TRƯỜNG ĐẠI HỌC MỞ HÀ NỘI

**KHOA ĐIỆN - ĐIỆN TỬ**



ĐỒ ÁN 3

**THIẾT KẾ MẠCH ĐO VÀ THU THẬP GIÁ TRỊ NHIỆT ĐỘ, ĐỘ ẨM VÀ NỒNG ĐỘ BỤI**

Lớp học phần:

K25-ĐT2

Sinh viên thực hiện:

22A1201D0169

Ngô Thị Minh Ánh

22A1201D0031

Nguyễn Thị Ngọc Huệ

22A1201D0107

Giảng viên hướng dẫn:

Từ Viết Hoàng

HÀ NỘI , 09-2025

Nguyễn Thị Nhung

**LỜI NÓI ĐẦU**

Trong bối cảnh khoa học công nghệ phát triển mạnh mẽ, các hệ thống giám sát và điều khiển tự động ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực như nông nghiệp thông minh, y tế, môi trường và đời sống hằng ngày. Việc theo dõi các thông số môi trường như nhiệt độ, độ ẩm và nồng độ bụi đóng vai trò quan trọng trong việc đánh giá chất lượng không khí, bảo vệ sức khỏe con người cũng như nâng cao năng suất sản xuất.

Xuất phát từ nhu cầu thực tế đó, nhóm chúng em đã lựa chọn đề tài “Thiết kế mạch đo và thu thập giá trị nhiệt độ, độ ẩm và nồng độ bụi”. Đề tài nhằm mục tiêu xây dựng một hệ thống có khả năng đo lường chính xác các thông số môi trường, hiển thị và lưu trữ dữ liệu phục vụ cho việc giám sát.

Thông qua đồ án này, nhóm mong muốn củng cố kiến thức đã học về điện tử, vi điều khiển và IoT, đồng thời rèn luyện kỹ năng thiết kế mạch, lập trình và phân tích dữ liệu. Đề tài không chỉ mang tính học thuật mà còn có ý nghĩa thực tiễn cao, có thể phát triển và ứng dụng vào nhiều lĩnh vực khác nhau.

**LỜI CẢM ƠN**

Trước tiên, chúng em xin gửi lời cảm ơn sâu sắc đến thầy Từ Viết Hoàng, người đã tận tình hướng dẫn, truyền đạt kiến thức, và luôn sẵn lòng hỗ trợ nhóm em trong suốt quá trình thực hiện đồ án này. Sự chỉ bảo tận tâm của thầy không chỉ giúp chúng em hoàn thiện đồ án mà còn giúp chúng em hiểu sâu hơn về kiến thức chuyên ngành và rèn luyện tư duy kỹ thuật.

Chúng em cũng xin chân thành cảm ơn các thầy cô trong khoa Điện - Điện Tử trường Đại Học Mở Hà Nội, đã giảng dạy, truyền đạt kiến thức nền tảng và tạo điều kiện thuận lợi cho chúng em trong suốt quá trình học tập và nghiên cứu.

Đặc biệt, chúng em xin cảm ơn các bạn lớp K25-ĐT2, những người đã luôn động viên, chia sẻ tài liệu và hỗ trợ nhóm em trong suốt thời gian thực hiện đồ án. Những ý kiến đóng góp và sự hỗ trợ của các bạn đã giúp nhóm em vượt qua những khó khăn và hoàn thành tốt nhiệm vụ.

Cuối cùng, chúng em xin gửi lời tri ân sâu sắc đến gia đình, những người luôn là nguồn động viên to lớn, ủng hộ chúng em cả về vật chất lẫn tinh thần trong suốt chặng đường học tập và nghiên cứu.

Do năng lực bản thân còn nhiều hạn chế, đồ án của nhóm em chắc chắn không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự góp ý từ thầy cô và các bạn để tiếp tục hoàn thiện hơn trong tương lai.

Em xin chân thành cảm ơn!

