữ liệu giữa các thiết bị có băng thông thấp hoặc kết nối không ổn định, đặc biệt khi tài nguyên mạng có giới hạn hoặc cần đảm bảo tính đáng tin cậy và hiệu quả. Giao thức này thường được sử dụng trong Internet of Things (IoT), máy chủ tin nhắn, ứng dụng cần gửi và nhận dữ liệu theo thời gian thực.

Một số điểm quan trọng về giao thức MQTT:

- **Mô hình Publish/Subscribe:** MQTT dựa trên mô hình Publish/Subscribe, trong đó các thiết bị hoặc ứng dụng đăng ký (subscribe) vào các chủ đề (topics) và nhận thông tin từ các chủ đề đó. Các thiết bị hoặc ứng dụng khác gửi thông tin (publish) đến các chủ đề, và MQTT đảm bảo rằng những người đã đăng ký nhận được thông tin này.

- **Nhẹ và hiệu quả:** MQTT được thiết kế để hoạt động trên các kết nối có băng thông thấp hoặc kết nối không ổn định, giúp giảm thiểu lưu lượng dữ liệu và tối ưu hóa việc truyền tải thông tin.
- **Bảo mật:** MQTT có hỗ trợ bảo mật, bao gồm chứng thực người dùng và sử dụng giao thức TLS/SSL để mã hóa thông tin trên mạng. Điều này đảm bảo rằng dữ liệu được truyền đi an toàn.
- **QoS (Quality of Service):** MQTT hỗ trợ ba mức độ QoS để đảm bảo việc truyền tải dữ liệu đáng tin cậy. Các mức độ QoS khác nhau cho phép bạn chọn mức độ đảm bảo dữ liệu sẽ đến đúng đích và có được xử lý như thế nào.
- **Làm việc trên nhiều nền tảng:** MQTT được hỗ trợ trên nhiều nền tảng, từ thiết bị nhúng đến máy chủ mạng, và có các thư viện và API cho nhiều ngôn ngữ lập trình khác nhau.
- **Retained Messages:** MQTT cho phép các thông điệp được gửi với tùy chọn "Retained", có nghĩa là thông điệp sẽ được giữ lại trên chủ đề và được gửi đến tất cả các thiết bị hoặc ứng dụng mới đăng ký vào chủ đề đó.

Đối với đề tài “Xây dựng mô hình tưới nước tự động”, chúng tôi thực hiện kết nối với giao thức MQTT Explorer.

MQTT Explorer là một ứng dụng trực quan thực hiện các hoạt động liên quan đến giao thức MQTT, chẳng hạn như theo dõi, gửi và nhận thông điệp trên các chủ đề MQTT.

MQTT Explorer giúp người phát triển và quản trị viên hệ thống IoT và mạng MQTT dễ dàng theo dõi và tương tác với các thiết bị và ứng dụng MQTT. Nó cung cấp một cách thuận tiện để thử nghiệm và kiểm tra giao tiếp MQTT.

Một số tính năng quan trọng của MQTT Explorer:

- **Kết nối MQTT Broker:** MQTT Explorer cho phép bạn thiết lập và quản lý kết nối tới MQTT broker. Bạn có thể chỉ định thông tin về broker, bao gồm địa chỉ IP hoặc tên miền, cổng kết nối và thông tin đăng nhập nếu cần thiết.
- **Xem các chủ đề (Topics):** Ứng dụng cho phép bạn xem danh sách các chủ đề MQTT mà bạn có thể đăng ký (subscribe) hoặc gửi thông tin (publish). Bạn có thể thấy thông tin về mỗi chủ đề và trạng thái của chúng.
- **Publish và Subscribe:** Bạn có thể gửi (publish) thông điệp lên các chủ đề MQTT và đăng ký (subscribe) vào các chủ đề để nhận thông tin từ chúng. MQTT Explorer cung cấp giao diện trực quan để thực hiện các hoạt động này.
- **Chế độ xem Real-Time:** MQTT Explorer cho phép bạn xem thông điệp MQTT được gửi và nhận trực tiếp trong thời gian thực. Điều này rất hữu ích để theo dõi và kiểm tra giao tiếp MQTT.

- **Lưu và tải cấu hình:** Bạn có thể lưu cấu hình kết nối và thiết lập cài đặt MQTT để sử dụng sau này và chia sẻ cấu hình với người khác.
- **Bảo mật:** MQTT Explorer hỗ trợ các tùy chọn bảo mật như sử dụng giao thức TLS/SSL để mã hóa thông tin, cũng như xác thực bằng tên người dùng và mật khẩu.
- **Hỗ trợ nhiều nền tảng:** MQTT Explorer có phiên bản cho nhiều nền tảng, bao gồm Windows, macOS và Linux.

Một thông điệp tiêu biểu trong giao thức MQTT:

PINGRESP là một trong những loại thông điệp (message) trong giao thức MQTT, phản hồi từ máy chủ MQTT đến client MQTT sau khi client đã gửi một gói tin PINGREQ để kiểm tra tính sống còn của kết nối.

Cụ thể:

- Khi một client MQTT muốn kiểm tra xem kết nối tới máy chủ MQTT có còn sống hay không, nó gửi một gói tin PINGREQ đến máy chủ.
- Máy chủ MQTT nhận được gói tin PINGREQ và sau khi kiểm tra kết nối, nó sẽ gửi một phản hồi dưới dạng gói tin PINGRESP về cho client.
- Khi client nhận được gói tin PINGRESP, nó biết rằng kết nối vẫn còn và không cần phải thực hiện lại việc kết nối lại.

PINGRESP được sử dụng để duy trì kết nối giữa client và máy chủ MQTT bằng cách thường xuyên gửi các gói tin PINGREQ và kiểm tra phản hồi PINGRESP từ máy chủ. Nếu kết nối bị ngắt mất hoặc máy chủ không phản hồi đúng cách trong khoảng thời gian quy định, client có thể thực hiện lại việc kết nối để duy trì tính kết nối với MQTT.

2.11. Ứng dụng di động MyMQTT

Link tải MyMQTT: [MyMQTT](#)

MyMQTT là một ứng dụng MQTT client dành riêng cho thiết bị di động Android và có thể tải về từ Google Play Store. Ứng dụng này cho phép bạn thực hiện các hoạt động liên quan đến giao thức MQTT trực tiếp từ điện thoại hoặc máy tính bảng Android của bạn.

Một số tính năng quan trọng của ứng dụng MyMQTT:

- **Kết nối MQTT Broker:** MyMQTT cho phép thiết lập và quản lý kết nối tới MQTT broker thông qua giao thức MQTT hoặc MQTT over WebSocket. Bạn có thể chỉ định thông tin về broker, bao gồm địa chỉ IP hoặc tên miền, cổng kết nối và thông tin đăng nhập nếu cần thiết.
- **Publish và Subscribe:** Bạn có thể gửi (publish) thông điệp lên các chủ đề MQTT và đăng ký (subscribe) vào các chủ đề để nhận thông tin từ chúng.
- **Real-Time Monitoring:** MyMQTT cho phép bạn xem thông điệp MQTT được gửi và nhận trực tiếp trong thời gian thực, theo dõi và kiểm tra giao tiếp MQTT.
- **Cấu hình tiên bộ:** Bạn có thể tùy chỉnh cài đặt MQTT và quản lý nhiều cấu hình kết nối khác nhau.

