TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA ĐIỆN ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG



ĐỒ ÁN MÔN HỌC 1

**HỆ THỐNG GIÁM SÁT VÀ ĐIỀU KHIỂN CHĂM SÓC VƯỜN RAU IOT**

**NGÀNH CÔNG NGHỆ KỸ THUẬT MÁY TÍNH**

Sinh viên: **NGÔ THANH TÂN**

MSSV: 22119227

**TRỊNH THIẾT TRÌNH**

MSSV: 22119243

Hướng dẫn**: PGS.TS PHAN VĂN CA**

TP. HỒ CHÍ MINH – 06/2025

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA ĐIỆN ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

ĐỒ ÁN MÔN HỌC 1

**HỆ THỐNG CHĂM SÓC VƯỜN RAU**

**THÔNG MINH**

**NGÀNH CÔNG NGHỆ KỸ THUẬT MÁY TÍNH**

Sinh viên: **NGÔ THANH TÂN**

MSSV: 22119227

**TRỊNH THIẾT TRÌNH**

MSSV: 22119243

Hướng dẫn**: PGS.TS PHAN VĂN CA**

TP. HỒ CHÍ MINH – 06/2025

**BẢN NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

**….………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………**

**MỤC LỤC**

[LỜI CẢM ƠN 1](#_Toc12473)

[LỜI MỞ ĐẦU 2](#_Toc26724)

[CHƯƠNG 1 GIỚI THIỆU 4](#_Toc31121)

[1.1 Mục tiêu đề tài 4](#_Toc31276)

[1.2 Phạm vi giới hạn đề tài 4](#_Toc1509)

[1.3 Phương pháp nghiên cứu 4](#_Toc31819)

[1.4 Đối tượng và phạm vi nghiên cứu 5](#_Toc17882)

[1.4.1 Đối tượng nghiên cứu 5](#_Toc13977)

[1.4.2 Phạm vi nghiên cứu 6](#_Toc16646)

[1.5 Bố cục quyển báo cáo 7](#_Toc31984)

[CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 8](#_Toc9234)

[2.1 Tổng quan ESP32 8](#_Toc6334)

[2.2.1 Giới thiệu 8](#_Toc19569)

[2.2.2 Đặc điểm 9](#_Toc19555)

[2.2 Kiến trúc ESP32 11](#_Toc19159)

[2.2.1 Thông số kĩ thuật cơ bản 11](#_Toc9810)

[2.2.2 Chức năng tích hợp 12](#_Toc23117)

[2.2.3 Chức năng các chân 13](#_Toc31008)

[2.3 Thiết bị ngoại vi và giao thức 15](#_Toc18686)

[2.3.1 Wifi 15](#_Toc31220)

[2.3.2 GPIO (General Purpose Input/Output Interface) 15](#_Toc9100)

[2.3.3 ADC (Analog to Digital Converter) 15](#_Toc22213)

[2.3.4 DAC (Digital to Analog Converter) 16](#_Toc15562)

[2.3.5 UART (Universal Asynchronous Receiver Transmitter) 16](#_Toc7324)

[2.3.6 I2C Interface (Inter-Integrated Circuit) 16](#_Toc21681)

[2.4 App Blynk 19](#_Toc15938)

[2.4.1 Cấu trúc của Blynk 19](#_Toc17110)

[2.4.2. Virtual Pins (Chân ảo) 20](#_Toc19767)

[2.4.3. Hỗ trợ phần cứng 20](#_Toc4587)

[2.4.4. Giao tiếp và bảo mật 21](#_Toc21432)

[CHƯƠNG 3 THIẾT KẾ HỆ THỐNG 22](#_Toc27722)

[3.1 Định nghĩa hệ thống 22](#_Toc8088)

[3.2 Đặc tả kỹ thuật 22](#_Toc2512)

[3.2.1 Chức năng kỹ thuật 22](#_Toc3380)

[3.2.2 Thông số kỹ thuật 23](#_Toc4495)

[3.2.2.1 Thông số phần cứng 23](#_Toc6870)

[3.2.2.2 Kích thước, vật liệu và khối lượng mô hình 23](#_Toc13863)

[3.3 Thiết kế kiến trúc 23](#_Toc4148)

[3.3.1 Sơ đồ kiến trúc tổng quan 23](#_Toc18880)

[3.3.2 Mô tả các thành phần chính 24](#_Toc8949)

[3.4 Thiết kế chi tiết 25](#_Toc12027)

[3.4.1 Khối nguồn 25](#_Toc2151)

[3.4.2 Khối cảm biến đo dữ liệu môi trường 25](#_Toc9667)

[3.4.2.1 Cảm biến nhiệt độ, độ ẩm: 25](#_Toc13294)

[3.4.2.2 Cảm biến độ ẩm đất 26](#_Toc23477)

[3.4.2.3 Cảm biến chất lượng không khí 26](#_Toc14141)

[3.4.2.4 Cảm biến mưa 27](#_Toc24528)

[3.4.2.5 Cảm biến ánh sáng 28](#_Toc24447)

[3.4.2.6 Danh sách các cảm biến phù hợp với hệ thống 29](#_Toc23245)

[3.4.2.7 Danh sách tất cả các linh kiện có trong hệ thống 30](#_Toc18371)

[3.4.3 Khối xử lí dữ liệu 31](#_Toc14960)

[3.4.4 Khối điều khiển từ xa 32](#_Toc29480)

[3.4.5 Khối truyền nhận dữ liệu 33](#_Toc7799)

[3.4.6 Khối điều khiển động cơ 34](#_Toc17690)

[3.4.7 Hiển thị dữ liệu 35](#_Toc7346)

[3.5 Thiết kế phần mềm 36](#_Toc9172)

[3.6 Lưu đồ và giải thuật của hệ thống 41](#_Toc21340)

[3.6.1 Lưu đồ hệ thống 41](#_Toc27119)

[3.6.2 Giải thuật của hệ thống 42](#_Toc6882)

[3.6.2.1 Giới Thiệu 42](#_Toc15727)

[3.6.2.2 Thành Phần Hệ Thống 42](#_Toc16434)

[3.6.2.3 Chế Độ Hoạt Động 43](#_Toc25621)

[3.6.2.4 Phân Tích Các Task (Nhiệm Vụ) 43](#_Toc31296)

[3.6.2.5. Giải Thuật Điều Khiển 44](#_Toc21713)

[3.6.2.6. Tích Hợp Blynk 46](#_Toc6071)

[CHƯƠNG 4 KẾT QUẢ 48](#_Toc13948)

[4.1 Nhận xét 48](#_Toc16505)

[4.2 Thực nghiệm 48](#_Toc28955)

[4.2.1 Mô hình hệ thống 48](#_Toc24726)

[4.2.2 Giao diện Blynk 49](#_Toc1644)

[4.2.3 Đánh giá hệ thống 51](#_Toc3784)

[CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 52](#_Toc17417)

[5.1. Kết luận 52](#_Toc17782)

[5.2. Hướng phát triển 53](#_Toc30990)

[PHỤ LỤC 54](#_Toc9303)

[Source code 54](#_Toc18233)

[TÀI LIỆU THAM KHẢO 70](#_Toc19341)

**LỜI CẢM ƠN**

Trong quá trình thực hiện đồ án “Hệ thống chăm sóc vườn rau thông minh”, nhóm nghiên cứu xin bày tỏ lòng biết ơn sâu sắc đến:

PGS.TS. Phan Văn Ca – Giảng viên hướng dẫn, người đã tận tình hỗ trợ, định hướng phương pháp nghiên cứu và góp ý chi tiết giúp nhóm hoàn thiện ý tưởng, tối ưu thuật toán và nâng cao chất lượng hệ thống.

Gia đình và bạn bè – Luôn đồng hành, động viên về tinh thần và tạo điều kiện để mỗi thành viên có thể tập trung cao độ cho đồ án.

Sự giúp đỡ và đóng góp quý báu của quý thầy cô, anh chị và các bạn chính là nguồn động lực lớn để nhóm nghiieen cứu hoàn thành đồ án đúng tiến độ và đạt được kết quả tốt đẹp. Xin chân thành cảm ơn!

Tp. Hồ Chí Minh, tháng 06 năm 2025

Nhóm thực hiện

Ngô Thanh Tân – 22119227

Trịnh Thiết Trình – 22119243

**LỜI MỞ ĐẦU**

Trong thời đại khoa học công nghệ phát triển từng ngày như hiện nay, việc mọi người có thể tiếp cận với các thiết bị điện tử IoT (Internet Of Things) không còn gì mới mẻ và xa lạ trong mọi lĩnh vực đời sống như chăn nuôi, trồng trọt, thiết kế nhà cửa, kỹ thuật,… để phục vụ cho nhu cầu của mình. Nhận thấy nhu cầu đó, lĩnh vực IoT ngày càng phát triển và không ngừng nghiên cứu, phát triển, ứng dụng các công nghệ vào các thiết bị, động cơ để chúng có thể giao tiếp với nhau, điều khiển nhau thông qua các cảm biến 1 cách tự động, ngoài ra còn hỗ trợ cho con người điều khiển các thiết bị thông qua các ứng dụng chuyên dụng mà không cần phải trực tiếp điều khiển thiết bị.

Qua quan sát thực tế, việc trồng trọt của người nông dân rất vất vả vì phải đối diện với điều kiện thời tiết khắc nghiệt như hiện nay, sự biến đổi khí hậu làm nhiệt độ, độ ẩm thay đổi bất thường. Khô hạn kéo dài làm chất lượng đất canh tác không được đảm bảo tốt cho cây trồng sinh trưởng. Nắng mưa thất thường, ô nhiễm không khí cũng là nguyên nhân dẫn đến cây trồng không đảm bảo được năng suất. Đây là bái toán lớn đặt ra cho người nông dân giải quyết để cây trồng được đạt năng suất tốt đảm bảo tài chính gia đình.

Nhận ra được nỗi lo đó, nhóm nghiên cứu đã nảy sinh 1 ý tưởng để thiết kế “***Hệ thống chăm sóc vườn rau thông minh***” giúp người nông dân - đối tượng hướng đến là người nông dân trồng rau xanh - có thể theo dõi được các điều kiện thời tiết như nắng mưa, giám sát chất lượng không khí vầ độ ẩm trong đất trồng để tiện chăm sóc cho vườn rau của mình. Hệ thống ứng dụng lĩnh vực IoT vào để có thể tự động hoạt động mà không cần phụ thuộc vào sức người quá nhiều, điều này giúp giảm bớt sức lao động của người nông dân nhưng vẫn đảm bảo vườn ra được chăm sóc 1 cách cẩn thận để đảm bảo năng suất.

Đó là lý do nhóm nghiên cứu chọn đề tài cho đồ án là “***Hệ thống chăm sóc vườn ra thông minh***”.

# CHƯƠNG 1 GIỚI THIỆU

## 1.1 Mục tiêu đề tài

Đề tài “Hệ thống chăm sóc vườn rau thông minh” được đặt ra nhằm thiết kế, thi công hệ thống “Chăm sóc vườn rau thông minh” có các chức năng:

- Chăm sóc vườn rau 1 cách tự động qua việc giám sát, theo dõi các biến đổi từ môi trường qua các cảm biến.

- Điều khiển các động cơ, thiết bị có trong hệ thống để phục vụ cho việc trồng trọt như tưới tiêu, điều hòa không khí, che chắn mưa và ánh sáng mạnh.

- Có 2 chế độ hoạt động là tự động và điều khiển từ xa để điều khiển các động cơ bên trong hệ thống.

- Theo dõi và giám sát các rườn rau trên điện thoại hoặc trên máy tính.

## 1.2 Phạm vi giới hạn đề tài

Hệ thống “Chăm sóc vườn rau thông minh” cũng có các phạm vi giới hạn như:

- Nhiệt độ mà hệ thống có thể giám sát được trong vườn không quá 50oC.

- Hệ thống hoạt động ở chế độ điều khiển từ xa có thể bị ngắt nếu không đảm bảo kết nối Internet.

- Độ chính xác mà dữ liệu cảm biến đọc về có thể bị sai lệch so với thực tế.

- Dữ liệu đầu vào mà cảm biến đọc phải là dữ liệu thô.

## 1.3 Phương pháp nghiên cứu

Để thực hiện đề tài, nhóm đã sử dụng các phương pháp nghiên cứu sau:

- Phương pháp tổng hợp tài liệu lý thuyết:

+ Thu thập và nghiên cứu các tài liệu kỹ thuật liên quan đến: nguyên lý hoạt động và ứng dụng của các loại cảm biến (nhiệt độ, độ ẩm, độ ẩm đất, mưa, ánh sáng, chất lượng không khí).

+ Điều khiển thiết bị như máy bơm, quạt, servo.

+ Tổng hợp kiến thức về kiến trúc hệ thống IoT và quy trình xây dựng mô hình tự động hóa.

- Phương pháp khảo sát hệ thống thực tế:

+ Tham khảo các mô hình vườn rau thông minh đã được triển khai thực tế trên Internet, Youtube.

+ Đánh giá hiệu quả, ưu nhược điểm và khả năng ứng dụng của các cảm biến và động cơ trong hệ thống hiện có để làm cơ sở thiết kế mô hình phù hợp với điều kiện nghiên cứu.

- Phương pháp mô phỏng và thực nghiệm:

+ Lắp ráp hệ thống thực tế trên mô hình nhỏ (thực nghiệm), tiến hành kiểm tra chức năng các cảm biến và thiết bị điều khiển.

+ Nạp và kiểm tra code để hệ thống hoạt động.

+ Ghi nhận và đánh giá khả năng hoạt động tự động của hệ thống trong các điều kiện khác nhau.

- Phương pháp tham khảo ý kiến chuyên gia

+ Tham khảo ý kiến từ giảng viên, các video trên Youtube, trên Internet để hoàn thiện thiết kế, chọn giải pháp phù hợp và đảm bảo tính thực tiễn.

## 1.4 Đối tượng và phạm vi nghiên cứu

**1.4.1 Đối tượng nghiên cứu:** việc thu thập dữ liệu từ cảm biến và điều khiển thiết bị tự động để chăm sóc vườn rau một cách thông minh, tối ưu.

- Hệ thống IoT (Internet of Things) trong nông nghiệp, cụ thể là ứng dụng vào chăm sóc vườn rau.

- Các cảm biến phục vụ việc thu thập dữ liệu môi trường như:

+ Cảm biến nhiệt độ, độ ẩm.

+ Cảm biến độ ẩm đất.

+ Cảm biến mưa.

+ Cảm biến chất lượng không khí.

+ Cảm biến ánh sáng.

- Các thiết bị điều khiển tự động như:

+Máy bơm nước.

+ Quạt thông gió.

+ Servo (dùng để điều khiển mái che).

**1.4.2 Phạm vi nghiên cứu:** giới hạn trong việc thiết kế và triển khai một hệ thống chăm sóc vườn rau quy mô nhỏ sử dụng IoT, tích hợp cảm biến và thiết bị điều khiển, có khả năng hoạt động tự động và thông minh.

- Phạm vi kỹ thuật phần cứng:

+ Lắp đặt và tích hợp các loại cảm biến nói trên.

+ Kết nối các cảm biến với vi điều khiển ESP32.

+ Điều khiển các thiết bị đầu ra (máy bơm, quạt, servo) qua tín hiệu điều khiển tự động dựa trên dữ liệu cảm biến.

- Phạm vi kỹ thuật phần mềm:

+ Viết chương trình xử lý dữ liệu cảm biến và ra quyết định điều khiển thiết bị.

+ Điều khiển từ xa qua App Blynk.

+ Giao diện hiển thị thông tin: dashboard theo dõi nhiệt độ, độ ẩm, độ ẩm đất, chất lượng không khí và điều khiển các động cơ.

- Phạm vi ứng dụng thực tế:

+ Mô hình hoặc khu vườn rau quy mô nhỏ (vườn hộ gia đình, mô hình trồng rau sạch trong nhà).

+ Chăm sóc tự động trong các điều kiện thời tiết khác nhau (nắng, mưa, nhiệt độ cao/thấp, đất khô/ướt, chất lượng không khí tốt/xấu).

+ Chăm sóc thủ công qua điều khiển từ xa trên điện thoại/máy tính.