**MỤC LỤC**

[Chương I. TỔNG QUAN VỀ ĐỀ TÀI NGHIÊN CỨU 1](#_Toc209713862)

[1. ĐẶT VẤN ĐỀ 1](#_Toc209713863)

[2. MỤC TIÊU VÀ PHẠM VI NGHIÊN CỨU ĐỀ TÀI 1](#_Toc209713864)

[2.1 Mục tiêu nghiên cứu 1](#_Toc209713865)

[2.2 Phạm vi nghiên cứu 1](#_Toc209713866)

[Chương II. CƠ SỞ LÝ THUYẾT 1](#_Toc209713867)

[1. TỔNG QUAN VỀ IOT 1](#_Toc209713868)

[1.1 Giới thiệu về Internet of Things (IoT) 2](#_Toc209713869)

[1.2 Lịch sử và sự phát triển của IoT 2](#_Toc209713870)

[1.3 Các đặc điểm của IoT 3](#_Toc209713871)

[1.4 Ứng dụng của IoT 4](#_Toc209713872)

[2. GIỚI THIỆU VỀ CÁC MODULE VÀ PHẦN MỀM SỬ DỤNG TRONG ĐỒ ÁN 6](#_Toc209713873)

[2.1 Vi điều khiển ESP32 6](#_Toc209713874)

[2.2 Opto PC817(4N35) 9](#_Toc209713875)

[2.3 Relay 5VDC 10](#_Toc209713876)

[2.4 Diode 1N4007 12](#_Toc209713877)

[2.5 Cảm biến DHT11 13](#_Toc209713878)

[2.6 Phần mềm PlatformIO 14](#_Toc209713879)

[Chương III. THIẾT KẾ VÀ THI CÔNG HỆ THỐNG 15](#_Toc209713880)

[1. NGUYÊN LÝ HOẠT ĐỘNG 15](#_Toc209713881)

[2. SƠ ĐỒ MẠCH 16](#_Toc209713882)

[3. MẠCH IN 17](#_Toc209713883)

[4. CODE PlatformIO 17](#_Toc209713884)

[Chương IV. Kết quả 32](#_Toc209713885)

[Chương V. KẾT LUẬN 32](#_Toc209713886)

[Chương VI. ỨNG DỤNG 33](#_Toc209713887)

MỤC LỤC HÌNH ẢNH

[Hình 1: Quá trình phát triển của IOT 2](#_Toc209713849)

[Hình 2: Ứng dụng của IoT 5](#_Toc209713850)

[Hình 3: Vi điều khiển ESP32 7](#_Toc209713851)

[Hình 4: Cấu hình chân của vi điều khiến ESP32 8](#_Toc209713852)

[Hình 5: Cấu tạo link kiện Opto PC817 9](#_Toc209713853)

[Hình 6: Relay 10](#_Toc209713854)

[Hình 7: Diode 1N4007 12](#_Toc209713855)

[Hình 8: Cảm biến DHT11 13](#_Toc209713856)

[Hình 9: Phần mềm PlatformIO 14](#_Toc209713857)

[Hình 10 : Sơ đồ mạch nguyên lí 16](#_Toc209713858)

[Hình 11: Sơ đồ mạch in 17](#_Toc209713859)

[Hình 12 : Sơ đồ mạch 32](#_Toc209713860)

1. TỔNG QUAN VỀ ĐỀ TÀI NGHIÊN CỨU
   1. ĐẶT VẤN ĐỀ

Ngày nay, vấn đề ô nhiễm môi trường và biến đổi khí hậu ngày càng trở nên nghiêm trọng, ảnh hưởng trực tiếp đến sức khỏe con người cũng như đời sống sản xuất. Các thông số môi trường như nhiệt độ, độ ẩm và nồng độ bụi là những chỉ số quan trọng để đánh giá chất lượng không khí, điều kiện sinh hoạt và sản xuất.

Trong khi đó, nhu cầu giám sát môi trường liên tục và chính xác ngày càng cấp thiết, đặc biệt trong các lĩnh vực như:

* Nông nghiệp: giám sát độ ẩm, nhiệt độ để tối ưu hóa việc tưới tiêu, trồng trọt.
* Y tế – sức khỏe: theo dõi chất lượng không khí để cảnh báo sớm về các tác nhân gây bệnh.
* Đời sống đô thị: giám sát ô nhiễm bụi mịn nhằm nâng cao chất lượng sống.

Xuất phát từ yêu cầu đó, đề tài “Thiết kế mạch đo và thu thập giá trị nhiệt độ, độ ẩm và nồng độ bụi” được thực hiện, với mong muốn xây dựng một hệ thống có khả năng đo, thu thập và hiển thị các thông số môi trường một cách trực quan và chính xác.

* 1. MỤC TIÊU VÀ PHẠM VI NGHIÊN CỨU ĐỀ TÀI
     1. Mục tiêu nghiên cứu

Thiết kế và xây dựng mạch điện tử có khả năng đo các thông số: nhiệt độ, độ ẩm, nồng độ bụi.