- **Lưu lịch sử:** MyMQTT có khả năng lưu lịch sử của các thông điệp MQTT, cho phép bạn xem và thử nghiệm lại thông điệp trước đó.
- **Chế độ bảo mật:** Ứng dụng hỗ trợ bảo mật bằng cách sử dụng giao thức TLS/SSL để mã hóa thông tin và cung cấp tùy chọn xác thực bằng tên người dùng và mật khẩu.
- **Hỗ trợ nhiều chủ đề:** MyMQTT cho phép bạn làm việc với nhiều chủ đề MQTT khác nhau và theo dõi chúng một cách dễ dàng.

MyMQTT

Please read and accept the privacy policy.

☒ I agree to the privacy policy

Let's go

MQTT Broker

Host

Port 1883 ☐ SSL

MQTT V3 **MQTT V5**

Credentials

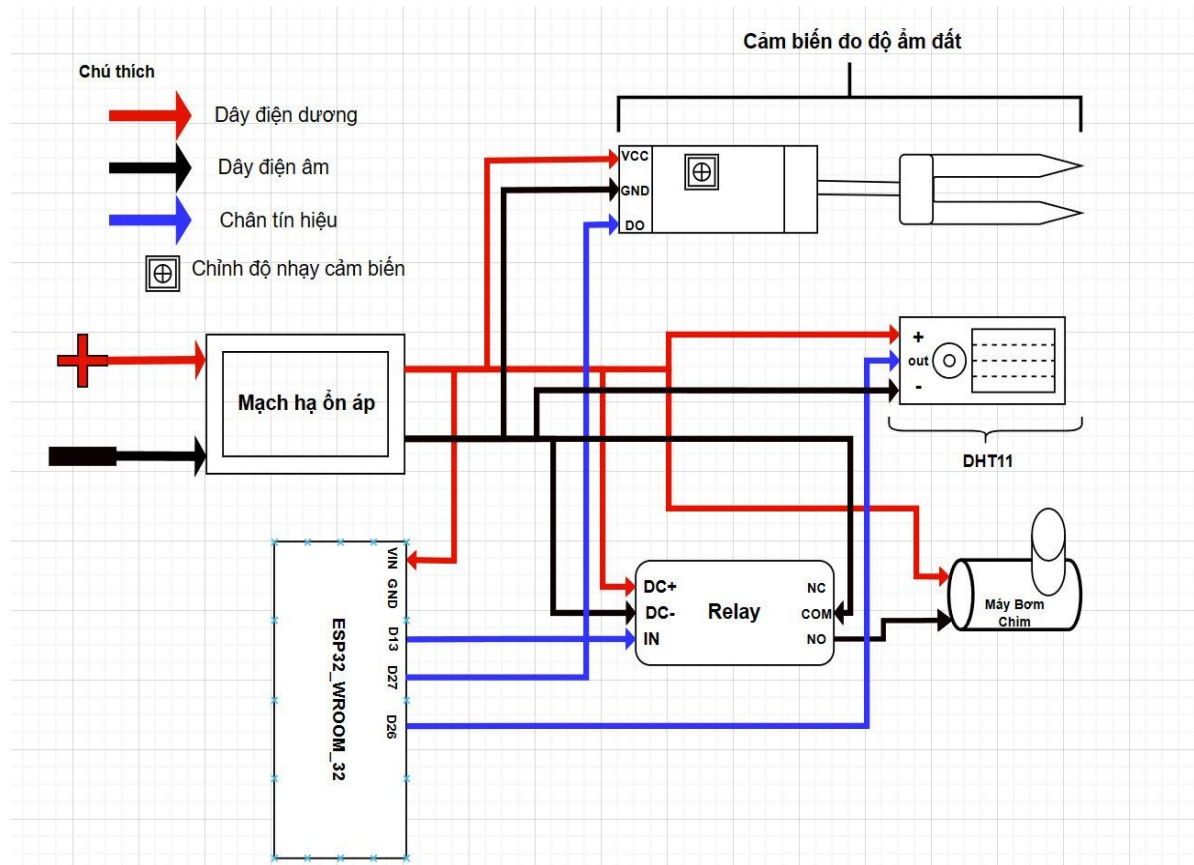
Username (optional)

Password (optional)

Connect

Hình 2.11.1. Giao diện MyMQTT sau khi đã tải về máy

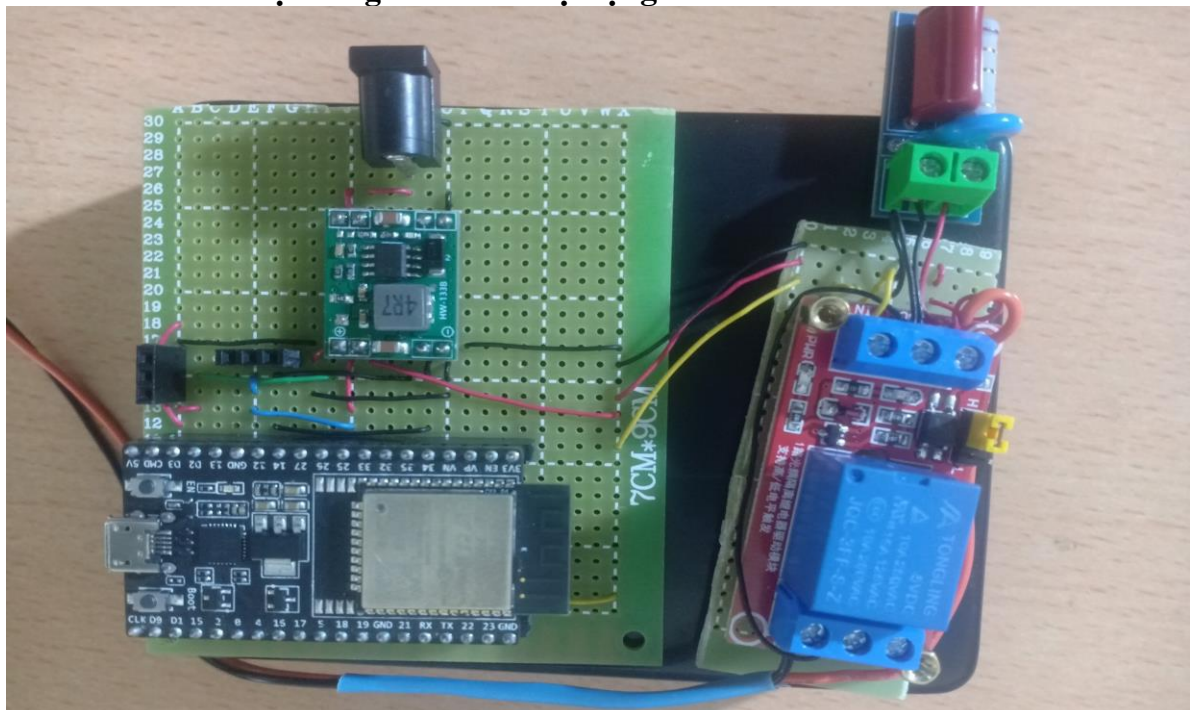
CHƯƠNG 3: MÔ HÌNH



Hình 3.1. Hình vẽ mô phỏng sơ đồ kết nối mạch

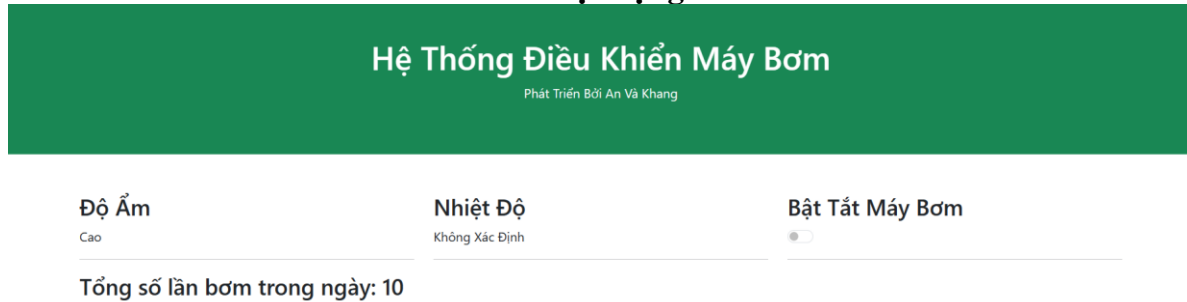
CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC (Link github - Hình ảnh bảng mạch và giao diện web)