## 1.5 Bố cục quyển báo cáo

- Chương 1. GIỚI THIỆU: trình bày mục tiêu đề tài, phạm vi giới hạn đề tài, phương pháp nghiên cứu, đối tượng và phạm vi nghiên cứu nhằm xây dựng hệ thống chăm sóc vườn rau thông minh ứng dụng IoT.

- Chương 2: CƠ SỞ LÝ THUYẾT: tổng hợp các kiến thức nền tảng về IoT, nguyên lý hoạt động của các loại cảm biến và thiết bị điều khiển, cũng như các công nghệ liên quan phục vụ cho việc thiết kế hệ thống.

- Chương 3: THIẾT KẾ HỆ THỐNG: mô tả chi tiết kiến trúc phần cứng và phần mềm của hệ thống, cách kết nối các cảm biến, thiết bị điều khiển, cũng như quy trình xử lý dữ liệu và vận hành hệ thống.

- Chương 4: KẾT QUẢ: trình bày kết quả thực nghiệm của hệ thống, đánh giá hiệu quả hoạt động trong thực tế và khả năng đáp ứng nhu cầu chăm sóc vườn rau tự động.

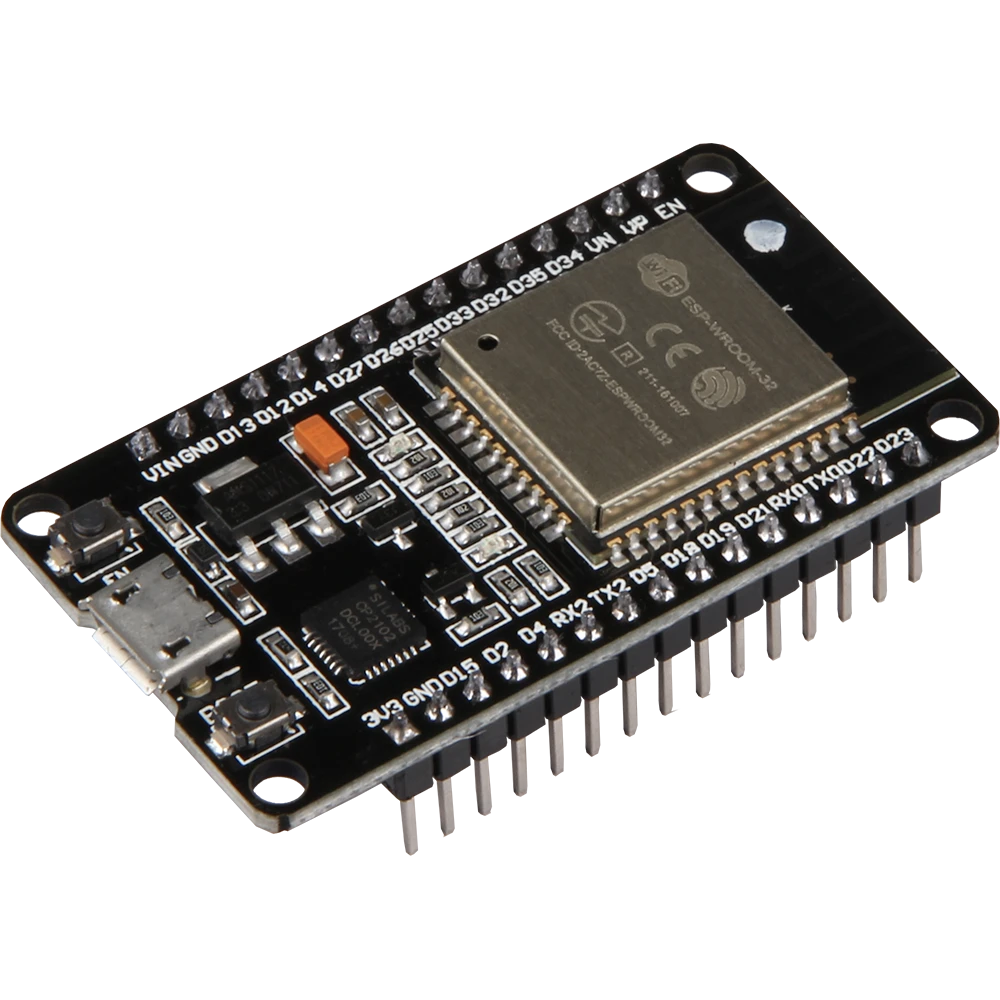
- Chương 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN: đưa ra kết luận về những gì đã đạt được, các hạn chế còn tồn tại, đồng thời đề xuất hướng phát triển và cải tiến hệ thống trong tương lai.

# CHƯƠNG 2 CƠ sở lý thuyết

## 2.1 Tổng quan ESP32

### 2.2.1 Giới thiệu

ESP32 là một chip được tích hợp công nghệ Wifi và Bluetooth với công nghệ tiêu thụ năng lượng cực thấp. Nó cung cấp một nền tảng tích hợp mạnh mẽ, đáp ứng nhu cầu hiệu năng tốt nhất, tính linh hoạt, thiết kế nhỏ gọn, hiệu suất cao và độ tin cậy trong nhiều ứng dụng. Các dòng chip ESP32 bao gồm ESP32-D0WDQ6,ESP32-D0WD,ESP32-D2WD và ESP32-S0WD. Espressif đã thiết kế và sản xuất ra một số module để người dùng dễ dàng tiếp cận hơnvới dòng chip ESP32. Các thành phần chính trên những module này bao gồm chip ESP32, bộ tạo dao động thạch anh, mạch ănten, chỉ khác nhau về một số chức năng tùy từng phiên bản như số lượng chân GPIO, các thiết bị ngoại vi được thêm vào như: màn hình LCD, bảng cảm ứng, khe cắm thẻ SD, module máy ảnh,…dưới đây là một số phiênbản đã được phát triển và đưa vào sử dụng.Phiên bản ESP32-WOOM-32 là một module vi điều khiển (MCU), Wifi( Wireless Fidelity), BT (Bluetooth), BTE (Bluetooth Low Energy) phổ biến và mạnh mẽ phục vụ cho nhiều ứng dụng khác nhau từ những ứng dụng đơn giản như điều khiển thiết bị, đọc giá trị cảm biến đến những nhiệm vụ phức tạp như mã hóa giọng nói, phát nhạc trực tuyến, giải mã MP3,…

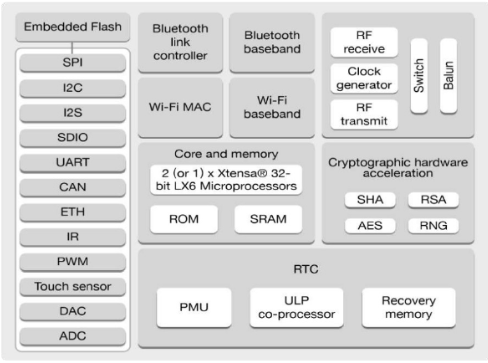


Hình 1. ESP32-WOOM-32

Module này được xây dựng với lõi nhân là chip ESP32-D0WDQ6, chip được thiết kế để có thể mở rộng. Có 2 lõi CPU (Central Processing Unit) có thể kiểm soát riêng biệt và tần số xung đồng hồ dao động từ 80MHz đến 240MHz. Do tích hợp Bluetooth, Bluetooth LE, Wifi nên rất nhiều ứng dụng có thể thực hiện được trên module. Sử dụng Wifi cho phép kết nối trong phạm vi rộng, không giới hạn, sử dụng Bluetooth cho phép kết nối một cách dễ dàng. Module cũng hổ trợ truyền dữ liệu lên đến 150Mbps.

### 2.2.2 Đặc điểm

- Kiến trúc:



Hình 2. Kiến trúc ESP32-WOOM-32

- Ứng dụng

+ Nhà thông minh

+ Chăm sóc vườn rau

+ Robot công nghiệp

+ Nhận dạng giọng nói

+ Nhận dạng hình ảnh

+ Ứng dụng chăm sóc sức khỏe

- Mức năng lượng thấp

Chip ESP32 được thiết kế cho các ứng dụng di động, điện tử, ứng dụng IoT. ESP32 có tất cả các đặc tính của một chip tiêu thụ mức năng lượng thấp, nhiều chế độ hoạt động giúp giảm thiểu tối đa năng lượng mà chip tiêu tốn. Ví dụ cảm biến trong ứng dụng IoT, ESP32 được đánh thức theo chu kỳ và chỉ phát hiện khi có một điều kiện cụ thể.

- Tích hợp hoàn chỉnh

Chip ESP32 có khả năng tích hợp cao Wifi và Bluetooth cho các ứng dụng IoT, với khoảng 20 thiết bị ngoại vi. ESP32 tích hợp ăngten, sóng RF (Radiofrequency), bộ khuếch đại công suất, bộ khuếch đại thu tiếng ồn, module quản lý năng lượng, …

## 2.2 Kiến trúc ESP32

### 2.2.1 Thông số kĩ thuật cơ bản

Bảng 1. Bảng thông số kỹ thuật ESP32-WOOM-32

|  |  |  |
| --- | --- | --- |
| Loại | Đặc trưng | Thông số |
| WIFI | Giao thức | 802.11 b/g/n (802.11n lên đến 150 Mbps) |
| Tần số | 2.4GHz ~ 2.5GHz |
| Phần cứng | Giao tiếp module | Thẻ SD, UART, SPI, SDIO, I2C, Led PWM,  Motor PWM, I2S  IR, Counter, GPIO, cảm biến cảm ứng,  ADC, DAC |
| Chip cảm biến | Hall sensor (từ trường) |
| Thạch anh | 40MHz |
| SPI Flash | 4MB |
| Điện áp hệ thống | 2.7V ~ 3.6V |
| Dòng hệ thống | Trung bình: 80mA |
| Dòng điện tối thiểu của nguồn cung cấp | 500mA |
| Nhiệt độ hoạt động | -40℃ ~ +85℃ |
| Kích thước | (18.0) mm x (25.5) mm x (3.1) mm |

### 2.2.2 Chức năng tích hợp

- CPU: Chip ESP32 là bộ xử lý lõi kép (Dual core) của vi điều khiển Xtensa® 32bit LX6 với các đặc trưng:

+ Hổ trợ xung nhịp lên đến 240MHz.

+ Hổ trợ DSP như bộ nhân 32bit, bộ chia 32bit, MAC 40bit.

+ Hổ trợ 32 vector interrupt.

- Bộ nhớ nội (Internal Memory) của ESP32 bao gồm:

+ 448KB ROM cho việc khởi động và các chức năng cốt lõi.

+ 520KB SRAM trên chip cho dữ liệu.

+ 8KB SRAM trong RTC (Real Time Control): được gọi là bộ nhớ RTC FAST.

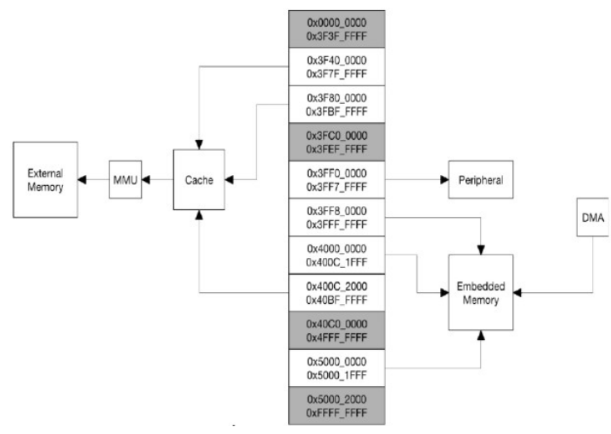
+ 8KB SRAM trong RTC: được gọi là bộ nhớ RTC LOW.

+ 1Kbit eFuse: 256bit sử dụng cho hệ thống, 768bit chuyển đổi cho ứng dụng.

- External Flash (bộ nhớ flash ngoài) và SRAM

- ESP32 hổ trợ nhiều external QSPI flash và SRAM, có thể truy cập ở tốc độ cao. SRAM được hổ trợ lên đến 8MB, đọc và ghi 8bit, 16bit, 32bit.

- Tổ chức bộ nhớ



Hình 3. Cấu trúc và địa chỉ bộ nhớ của ESP32-WOOM-32

- Hệ thống xung đồng hồ CPU Clock

+ Khi được reset, dao động thạch anh bên ngoài được đặt lại mặc định cho CPU.Dao động thạch anh kết nối với PLL (Phase-Locked Loop) để tạo ra xung tần số cao(160MHz).

+ Ngoài ra, ESP32 có bộ dao động nội 8MHz và có thể sử dụng được bộ dao động này cho một số ứng dụng nhất định.

- RTC Clock (Real Time Clock) có các nguồn hoạt động:

+ Thạch anh bên ngoài tốc độ thấp (32KHz).

+ Thạch anh bên ngoài với bộ chia 4.

+ Bộ dao động nội RC (150KHz, có thể điều chỉnh được).

+ Bộ dao động nội 8MHz.

+ Xung nội 32.25KHz.

- Khi ở chế độ bình thường, cần truy cập CPU nhanh hơn, ứng dụng có thể chọn xung tốc độ cao bên ngoài với bộ chia 4 hoặc dao động nội 8MHz. Khi hoạt động ở chế độ năng lượng thấp, ứng dụng có thể chọn thạch anh bên ngoài tốc độ thấp(32KHz), dao động nội RC hoặc xung nội 31.25KHz.

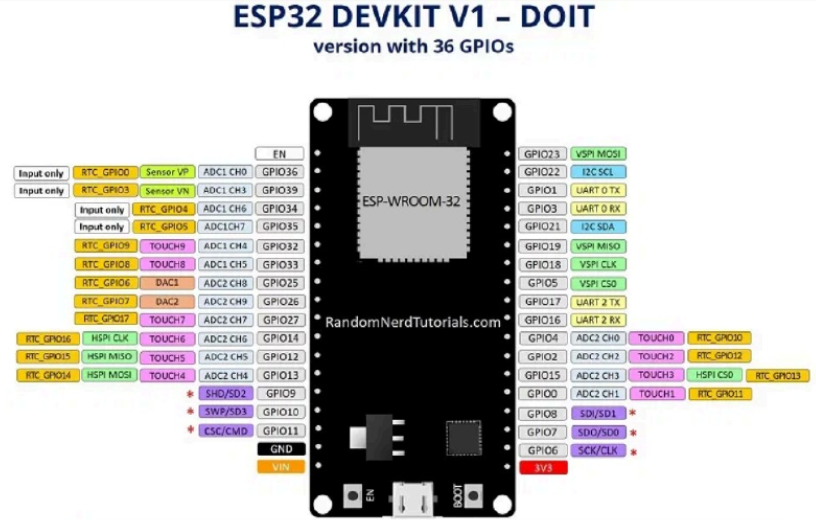
### 2.2.3 Chức năng các chân

Bảng 2. Chức năng các chân ESP32-WOOM-32 được sử dụng

|  |  |
| --- | --- |
| Chân (GPIO) | Chức năng (theo code/ứng dụng) |
| EN (CHIP\_EN) | Enable toàn bộ module ESP32; kéo HIGH để bật, kéo LOW để tắt (shutdown). |
| 3V3 | Nguồn 3.3 V cấp cho module ESP-WROOM-32. |
| GND | Mass (đất) cho module. |
| GPIO2 | LED trạng thái (HIGH = đã kết nối Blynk/Wi-Fi → LED sáng); là chân strapping khi boot (phải giữ HIGH để khởi động). |
| GPIO25 | Dữ liệu DHT11 (đọc nhiệt độ + độ ẩm). |
| GPIO36 | Đọc tín hiệu analog từ Soil Moisture Sensor (ADC1\_CH0). |
| GPIO34 | Đọc tín hiệu analog từ Rain Sensor (ADC1\_CH6). |
| GPIO35 | Đọc tín hiệu analog từ MQ-135 Air Quality Sensor (ADC1\_CH7). |
| GPIO17 | Điều khiển Relay Pump (inverted logic: LOW → bật bơm; HIGH → tắt bơm). |
| GPIO16 | Điều khiển Relay Fan (inverted logic: LOW → bật quạt; HIGH → tắt quạt). |
| GPIO18 | PWM điều khiển Servo Canopy (0° → mở hoàn toàn; 180° → đóng hoàn toàn). |
| GPIO21 (SDA) | I²C SDA cho màn hình OLED (Adafruit\_SSD1306). |
| GPIO22 (SCL) | I²C SCL cho màn hình OLED (Adafruit\_SSD1306). |

- Ngoài ra tất cả các chân GPIO đều có thể cấu hình thành ngắt (interrupt) và 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)

- Sơ đồ chân



Hình 4. Sơ đồ chân ESP32-WOOM-32

## 2.3 Thiết bị ngoại vi và giao thức

### 2.3.1 Wifi

ESP32 thực hiện giao thức TCP/IP và Wifi 802.11 b/g/n, quản lý năng lượng được xử lý để giảm thiểu thời gian thực hiện các tác vụ. Các thư viện của Wifi cung cấp để định cấu hình và giám sát chức năng kết nối mạng ESP32, nó cấu hình cho:

- Chế độ trạm (hay chế độ STA hoặc chế độ Wifi client): ESP32 kết nối với một điểm truy cập.