Thu thập, xử lý và hiển thị dữ liệu cảm biến lên màn hình hoặc gửi đến máy tính.

Ứng dụng vi điều khiển ESP32 để điều khiển, lập trình và quản lý dữ liệu.

Đảm bảo mạch hoạt động ổn định, có thể mở rộng cho các nghiên cứu hoặc ứng dụng thực tế khác.

* + 1. Phạm vi nghiên cứu

Đối tượng nghiên cứu: các cảm biến đo nhiệt độ, độ ẩm (DHT11/DHT22) và cảm biến bụi (Sharp GP2Y10 hoặc tương tự).

Chỉ tập trung vào việc đo và thu thập dữ liệu môi trường cơ bản (nhiệt độ, độ ẩm, nồng độ bụi).

Kết quả được hiển thị trên giao diện Web hoặc Blynk để quan sát.

Không đi sâu vào các giải pháp xử lý, cải thiện môi trường hoặc các hệ thống IoT quy mô lớn.

1. CƠ SỞ LÝ THUYẾT
   1. TỔNG QUAN VỀ IOT
      1. Giới thiệu về Internet of Things (IoT)

Internet vạn vật (IoT - Internet of Things) là mạng kết nối các đồ vật, thiết bị thông qua cảm biến, phần mềm và nhiều công nghệ khác, cho phép những đồ vật, thiết bị đó thu thập, trao đổi dữ liệu với nhau.

Internet vạn vật lan tỏa lợi ích của mạng internet tới mọi đồ vật được kết nối, không chỉ dừng lại ở phạm vi một chiếc máy tính. Khi được kết nối với internet, nó sẽ trở nên thông minh hơn nhờ khả năng gửi, nhận thông tin và tự động hoạt động dựa trên thông tin đó.

Thiết bị IoT có thể là đồ vật được gắn cảm biến để thu thập dữ liệu về môi trường xung quanh (như những giác quan), máy tính, bộ điều khiển tiếp nhận dữ liệu, sau đó ra lệnh cho các thiết bị khác, hoặc cũng có thể là đồ vật được tích hợp cả hai tính năng trên.

Hệ thống IoT hoàn chỉnh đều cần phải thực hiện qua trình đủ 4 bước: Thu thập dữ liệu, chia sẻ dữ liệu, xử lý dữ liệu và đưa ra quyết định.

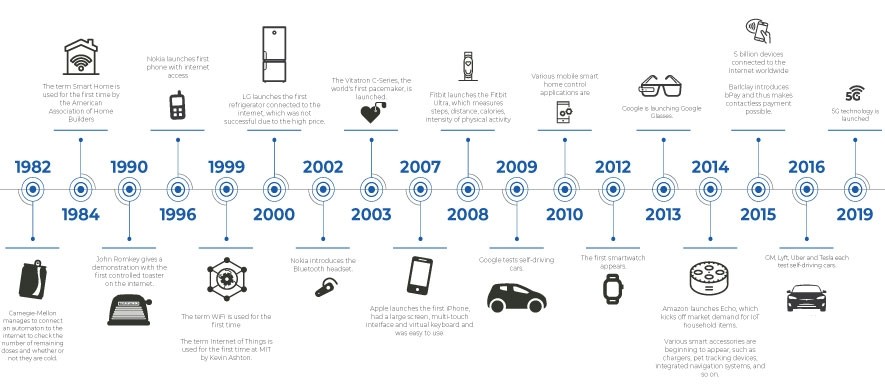
* + 1. Lịch sử và sự phát triển của IoT

**Năm 1982 - 1990: Giai đoạn sơ khai**

Trong những năm từ 1982 đến 1990, IoT lúc này chưa có gì là rõ ràng khi đây vẫn chỉ là ý tưởng đưa các cảm biến, trí thông minh vào các vật, mang đến một mạng lưới thiết bị thông minh. Tuy nhiên tiến độ thực hiện các ý tưởng này được diễn ra khá chậm, vì bấy giờ công nghệ chưa đạt đủ trình độ để thực hiện hóa.

**Năm 1991 - 1994: Những khái niệm phổ quát đầu tiên**

Năm 1991, Mark Weiser đã đưa ra khái niệm về điện toán phổ quát, cho thấy tầm nhìn của một môi trường máy tính toàn cầu với các thiết bị nhúng thông minh. Năm 1994, Reza Raji mô tả khái niệm IoT như "chuyển các gói dữ liệu nhỏ sang tập hợp các nút mạng lớn, để tích hợp và tự động hóa mọi thứ từ các thiết bị gia dụng với cả một nhà máy sản xuất".