4.1. Hình ảnh hệ thống tưới nước tự động



Hình 4.1.1. Hệ thống tưới nước tự động

4.2. Hình ảnh website điều khiển hoạt động tưới nước từ xa



Hình 4.2.1. Website điều khiển hoạt động tưới nước từ xa

CHƯƠNG 5: DEMO

5.1. Lập trình ESP32, kết nối với Wifi và MQTT

Chúng ta cần bắt đầu bằng việc nhập các thư viện cần thiết để xây dựng ứng dụng của mình. Sử dụng thư viện TimeLib để quản lý thời gian, thư viện WiFi để kết nối với mạng WiFi, thư viện PubSubClient để giao tiếp với máy chủ MQTT và cuối cùng là thư viện DHT để đọc dữ liệu từ cảm biến DHT11.

Việc này sẽ giúp chúng ta tương tác với thời gian, kết nối với mạng, truyền thông với máy chủ MQTT và thu thập dữ liệu từ cảm biến DHT11 một cách hiệu quả. Điều này sẽ là bước quan trọng đầu tiên để chúng ta có thể xây dựng một ứng dụng IoT hoàn chỉnh với đầy đủ chức năng.

```
#include <TimeLib.h> //thư viện quản lý thời gian
#include <WiFi.h>      //thư viện cho kết nối wifi
#include <string.h>
#include <PubSubClient.h> //thư viện MQTT
#include <DHT.h> //thư viện cảm biến nhiệt độ DHT11
```

Để thiết lập kết nối Wifi và MQTT cho dự án, chúng ta cần thực hiện một loạt các bước quan trọng. Đầu tiên, cấu hình kết nối Wifi, đảm bảo rằng thiết bị của chúng ta có thể truy cập mạng Wifi một cách dễ dàng. Chúng ta phải xác định các thông số quan trọng như tên mạng Wifi (SSID) và mật khẩu của mạng Wifi. Thông tin này cần được nhập vào chương trình của chúng ta để thiết bị có thể kết nối đến mạng Wifi được chọn.

```
//Cấu hình Wifi và MQTT
// const char *ssid = "Hana";
// const char *password = "66622888";
// const char *ssid = "Sinh vien Thuy Loi";
// const char *password = "Thuyloi2023";
const char *ssid = "TLU -CHUYENDOISO";
const char *password = "chuyendoiso@2023tlus";
```

Thiết lập kết nối MQTT để có thể gửi/nhận dữ liệu từ và đến máy chủ MQTT một cách hiệu quả và đáng tin cậy. Chúng ta xác định các thông số như địa chỉ máy chủ MQTT, cổng, tên người dùng và mật khẩu. Các thông số này đóng vai trò quan trọng

trong việc xác định máy chủ MQTT mà chúng ta muốn kết nối và xác thực với nó. Điều này cho phép chúng ta gửi và nhận dữ liệu một cách an toàn và đáng tin cậy thông qua giao thức MQTT.

```
const char *mqtt_server = "phuongnamdts.com";
const int mqtt_port = 4783;
const char *mqtt_user = "baonammqtt";
const char *mqtt_pass = "mqtt@d1git";
```

Tiến hành gán địa chỉ dẫn tới các cổng chủ đề trong giao thức MQTT, nhằm xác định cách mà dữ liệu được tổ chức và xác định rõ nội dung của thông điệp. Quá trình này đóng vai trò quan trọng trong việc xây dựng một hệ thống giao tiếp IoT hiệu quả và thông minh.

Thiết bị cảm biến đất "SOILSENSOR" sẽ đăng ký **root_topic_subscribe** để nhận các yêu cầu (commands) từ máy chủ MQTT hoặc các thiết bị khác gửi đến nó thông qua chủ đề "**cmnd/SOILSENSOR/command**"

Thiết bị cảm biến đất **root_topic_publish** đăng/gửi (publish) giá trị hoặc dữ liệu của nó lên máy chủ MQTT qua chủ đề "**HQT/SOILSENSOR/value**"

```
//Các cổng chủ đề trong MQTT
const char *root_topic_subscribe = "cmnd/SOILSENSOR/command";
const char *root_topic_publish = "HQT/SOILSENSOR/value";
```

Kế đến, khởi tạo đối tượng kết nối Wifi, một bước quan trọng để thiết lập liên kết không dây với mạng. Điều này được thực hiện thông qua sử dụng thư viện WiFiClient, cho phép thiết bị kết nối một cách an toàn và ổn định.

Chúng ta sẽ sử dụng thư viện PubSubClient để khởi tạo đối tượng kết nối MQTT. Giao thức MQTT rất quan trọng trong việc truyền thông giữa các thiết bị IoT và máy chủ MQTT. Điều này giúp chúng ta chuyển thông tin, dữ liệu từ cảm biến và điều khiển các thiết bị ngoại vi như relay, cảm biến dòng, cảm biến độ ẩm đất và cảm biến nhiệt độ DHT11.

```
//Đối tượng kết nối Wifi và MQTT
WiFiClient espClient;
PubSubClient client(espClient);
```

Các chân được kết nối cho các cảm biến và thiết bị ngoại vi này là một phần quan trọng của quá trình triển khai. Chúng ta cần đảm bảo rằng mọi kết nối vật lý được thiết lập đúng cách để đảm bảo thu thập dữ liệu và điều khiển các thiết bị ngoại vi một cách hiệu quả.

Khai báo biến relayPin để lưu trữ số chân GPIO (General Purpose Input/Output) được sử dụng để kết nối với một relay trong hệ thống. Biến flowsensorPin xác định số chân

GPIO được sử dụng để kết nối với cảm biến dòng, đo dòng điện chảy qua một mạch điện.

Biến `soilsensorPin` xác định số chân GPIO được sử dụng để kết nối với cảm biến độ ẩm đất để đo mức độ ẩm của đất, thông qua điện cực chì hoặc điện cực khác, để xác định xem cây trồng cần tưới nước hay không.

Biến `dht11` xác định số chân GPIO được sử dụng để kết nối với cảm biến nhiệt độ và độ ẩm DHT11. Cảm biến DHT11 là một cảm biến kết hợp có khả năng đo nhiệt độ và độ ẩm trong môi trường.

```
//Các chân cảm biến
const int relayPin = 17;//19
const int flowsensorPin = 35;
const int soilsensorPin = 36;//27
const int dht11 = 10;//26
```

Khai báo một đối tượng cảm biến DHT với tên là `dht`. Khởi tạo nó bằng cách gọi constructor của lớp DHT và truyền hai tham số cho constructor: chân kết nối (pin) 10 trên mạch điều khiển mà cảm biến DHT11 được kết nối và loại cảm biến bạn đang sử dụng (ở đây là DHT11) để xác định cách đọc và giải mã dữ liệu từ cảm biến cụ thể này.

```
//Đối tượng cảm biến DHT
DHT dht(dht11, DHT11);
```

Khai báo và khởi tạo các biến dùng để quản lý thời gian, lưu trữ thời điểm mà các sự kiện xảy ra hoặc được kích hoạt trong chương trình.