- Chế độ AP (hay chế độ Soft-AP hoặc chế độ điểm truy cập): các trạm kết nối với ESP32.

- Chế độ kết hợp AP-STA (ESP32 đồng thời là điểm truy cập và là trạm được kết nối với các điểm truy cập khác).

- Các chế độ bảo mật khác nhau cho các chế độ trên (WPA, WPA2, WEP, …).

- Quét các điểm truy cập (chủ động và thụ động).

- Chế độ giám sát các gói Wifi của tiêu chuẩn IEEE802.11.

### 2.3.2 GPIO (General Purpose Input/Output Interface)

ESP32 với số lượng lớn chân GPIO tùy thuộc vào từng phiên bản module, mỗi chân được tích hợp nhiều chức năng khác nhau bằng cách lập trình thanh ghi. Có một số loại chân GPIO như: chỉ đọc dữ liệu số (0 hoặc 1), cho phép analog, cho phép cảm ứng điện dung,… Hầu hết các chân GPIO số (digital) có thể được cấu hình điện trở kéo lên (pull- up) hoặc kéo xuống (pull-down) hoặc trở kháng cao. Mỗi chân có thể có nhiều chức năng khác nhau như SDIO, UART, SPI, …

### 2.3.3 ADC (Analog to Digital Converter)

ESP32 tích hợp ADC 12bit và hổ trợ đo 18 kênh. Với thiết lập thích hợp, các ADC có thể được cấu hình để đo điện áp tối đa 18 chân.

### 2.3.4 DAC (Digital to Analog Converter)

ESP32 có hai kênh DAC 8bit được sử dụng để chuyển đổi tín hiệu số sang tín hiệu analog, được tích hợp điện trở và bộ đệm.

### 2.3.5 UART (Universal Asynchronous Receiver Transmitter)

UART truyền lấy byte dữ liệu và truyền các bit ở dạng tuần tự. Máy phát thứ hai là máy thu tập hợp lại các bit thành một byte hoàn chỉnh. Truyền dữ liệu nối tiếp qua một dây thực sự tiết kiệm chi phí hơn so với truyền song song qua nhiều dây.

ESP32 tích hợp ba UART: UART1, UART2, UART3 cung cấp chuẩn giao tiếp bất đồng bộ (RS232 và RS485) với tốc độ lên đến 5Mbps, có thể truy cập bởi bộ điều khiển DMA hoặc trực tiếp bởi CPU.

Cấu trúc của giao thức UART:

- Một UART chứa một máy phát điện đồng hồ. Điều này cho phép lấy mẫu trong một khoảng thời gian ngắn.

- Nó cũng chứa các thanh ghi cả đầu vào và đầu ra.

- Điều khiển truyền và nhận.

- Một logic điều khiển đọc hoặc ghi.

- Các thành phần tùy chọn khác của UART là: truyền hoặc nhận bộ đệm, bộ nhớ đệm FIFO và bộ điều khiển DMA.

### 2.3.6 I2C Interface (Inter-Integrated Circuit)

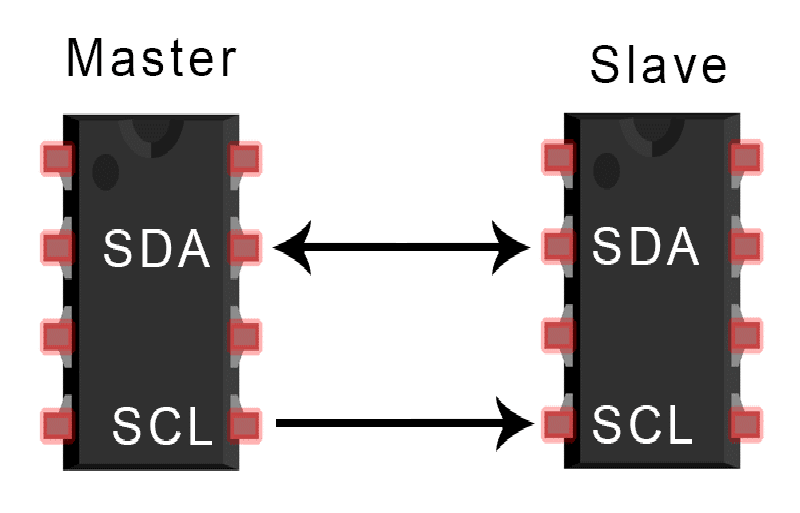
ESP32 có hai giao tiếp I2C, có thể vận hành ở chế độ master hoặc slave phụ thuộc vào người sử dụng cấu hình, nó hổ trợ:

- Chế độ tiêu chuẩn (100Kbit/s).

- Chế độ nhanh (400Kbit/s).

- Tốc độ lên đến 5MHz.

- Chế độ kép



Hình 5. Giao thức I2C

Giống như giao tiếp UART, I2C chỉ sử dụng hai dây để truyền dữ liệu giữa các thiết bị:

- SDA (Serial Data) - đường truyền cho master và slave để gửi và nhận dữ liệu.

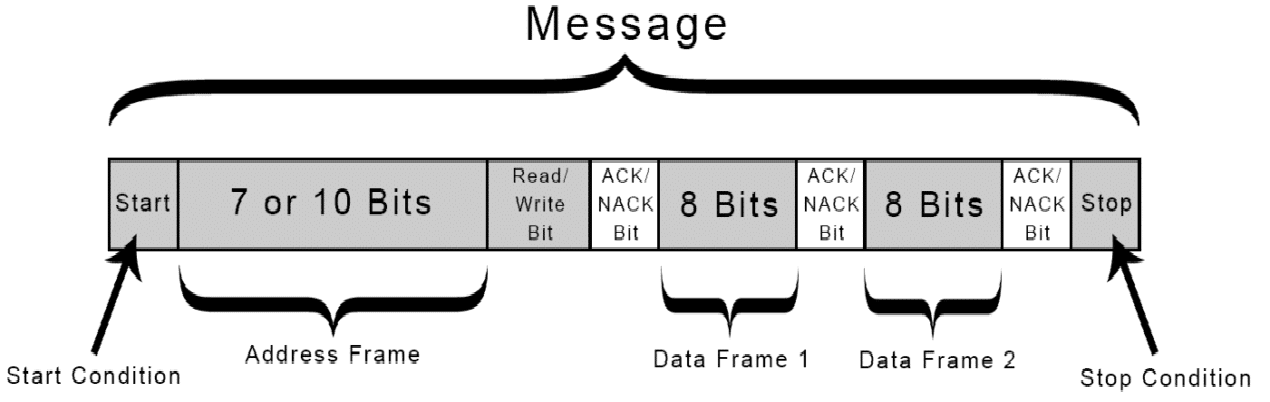
- SCL (Serial Clock) - đường mang tín hiệu xung nhịp.

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

Giống như SPI, I2C là đồng bộ, do đó đầu ra của các bit được đồng bộ hóa với việc lấy mẫu các bit bởi một tín hiệu xung nhịp được chia sẻ giữa master và slave. Tín hiệu xung nhịp luôn được điều khiển bởi master.

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 6. Khung Frame của 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: 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.

- 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 Slave Select 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.

## 2.4 App Blynk

Blynk là một nền tảng IoT (Internet of Things) được thiết kế để hỗ trợ người dùng kết nối, điều khiển và giám sát các thiết bị từ xa thông qua ứng dụng di động hoặc web. Với mục tiêu đơn giản hóa quá trình phát triển IoT, Blynk phù hợp cho cả người mới bắt đầu lẫn các nhà phát triển chuyên nghiệp, không đòi hỏi kiến thức lập trình sâu rộng. Dưới đây là những khái niệm cơ bản về lý thuyết của Blynk:

**2.4.1 Cấu trúc của Blynk**

Hệ sinh thái Blynk bao gồm ba thành phần chính, hoạt động phối hợp để tạo ra một giải pháp IoT hoàn chỉnh:

- Blynk App: đây là ứng dụng di động (hỗ trợ iOS và Android) cung cấp giao diện người dùng để tạo các bảng điều khiển (dashboard) tùy chỉnh. Người dùng có thể sử dụng các widget (như nút bấm, thanh trượt, biểu đồ) thông qua cơ chế kéo-thả mà không cần viết mã. Ứng dụng này đóng vai trò là cầu nối trực tiếp giữa người dùng và thiết bị.

- Blynk Server: máy chủ trung gian chịu trách nhiệm xử lý giao tiếp giữa ứng dụng và phần cứng. Blynk cung cấp Blynk Cloud (máy chủ đám mây miễn phí) để người dùng dễ dàng sử dụng, đồng thời hỗ trợ tùy chọn thiết lập máy chủ cục bộ (local server) nhằm tăng cường kiểm soát và bảo mật. Server đảm nhận các nhiệm vụ như xác thực (authentication), lưu trữ dữ liệu và chuyển tiếp lệnh.

- Blynk Library: thư viện mã nguồn mở được cài đặt trên phần cứng (ví dụ: Arduino, ESP8266/ESP32). Thư viện này cho phép thiết bị giao tiếp với Blynk Server, xử lý các lệnh từ ứng dụng và gửi dữ liệu (như giá trị cảm biến) ngược lại server để hiển thị trên app.

- Cách hoạt động: khi người dùng nhấn nút trên ứng dụng, lệnh được gửi đến Blynk Server. Server sau đó chuyển tiếp lệnh này đến phần cứng thông qua Blynk Library. Ngược lại, dữ liệu từ phần cứng được gửi về server và cập nhật trên ứng dụng theo thời gian thực.

**2.4.2. Virtual Pins (Chân ảo)**

Một khái niệm cốt lõi của Blynk là Virtual Pins – các kênh dữ liệu trừu tượng giúp trao đổi thông tin giữa ứng dụng và phần cứng mà không cần can thiệp trực tiếp vào các chân vật lý của thiết bị. Điều này mang lại sự linh hoạt và đơn giản trong quá trình phát triển.

Ví dụ về điều khiển: một widget nút bấm trên ứng dụng được liên kết với Virtual Pin V1. Khi nút được nhấn, giá trị (1 hoặc 0) được gửi đến V1. Phần cứng đọc giá trị này và thực hiện hành động tương ứng (như bật/tắt đèn LED).

Ví dụ về hiển thị dữ liệu: phần cứng ghi giá trị từ cảm biến (như nhiệt độ) vào Virtual Pin V2. Ứng dụng sử dụng widget biểu đồ để hiển thị dữ liệu từ V2 theo thời gian thực.

**2.4.3. Hỗ trợ phần cứng**

Blynk là một nền tảng hardware-agnostic (không phụ thuộc phần cứng cụ thể), hỗ trợ nhiều thiết bị IoT phổ biến, bao gồm:

- Arduino

- Raspberry Pi

- ESP8266/ESP32

- Particle

- Và nhiều loại khác

Người dùng có thể kết nối thiết bị qua Wi-Fi, Ethernet, hoặc các phương thức khác, tùy thuộc vào phần cứng sử dụng.

**2.4.4. Giao tiếp và bảo mật**

Blynk sử dụng các giao thức giao tiếp như TCP/IP, HTTP, hoặc MQTT để truyền dữ liệu giữa ứng dụng, server và phần cứng. Tuy nhiên, chi tiết giao thức được Blynk Library xử lý, giúp người dùng tập trung vào phát triển thay vì cấu hình mạng. Về bảo mật, Blynk cung cấp:

- Mã hóa dữ liệu đầu cuối.

- Giám sát sự cố 24/7 (trên Blynk Cloud).

# CHƯƠNG 3 THIẾT KẾ HỆ THỐNG

## 3.1 Định nghĩa hệ thống

- Chăm sóc vườn rau bằng cách đo nhiệt độ, độ ẩm, độ ẩm đất, thông báo có mưa từ môi trường.

- Phản ứng nhanh với các thông tin đo được từ môi trường để chăm sóc vường rau.

- Giám sát và điều khiển các động cơ bên trong hệ thống để chăm sóc cho vườn rau.

- Có 2 chế độ điều khiển hệ thống là tự động và điều khiển từ xa.

- Tính năng của hệ thống:

+ Có mái che để che nắng, mưa để bảo vệ phía trên vườn rau.

+ Có máy bơm nước để bơm nước cung cấp nước cho đất trồng khi đất bị khô.

+ Có quạt để điều hòa không khí trong vườn khi không khí trong vườn có chất lượng không tốt.

+ Hiển thị thông tin nhiệt độ, độ ẩm, độ ẩm đất,…

## 3.2 Đặc tả kỹ thuật

**3.2.1 Chức năng kỹ thuật**

- Thu thập dữ liệu từ môi trường như nhiệt độ, độ ẩm, độ ẩm đất, chất lượng CO2 trong không khí, cường độ ánh sáng và mưa từ môi trường.

- Truyền nhận dữ liệu.

- Xử lý dữ liệu.

- Hiển thị dữ liệu.

- Điều khiển các động cơ như quạt, máy bơm, servo.

- Điều khiển từ xa.

**3.2.2 Thông số kỹ thuật**

**3.2.2.1 Thông số phần cứng**

- Điện áp hoạt động: 3.3VDC đến 5VDC.

- Công suất tiêu thụ: 5W.

- Độ chính xác của cảm biến đo nhiệt độ: 土 2oC, tầm hoạt động từ 0oC đến 50oC.

- Độ chính xác của cảm biến đo độ ẩm 土 5RH, tầm hoạt động từ 20% đến 80%RH.

- Độ chính xác của cảm biến độ ẩm đất: ± 3% trong khoảng 0 – 53% và ± 5% trong khoảng 53 – 100%.

**3.2.2.2 Kích thước, vật liệu và khối lượng mô hình**

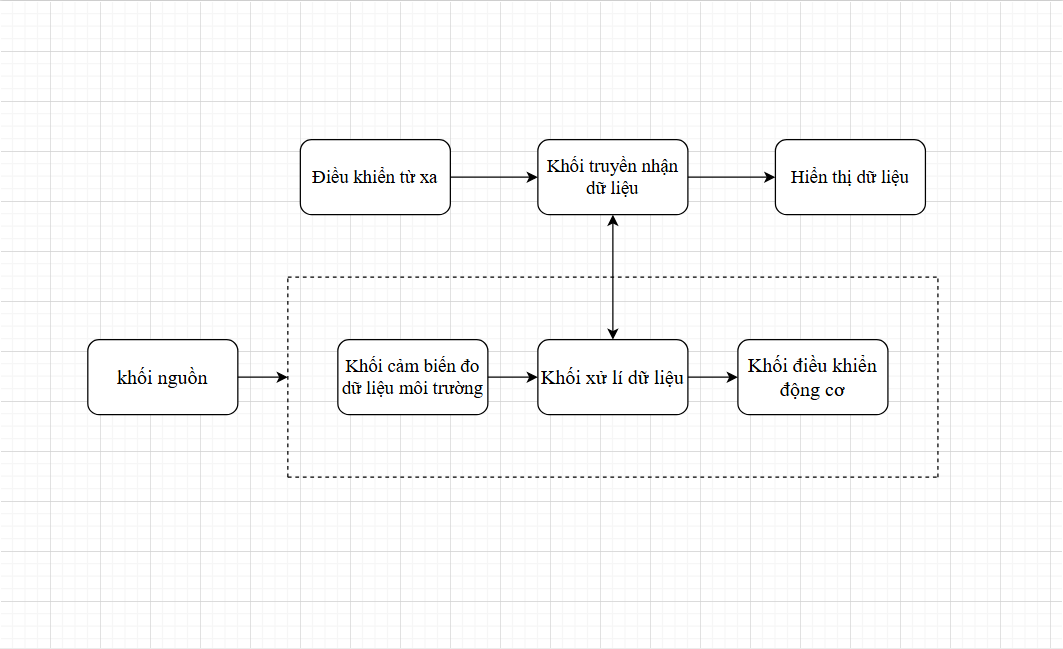
- Kích thước của mô hình hệ thống: 40x20x10 cm.