Hình 1: Quá trình phát triển của IOT

**Năm 1999: Khái niệm IoT ra đời và bắt đầu phổ biến**

Khái niệm IoT được đưa ra lần đầu tiên vào năm 1999 bởi Kevin Ashton, một nhà nghiên cứu tại MIT. Tuy nhiên, công nghệ cần thiết để thực hiện ý tưởng này chưa sẵn sàng cho đến khi Internet và công nghệ không dây trở nên phổ biến hơn.

Trong những năm tiếp theo, nhiều công ty đã đưa ra các giải pháp IoT trong đó có Microsoft và Novell là nổi bật nhất. Năm 1999, Bill Joy đã đề xuất phương thức truyền tải thiết bị-tới-thiết bị (D2D) trong bộ khung "Six Webs" của ông, được ông trình bày tại Diễn đàn Kinh tế Thế giới ở Davos.

Trong cùng năm 1999, Trung tâm Auto-ID tại Viện Công nghệ Massachusetts đã đưa ra khái niệm Internet Vạn Vật và công nghệ Nhận dạng qua tần số vô tuyến (RFID) được xem là một điều kiện tiên quyết cho IoT. Từ đó, IoT đã trở nên phổ biến và phát triển nhanh chóng, với sự hội tụ của nhiều công nghệ như truyền tải vô tuyến, phân tích dữ liệu thời gian thực, học máy, cảm biến và hệ thống nhúng.

**Năm 2000 - 2013: Tích hợp các công nghệ mới cho IoT**

Các công nghệ IoT đã được phát triển rộng rãi, bao gồm các chuẩn giao tiếp IoT như: MQTT, CoAP, Zigbee, Z-Wave, BLE, WiFi, và 6LoWPAN. Ngoài ra, việc tăng cường khả năng kết nối với IPv6 đã cho phép IoT trở nên phổ biến hơn và dễ dàng kết nối với các thiết bị khác nhau trên toàn cầu.

**Năm 2013-2016: Phát triển công nghệ IoT**

Trong giai đoạn này, IoT đã được tích hợp với các công nghệ mới như điện toán đám mây và Big Data để thu thập và phân tích dữ liệu từ các thiết bị IoT. Internet of Everything (IoE)\* và 5G cũng đã được giới thiệu vào năm 2016, đưa IoT đến một tầm cao mới về tốc độ truyền thông và khả năng kết nối.

*\*Chú thích: Internet of Everything (IoE) là một khái niệm mở rộng hơn, đưa ra một khái niệm về mạng lưới các thực thể được kết nối với nhau, bao gồm cả con người, các thiết bị, quy trình và dịch vụ, nhằm tạo ra các giá trị kinh tế và xã hội mới. IoE tập trung vào việc kết nối không chỉ các thiết bị IoT mà cả những người dùng, hệ thống và quy trình.*

**Năm 2016 - nay: Phát triển cùng cách mạng công nghiệp 4.0**

IoT đã trở thành một phần quan trọng trong cuộc cách mạng công nghiệp 4.0, với sự hội tụ của nhiều công nghệ như truyền tải vô tuyến, phân tích dữ liệu thời gian thực, học máy, cảm biến và hệ thống nhúng. IoT cũng đã được áp dụng rộng rãi trong nhiều lĩnh vực từ nhà thông minh đến tự động hóa công trình.

* + 1. Các đặc điểm của IoT

**Cấu trúc hệ thống IoT có thể được chia thành 5 thành phần chính, bao gồm:**

-Thiết bị IoT (IoT devices): Đây là các thiết bị vật lý được kết nối với Internet và có khả năng thu thập, xử lý và truyền dữ liệu. Một số vật dụng có thể là thiết bị IoT như: đồ dùng y tế, đồ gia định, vật dụng trong ngành công nghiệp,...

-Kết nối (connectivity): Đây là thành phần cung cấp khả năng kết nối giữa các thiết

bị IoT với nhau và với Internet. Hệ thống sẽ dùng các công nghệ riêng biệt để thực hiện kết nối IoT như: Bluetooth, Zigbee, Wifi, Lora,...