Biến `currentTimer` lưu thời điểm hiện tại dưới dạng số nguyên không dấu (unsigned long). Ban đầu, giá trị của biến này được đặt là 0, sử dụng để theo dõi thời gian hoặc đảm bảo rằng các hành động/sự kiện diễn ra sau một khoảng thời gian cố định.

Tương tự biến `currentTimer`, biến `relayStartTime` lưu thời điểm bắt đầu chạy máy bơm, sử dụng để đo thời gian chạy của máy bơm hoặc tính toán khoảng thời gian giữa các lần gửi dữ liệu.

Biến `DHTStartTime` lưu thời điểm bắt đầu thực hiện đọc dữ liệu từ cảm biến nhiệt độ và độ ẩm DHT11, nó sẽ được cập nhật khi bạn bắt đầu đọc dữ liệu từ cảm biến.

Tương tự như `DHTStartTime`, biến `SoilStartTime` được sử dụng để lưu thời điểm bắt đầu đọc dữ liệu từ cảm biến độ ẩm đất, nó sẽ được cập nhật khi bạn bắt đầu đọc dữ liệu từ cảm biến độ ẩm đất.

Biến `MQTTStartTime` lưu thời điểm bắt đầu thực hiện các hoạt động liên quan đến gửi dữ liệu lên MQTT, nó được cập nhật khi bạn bắt đầu gửi dữ liệu lên MQTT.

```
//Biến lưu thời điểm lần cuối gửi dữ liệu lên MQTT
unsigned long currentTimer = 0;
unsigned long relayStartTime = 0; //thời gian bắt đầu chạy relay máy bơm
unsigned long DHTStartTime = 0;
unsigned long SoilStartTime = 0;
unsigned long MQTTSendTime = 0;
```

Định nghĩa một số hằng số (constants) để thiết lập khoảng thời gian giữa các lần gửi dữ liệu trong chương trình. Mỗi hằng số đại diện cho một loại dữ liệu cụ thể mà bạn muốn gửi hoặc cập nhật và định nghĩa khoảng thời gian giữa các lần gửi dữ liệu đó.

Interval là khoảng thời gian chung giữa các lần gửi dữ liệu, là khoảng thời gian cố định giữa các lần cập nhật dữ liệu chung trong chương trình. Trong trường hợp này, giá trị của interval được thiết lập là 30.

Relayinterval là khoảng thời gian giữa các lần cập nhật trạng thái của một relay. Giá trị của relayinterval được thiết lập là 10.

Tương tự với thời gian thiết lập là 10, Soilinterval là khoảng thời gian giữa các lần đo độ ẩm của đất hoặc thông tin liên quan đến đất. DHTinterval là khoảng thời gian giữa các lần đo nhiệt độ và độ ẩm bằng cảm biến DHT. MQTTinterval là khoảng thời gian giữa các lần gửi dữ liệu thông qua giao thức MQTT

```
//Khoảng thời gian thiết lập giữa các lần gửi dữ liệu
const int interval = 30;
const int relayinterval = 10;
const int Soilinterval = 10;
const int DHTinterval = 10;
const int MQTTinterval = 10;
```

Thực hiện theo dõi thời gian trôi qua kể từ khi chương trình bắt đầu chạy. Cụ thể, các biến elapsedRelayTime1 và elapsedRelayTime2 được sử dụng để kiểm tra thời gian cần thiết để bật hoặc tắt máy bơm.

elapsedDHTTime và elapsedSoilTime dùng để đo thời gian giữa các lần đọc dữ liệu từ cảm biến nhiệt độ/ độ ẩm và cảm biến đất.

Cuối cùng, elapsedMQTTTime có thể sử dụng để kiểm tra thời gian trôi qua giữa các lần truyền dữ liệu thông qua giao thức MQTT, chẳng hạn như gửi hoặc nhận dữ liệu qua MQTT.

Những biến thời gian này giúp điều khiển và lên lịch các hoạt động trong chương trình dựa trên thời gian.

Các biến unsigned long được sử dụng thay vì int để đảm bảo rằng chúng có khả năng theo dõi thời gian trôi qua trong một khoảng thời gian dài.

```
// Biến so sánh
//relay máy bơm
    unsigned long elapsedRelayTime1 = 0;
    unsigned long elapsedRelayTime2 = 0;
// các biến so sánh còn lại
    unsigned long elapsedDHTTime = 0;
    unsigned long elapsedSoilTime = 0;
    unsigned long elapsedMQTTTime = 0;
```

Cuối cùng, trước khi setup, định nghĩa các biến để kiểm tra và lưu trữ các trạng thái và giá trị khác nhau trong chương trình.

Biến isRelayON kiểm tra trạng thái hiện tại của relay với hai trạng thái: BẬT (HIGH) hoặc TẮT (LOW), sử dụng để kiểm tra hoặc điều khiển relay tùy theo 2 trạng thái này.

Mảng ký tự sensordht có độ dài tối đa là 15 ký tự, được sử dụng để lưu trạng thái độ ẩm đo được từ cảm biến DHT.

Một chuỗi ký tự soilMoisture lưu trữ trạng thái độ ẩm đất từ một cảm biến độ ẩm đất. Tương tự, biến mảng ký tự soilMoistureStr với kích thước là 20 được sử dụng để lưu trạng thái độ ẩm đất.

Biến ibusy kiểm tra trạng thái hoạt động của chương trình, quản lý việc thực hiện các tác vụ đồng thời. Nếu ibusy có giá trị true, thì chương trình đang bận rộn thực hiện một tác vụ nào đó, ngược lại, nếu là false, chương trình có thể thực hiện các tác vụ khác.

```
//Biến quy định, kiểm tra trạng thái relay
    int isRelayON = LOW;
// biến chuỗi khai báo trạng thái độ ẩm DHT11
    char sensordht[15];
    char soilMoisture;
    char soilMoistureStr[20];
    bool isbusy = false;
```

Hàm setup() thiết lập kết nối WiFi, MQTT và khởi tạo cảm biến DHT11 đọc dữ liệu nhiệt độ và độ ẩm, được thực thi một lần duy nhất khi nạp chương trình vào thiết bị.