- Vật liệu: bìa cứng.

- Khối lượng: khoảng 1 kg.

## 3.3 Thiết kế kiến trúc

**3.3.1 Sơ đồ kiến trúc tổng quan**



Hình 7. Sơ đồ khối của hệ thống

**3.3.2 Mô tả các thành phần chính**

*-* Khối nguồn: cung cấp nguồn cho toàn hệ thống hoạt động.

- Khối cảm biến đo dữ liệu môi trường: thu nhập dữ liệu nhiệt độ, độ ẩm, độ ẩm đất, chất lượng không khí(cụ thể là CO2), mưa và cường độ ánh sáng, sau đó gửi đến khối xử lí dữ liệu.

- Khối điều khiển từ xa: điều khiển các động cơ trong hệ thống bằng điện thoại.

- Khối xử lí dữ liệu: xử lí dữ liệu là các giá trị cảm biến bằng các phép tính toán, biến đổi các dữ liệu được khối cảm biến đo dữ liệu môi trường gửi đến, so sánh các giá trị cảm biến với các ngưỡng được cài đặt trước trong hệ thống để điều khiển các động cơ ở khối điều khiển động cơ, đồng thời khối này còn gửi dữ liệu đến khối truyền nhận dữ liệu.

- Khối điều khiển động cơ: nhận tín hiệu điều khiển từ khối xử lí dữ liệu để điều khiển các động cơ trong hệ thống.

- Khối truyền nhận dữ liệu: nhận dữ liệu từ khối điều khiển từ xa và khối cảm biến đo dữ liệu môi trường và gửi dữ liệu đến khối xử lí dữ liệu, sau khi xử lí xong có thể nhận dữ liệu từ khối xử lí dữ liệu truyền đến khối hiển thị dữ liệu để hiển thị.

- Khối hiển thị dữ liệu: hiển thị các giá trị cảm biến như nhiệt độ, độ ẩm, độ ẩm đất,… trên màn hình OLED 0.96 inch, điện thoại.

## 3.4 Thiết kế chi tiết

### **3.4.1 Khối nguồn**

Bảng 3. So sánh các loại pin

|  |  |  |  |
| --- | --- | --- | --- |
| Đặc điểm | Pin 18650 | Pin 26650 | Pin 14500 |
| Dung lượng | ~2200-3500mAh | ~3000-5000mAh | ~600-1000mAh |
| Nguồn cung cấp | 3.7V | 3.7V | 3.7V |

- Chọn Pin 18650 vì:

+ Có dung lượng tốt phù hợp cho hệ thống (hoạt động trong khoảng 2-5h).

+ Giá thành hợp lí.

**3.4.2 Khối cảm biến đo dữ liệu môi trường**

**3.4.2.1 Cảm biến nhiệt độ, độ ẩm:**

Để thu thập dữ liệu nhiệt độ, độ ẩm của môi trường, ta có các loại cảm biến nhiệt độ, độ ẩm như: DHT11, DHT22… Mỗi loại cảm biến có các thông số như bảng dưới:

Bảng 4. Thông số của cảm biến nhiệt độ DHT11, DHT22

|  |  |  |
| --- | --- | --- |
| Cảm biến | DHT11 | DHT22 |
| Đo lường | Nhiệt độ  Độ ẩm | Nhiệt độ  Độ ẩm |
| Giao thức truyền thông | one-wire | one-wire |
| Điện áp nguồn | 3.3V đến 5V DC | 3 đến 6V DC |
| Khoảng nhiệt độ | 0 đến 50ºC | -40 đến 80ºC |
| Độ chính xác | +/- 2ºC (ở 0 đến 50ºC) | +/- 0,5ºC (ở -40 đến 80ºC) |
| Hỗ trợ (Arduino IDE) | Thư viện DHT Adafruit  Thư viện cảm biến hợp nhất Adafruit | Thư viện DHT Adafruit  Thư viện cảm biến hợp nhất Adafruit |

Từ bảng thông số trên, lựa chọn sử dụng cảm biến DHT11 để thu thập dữ liệu nhiệt độ, độ ẩm từ môi trường. Vì trên thực tế để áp dụng đo nhiệt độ, độ ẩm của vườn rau thì nhiệt độ không thể để vượt quá 500C được nên chọn DHT11 thay vì DHT22 là hợp lý.

**3.4.2.2 Cảm biến độ ẩm đất**

Cảm biến độ ẩm đất Soil Moisture Sensor V1.2 có thiết kế đơn giản, dễ sử dụng, đặc biệt phần điện cực cắm xuống đất được phủ lớp sơn chống ăn mòn nên cảm biến có độ bền và độ ổn định cao hơn các loại mạ kẽm thông thường, cảm biến trả giá trị độ ẩm đất tương ứng với điện áp qua chân Analog, thích hợp cho các ứng dụng tưới cây tự động, vườn thông minh.

**3.4.2.3 Cảm biến chất lượng không khí**

Để thu thập dữ liệu độ ẩm đất của đất trồng, ta có các loại cảm biến như: MQ-135, CCS811… Mỗi loại cảm biến có các thông số như bảng dưới:

Bảng 5. Thông số của cảm biến độ ẩm đất MQ-135, CCS811

|  |  |  |
| --- | --- | --- |
| Loại cảm biến | MQ-135 | CCS811 |
| Thông số đo | CO2, NH3, NOx, Alcohol, Benzene | eCO2, TVOC |
| Giao thức truyền thông | Analog | I2C |
| Điện áp hoạt động | 5V | 3.3V |
| Chất lượng đo (ppm) | 10 – 1000+ | 400 – 8192 eCO2 |
| Độ chính xác | ±10% ~ ±20% | ±5% |
| Hỗ trợ Arduino IDE | Có thư viện | Có thư viện |

Từ bảng thông số trên, lựa chọn sử dụng cảm biến MQ-135 để thu thập dữ liệu độ ẩm đất của đất trồng. Vì sử dụng đọc dữ liệu thu thập về là analog sẽ dễ tính toán hơn, giá thành rẻ hơn.

**3.4.2.4 Cảm biến mưa**

Để thu thập dữ liệu thời tiết có mưa hay không, ta có các loại cảm biến như: FC-37/YL-83, RG11… Mỗi loại cảm biến có các thông số như bảng dưới:

Bảng 6. Thông số của cảm biến mưa FC-37/YL-83, RG11

|  |  |  |
| --- | --- | --- |
| Loại cảm biến | FC-37/YL-83 | RG11 |
| Nguyên lý hoạt động | Phát hiện mưa bằng điện trở bề mặt | Dùng hồng ngoại – đo chính xác |
| Giao thức truyền thông | Analog + digital | Digital / Serial |
| Điện áp hoạt động | 3.3V đến 5V | 5V – 12V |
| Tín hiệu đầu ra | A0(mức mưa), D0(ngưỡng) | Pulse / Serial |
| Độ chính xác | Thấp – chỉ phát hiện có mưa | Cao (tự động hiệu chuẩn) |
| Hỗ trợ Arduino IDE | Có thư viện | Không có thư viện sẵn |

Từ bảng thông số trên, lựa chọn sử dụng cảm biến FC-37/YL-83 để thu thập dữ liệu thời tiết có mưa hay không. Vì dùng cảm biến này sử dụng điện trở bề mặt sẽ tiện lợi hơn dùng hồng ngoại để phát hiện, ngoài ra mức điện áp hoạt động cũng thấp, phù hợp với mô hình hệ thống, giá thành rẻ hơn so với RG11 và đặc biệt FC-37/YL-83 có hỗ trợ thư viện cho Arduino IDE.

**3.4.2.5 Cảm biến ánh sáng**

Bảng 7. Thông số của các cảm biến ánh sáng

|  |  |  |  |
| --- | --- | --- | --- |
| Tiêu chí | **BH1750** | **TSL2561** | **TEMT6000** |
| **Điện áp hoạt động** | 3.0V – 5.0V | 3.0V – 5.0V | 3.0V – 5.0V |
| **Nguyên lý hoạt động** | Photodiode đo ánh sáng, tích hợp ADC và I2C | 2 photodiode lọc IR & visible light, có ADC và I2C | Phototransistor chuyển ánh sáng thành dòng |
| **Tín hiệu thu được** | **Digital (I2C)** – giá trị lux trực tiếp | **Digital (I2C)** – lux (cần xử lý thêm) | **Analog** – điện áp tỉ lệ cường độ sáng |
| **Thư viện trên ESP32** | Có sẵn, dễ dùng (BH1750.h) | Có sẵn (Adafruit\_TSL2561\_U.h) nhưng phức tạp hơn | Không cần thư viện, dùng ADC trực tiếp |

Từ bảng so sánh thông số, lựa chọn sử dụng cảm biến BH1750 để đo cường độ ánh sáng trong hệ thống là hoàn toàn hợp lý. Cảm biến này sử dụng photodiode tích hợp sẵn ADC và giao tiếp I2C, giúp đọc trực tiếp giá trị ánh sáng theo đơn vị lux, tiện lợi hơn nhiều so với cảm biến analog cần xử lý thêm.

Bên cạnh đó, BH1750 có mức điện áp hoạt động thấp (3.0V–5.0V), tương thích tốt với vi điều khiển như ESP32, phù hợp với mô hình hệ thống hiện tại. Ngoài ra, thư viện hỗ trợ đơn giản, dễ dùng, giúp việc lập trình nhanh chóng và ổn định hơn so với các cảm biến khác.

**3.4.2.6 Danh sách các cảm biến phù hợp với hệ thống**

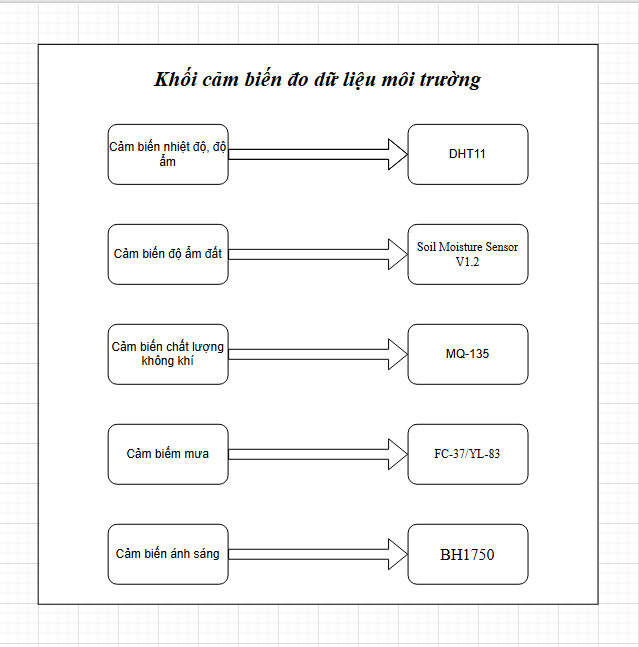
Bảng 8. Danh sách các cảm biến

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên cảm biến | Số lượng | Loại cảm biến |
| 1 | DHT11 | 1 | Cảm biến nhiệt độ, độ ẩm |
| 2 | Soil Moisture Sensor V1.2 | 1 | Cảm biến độ ẩm đất |
| 3 | MQ-135 | 1 | Cảm biến chất lượng không khí |
| 4 | FC-32/YL-83 | 1 | Cảm biến mưa |
| 5 | BH1750 | 1 | Cảm biến ánh sáng |

#### 3.4.2.7 Danh sách tất cả các linh kiện có trong hệ thống

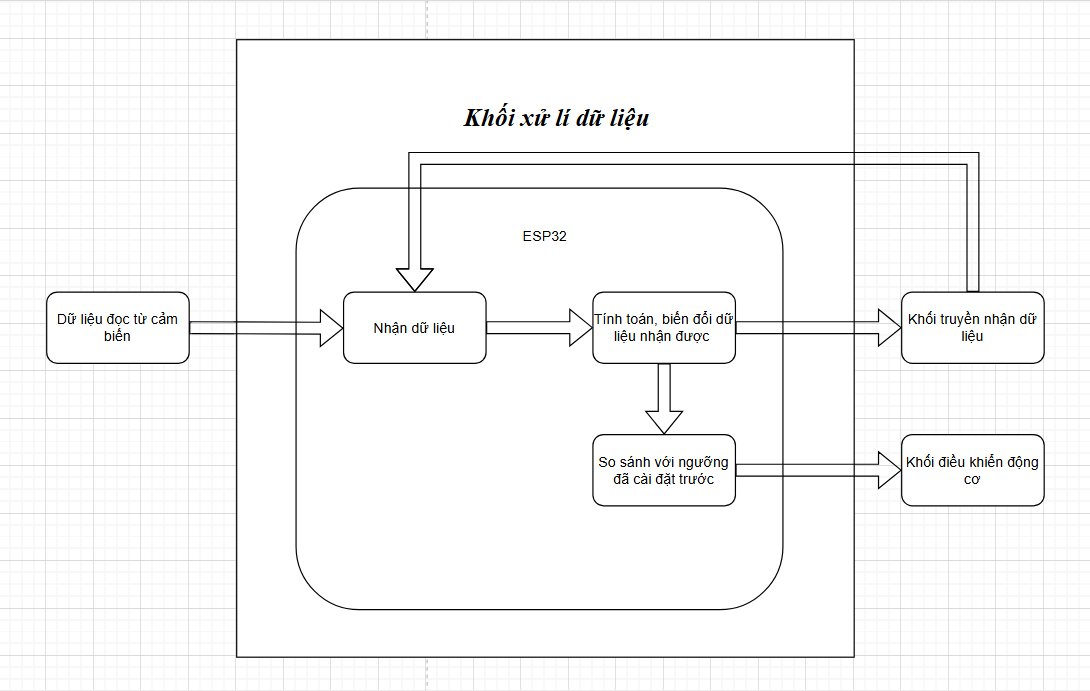
Bảng 9. Tất cả những linh kiện phần cứng được sử dụng

|  |  |  |
| --- | --- | --- |
| STT | Tên linh kiện | Số lượng |
| 1 | ESP32 | 1 |
| 2 | Cảm biến nhiệt độ, độ ẩm DHT11 | 1 |
| 3 | Cảm biến chất lượng không khí MQ135 | 1 |
| 4 | Cảm biến mưa  FC-37/YL-83 | 1 |
| 5 | Cảm biến độ ẩm đất Soil Moisture Sensor V1.2 | 1 |
| 6 | Cảm biến ánh sáng BH1750 | 1 |
| 7 | Màn hình OLED 0.96inch | 1 |
| 8 | Pin 18650 | 2 |
| 9 | Đế pin 18650 | 2 |
| 10 | Công tắc pin 18650 | 1 |
| 11 | Relay | 2 |
| 12 | Servo SG90 | 1 |
| 13 | Máy bơm 5VDC | 1 |
| 14 | Quạt 5VDC | 1 |



Hình 7. Sơ đồ khối cảm biến đo dữ liệu môi trường

**3.4.3 Khối xử lí dữ liệu**



Hình 8. Sơ đồ khối Khối xử lí dữ liệu

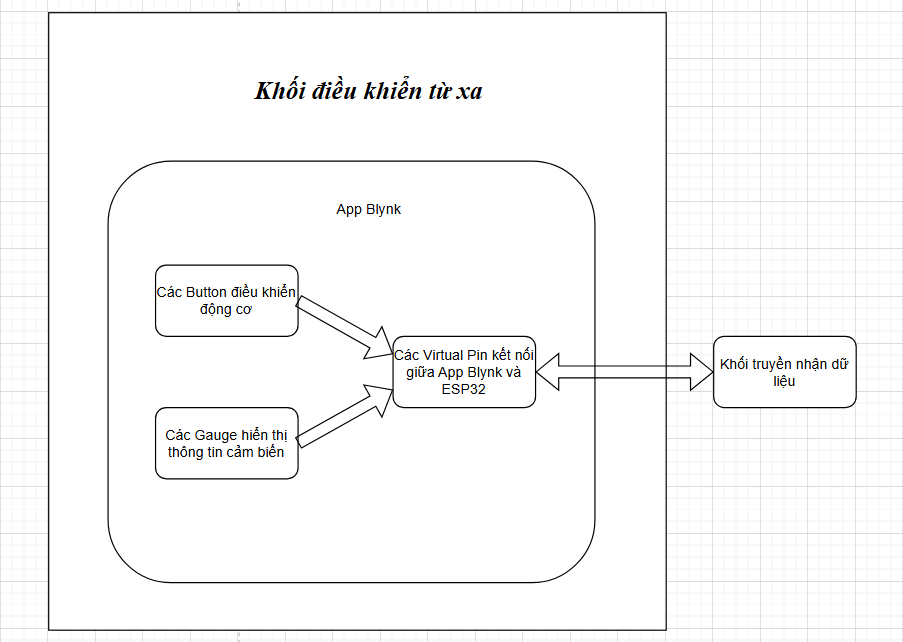
- Dùng ESP32 làm vi điều khiển chính để điều khiển toàn bộ hệ thống vì ESP32 có giá thành phù hợp, nhỏ gọn và có hỗ trợ kết nối Internet.