-Cơ sở hạ tầng (infrastructure): Đây là thành phần cung cấp nền tảng cho hoạt động của hệ thống IoT. Một cơ sở hạ tầng của IoT tập hợp những yếu tố gồm: trung tâm dữ liệu, mạng lưới,...

-Dữ liệu (data): Đây là thành phần quan trọng nhất của hệ thống IoT. Các dữ liệu được thu thập từ các thiết bị IoT sẽ phục vụ cho việc phân tích, triển khai hành động và đưa ra quyết định.

-Ứng dụng (applications): Đây là thành phần sử dụng dữ liệu từ các thiết bị IoT để cung cấp các giá trị cho người dùng. Bạn có thể sử dụng IoT trong nhiều lĩnh vực như: chăm sóc sức khỏe, y tế, nông nghiệp, sản xuất, công nghiệp,..

**Ưu điểm:**

-Tăng khả năng truy cập thông từ mọi thiết bị, ở mọi lúc và mọi nơi.

-Các thiết bị điện tử có thể kết nối với nhau một cách dễ dàng.

-Chuyển các gói dữ liệu qua mạng được kết nối tiết kiệm thời gian và tiền bạc.

-Hệ thống IoT giúp chất lượng dịch vụ của doanh nghiệp được cải thiện, đồng thời giảm sự can thiệp của người dùng thông qua việc tự động hóa các nhiệm vụ.

**Nhược điểm:**

-Rủi ro hacker đánh cắp thông tin mật tăng cao vì số lượng các thiết bị kết nối và chia sẻ thông tin tăng.

-Các doanh nghiệp gặp phải thách thức lớn trong việc thu thập, phân tích và quản lý dữ liệu từ mọi thiết bị.

-Các thiết bị từ các nhà sản xuất khó cơ cơ hội liên kết với nhau, vì hệ thống IoT chưa đạt tiêu chuẩn quốc tế về khả năng tương thích.

* + 1. Ứng dụng của IoT

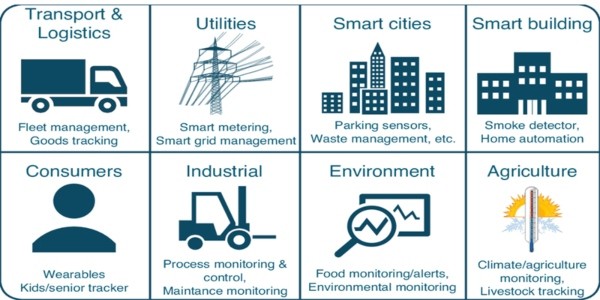
Tính linh hoạt của IoT làm cho nó trở thành công cụ đắc lực đối với rất nhiều doanh nghiệp, tổ chức, cơ quan chính phủ,... Dưới đây là một số ví dụ rõ nét nhất về ứng dụng của Internet vạn vật:

**Ứng dụng IoT trong nông nghiệp thông minh:** Đối với việc trồng trọt trong nhà kính, IoT biến việc giám sát, quản lý các điều kiện khí hậu vi mô thành hiện thực, từ đó giúp tăng sản lượng. Đối với việc trồng cây ngoài trời, thiết bị sử dụng công nghệ IoT có thể cảm nhận được độ ẩm, chất dinh dưỡng của đất, kết hợp với dữ liệu thời tiết giúp kiểm soát tốt hơn hệ thống tưới tiêu và bón phân thông minh.

**Ứng dụng IoT trong ngôi nhà thông minh:**

Thiết bị đeo bao gồm điện thoại, đồng hồ, kính thực tế ảo, máy theo dõi sức khỏe,... cải thiện khả năng giải trí, kết nối mạng, theo dõi sức khỏe và luyện tập thể chất.

Nhà thông minh đảm nhận những việc như kích hoạt kiểm soát môi trường để ngôi nhà luôn ở trạng thái thoải mái nhất cho người ở. Bữa tối được nấu từ xa, thức ăn đã sẵn sàng trước khi chủ nhân quay trở về. An ninh cũng dễ dàng hơn nhờ khả năng điều khiển thiết bị theo dõi, cảnh báo và hệ thống điện từ xa, đồng thời kích hoạt khóa thông minh, cho phép những người thích hợp ra vào nhà ngay cả khi họ không có chìa khóa.



Hình 2: Ứng dụng của IoT

**Ứng dụng IoT trong Y tế:**

Đầu tiên và quan trọng nhất, thiết bị đeo IoT cho phép bệnh viện, chuyên gia theo dõi sức khỏe của bệnh nhân tại nhà. Việc đưa cảm biến IoT vào các thiết bị quan trọng khác cũng giúp giảm tình trạng hỏng hóc và tăng độ tin cậy khi khám chữa bệnh.