Serial.begin(9600): khởi tạo cổng giao tiếp Serial với tốc độ baud rate là 9600. Trong đó Serial là cổng giao tiếp thông qua cổng USB hoặc cổng nối tiếp trên Arduino, cho phép bạn gửi và nhận dữ liệu từ và đến máy tính hoặc một thiết bị ngoại vi khác. Serial Monitor sẽ được sử dụng để hiển thị thông tin debug và giám sát trong quá trình chạy chương trình.

setup_wifi(): thực hiện cấu hình kết nối WiFi, đã được định nghĩa ngoài hàm setup().

```
void setup() {
    // Khởi tạo cổng giao tiếp Serial, kết nối WiFi và cài đặt cổng cho cảm biến
    Serial.begin(9600);
    setup_wifi();
```


pinMode(2, OUTPUT) cấu hình chân số 2 của mạch Arduino thành chế độ OUTPUT, chân này sẽ được sử dụng để điều khiển đèn D2 của mạch. Còn **digitalWrite(2, LOW)** thiết lập chân số 2 ở mức thấp (LOW) để tắt đèn LED.

```
//đèn D2 của mạch
pinMode(2, OUTPUT);
digitalWrite(2, LOW);
```

Thiết lập và cấu hình một đối tượng MQTT client bằng cách đặt thời gian kết nối, xác định máy chủ MQTT và cổng kết nối, cũng như thiết lập hàm gọi lại để xử lý dữ liệu nhận được từ máy chủ MQTT.

client.setKeepAlive(90) cài đặt thời gian giữa các gói tin trong kết nối MQTT, cứ sau 90 giây, client MQTT sẽ gửi một gói tin PINGREQ đến máy chủ MQTT để duy trì kết nối. Nếu không nhận được PINGRESP phản hồi từ máy chủ trong khoảng thời gian này, kết nối sẽ bị đóng.

client.setServer cài đặt máy chủ MQTT và cổng kết nối MQTT để client xác định nơi nó sẽ gửi và nhận các thông điệp MQTT. client.setCallback cài đặt hàm callback cho client MQTT để khi client nhận được các thông điệp từ MQTT, nó sẽ gọi hàm này để xử lý.

```
//thiết lập kết nối MQTT
client.setKeepAlive(90);
client.setServer(mqtt_server, mqtt_port);
client.setCallback(callback);
```

Tiếp tục cấu hình chân relayPin chế độ OUTPUT với pinMode, để sẵn sàng điều khiển relay bật/tắt máy bơm. Thiết lập chân relayPin ở mức thấp (LOW) với digitalWrite khi chương trình bắt đầu chạy.

Tương tự relayPin, thiết lập chế độ hoạt động của chân dht11 là INPUT để đọc dữ liệu từ cảm biến DHT11. Cảm biến DHT11 có thể gửi dữ liệu nhiệt độ và độ ẩm đến Arduino thông qua chân này. Đặt chân dht11 ở mức cao (HIGH), có thể dùng để kích hoạt cảm biến DHT11. Quá trình này làm cho cảm biến hoạt động chính xác hơn trong quá trình khởi động.

Phương thức begin() gọi trên đối tượng để khởi động cảm biến DHT11, chuẩn bị cho việc đọc dữ liệu từ nó.

```
//cài đặt chân relay
//relay máy bơm
pinMode(relayPin, OUTPUT);
digitalWrite(relayPin, LOW);
//khởi động cảm biến DHT11
pinMode(dht11, INPUT);
digitalWrite(dht11, HIGH);
dht.begin();
```

Triển khai hàm loop, biến currentTimer được tăng lên một đơn vị để theo dõi thời gian chạy của chương trình. Sau đó, các biến thời gian đã trôi qua từ các sự kiện trước đó được cập nhật.

Biến elapsedRelayTime1 tính và lưu thời gian đã trôi qua từ thời điểm relayStartTime đến thời điểm hiện tại tại currentTimer, giúp bạn biết được bao lâu kể từ khi bạn bắt đầu đo thời gian tới thời điểm hiện tại.

Tương tự, biến elapsedDHTTime tính và lưu thời gian đã trôi qua từ DHTStartTime đến currentTimer để đo thời gian giữa các sự kiện liên quan đến cảm biến DHT (DHT sensor).

Biến elapsedSoilTime tính và lưu thời gian đã trôi qua từ SoilStartTime đến currentTimer để đo thời gian giữa các sự kiện liên quan đến cảm biến độ ẩm đất (soil moisture sensor).

Cuối cùng, biến elapsedMQTTTime tính và lưu thời gian đã trôi qua từ MQTTStartTime đến currentTimer để đo thời gian giữa các sự kiện liên quan đến giao tiếp MQTT.

```
void loop() {  
  //khởi tạo các biến lưu thời gian cảm biến  
  currentTimer++;  
  elapsedRelayTime1 = (currentTimer - relayStartTime);  
  elapsedDHTTime = (currentTimer - DHTStartTime);  
  elapsedSoilTime = (currentTimer - SoilStartTime);  
  elapsedMQTTTime = (currentTimer - MQTTStartTime);
```

Chương trình kiểm tra điều kiện để cập nhật trạng thái của relay máy bơm và gửi dữ liệu lên MQTT. Nếu đã đủ thời gian kể từ lần cuối cập nhật là 30 giây, relay máy bơm sẽ được bật. Lúc này, thời điểm bắt đầu thực hiện đọc dữ liệu từ cảm biến sẽ được tính là thời gian chạy hiện tại.

Sau đó, dữ liệu từ cảm biến DHT được đọc và gửi lên MQTT, màn hình hiện thông báo "Bật relay bơm" trên cổng nối tiếp (Serial).

```
// kiểm tra thời gian để cập nhật trạng thái relay máy bơm và gửi dữ liệu lên MQTT  
if (elapsedRelayTime1 >= interval && isRelayON == LOW) {  
  relayStartTime = currentTimer;  
  digitalWrite(relayPin, HIGH);  
  isRelayON = HIGH;  
  isbusy = true;  
  Serial.println("Bật relay bơm");  
}
```

Biến elapsedRelayTime2 tính và lưu thời gian đã trôi qua từ khi relay bắt đầu được bật relayStartTime đến thời điểm hiện tại tại currentTimer.

Bắt đầu kiểm tra xem đã đủ thời gian để tắt relay hay chưa bằng cách kiểm tra xem thời gian đã trôi qua `elapsedRelayTime2` đã lớn hơn hoặc bằng 10 và trạng thái hiện tại của relay đã và đang bật chưa. Điều này đảm bảo rằng đã đủ thời gian để tắt relay sau khi nó được bật.

Relay sau đó được tắt, biến `isbusy` đánh dấu rằng quá trình bơm đã hoàn thành, relay đã tắt và chuyển về false để đánh dấu rằng không có tác vụ nào đang thực hiện. Màn hình hiện thông báo "Tắt relay bơm" trên cổng nối tiếp (Serial). Cập nhật lại thời gian bắt đầu của relay bằng thời gian hiện tại `currentTimer`. Điều này đặt lại đồng hồ bắt đầu cho lần bật relay tiếp theo.

```
elapsedRelayTime2 = (currentTimer - relayStartTime);  
// Kiểm tra nếu đã đủ thời gian bơm (10 giây)  
if (elapsedRelayTime2 >= relayinterval && isRelayON == HIGH) {  
    digitalWrite(relayPin, LOW); // Tắt relay sau khi đã bơm đủ thời gian  
    isRelayON = LOW;  
    isbusy = false;  
    Serial.println("tắt relay bơm");  
    relayStartTime = currentTimer;  
}
```

Thực hiện kiểm tra xem biến `elapsedSoilTime` có lớn hơn hoặc bằng 10 không. Nếu điều kiện này đúng, tức là đã đủ thời gian trôi qua để đọc dữ liệu độ ẩm đất từ cảm biến độ ẩm đất và xử lý.

Cảm biến bắt đầu độ ẩm đất đọc giá trị 0 (ẩm) hoặc 1 (khô) và lưu trữ nó vào biến `soilMoisture`. Sau đó in ra cổng Serial một chuỗi ký tự "Soil Moisture" để xác định dữ liệu mà bạn đang ghi và in giá trị độ ẩm đất (biến `soilMoisture`) dưới dạng chuỗi, dùng hệ cơ số HEX (hệ 16).