- ESP32 sẽ đọc dữ liệu của các cảm biến thu về, xử lí dữ liệu này từ analog sang digital, tính toán các giá trị dữ liệu phù hợp.

- Ngoài ra ESP32 còn dùng các giá trị tính toán được từ dữ liệu của cảm biến thu về để điều khiển các động cơ như máy bơm, quạt, servo trong hệ thống.

- ESP32 sử dụng các giá trị cảm biến này để truyền đến khối truyền nhận dữ liệu để thực hiện hiển thị lên màn hình OLED 0.96 inch và đồng bộ lên Blynk.

**3.4.4 Khối điều khiển từ xa**



Hình 9. Sơ đồ khối khối điều khiển từ xa

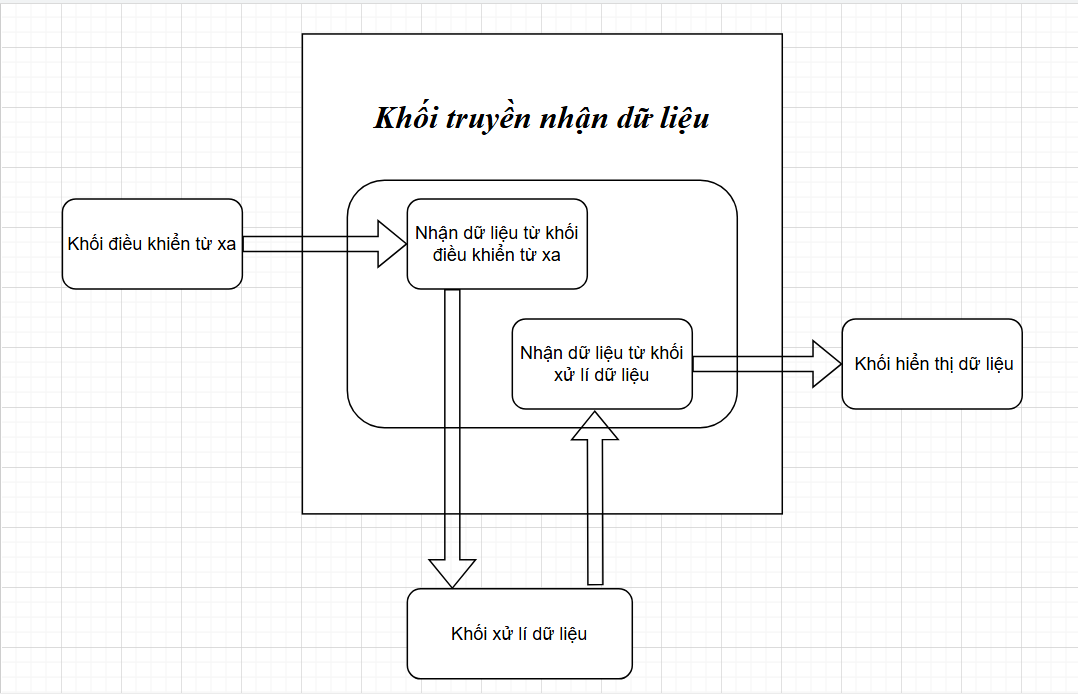
- Chọn ứng dụng Blynk để điều khiển hệ thống từ xa vì có hỗ trợ kết nối qua Internet.

- Thiết kế giao diện của ứng dụng gồm các Switch để điều khiển các động cơ như máy bơm, quạt, servo trong hệ thống.

- Thiết kế các Gauge để hiển thị biểu đồ giá trị của nhiệt độ, độ ẩm, độ ẩm đất, chất lượng không khí.

- Cầu nối giao tiếp giữa ESP32 và App Blynk là các Virtual Pin để gửi và nhận dữ liệu.

**3.4.5 Khối truyền nhận dữ liệu**



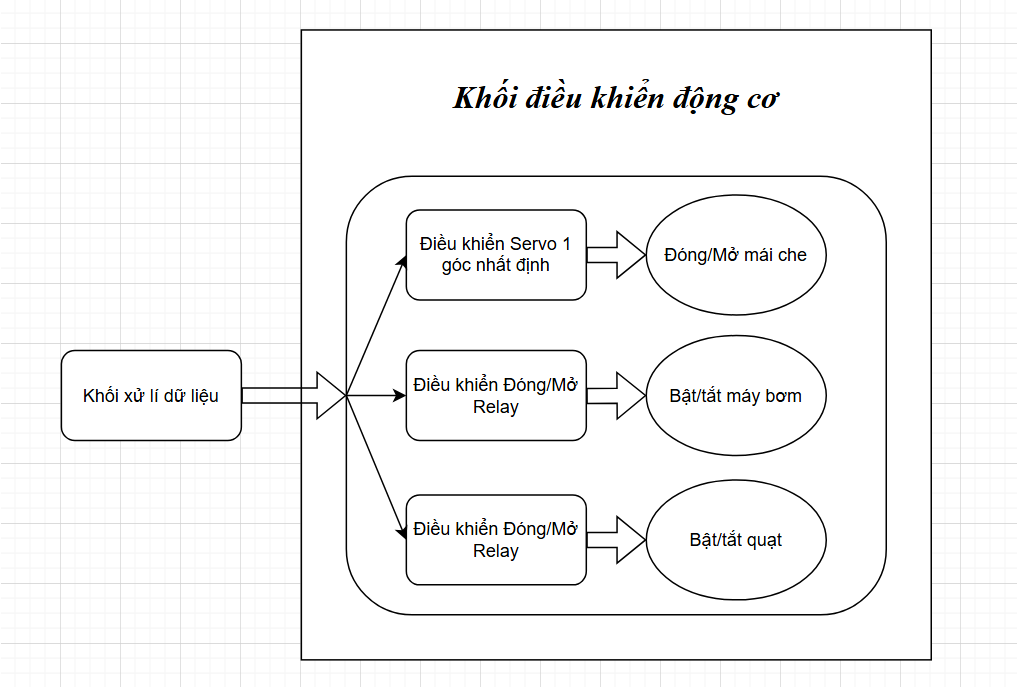
Hình 10. Sơ đồ khối khối truyền nhận dữ liệu

- Khối này thực hiện 2 chức năng:

+ Truyền nhận dữ liệu giữa Blynk và ESP32 thông qua kết nối Wifi để có thể điều khiển hệ thống từ xa và hiển thị đồng bộ dữ liệu lên Blynk.

+ Truyền dữ liệu là các giá trị của cảm biến đến màn hình OLED 0.96 inch để hiển thị các thông tin nhiệt độ, độ ẩm, độ ẩm đất, chất lượng không khí.

**3.4.6 Khối điều khiển động cơ**



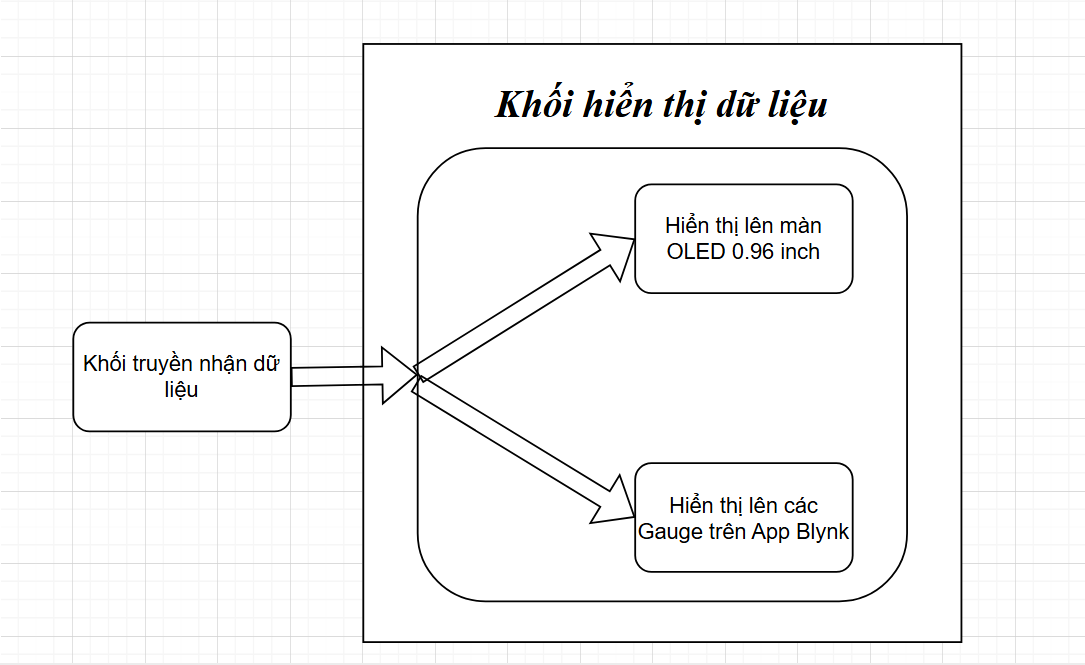
Hình 11. Sơ đồ khối khối điều khiển động cơ

- Máy bơm sẽ được ESP32 điều khiển qua relay bằng cách đọc giá trị dữ liệu từ cảm biến độ ẩm đất và so sánh với giá trị ngưỡng đã được cài đặt trước trong hệ thống hoặc sẽ được điều khiển từ xa bằng ứng dụng Blynk.

- Servo kết nối với mái che được điều khiển bằng ESP32 sau khi đọc và so sánh các giá trị đọc được từ cảm biến mưa và cảm biến ánh sáng để đóng mở mái che, ngoài ra nó còn được điều khiển thông qua Blynk.

- Quạt sẽ được ESP32 điều khiển qua relay bằng cách đọc giá trị dữ liệu từ cảm biến chất lượng không khí, quạt còn có thể được bật thông qua điều khiển từ xa trên Blynk.

**3.4.7 Hiển thị dữ liệu**



Hình 12. Sơ đồ khối khối hiển thị dữ liệu

- Các dữ liệu đã được ESP32 xử lí, tính toán sẽ hiển thị lên màn OLED 0.96 inch các thông tin nhiệt độ, độ ẩm, độ ẩm đất.

- Hiển thị đồng bộ các thông tin này lên các Gauge trong App Blynk trong khối điều khiển từ xa.

## 3.5 Thiết kế phần mềm

- Yêu cầu: điều khiển hoạt động của các động cơ trong hệ thống. Hiển thị thông tin nhiệt độ, độ ẩm, độ ẩm đất, chất lượng không khí.

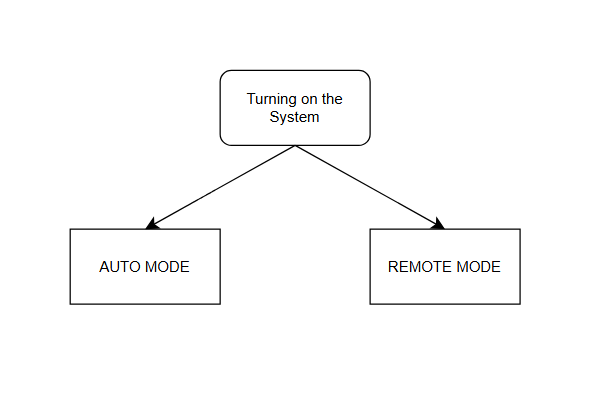
- Chức năng:

+ Chuyển đổi trạng thái hệ thống là AUTO hay REMOTE bằng nút Switch trên App.

+ Điều khiển Servo SG90 để đóng mở mái che, Relay để bật/tắt máy bơm và quạt thông qua các nút Switch trên App.

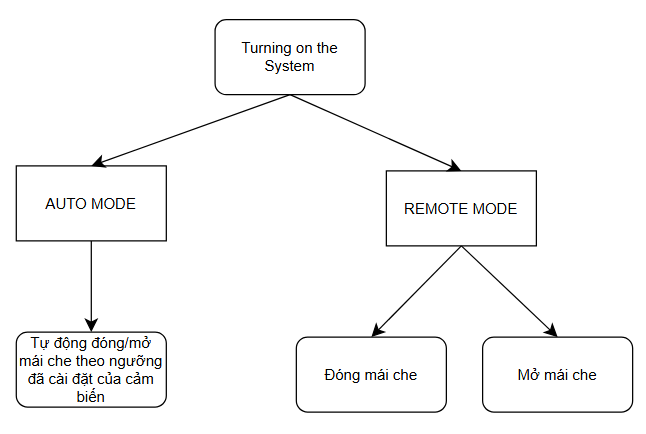
+ Hiển thị thông tin dữ liệu nhiệt độ, độ ẩm, độ ẩm đất, chất lượng không khí được đọc từ các cảm biến lên App.

- Mô tả chi tiết chức năng:



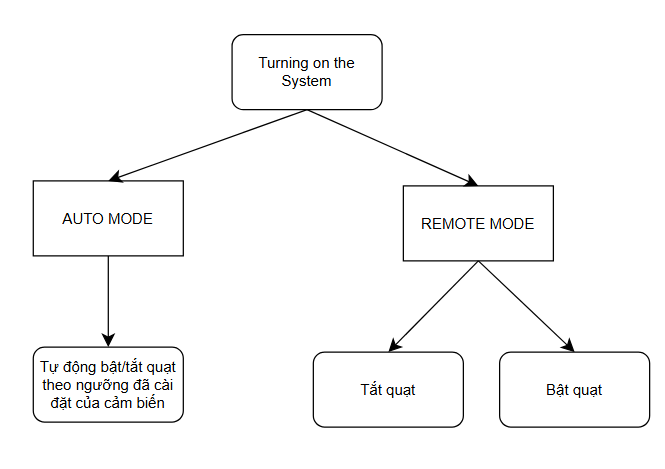
Hình 11. Mô tả hoạt động của phần mềm chế độ hoạt động hệ thống

Khi hệ thống được bật và bắt đầu khởi động. Hệ thống sẽ mặc định ở AUTO MODE, đây là chế độ hệ thống hoạt động theo các ngưỡng của các cảm biến đã được cài đặt từ trước. Trên Blynk có nút Switch để điều khiển chuyển chế độ hệ thống. Khi bật Switch thì hệ thống sẽ chuyển sang REMOTE MODE cho phép điều khiển các thiết bị động cơ bên trong hệ thống.



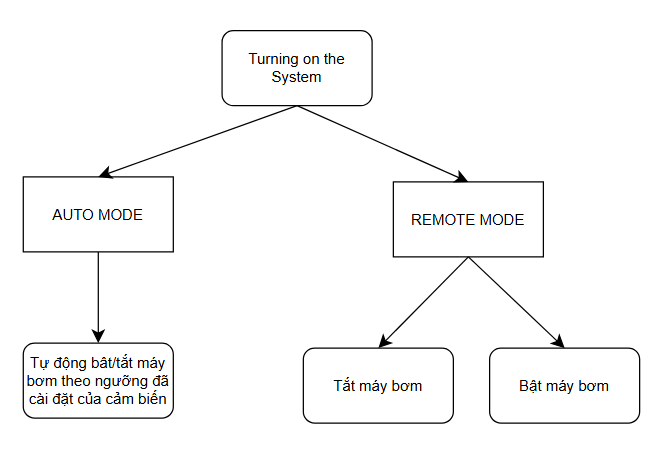
Hình 12. Sơ đồ mô tả hoạt động của phần mềm điều khiển mái che.