Điều này góp phần giảm thời gian điều trị nội trú trong nhờ cung cấp thông tin theo thời gian thực, chính xác đến từng phút, đảm bảo bệnh nhân được chăm sóc tốt khi điều trị ngoại trú. Trong bệnh viện, giường thông minh thông báo cho nhân viên y tế về tình trạng sẵn có nhằm cắt giảm thời gian chờ đợi.

Quá trình chăm sóc người cao tuổi trở nên thoải mái hơn. Ngoài việc theo dõi thời gian thực tại nhà nêu trên, cảm biến cũng có thể xác định bệnh nhân có bị ngã hay bị đau tim hay không.

**Ứng dụng IoT trong sản xuất:**

Công nghệ RFID và GPS có thể giúp doanh nghiệp theo dõi sản phẩm từ khi bắt đầu sản xuất trên dây chuyền ở nhà máy đến vị trí bày bán tại cửa hàng đích, toàn bộ chuỗi cung ứng từ đầu đến cuối. Những cảm biến này thu thập thông tin về thời gian di chuyển, tình trạng sản phẩm và điều kiện môi trường mà sản phẩm phải chịu.

Cảm biến được gắn vào thiết bị trong nhà máy giúp xác định "nút thắt cổ chai" (Bottleneck) trong dây chuyền, do đó giảm thiểu thời gian sản xuất, tránh lãng phí. Một số cảm biến khác cũng có thể theo dõi hiệu suất của máy, dự đoán khi nào thiết bị sẽ cần bảo trì, do đó ngăn ngừa hỏng hóc nặng nề, gây tốn kém nhiều chi phí.

**Ứng dụng IoT trong Giao thông vận tải:**

Đến thời điểm này, hầu hết mọi người đã nghe nói về những tiến bộ đạt được với ô tô tự lái. Nhưng đó chỉ là một phần của tiềm năng to lớn trong lĩnh vực giao thông vận tải. GPS, một ví dụ khác của IoT, đang được sử dụng để giúp các công ty vận tải vạch ra tuyến đường nhanh hơn, hiệu quả hơn cho xe tải vận chuyển hàng hóa, nhờ đó đẩy nhanh thời gian giao hàng.

Ngoài ra, nhà quy hoạch thành phố cũng có thể sử dụng dữ liệu GPS đó nhằm xác định mô hình giao thông, nhu cầu chỗ đậu xe cũng như xây dựng và bảo trì đường bộ.

* 1. GIỚI THIỆU VỀ CÁC MODULE VÀ PHẦN MỀM SỬ DỤNG TRONG ĐỒ ÁN
     1. Vi điều khiển ESP32

ESP32 WROOM 32 Micro được tích hợp anten và RF, hoạt động tiết kiệm năng lượng, ổn định, chống nhiễu tốt, đây là giải pháp chi phí thấp nhất cho 1 dự án với một mạch sử dụng wifi 2.4Ghz và bluethooth TSMC công nghệ 40nm năng lượng thấp.

**Một số ứng dụng chính:** ESP32 là một vi điều khiển mạnh mẽ tích hợp WiFi và Bluetooth, được sử dụng rất rộng rãi trong các dự án IoT và hệ thống nhúng. Một số ứng dụng tiêu biểu:

1. Smart Home (Nhà thông minh):

* Điều khiển thiết bị điện qua điện thoại (đèn, quạt, điều hòa...).
* Hệ thống khóa cửa thông minh, giám sát an ninh bằng cảm biến.

1. Nông nghiệp thông minh:

* Giám sát nhiệt độ, độ ẩm đất, ánh sáng, nồng độ bụi.
* Tự động tưới tiêu, chiếu sáng cây trồng.

1. Giám sát môi trường:

* Đo và cảnh báo nồng độ bụi mịn, khí độc hại.
* Trạm thời tiết mini (đo nhiệt độ, độ ẩm, áp suất).

1. Thiết bị đeo thông minh (Wearables):

* Vòng tay theo dõi sức khỏe, nhịp tim.
* Thiết bị thể dục, đếm bước, kết nối Bluetooth với điện thoại.