Chuyển giá trị số của biến `soilMoisture` thành một chuỗi và lưu trữ trong biến `soilMoistureStr`. Hàm `sprintf` cho phép định dạng chuỗi dựa trên một mẫu và biến đầu vào rồi cập nhật thời điểm bắt đầu đọc dữ liệu từ cảm biến độ ẩm đất `SoilStartTime` bằng thời gian hiện tại `currentTimer`.

```
// Đọc độ ẩm đất từ cảm biến độ ẩm đất  
if (elapsedSoilTime >= Soilinterval) {  
    soilMoisture = digitalRead(soilsensorPin);  
    Serial.print("Soil Moisture");  
    Serial.println(String(soilMoisture, HEX));  
    sprintf(soilMoistureStr, "%d", soilMoisture);  
    SoilStartTime = currentTimer;  
}
```

Khi đã hoàn thành việc đọc độ ẩm đất, chúng ta sẽ kiểm tra độ ẩm đất và điều khiển relay. Nếu `isbusy` có giá trị là false, tức là chương trình đang không xử lý công việc

gì, kiểm tra trạng thái độ ẩm đất soilMoisture, nếu có giá trị bằng 1 (độ ẩm thấp) và relay đang bật thì tiếp tục chạy máy bơm. Ngược lại, tắt relay ngừng bơm.

```
// Kiểm tra độ ẩm đất và điều khiển relay
if (!isbusy) {
    if (soilMoisture == 1 && isRelayON == HIGH) {
        digitalWrite(relayPin, HIGH);
    } else {
        digitalWrite(relayPin, LOW);
    }
}
```

Kiểm tra xem kết nối MQTT nếu chưa được kết nối thì gọi hàm reconnect() để thử kết nối lại. Hàm reconnect() sẽ được định nghĩa ngoài vòng lặp, có nhiệm vụ thực hiện kết nối lại với máy chủ MQTT khi bị mất kết nối.

Sau khi thực hiện kết nối MQTT thành công và thời gian đã trôi qua elapsedMQTTTime có lớn hơn hoặc bằng 10, sao chép chuỗi dữ liệu từ biến soilMoistureStr và sensordht vào chuỗi str. Chuỗi này được chuyển thành mảng ký tự msg có độ dài tối đa 36 ký tự. Mục đích của việc này là để chuẩn bị dữ liệu để gửi lên máy chủ MQTT.

Mảng ký tự msg đã sao chép ở trên sẽ được gửi lên máy chủ MQTT với chủ đề (topic) được xác định bởi biến root_topic_publish, thực hiện cập nhật lại thời gian bắt đầu thời gian MQTTStartTime thành thời điểm hiện tại currentTimer để tính toán thời gian kế tiếp khi phải gửi dữ liệu lên MQTT.

Cho phép chương trình chờ đợi 1 giây trước khi tiếp tục thực hiện vòng lặp kế tiếp. Điều này giúp kiểm soát tần suất gửi dữ liệu lên MQTT và tránh gửi dữ liệu quá nhanh. Hàm loop() của đối tượng MQTT client được gọi để xử lý các sự kiện và thông báo từ máy chủ MQTT. Nó đảm bảo rằng chương trình sẽ tiếp tục nhận và gửi dữ liệu từ MQTT trong vòng lặp chính.

```
// Kiểm tra kết nối MQTT và gửi dữ liệu
if (!client.connected()) {
    reconnect();
}
if (client.connected() && elapsedMQTTTime >= MQTTinterval) {
    char msg[36];
    String str = String(soilMoistureStr) + ";" + String(sensordht);
    str.toCharArray(msg, 36);
    //Gửi dữ liệu lên MQTT
    client.publish(root_topic_publish, msg);
    MQTTStartTime = currentTimer;
}
delay(1000);
//lặp lại việc nhận và gửi dữ liệu từ MQTT
client.loop();
```

Hàm `setup_wifi()` thực hiện cài đặt và thiết lập kết nối WiFi cho thiết bị.

Đầu tiên, trên cổng giao tiếp serial sẽ hiển thị thông báo “Dang ket noi wifi” để mô tả quá trình kết nối Wi-Fi đang diễn ra, kèm theo là tên mạng Wi-Fi (SSID) mà bạn đang cố gắng kết nối.

Kế đến sử dụng `WiFi.begin(ssid, password)` để bắt đầu quá trình kết nối WiFi với tên mạng `ssid` và mật khẩu `password`. Tiếp tục kiểm tra trạng thái kết nối WiFi bằng `WiFi.status()` và nó sẽ in ra dấu "." mỗi 0.5 giây nếu vẫn chưa kết nối thành công (`WL_CONNECTED`). Khi kết nối được thiết lập thành công, nó in ra "Da ket noi!" và hiển thị địa chỉ IP cục bộ của thiết bị.

```
// Thiết lập kết nối WiFi
void setup_wifi() {
  Serial.println();
  Serial.print("Dang ket noi wifi: ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("Da ket noi!");
  Serial.println("IP: ");
  Serial.println(WiFi.localIP());
}
```

Tạo hàm `reconnect()` để kết nối lại với máy chủ MQTT khi bị mất kết nối. Hàm sử dụng một vòng lặp `while` để kiểm tra trạng thái kết nối MQTT. Nếu kết nối không thành công (`!client.connected()`), trên cổng Serial sẽ in ra dòng "Ket noi den server MQTT..." và tiến hành kết nối lại bằng cách tạo một `clientId` và sử dụng nó để kết nối.

Thử kết nối lại với máy chủ MQTT bằng cách sử dụng thông tin `clientId` vừa tạo trước đó, tên người dùng `mqtt_user`, và mật khẩu MQTT `mqtt_pass` đã được khai báo trước đây. Hàm `connect` trả về `true` nếu kết nối thành công.

Tiếp đến đăng ký chủ đề MQTT (`root_topic_subscribe`) để nhận dữ liệu từ máy chủ MQTT. Nếu không kết nối được, hàm in ra thông báo lỗi và thử kết nối lại sau 5 giây (`delay(5000)`).

Hàm `reconnect` được sử dụng để đảm bảo kết nối MQTT luôn ổn định và có khả năng tự động kết nối lại sau khi mất kết nối.

```
// Kết nối lại khi mất kết nối MQTT
void reconnect() {
    while (!client.connected()) {
        Serial.print("Ket noi den server MQTT...");
        String clientId = "ESP32_WROOM_KHANG";

        if (client.connect(clientId.c_str(), mqtt_user, mqtt_pass)) {
            Serial.println("Da ket noi MQTT!");

            if (client.subscribe(root_topic_subscribe)) {
                Serial.println("Dang ky OK");
            } else {
                Serial.println("Loi dang ky");
            }
        } else {
            Serial.print("loi ket noi -> ");
            Serial.print(client.state());
            Serial.println(" thu lai sau 5 giay ");
            delay(5000);
        }
    }
}
}
```

Cuối cùng tạo hàm callback để thực hiện hành động điều khiển relay (BẬT hoặc TẮT) mỗi khi thiết bị nhận được một tin nhắn từ máy chủ MQTT.

Hàm callback sử dụng các tham số: topic chứa tên chủ đề MQTT mà tin nhắn được gửi đến, chủ đề này xác định loại dữ liệu hoặc chức năng mà tin nhắn đang đề cập; payload là một mảng byte chứa dữ liệu của tin nhắn MQTT; length chứa độ dài của payload, tức là số lượng byte dữ liệu trong tin nhắn.