Khi hệ thống khởi động, nếu đang ở chế độ AUTO thì mái che sẽ được điều khiển bởi cảm biến mưa và cảm biến ánh sáng đểđóng/mở tự động thông qua ngưỡng đã được cài đặt trước. Nếu khi chuyển chế độ sang REMOTE, lúc này điều khiển mái che thủ công bằng nút Switch trên App Blynk.



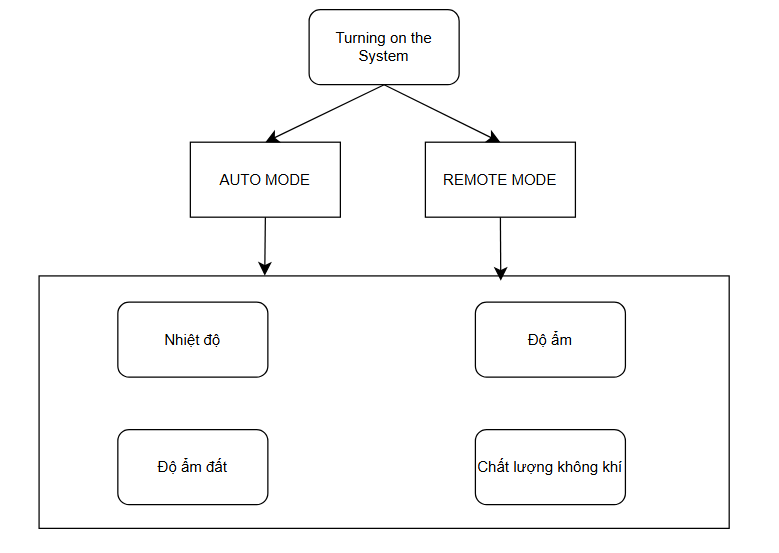
Hình 13. Sơ đồ mô tả hoạt động của phần mềm điều khiển quạt

Khi hệ thống khởi động, nếu đang ở chế độ AUTO thì quạt sẽ được điều khiển bởi cảm biến chất lượng không khí MQ135 để bật/tắt tự động thông qua ngưỡng đã được cài đặt trước. Nếu khi chuyển chế độ sang REMOTE, lúc này điều khiển quạt bât/tắt thủ công bằng nút Switch trên App Blynk.



Hình 14. Sơ đồ mô tả hoạt động của phần mềm điều khiển máy bơm

Khi hệ thống khởi động, nếu đang ở chế độ AUTO thì máy bơm sẽ được điều khiển bởi cảm biến độ ẩm đất để bật/tắt tự động thông qua ngưỡng đã được cài đặt trước. Nếu khi chuyển chế độ sang REMOTE, lúc này điều khiển máy bơm bât/tắt thủ công bằng nút Switch trên App Blynk.

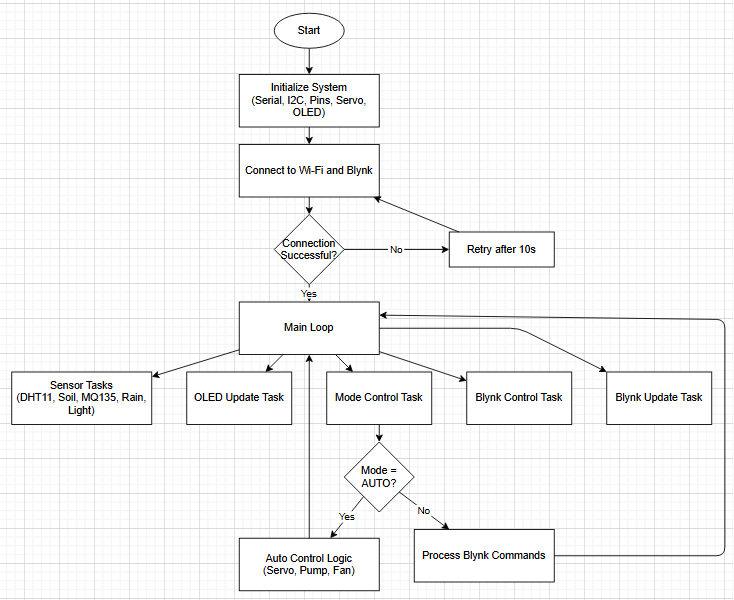


Hình 15. Sơ đồ mô tả hoạt động của phầm mềm hiển thị dữ liệu.

Khi hệ thống khởi động, dù đang ở chế độ AUTO hay REMOTE thì trên App Blynk đều hiển thị thông tin về: nhiệt độ, độ ẩm đọc từ cảm biến nhiệt độ, độ ẩm DHT11; Độ ẩm đất đọc từ cảm biến độ ẩm đất; Chất lượng không khí đọc từ cảm biến chất lượng không khí MQ135. Các thông tin này đồng bộ 2 chiều giữa hệ thống và Blynk.

**3.6 Lưu đồ và giải thuật của hệ thống**

**3.6.1 Lưu đồ hệ thống**



Hình 16. Lưu đồ giải thuật của hệ thống

**3.6.2 Giải thuật của hệ thống**

**3.6.2.1 Giới Thiệu**

Hệ thống "Smart Vegetable" là một dự án IoT sử dụng vi điều khiển ESP32 để giám sát và điều khiển môi trường trồng rau thông minh. Hệ thống kết nối Wi-Fi và tích hợp với nền tảng Blynk để theo dõi từ xa và điều khiển các thiết bị. Các thông số môi trường như nhiệt độ, độ ẩm, độ ẩm đất, chất lượng không khí, cường độ ánh sáng và mưa được giám sát liên tục. Dựa trên dữ liệu này, hệ thống điều khiển servo (mái che), bơm nước và quạt ở hai chế độ: AUTO (tự động dựa trên cảm biến) và REMOTE (điều khiển thủ công qua Blynk).

**3.6.2.2 Thành Phần Hệ Thống**

- Cảm Biến:

+ DHT11: Đo nhiệt độ và độ ẩm không khí.

+ Cảm biến độ ẩm đất: Đo độ ẩm đất (ADC).

+ MQ135: Đo chất lượng không khí (ADC).

+ Cảm biến mưa: Phát hiện mưa (ADC).

+ BH1750: Đo cường độ ánh sáng (I2C).

- Thiết Bị Điều Khiển:

+ Servo: Điều khiển mái che (mở/đóng).

+ Bơm: Tưới nước (Relay).

+ Quạt: Làm mát hoặc thông gió (Relay).

- Hiển Thị:

+ Màn hình OLED (SSD1306): Hiển thị dữ liệu cảm biến và trạng thái thiết bị (I2C).

- Kết Nối:

+ Wi-Fi: Kết nối internet.

+ Blynk: Giám sát và điều khiển từ xa qua ứng dụng.

**3.6.2.3 Chế Độ Hoạt Động**

- AUTO: Hệ thống tự động điều khiển thiết bị dựa trên dữ liệu cảm biến và ngưỡng định trước.

- REMOTE: Người dùng điều khiển thiết bị thủ công qua ứng dụng Blynk.

**3.6.2.4 Phân Tích Các Task (Nhiệm Vụ)**

Hệ thống sử dụng FreeRTOS để quản lý các nhiệm vụ song song, đảm bảo hoạt động hiệu quả và không bị chặn.

- DHTsensorTask: Đọc nhiệt độ và độ ẩm mỗi 2 giây.

- soilMoistureTask: Đọc độ ẩm đất mỗi 2 giây.

- airQualityTask: Đọc chất lượng không khí mỗi 2 giây.

- rainSensorTask: Đọc cảm biến mưa mỗi 2 giây.

- lightSensorTask: Đọc cường độ ánh sáng mỗi 2 giây.

- oledUpdateTask: Cập nhật màn hình OLED mỗi 2 giây.

- modeControlTask: Quản lý chế độ AUTO/REMOTE và điều khiển thiết bị.

- blynkControlTask: Xử lý lệnh từ Blynk mỗi 50ms.

- TaskConnectToWiFiAndBlynk: Kết nối Wi-Fi và Blynk (chạy một lần).

- blynkUpdateTask: Cập nhật dữ liệu lên Blynk mỗi 10 giây.

- displayInfoTask: In dữ liệu ra Serial Monitor mỗi 2 giây.

**3.6.2.5. Giải Thuật Điều Khiển**

**A.Khởi Tạo (setup function)**

- Khởi tạo giao tiếp Serial, I2C, các chân I/O, servo, mutex và OLED.

- Đặt relay (bơm, quạt) ở trạng thái OFF ban đầu và servo ở góc mở (0°).

- Tạo các task và gán vào lõi 0 hoặc 1 của ESP32.

**B.Đọc Dữ Liệu Cảm Biến**

- Mỗi task cảm biến đọc dữ liệu định kỳ (2 giây) và cập nhật biến toàn cục:

+ DHT11: Nhiệt độ (temperature), độ ẩm (humidity).

+ Soil Moisture: Độ ẩm đất (soilMoisturePercent) tính theo phần trăm từ giá trị ADC.

+ MQ135: Chất lượng không khí (airQuality) từ 0-100%.

+ Rain Sensor: Giá trị thô (rainRawGlobal) từ ADC.

+ BH1750: Cường độ ánh sáng (lux).

**C.Hiển Thị Trên OLED**

- Task oledUpdateTask chạy mỗi 2 giây:

+ Sử dụng mutex để truy cập I2C an toàn.

+ Hiển thị chế độ, dữ liệu cảm biến và trạng thái thiết bị.

**D.Quản Lý Chế Độ (modeControlTask)**

- Chạy mỗi 500ms:

+ Kiểm tra thay đổi chế độ:

* Nếu chuyển sang AUTO: Đồng bộ trạng thái thiết bị lên Blynk.
* Nếu chuyển sang REMOTE: Reset widget Blynk về 0, tắt tất cả thiết bị và cập nhật trạng thái.

- Nếu ở AUTO, gọi autoControlLogic().

**E.Logic Điều Khiển Tự Động (autoControlLogic)**

- Chỉ chạy ở chế độ AUTO:

+ Servo (Mái Che):

* Đóng (closeAngle = 180°) nếu ánh sáng > 200 lux hoặc mưa (rainRawGlobal < 2500).
* Mở (openAngle = 0°) nếu không.

+ Bơm:

* Bật nếu độ ẩm đất từ 10% đến 30%.
* Tắt nếu ngoài khoảng này.

+ Quạt:

* Bật nếu nhiệt độ > 35°C hoặc chất lượng không khí ≤ 85%.
* Tắt nếu không.

+ Đồng bộ trạng thái lên Blynk.

**F.Điều Khiển Từ Blynk (blynkControlTask)**

- Chạy mỗi 50ms:

+ Nếu ở chế độ REMOTE và có lệnh từ Blynk:

* Cập nhật trạng thái servo, bơm, quạt theo giá trị từ ứng dụng.

**G.Callback Từ Blynk**

- Các hàm BLYNK\_WRITE xử lý đầu vào từ ứng dụng:

- Cập nhật chế độ và trạng thái thiết bị vào struct blynkControl.

**H.Kết Nối Wi-Fi và Blynk**

- Task TaskConnectToWiFiAndBlynk:

+ Thử kết nối Wi-Fi trong 10 giây, sau đó kết nối Blynk trong 5 giây.

+ Nếu thành công, bật LED và khởi tạo virtual pins.

**I.Cập Nhật Dữ Liệu Lên Blynk**

- Task blynkUpdateTask chạy mỗi 10 giây:

+ Gửi nhiệt độ, độ ẩm, độ ẩm đất, chất lượng không khí lên Blynk.

**K.Hiển Thị Thông Tin**

- Task displayInfoTask in dữ liệu cảm biến và trạng thái thiết bị ra Serial Monitor mỗi 2 giây.

**3.6.2.6. Tích Hợp Blynk**

- Giám sát: Dữ liệu cảm biến được cập nhật lên Blynk mỗi 10 giây.

- Điều khiển: Ở chế độ REMOTE, người dùng điều khiển servo, bơm, quạt qua virtual pins (V0, V1, V2, V7).

**CHƯƠNG 4 KẾT QUẢ**

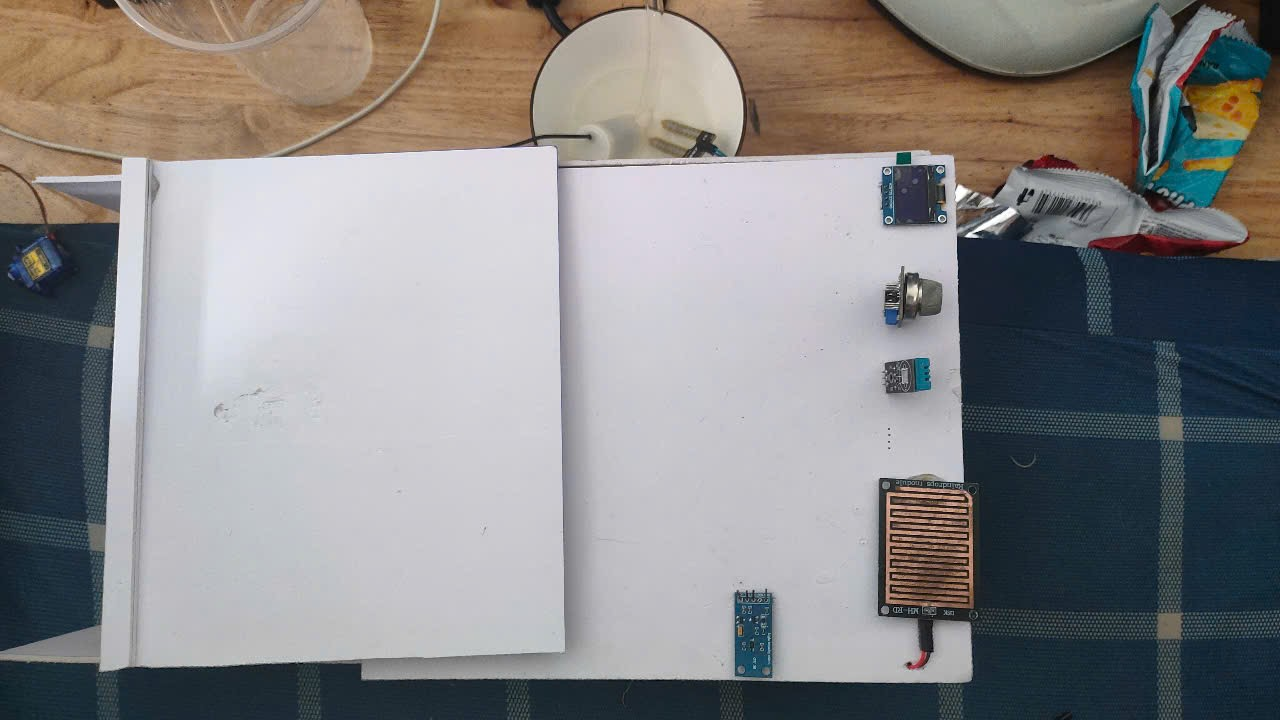
## 4.1 Nhận xét

- Sau thời gian 15 tuần thực hiện đề tài, qua quá trình nghiên cứu và tìm hiểu nhóm đã học hỏi được nhiều kiến thức mới, công nghệ mới. Đồng thời áp dụng được nhiều lý thuyết vào dự án thực tế.

- Board ESP32-WOOM-32 là mạch điều khiển chính trong hệ thống. Nhóm đã biết cách ứng dụng chức năng tích hợp Wifi trong board để phát triển hệ thống điều khiển thiết bị từ xa. Từ đó nắm được bản chất điều khiển, để có thể mở rộng cho các ứng dụng IoT sau này, điều khiển được một hệ thống lớn. Thông qua đề tài nhóm đã biết cách điều khiển và giám sát thiết bị qua Internet (Wifi). Đồng thời thiết kế được giao diện Web, ứng dụng Blynk cho hệ thống điều khiển. Đề tài có sử dụng nhiều cảm biến qua đó nhóm biết được cấu tạo, nguyên lý hoạt động cũng như cách sử dụng các cảm biến. Các cảm biến này rất phổ biến, thích hợp cho việc giám sát thông tin dữ liệu môi trường trên các hệ thống nhỏ.