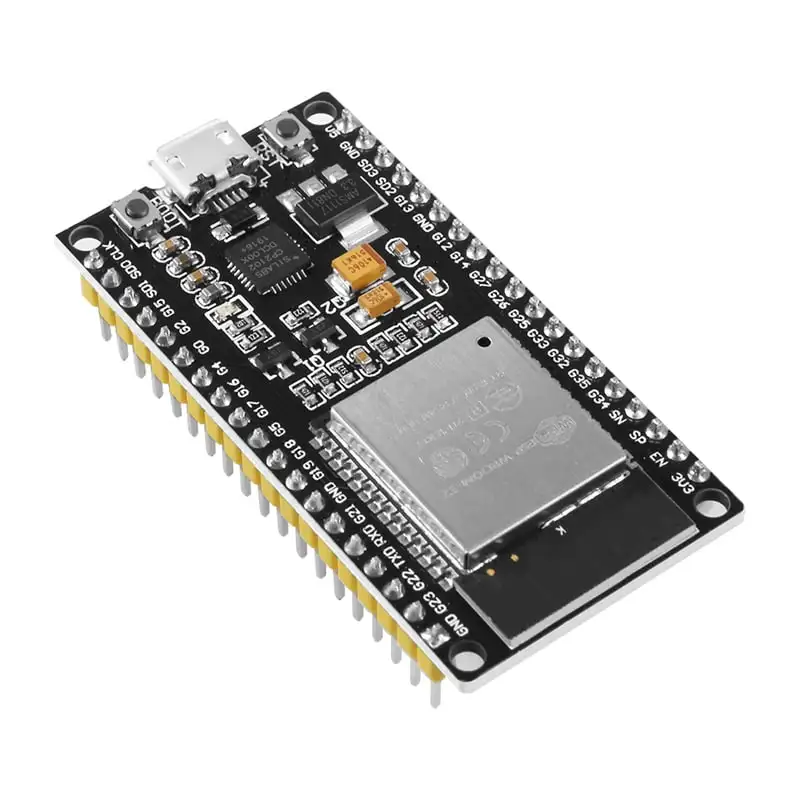
1. Công nghiệp – Tự động hóa:

* Kết nối cảm biến trong dây chuyền sản xuất.
* Gửi dữ liệu về server qua WiFi/MQTT.

1. Các ứng dụng học tập và nghiên cứu:

* Học lập trình IoT.
* Xây dựng prototype (mẫu thử) cho sản phẩm.

**Mạch nạp:** ESP32 có thể nạp chương trình bằng cổng USB to UART( CP2101/CH340/FT232 ) có tích hợp sẵn trên board phát triển (ESP32 Devkit V1 , NodeMCU-32S…).



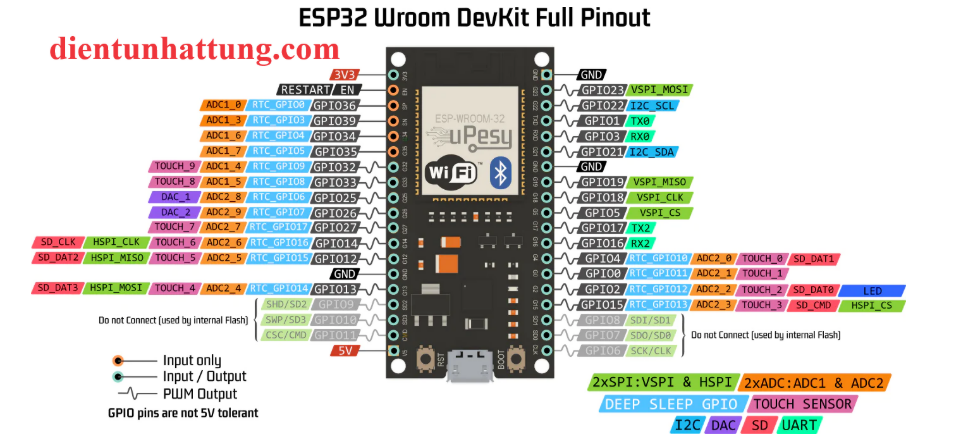
Hình 3: Vi điều khiển ESP32

**Cấu hình chi tiết của ESP32:**

**Bộ nhớ**

* 448KByte bộ nhớ Plash ROM
* 8KByte cho SRAM  in RTC
* 250KB SRAM (16 KB for cache)

.