Các dữ liệu khi được ghi vào sẽ hiển thị thông báo "Du lieu duoc ghi vao" và tên chủ đề của tin nhắn để theo dõi thông tin.

Khởi tạo biến incoming lưu trữ nội dung của tin nhắn MQTT. Sử dụng một vòng lặp để duyệt qua từng byte dữ liệu trong payload (dữ liệu của tin nhắn), thêm mỗi byte từ payload vào biến incoming và chuyển chúng thành một chuỗi ký tự (incoming).

Loại bỏ các khoảng trắng (nếu có) ở đầu và cuối của biến incoming để làm sạch dữ liệu bằng hàm trim(). Lúc này tại cổng Serial sẽ hiển thị thông báo đang đọc dữ liệu với thông tin dữ liệu đã đọc được trong biến incoming.

Từ đây, hàm kiểm tra giá trị của chuỗi incoming để xác định hành động tiếp theo. Nếu chuỗi incoming là "ON", tức là relay đang được bật, chương trình bắt đầu chờ đợi, sau 10 giây thì thực hiện tắt relay. Ngược lại thì ghi nhận relay ở trạng thái LOW (TẮT relay).

Trong trường hợp không khớp với "ON" hoặc "OFF", hàm cũng đặt relayPin ở trạng thái LOW (TẮT relay).

```
//Hàm xử lý khi nhận được tin nhắn từ MQTT gửi
void callback(char *topic, byte *payload, unsigned int length) {
    String incoming = "";
    Serial.print("Du lieu duoc ghi vao -> ");
    Serial.print(topic);
    Serial.println("");
    for (int i = 0; i < length; i++) {
        incoming += (char)payload[i];
    }
    incoming.trim();
    Serial.println("Dang doc du lieu -> " + incoming);

    if (incoming == "ON") {
        digitalWrite(relayPin, HIGH);
        delay(10000);
        digitalWrite(relayPin, LOW);
    } else if (incoming == "OFF") {
        digitalWrite(relayPin, LOW);
    } else {
        digitalWrite(relayPin, LOW);
    }
}
```

5.2. Lập trình website

Trang web sử dụng một đoạn mã HTML và JavaScript với mục đích hiển thị thông tin về độ ẩm và nhiệt độ, đồng thời cung cấp khả năng điều khiển máy bơm thông qua giao diện người dùng. Sự tương tác này không chỉ mang lại thông tin về môi trường, mà còn mang tính ứng dụng cao, giúp người dùng quản lý hiệu quả các thay đổi trong môi trường.

Để tạo ra giao diện người dùng thân thiện và hấp dẫn, trang web sử dụng Bootstrap, một framework phổ biến cho việc phát triển giao diện web, giúp tăng cường trải nghiệm của người dùng và làm cho dữ liệu dễ dàng đọc hiểu.

❖ Trong mã HTML:

➤ Trong phần head của trang web

Trước tiên khai báo doctype cho trình duyệt để trình duyệt biết rằng trang web này được viết bằng phiên bản HTML5. Thẻ mở đầu của một tài liệu HTML cho biết rằng nội dung của trang web này sẽ được viết bằng ngôn ngữ tiếng Anh (en).

Trong phần head, trên thanh tiêu đề trang web hiển thị tiêu đề "Hệ Thống Điều Khiển Máy Bơm". Sử dụng thẻ meta chỉ định bảng mã ký tự UTF-8 để đảm bảo hiển thị chữ cái và meta khung nhìn giúp trang web tự động điều chỉnh kích thước phù hợp với kích thước màn hình của các thiết bị khác nhau.

Thêm một liên kết đến tập tin CSS của Bootstrap để tùy chỉnh giao diện của trang web theo cách của Bootstrap và liên kết đến tập tin JavaScript của Bootstrap bởi vì đối với Bootstrap, JavaScript là bắt buộc để một số tính năng hoạt động, vì vậy cần phải bao gồm nó.


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Hệ Thống Điều Khiển Máy Bơm</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
      rel="stylesheet"
    />
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
  </head>
```

➤ Trong phần thân của trang web:

Tạo một hộp chứa có độ rộng tối đa trên nền (bằng cách sử dụng lớp CSS "container-fluid"), với một lớp CSS "p-5" để thêm khoảng cách xung quanh. Đặt màu nền thành màu xanh lá cây bằng cách sử dụng lớp "bg-success", và đặt màu văn bản là trắng với lớp "text-white". Để căn giữa nội dung bên trong, sử dụng lớp "text-center".

Bên trong hộp chứa này, đặt một tiêu đề cấp 1 (<h1>) và một đoạn văn bản đơn giản bên trong thẻ <p>. Nội dung này sẽ hiển thị trên trang web với kiểu chữ mặc định của trình duyệt.

```
<body>
  <div class="container-fluid p-5 bg-success text-white text-center">
    <h1>Hệ Thống Điều Khiển Máy Bơm</h1>
    <p>Phát Triển Bởi An Và Khang</p>
  </div>
```

Tương tự, bạn có thể tạo một phần tử trong trang web của mình với việc tạo một vùng chứa bằng cách sử dụng thẻ "div" có lớp CSS là "container". Điều này giúp tạo một phần tử có chiều rộng tối đa và thêm khoảng cách ở phía trên (margin-top) cho nó cùng với các phần tử bên trong.

Trong phần tử "container" này, bạn tiếp tục tạo một phần tử con bằng thẻ "div" với lớp CSS là "row". Phần tử này giúp bạn tạo một hàng để chứa các cột trong một bố cục dạng lưới.

Bên trong phần tử "row", bạn tạo 3 cột sử dụng lớp CSS "col-sm-4". Điều này sẽ giúp bạn có 3 cột có chiều rộng tỷ lệ 1/3 trên màn hình khi độ rộng màn hình là nhỏ (small). Các cột sẽ có nội dung theo trình tự sau: một tiêu đề cấp 3, một thẻ <p> không có nội dung ban đầu nhưng được đặt một ID để bạn có thể thay đổi nội dung bằng JavaScript.

Để tạo công cụ kiểm tra dạng công tắc để bật/tắt máy bơm, bạn có thể tạo một ô kiểm tra <input type="checkbox">. Bọc nó bằng thẻ "form-check" và "form-switch" để tạo giao diện công tắc. Sử dụng JavaScript để xử lý việc bật/tắt máy bơm khi người dùng thay đổi trạng thái của công tắc. Sự thay đổi trạng thái này sẽ được xử lý bởi hàm "togglePump()".

Cuối cùng, bạn có thể tạo một phần tử "div" riêng biệt bên trong phần tử con ở bước 3, chứa một tiêu đề cấp 3 và một phần tử có ID. Nội dung ban đầu của phần

từ `` có thể được đặt là "0", và bạn có thể sử dụng JavaScript để cập nhật giá trị của nó theo nhu cầu.