## 4.2 Thực nghiệm

**4.2.1 Mô hình hệ thống**



Hình 17. Mô hình mô phỏng hệ thống

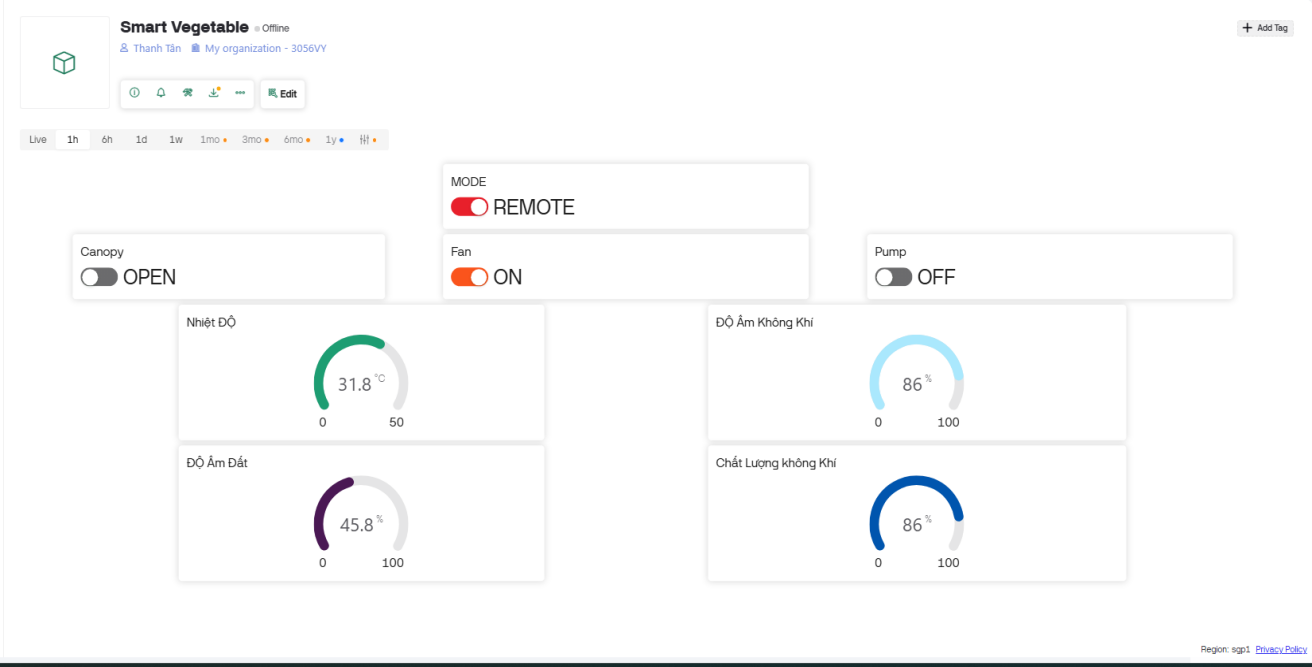
Đây là mô hình hệ thống “ Chăm sóc vườn rau thông minh” được gắn các cảm biến DHT11, MQ135, cảm biến mưa, cảm biến ánh sáng BH1750, cảm biến độ ẩm đất để điều khiển các thiết bị như quạt, máy bơm, servo điều khiển mái che.

**4.2.2 Giao diện Blynk**



Hình 18. Giao diện Blynk trên điện thoại

Giao diện App Blynk hiển thị trạng thái thiết bị và thông tin được cảm biến đo được ở chế độ AUTO trên điện thoại.



Hình 19. Giao diện Blynk trên máy tính

Giao diện Wrbsite Blynk hiển thị trạng thái thiết bị và thông tin được cảm biến đo được ở chế độ REMOTE trên máy tính.

**4.2.3 Đánh giá hệ thống**

\* Ưu điểm:

- Hệ thống hoạt động ổn định.

- Thực hiện đầy đủ các tính năng có trong yêu cầu hệ thống.

- Có thể phát triển thêm các chức năng khác nếu muốn.

\* Nhược điểm:

- Thời gian nhận tín hiệu điều khiển từ Blunk xuống còn bị chậm.

- Yêu cầu khả năng kết nối các cảm biến, các thiết bị và vi điều khiển còn gặp nhiều khó khăn.

- Chưa tối ưu tốt nguồn điện của hệ thống.

# chương 5 kết luận và hướng phát triển

## 5.1. Kết luận

- Đã triển khai thành công một hệ thống IoT giúp giám sát và điều khiển việc chăm sóc vườn rau một cách “thông minh” ở quy mô mô hình nhỏ.

- Sử dụng ESP32 làm vi điều khiển trung tâm để thu thập dữ liệu từ các cảm biến: DHT11 (nhiệt độ, độ ẩm không khí), Soil Moisture Sensor V1.2 (độ ẩm đất), MQ-135 (chất lượng không khí), FC-32 (cảm biến mưa), BH1750 (cảm biến ánh sáng).

- Dữ liệu thu thập được xử lý trên ESP32, hiển thị lên màn hình OLED 0.96 inch và đồng thời truyền song song về ứng dụng Blynk để người dùng có thể quan sát, đặt ngưỡng và điều khiển từ xa (AUTO/REMOTE) các thiết bị: máy bơm, quạt, servo điều khiển mái che.

- Khi hệ thống ở chế độ AUTO, ESP32 tự động so sánh giá trị cảm biến với ngưỡng cài sẵn, sau đó bật/tắt thiết bị tương ứng (ví dụ: tự động bật bơm khi độ ẩm đất < ngưỡng; đóng/mở mái che khi trời mưa/thiếu sáng; bật quạt khi chất lượng không khí kém).

- Khi chuyển sang chế độ REMOTE, người dùng có thể dùng giao diện Blynk trên điện thoại để bật/tắt máy bơm, quạt, servo ngay lập tức. Mọi trạng thái ON/OFF, giá trị cảm biến đều hiển thị trên app theo thời gian thực.

- Qua thử nghiệm thực tế trong điều kiện phòng lab mô phỏng vườn rau quy mô nhỏ (kích thước ~ 40×20×10 cm), hệ thống hoạt động ổn định.

- Dữ liệu sensor cập nhật lên màn hình OLED và Blynk trong vòng 2 giây.

- Độ chính xác đo đạc nằm trong phạm vi sai số của từng loại cảm biến: ±2 °C (DHT11), ±3–5 % (Soil Moisture Sensor), ±5 % (MQ-135), v.v.

- Thời gian phản hồi lệnh control từ app ~ 500 ms – 1 s, phụ thuộc chất lượng kết nối Wi-Fi.

- Vì dùng FreeRTOS nên ESP32 và các module hoạt động bền bỉ, không gặp sự cố treo máy hay ngắt kết nối thường xuyên.

- Ứng dụng Blynk được thiết kế với các Gauge, Switch đơn giản, trực quan, giúp người dùng dễ dàng theo dõi và điều khiển.

- Màn hình OLED cung cấp thông tin cơ bản ngay tại chỗ, thuận tiện cho việc quan sát trực tiếp mà không cần điện thoại.

## 5.2. Hướng phát triển

- Bổ sung cảm biến pH và EC để theo dõi chất lượng dinh dưỡng đất, giúp điều chỉnh lượng phân bón kịp thời; thêm cảm biến nhiệt độ đất (soil temp) và cảm biến ánh sáng chuyên dụng để có dữ liệu toàn diện, từ đó tối ưu lịch tưới và che phủ. Việc này cho phép hệ thống đánh giá chính xác hơn điều kiện sinh trưởng, giảm rủi ro sai sót khi chỉ dùng cảm biến độ ẩm đất và DHT.

- Tự phát triển một dashboard web nhẹ (React/Vue) hiển thị biểu đồ lịch sử (7–14 ngày), đồng thời tích hợp cảnh báo qua email hoặc notification khi thông số vượt ngưỡng (độ ẩm quá thấp, ánh sáng yếu). Ứng dụng di động chỉ giữ hai màn hình chính: “Xem số liệu” (các giá trị lớn và công tắc) và “Lịch sử” (biểu đồ đơn giản), cùng tính năng hẹn giờ tưới cơ bản. Giao diện tập trung vào dễ nhìn, thao tác nhanh để người dùng hộ gia đình không mất nhiều thời gian làm quen.

- Xây dựng gói giải pháp IoT trọn gói (bộ kit cảm biến + ESP32 + pin/solar + hướng dẫn cài đặt) dành cho hộ nông dân và hợp tác xã, kèm gói bảo trì, hỗ trợ kỹ thuật. Kết nối với hệ thống quản lý nông nghiệp (Farm Management System) để chia sẻ dữ liệu sản xuất, dự báo năng suất, và tích hợp vào chuỗi cung ứng. Thử nghiệm tại các vườn nhà kính hoặc sân thượng để thu thập dữ liệu thực tế, từ đó hoàn thiện sản phẩm trước khi triển khai quy mô lớn.