```
<body>
  <div class="container-fluid p-5 bg-success text-white text-center">
    <h1>Hệ Thống Điều Khiển Máy Bơm</h1>
    <p>Phát Triển Bởi An Và Khang</p>
  </div>

  <div class="container mt-5">
    <div class="row">
      <div class="col-sm-4">
        <h3>Độ Ẩm</h3>
        <p id="doam"></p>
        <hr />
      </div>
      <div class="col-sm-4">
        <h3>Nhiệt Độ</h3>
        <p id="nhietdo"></p>
        <hr />
      </div>
      <div class="col-sm-4">
        <h3>Bật Tắt Máy Bơm</h3>
        <div class="form-check form-switch">
          <input
            id="flexSwitchCheckDefault"
            class="form-check-input"
            type="checkbox"
            role="switch"
            onchange="togglePump()"
          />
        </div>
        <hr />
      </div>
    </div>

    <div>
      <h3>Tổng số lần bơm trong ngày: <span id="pumpCount">0</span></h3>
    </div>
  </div>
```

❖ Phần JavaScript:

Tạo một hàm có tên là `togglePump`. Khi người dùng bấm vào công tắc trên trang web, hàm này sẽ được kích hoạt. Sử dụng `document.getElementById` để lấy thông tin về công tắc có id là "flexSwitchCheckDefault" đã đặt trong phần `<body>` của trang web. Sau đó, chúng ta sẽ lưu thông tin này vào biến `toggle` để kiểm soát việc bật/tắt máy bơm.

Tiếp theo, chúng ta sẽ kiểm tra xem công tắc có được bật hay không và lưu kết quả vào biến `isChecked`. Dựa trên giá trị của biến `isChecked`, chúng ta sẽ đặt biến `command` thành "ON" nếu công tắc được bật và thành "OFF" nếu người dùng tắt công tắc. Điều này sẽ quyết định lệnh mà chúng ta sẽ gửi đến máy bơm.

Cuối cùng, chúng ta sử dụng tham số `command` để gọi hàm `sendMqttRequest`, gửi yêu cầu đến máy bơm thông qua giao thức MQTT để bật hoặc tắt máy bơm.

```
<script>
var pumpCount = 0;
function togglePump() {
    var toggle = document.getElementById("flexSwitchCheckDefault");
    const isChecked = toggle.checked;
    const command = isChecked ? "ON" : "OFF";
    sendMqttRequest(command);
}
```

Bắt đầu kiểm tra xem công tắc (toggle) đã được bật (ON) chưa. Nếu máy bơm đang hoạt động, sau 10 giây kể từ khi chúng ta gọi hàm `setTimeout`, nó sẽ tự động tắt máy bơm bằng cách thay đổi trạng thái công tắc (toggle) trên trang web thành "tắt" (false) và gửi một yêu cầu tắt máy bơm thông qua MQTT bằng cách gọi hàm `sendMqttRequest` với tham số "OFF".

Sau khi máy bơm hoàn tất quá trình làm việc của nó, biến `pumpCount`, mà trước đó đã được khai báo, sẽ tăng giá trị lên một đơn vị. Nói cách khác, biến này sẽ được tăng lên để đếm số lần máy bơm đã hoạt động.

```
if (isChecked) {
    // Nếu máy bơm được bật, hẹn giờ để tự động tắt nó sau 10 giây.
    setTimeout(function () {
        toggle.checked = false; // Tắt switch trên web
        sendMqttRequest("OFF"); // Gửi yêu cầu MQTT để tắt máy bơm
    }, 10000); // 10 giây (10000 mili giây)

    // console.log(pumpCount);

    // Tăng biến đếm lần bơm
    pumpCount++;
    // document.getElementById("pumpCount").textContent = pumpCount;

    // console.log(pumpCount);
}
}
```

Hàm `sendMqttRequest(command)` thực hiện việc gửi một yêu cầu MQTT dựa trên giá trị `command`. Đầu tiên, hàm này tạo một URL từ `command` và sau đó sử dụng `fetch` để gửi một yêu cầu HTTP GET đến URL đó. Sau khi nhận được phản hồi từ máy chủ MQTT, nó kiểm tra xem yêu cầu có thành công không và ghi log kết quả.

Nếu phản hồi từ máy chủ MQTT cho thấy lỗi, hàm này sẽ hiển thị thông báo lỗi. Ngược lại, nó sẽ lấy dữ liệu trong phản hồi dưới dạng văn bản.

Tiếp theo, hàm sẽ in thông báo lên màn hình để cho biết lệnh MQTT đã được gửi thành công và cung cấp giá trị của tham số `command`. Nội dung của phản hồi từ máy chủ MQTT cũng sẽ được in ra màn hình để theo dõi dữ liệu trả về từ máy chủ.

Nếu có bất kỳ lỗi nào xảy ra trong quá trình gửi yêu cầu hoặc xử lý phản hồi, hàm xử lý lỗi sẽ được gọi để in ra màn hình thông báo lỗi và hiển thị thông tin về lỗi đó.

```
function sendMqttRequest(command) {
  const url = `https://mqtt.baonamdt.com/cmd/SOILSENSOR/command/${command}?p=iotdht1`;

  fetch(url)
    .then((response) => {
      if (!response.ok) {
        throw new Error("Network response was not ok");
      }
      return response.text();
    })
    .then((data) => {
      console.log(`Command sent: ${command}`);
      console.log("Response:", data);
    })
    .catch((error) => {
      console.error("Error sending command:", error);
    });
}
```

Thiết lập hàm setInterval() để thực hiện yêu cầu MQTT, cập nhật giá trị độ ẩm và nhiệt độ từ máy chủ MQTT mỗi giây.

Đầu tiên, chúng ta sử dụng hàm fetch để gửi một yêu cầu GET đến một URL cụ thể để lấy dữ liệu từ máy chủ. Sau khi nhận được phản hồi, chúng ta xử lý dữ liệu JSON từ phản hồi và chuyển đổi nó thành một đối tượng JavaScript.

Khi có dữ liệu, chúng ta trích xuất giá trị "value" từ đối tượng này và chia nó thành một mảng các phần tử bằng cách sử dụng dấu chấm phẩy (;) để tách chúng ra.

Tiếp theo, chúng ta kiểm tra độ dài của mảng dữ liệu bằng y.data.data.length và cập nhật nội dung cho phần tử có id là "pumpCount" trên trang web bằng số lượng phần tử trong mảng.

Sau đó, chúng ta kiểm tra giá trị tại vị trí thứ hai trong mảng data[1]. Nếu giá trị này là "nan", thì chúng ta hiển thị "Không Xác Định" trong phần tử có id là "nhietdo" trên trang web. Ngược lại, nếu giá trị không phải "nan", chúng ta hiển thị giá trị nhiệt độ từ data[1] cùng với đơn vị đo nhiệt độ.

Tương tự, chúng ta xác định độ ẩm bằng cách kiểm tra giá trị tại vị trí thứ nhất trong chuỗi giá trị nhận được. Nếu giá trị là 1, thì đó là độ ẩm thấp; nếu giá trị là 0, thì đó là độ ẩm cao.

```

setInterval(() => {
  fetch(
    "https://mqtt.baonamdts.com/?ps=10&topic=SOILSENSOR&token=a3519714-bc3c-4a43-abe0-792c4d5ce2a4"
  )
    .then((x) => x.json())
    .then((y) => {
      const data = y.data.data[0].value.split(";");

      document.getElementById("pumpCount").textContent = y.data.data.length;

      if (data[1] === "nan") {
        document.getElementById("nhietdo").innerHTML = `Không Xác Định`;
      } else {
        document.getElementById("nhietdo").innerHTML = `${data[1]} °C`;
      }

      if (data[0] === "1") {
        document.getElementById("doam").innerHTML = `Thấp`;
      } else {
        document.getElementById("doam").innerHTML = `Cao`;
      }
    });
}, 1000);
</script>
</body>
</html>

```

Trang web này hiển thị thông tin về độ ẩm và nhiệt độ và cho phép người dùng bật/tắt máy bơm thông qua giao diện người dùng. Dữ liệu độ ẩm và nhiệt độ được cập nhật từ máy chủ MQTT và hiển thị động trên trang web.