# PHỤ lục

**Source code**

|  |
| --- |
| //https://blynk.cloud/dashboard/662630/global/devices/1/organization/662630/devices/3222607/dashboard  #define BLYNK\_TEMPLATE\_ID "TMPL6OIwNHVmO"  #define BLYNK\_TEMPLATE\_NAME "Smart Vegetable"  #define BLYNK\_AUTH\_TOKEN "m3ZO2xEHHWmPrM-xp-\_gjHhoeCG4t6\_T"  #define BLYNK\_PRINT Serial  #include <Arduino.h>  #include <freertos/FreeRTOS.h>  #include <freertos/task.h>  #include <DHT.h>  #include <Wire.h>  #include <BH1750.h>  #include <ESP32Servo.h>  #include <WiFi.h>  #include <BlynkSimpleEsp32.h>  #include <Adafruit\_SSD1306.h>  // Wi-Fi credentials  char auth[] = BLYNK\_AUTH\_TOKEN;  char ssid[] = "S21 FE 5G";  char pass[] = "tuchoxiu";  // — Virtual Pins —  #define VP\_BUTTON\_CANOPY V0  #define VP\_BUTTON\_PUMP   V1  #define VP\_BUTTON\_FAN    V2  #define VP\_BUTTON\_MODE   V7  #define VP\_TEMP          V3  #define VP\_HUMID         V4  #define VP\_SOIL          V5  #define VP\_MQ135         V6  // Cờ kết nối Wifi và Blynk  bool wifiConnected  = false;  bool blynkConnected = false;  // Chế độ hệ thống  bool isAutoMode = true;  // true = AUTO, false = REMOTE  bool prevMode   = true;  // Trạng thái thiết bị (được cập nhật trong AUTO hoặc REMOTE)  bool servoClose = false;  bool pumpOn      = false;  bool fanOn       = false;  // — Cấu hình chân & hằng số —  #define LED\_PIN            2    // LED báo Blynk OK  #define DHTPIN             25   // Chân DHT11  #define DHTTYPE            DHT11  #define SOIL\_MOISTURE\_PIN  36   // ADC  #define MQ135\_PIN          35   // ADC  #define RAIN\_SENSOR\_PIN    34   // ADC  #define BH1750\_ADDR        0x23  #define OLED\_ADDR          0x3C  #define FAN\_RELAY\_PIN      16  #define PUMP\_RELAY\_PIN     17  #define SERVO\_PIN          18  // Ngưỡng điều khiển  #define TEMP\_THRESHOLD   35.0  #define MQ135\_THRESHOLD  85  #define SOIL\_THRESHOLD\_MIN 10.0  #define SOIL\_THRESHOLD\_MAX 30.0  #define LIGHT\_THRESHOLD  200.0  #define RAIN\_THRESHOLD   2500  #define openAngle        0  #define closeAngle       180  #define WIFI\_TIMEOUT       10000   // ms  #define BLYNK\_CONN\_TIMEOUT 5000    // ms  // — Định nghĩa OLED —  #define SCREEN\_WIDTH 128  #define SCREEN\_HEIGHT 64  Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);  // — Mutex cho I2C —  SemaphoreHandle\_t i2cMutex;  // — Đối tượng cảm biến —  DHT dht(DHTPIN, DHTTYPE);  BH1750 lightMeter(BH1750\_ADDR);  Servo myServo;  // — Biến toàn cục lưu dữ liệu cảm biến —  float temperature          = 0.0;  float humidity             = 0.0;  float soilMoisturePercent  = 0.0;  float airQuality           = 0.0;  float lux                  = 0.0;  // Giá trị hiệu chỉnh cho sensor (nếu cần)  const int dryValue         = 4095;  const int wetValue         = 0;  const int cleanAirValue    = 0;  const int pollutedAirValue = 4095;  // Biến đọc trực tiếp Rain  int rainRawGlobal = 4095;  // Thêm biến để lưu trạng thái điều khiển từ Blynk  struct BlynkControlState {    bool modeChanged = false;    bool canopyChanged = false;    bool pumpChanged = false;    bool fanChanged = false;    bool newMode = true;    bool newCanopy = false;    bool newPump = false;    bool newFan = false;  } blynkControl;  // --------------------------------------------------------------------  // 1) Hàm điều khiển thực tế  void smoothServoControl(int targetAngle, int stepDelay = 15) {    int currentAngle = myServo.read();      if (currentAngle < targetAngle) {      for (int angle = currentAngle; angle <= targetAngle; angle++) {        myServo.write(angle);        delay(stepDelay);      }    } else {      for (int angle = currentAngle; angle >= targetAngle; angle--) {        myServo.write(angle);        delay(stepDelay);      }    }  }  void controlServo(bool close) {    if (!close) {      smoothServoControl(openAngle);    } else {      smoothServoControl(closeAngle);    }  }  void controlPump(bool on) {    digitalWrite(PUMP\_RELAY\_PIN, on ? LOW : HIGH);  }  void controlFan(bool on) {    digitalWrite(FAN\_RELAY\_PIN, on ? LOW : HIGH);  }  // 2) Hàm đọc Rain cho mỗi lần cần (dùng trong autoControlLogic)  int readRainSensor() {    int raw = analogRead(RAIN\_SENSOR\_PIN);    rainRawGlobal = raw;    return raw;  }  // --------------------------------------------------------------------  // 3) Task đọc DHT11 (cập nhật temperature, humidity hàng 2s)  void DHTsensorTask(void \*pvParameters) {    dht.begin();    while (1) {      float t = dht.readTemperature();      float h = dht.readHumidity();      if (!isnan(t) && !isnan(h)) {        temperature = t;        humidity = h;      }      vTaskDelay(pdMS\_TO\_TICKS(2000));    }  }  // 4) Task đọc Soil Moisture (hàng 2s)  void soilMoistureTask(void \*pvParameters) {    while (1) {      int raw = analogRead(SOIL\_MOISTURE\_PIN);      raw = constrain(raw, wetValue, dryValue);      soilMoisturePercent = 100.0 \* (dryValue - raw) / (dryValue - wetValue);      vTaskDelay(pdMS\_TO\_TICKS(2000));    }  }  // 5) Task đọc MQ135 (hàng 2s)  void airQualityTask(void \*pvParameters) {    while (1) {      int raw = analogRead(MQ135\_PIN);      raw = constrain(raw, cleanAirValue, pollutedAirValue);      airQuality = map(raw, cleanAirValue, pollutedAirValue, 100, 0);      vTaskDelay(pdMS\_TO\_TICKS(2000));    }  }  // 6) Task đọc Rain Sensor (hàng 2s, chỉ in ra log)  void rainSensorTask(void \*pvParameters) {    while (1) {      int raw = analogRead(RAIN\_SENSOR\_PIN);      rainRawGlobal = raw;      vTaskDelay(pdMS\_TO\_TICKS(2000));    }  }  // 7) Task đọc BH1750 (hàng 2s)  void lightSensorTask(void \*pvParameters) {    while (!lightMeter.begin(BH1750::CONTINUOUS\_HIGH\_RES\_MODE)) {      vTaskDelay(pdMS\_TO\_TICKS(2000));    }    while (1) {      float val = lightMeter.readLightLevel();      if (val < 0) {        lux = 0.0;      } else {        lux = val;      }      vTaskDelay(pdMS\_TO\_TICKS(2000));    }  }  // 8) Task hiển thị trên OLED (hàng 2s)  void oledUpdateTask(void \*pvParameters) {    while (1) {      if (xSemaphoreTake(i2cMutex, portMAX\_DELAY) == pdTRUE) {        display.clearDisplay();        display.setTextSize(0.2);        display.setTextColor(SSD1306\_WHITE);        display.setCursor(0, 0);          // Dòng 1: Mode        display.print("Mode: ");        display.println(isAutoMode ? "AUTO" : "REMOTE");          // Dòng 2: Temp và Humd        display.print("T: ");        display.print(temperature, 1);        display.print("C| H: ");        display.print(humidity, 1);        display.println("%");          // Dòng 3: DoAmDat và Pump        display.print("DoAmDat: ");        display.print(soilMoisturePercent, 1);        display.println("%");          // Dòng 4: AQI        display.print("AQI: ");        display.print(airQuality, 1);        display.print("% -> ");        display.println(airQuality > MQ135\_THRESHOLD ? "GOOD" : "BAD");          // Dòng 5: Weather        display.print("Weather:");        display.println(rainRawGlobal < RAIN\_THRESHOLD ? "Rain" : "Sun");        // Dòng 6: Light        display.print("Light: ");        display.print(lux, 1);        display.println("lux");        // Dòng 7: Canopy và Fan states        display.print("Canopy:");        display.print(servoClose ? "Close" : "Open");        display.print("| Fan:");        display.println(fanOn ? "On" : "Off");        // Dòng 8: Pump state        display.print("Pump: ");        display.println(pumpOn ? "On" : "Off");          display.display();        xSemaphoreGive(i2cMutex);      }      vTaskDelay(pdMS\_TO\_TICKS(2000));    }  }  // --------------------------------------------------------------------  // 9) Hàm điều khiển tự động (gọi mỗi khi isAutoMode = true):  void autoControlLogic() {    // 9.1. Lấy giá trị cảm biến    float currentTemp  = temperature;    float currentSoil  = soilMoisturePercent;    float currentLux   = lux;    int   currentRain  = rainRawGlobal;    int   currentMQ    = airQuality;    // 9.2. Điều khiển Servo (mái che):    if (currentLux > LIGHT\_THRESHOLD || currentRain < RAIN\_THRESHOLD) {      // điều kiện đóng mái che      if (!servoClose) {        servoClose = true;        controlServo(true);      }    } else {      // điều kiện mở mái che      if (servoClose) {        servoClose = false;        controlServo(false);      }    }    // 9.3. Điều khiển Pump    if ((SOIL\_THRESHOLD\_MIN < currentSoil) && (currentSoil < SOIL\_THRESHOLD\_MAX)) {      if (!pumpOn) {        pumpOn = true;        controlPump(true);      }    } else {      if (pumpOn) {        pumpOn = false;        controlPump(false);      }    }    // 9.4. Điều khiển Fan    if (currentTemp > TEMP\_THRESHOLD || airQuality <= MQ135\_THRESHOLD) {      if (!fanOn) {        fanOn = true;        controlFan(true);      }    } else {      if (fanOn) {        fanOn = false;        controlFan(false);      }    }    // 9.5. Đồng bộ trạng thái lên Blynk    Blynk.virtualWrite(VP\_BUTTON\_CANOPY, servoClose);    Blynk.virtualWrite(VP\_BUTTON\_PUMP,    pumpOn);    Blynk.virtualWrite(VP\_BUTTON\_FAN,     fanOn);  }  // --------------------------------------------------------------------// 10) Task quản lý chế độ AUTO / REMOTE  void modeControlTask(void \*pvParameters) {    (void) pvParameters;    while (1) {      // Nếu vừa chuyển đổi chế độ      if (isAutoMode != prevMode) {        if (isAutoMode) {          Serial.println("→ Chuyển sang chế độ AUTO");          // Đồng bộ trạng thái ban đầu lên Blynk (nếu cần)          Blynk.virtualWrite(VP\_BUTTON\_CANOPY, servoClose);          Blynk.virtualWrite(VP\_BUTTON\_PUMP,    pumpOn);          Blynk.virtualWrite(VP\_BUTTON\_FAN,     fanOn);        } else {          Serial.println("→ Chuyển sang chế độ REMOTE");          // Reset các widget Blynk về 0          Blynk.virtualWrite(VP\_BUTTON\_CANOPY, 0);          Blynk.virtualWrite(VP\_BUTTON\_PUMP,   0);          Blynk.virtualWrite(VP\_BUTTON\_FAN,    0);          // Tắt hết thiết bị          controlServo(false);          controlPump(false);          controlFan(false);          // Cập nhật lại cờ          servoClose = false;          pumpOn      = false;          fanOn       = false;        }        prevMode = isAutoMode;      }      // Nếu đang ở AUTO → gọi autoControlLogic()      if (isAutoMode) {        autoControlLogic();      }      vTaskDelay(pdMS\_TO\_TICKS(500));    }  }  // --------------------------------------------------------------------  // 11) Task xử lý lệnh điều khiển từ Blynk  void blynkControlTask(void \*pvParameters) {    while (1) {      if (blynkConnected) {        if (blynkControl.modeChanged) {          isAutoMode = blynkControl.newMode;          blynkControl.modeChanged = false;        }          if (!isAutoMode) {  // Chỉ xử lý khi ở chế độ REMOTE          if (blynkControl.canopyChanged) {            servoClose = blynkControl.newCanopy;            controlServo(servoClose);            blynkControl.canopyChanged = false;          }            if (blynkControl.pumpChanged) {            pumpOn = blynkControl.newPump;            controlPump(pumpOn);            blynkControl.pumpChanged = false;          }            if (blynkControl.fanChanged) {            fanOn = blynkControl.newFan;            controlFan(fanOn);            blynkControl.fanChanged = false;          }        }      }      vTaskDelay(pdMS\_TO\_TICKS(50));  // Kiểm tra mỗi 50ms    }  }  // --------------------------------------------------------------------  // 12) Tối ưu hóa callback từ Blynk  BLYNK\_WRITE(VP\_BUTTON\_MODE) {    blynkControl.newMode = (param.asInt() == 0);    blynkControl.modeChanged = true;  }  BLYNK\_WRITE(VP\_BUTTON\_CANOPY) {    blynkControl.newCanopy = param.asInt();    blynkControl.canopyChanged = true;  }  BLYNK\_WRITE(VP\_BUTTON\_PUMP) {    blynkControl.newPump = param.asInt();    blynkControl.pumpChanged = true;  }  BLYNK\_WRITE(VP\_BUTTON\_FAN) {    blynkControl.newFan = param.asInt();    blynkControl.fanChanged = true;  }  // --------------------------------------------------------------------// 13) Task kết nối WiFi + Blynk  void TaskConnectToWiFiAndBlynk(void \*pvParameters) {    (void) pvParameters;    while (1) {      Serial.println();      Serial.printf("Connecting to WiFi: %s …", ssid);      WiFi.mode(WIFI\_STA);      WiFi.begin(ssid, pass);      unsigned long t0 = millis();      while ((millis() - t0 < WIFI\_TIMEOUT) && WiFi.status() != WL\_CONNECTED) {        Serial.print(".");        delay(500);      }      Serial.println();      if (WiFi.status() == WL\_CONNECTED) {        wifiConnected = true;        Serial.print("WiFi OK, IP: ");        Serial.println(WiFi.localIP());      } else {        Serial.println("WiFi FAILED (timeout)");        vTaskDelete(NULL);      }      if (wifiConnected) {        Blynk.config(auth);        Serial.println("Connecting to Blynk …");        if (Blynk.connect(BLYNK\_CONN\_TIMEOUT)) {          blynkConnected = true;          digitalWrite(LED\_PIN, HIGH);          Serial.println("Blynk connected!");          // Gửi giá trị khởi tạo          Blynk.virtualWrite(VP\_BUTTON\_MODE, 0);          Blynk.virtualWrite(VP\_TEMP,  0);          Blynk.virtualWrite(VP\_HUMID, 0);          Blynk.virtualWrite(VP\_SOIL,  0);          Blynk.virtualWrite(VP\_MQ135, 0);        } else {          digitalWrite(LED\_PIN, LOW);          Serial.println("Blynk FAILED (timeout)");        }      }      if (wifiConnected && blynkConnected) break;      Serial.println("Retrying WiFi/Blynk in 10s …");      vTaskDelay(pdMS\_TO\_TICKS(10000));    }    vTaskDelete(NULL);  }  // 14) Task cập nhật dữ liệu lên Blynk (hàng 10s)  void blynkUpdateTask(void \*pvParameters) {    const TickType\_t delayTicks = pdMS\_TO\_TICKS(10000);    while (1) {      if (blynkConnected) {        Blynk.run();        Blynk.virtualWrite(VP\_TEMP,  temperature);        Blynk.virtualWrite(VP\_HUMID, humidity);        Blynk.virtualWrite(VP\_SOIL,  soilMoisturePercent);        Blynk.virtualWrite(VP\_MQ135, airQuality);      }      vTaskDelay(delayTicks);    }  }  // 15) Task mới để hiển thị thông tin  void displayInfoTask(void \*pvParameters) {    while (1) {      Serial.println("\n----------------------------------------");      Serial.println(">>> SENSOR READINGS:");      Serial.printf("DHT11 → Temperature: %.1f°C  Humidity: %.1f%%\n", temperature, humidity);      Serial.printf("Soil Moisture → Raw: %d  Percentage: %.1f%%\n", analogRead(SOIL\_MOISTURE\_PIN), soilMoisturePercent);      Serial.printf("MQ135 → Raw: %d  AQI: %.1f%%  Status: %s\n",                   analogRead(MQ135\_PIN),                   airQuality,                   airQuality > MQ135\_THRESHOLD ? "GOOD" : "BAD");      Serial.printf("Rain Sensor → Raw: %d  Status: %s\n",                   rainRawGlobal,                   rainRawGlobal < RAIN\_THRESHOLD ? "RAINING" : "NO RAIN");      Serial.printf("BH1750 → Light Level: %.1f lux\n", lux);        Serial.println("\n>>> DEVICE STATUS:");      Serial.printf("Canopy: %s  |  Fan: %s  |  Pump: %s\n",                   servoClose ? "CLOSED" : "OPEN",                   fanOn ? "ON" : "OFF",                   pumpOn ? "ON" : "OFF");      Serial.println("----------------------------------------");        vTaskDelay(pdMS\_TO\_TICKS(2000));    }  }  // --------------------------------------------------------------------  void setup() {    Serial.begin(115200);    // Khởi tạo I2C    Wire.begin(21, 22);    // Khởi tạo LED báo kết nối    pinMode(LED\_PIN, OUTPUT);    digitalWrite(LED\_PIN, LOW);    // Khởi tạo chân Input/Output    pinMode(SOIL\_MOISTURE\_PIN, INPUT);    pinMode(MQ135\_PIN,         INPUT);    pinMode(RAIN\_SENSOR\_PIN,   INPUT);    pinMode(PUMP\_RELAY\_PIN,    OUTPUT);    pinMode(FAN\_RELAY\_PIN,     OUTPUT);    // Đặt relay ở trạng thái OFF (HIGH) ban đầu    digitalWrite(PUMP\_RELAY\_PIN, HIGH);    digitalWrite(FAN\_RELAY\_PIN,  HIGH);    // Khởi tạo Servo ở góc 5°    myServo.attach(SERVO\_PIN);    myServo.write(openAngle);    // Khởi tạo mutex cho I2C    i2cMutex = xSemaphoreCreateMutex();    // Khởi tạo OLED    if (!display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR)) {      Serial.println(F("SSD1306 allocation failed"));      for (;;);    }    display.clearDisplay();    display.display();    // 1) Task kết nối WiFi/Blynk (core 1)    xTaskCreatePinnedToCore(      TaskConnectToWiFiAndBlynk,      "WiFiBlynkTask",      4096,      NULL,      3,      NULL,      1    );    // 2) Task cập nhật Blynk (core 1)    xTaskCreatePinnedToCore(      blynkUpdateTask,      "Blynk Update Task",      4096,      NULL,      1,      NULL,      1    );    // 3) Task đọc DHT11 (core 0)    xTaskCreatePinnedToCore(      DHTsensorTask,      "DHT Sensor Task",      2048,      NULL,      2,      NULL,      0    );    // 4) Task đọc Soil Moisture (core 0)    xTaskCreatePinnedToCore(      soilMoistureTask,      "Soil Moisture Task",      2048,      NULL,      2,      NULL,      0    );    // 5) Task đọc MQ135 (core 0)    xTaskCreatePinnedToCore(      airQualityTask,      "Air Quality Task",      2048,      NULL,      2,      NULL,      0    );    // 6) Task đọc Rain Sensor (core 0)    xTaskCreatePinnedToCore(      rainSensorTask,      "Rain Sensor Task",      2048,      NULL,      2,      NULL,      0    );    // 7) Task đọc BH1750 (core 0)    xTaskCreatePinnedToCore(      lightSensorTask,      "Light Sensor Task",      2048,      NULL,      2,      NULL,      0    );    // 8) Task hiển thị OLED (core 0)    xTaskCreatePinnedToCore(      oledUpdateTask,      "OLED Update Task",      3072,      NULL,      1,      NULL,      0    );    // 9) Task quản lý chế độ AUTO/REMOTE (core 1)    xTaskCreatePinnedToCore(      modeControlTask,      "Mode Control Task",      4096,      NULL,      2,      NULL,      1    );    // 10) Task xử lý lệnh điều khiển Blynk (core 1)    xTaskCreatePinnedToCore(      blynkControlTask,      "Blynk Control Task",      2048,      NULL,      2,  // Priority cao hơn      NULL,      1    );    // 11) Task hiển thị thông tin    xTaskCreatePinnedToCore(      displayInfoTask,      "Display Info Task",      4096,      NULL,      1,      NULL,      0    );  }  void loop() {    // Không cần code trong loop() vì đã dùng FreeRTOS Task    vTaskDelay(pdMS\_TO\_TICKS(10000));  } |

# TÀI LIỆU THAM khảo

1. Datasheet vi điều khiển ESP32

<https://www.alldatasheet.com/datasheet-pdf/view/1148023/ESPRESSIF/ESP32.html>

1. Datasheet cảm biến nhiệt độ độ ẩm DHT11

<https://www.alldatasheet.com/datasheet-pdf/view/1440068/ETC/DHT11.html>

1. Datasheet cảm biến chất lượng không khí MQ135

<https://www.alldatasheet.com/datasheet-pdf/view/1307647/WINSEN/MQ135.html>

1. Datasheet cảm biến ánh sáng BH1750

<https://www.alldatasheet.vn/datasheet-pdf/view/338083/ROHM/BH1750FVI.html>

1. Video Youtube kết nối ESP32 với DHT11

<https://www.youtube.com/watch?v=YHF7lN6Pm9c&list=PLNm9ck8XgwvgbHfENuOW3YRiYKM_hW8Jx&index=33>

1. Video Youtube kết nối ESP32 với MQ135

<https://www.youtube.com/watch?v=zz9KBPVDMhE>

1. Video Youtube kết nối ESP32 với BH1750

<https://www.youtube.com/watch?v=CpTige8HVuU>

1. Video Youtube kết nối ESP32 với cảm biến độ ẩm đất

<https://www.youtube.com/watch?v=CpvR50PXW2s&list=PLNm9ck8XgwvgbHfENuOW3YRiYKM_hW8Jx&index=53>

1. Video Youtube kết nối ESP32 với cảm biến mưa

<https://www.youtube.com/watch?v=-dBaQaQSzsc&list=PLNm9ck8XgwvgbHfENuOW3YRiYKM_hW8Jx&index=55>

1. Video Youtube kết nối ESP32 điều khiển Servo SG90

<http://youtube.com/watch?v=HB3eel8eD0o&list=PLNm9ck8XgwvgbHfENuOW3YRiYKM_hW8Jx&index=38>

1. Video Youtube kết nối ESP32 điều khiển relay 1 kênh

<https://www.youtube.com/watch?v=5v3hguC41Kk>