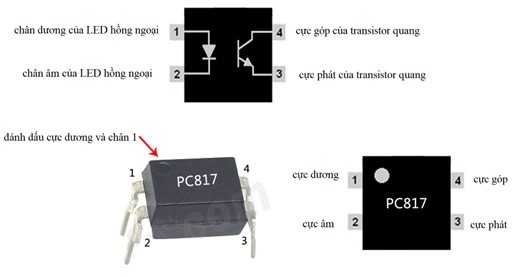
Hình 4: Cấu hình chân của vi điều khiến ESP32

**Thông số cơ bản ESP32 WROOM 32 Micro**

* **Điện áp nguồn (USB):** 5V DC
* **Đầu vào/Đầu ra điện áp:** 3.3V DC
* **Dòng điện :** 5μA trong hệ thống treo chế độ
* **In/OUT :**24
* **Mô hình :** ESP32 38 chân
* **Thạch anh :**240Mhz
* **Các chuẩn giao tiếp :**I²C, SPI, UART / USART, USB, CAN
* **Nhiệt độ hoạt động  :** -40°C ~ 85°C
* **Cổng ADC  :**12bit – 18 Kênh
* **Chip USB-Serial:** CP2102
* **Timer  :**Có

**Thông số Wifi và Bluetooth**

* Loại: Wifi + Bluetooth Module
* Hiệu suất: Lên đến 600 DMIPS
* Tần số: lên đến 240MHz
* Wifi: 802.11 B/g/n/E/I (802.11N @ 2.4 GHz lên đến 150 Mbit/S)
* Bluetooth: 4.2 BR/EDR BLE 2 chế độ điều khiển
* Ăng ten: PCB
* Bảo mật: IEEE 802.11, bao gồm cả WFA, WPA/WPA2 và WAPI
* Phần cứng tăng tốc mật mã học: AES, SHA-2, RSA, hình elip mật mã Đường Cong (ECC), số ngẫu nhiên Máy phát điện (RNG)
  + 1. Opto PC817(4N35)



Hình 5: Cấu tạo link kiện Opto PC817

PC817 bao gồm một diode phát quang hồng ngoại (IR LED) và một phototransistor được ghép quang học trong cùng một vỏ bọc nhỏ gọn. Nó hoạt động bằng cách truyền tín hiệu điện qua ánh sáng mà không cần kết nối điện trực tiếp giữa đầu vào và đầu ra, giúp cách ly hai phần mạch về mặt điện.

**Cấu trúc ( 4 chân ):**

-2 chân dành cho IR LED (Anode và Cathode).

-2 chân dành cho phototransistor (Collector và Emitter).

**Nguyên lý hoạt động:**

-Khi dòng điện chạy qua IR LED (thường cần điện áp khoảng 1.25V), nó phát ra ánh sáng hồng ngoại.

-Ánh sáng này kích hoạt phototransistor ở phía đầu ra, làm cho nó dẫn điện giữa Collector và Emitter, từ đó truyền tín hiệu mà không cần kết nối vật lý trực tiếp.

**Thông số kỹ thuật chính:**

-Điện áp cách ly: Lên đến 5kV (tùy phiên bản), đảm bảo an toàn cho các ứng dụng có sự chênh lệch điện áp lớn.

-Dòng điện tối đa qua phototransistor: 50mA.

-Thời gian đáp ứng: Rise time và fall time khoảng 18µs, phù hợp với tín hiệu tần số thấp đến trung bình (dưới 100kHz).

-Điện áp đầu vào (IR LED): Khoảng 1.2-1.5V, dòng điện khuyến nghị 5-20mA.

-Nhiệt độ hoạt động: Từ -30°C đến +100°C.

**Ứng dụng:**

-Cách ly tín hiệu: Được sử dụng trong giao tiếp giữa vi điều khiển (MCU) và các thiết bị ngoại vi để bảo vệ MCU khỏi điện áp cao hoặc nhiễu.

-Loại bỏ nhiễu: Giảm thiểu tác động của nhiễu điện từ trong các hệ thống điện tử.

-Điều khiển tải nhỏ: Dùng để bật/tắt các tải DC nhỏ hoặc làm cầu nối giữa các mức điện áp khác nhau (ví dụ: từ 5V sang 24V).

-Ứng dụng phổ biến: Nguồn cấp điện switching (SMPS), mạch giao tiếp I/O, cảm biến cách ly.

**Ưu điểm:**

-Kích thước nhỏ gọn: Đóng gói dạng DIP-4, dễ tích hợp vào các bo mạch.

-Giá thành rẻ: Là một trong những optocoupler phổ thông và kinh tế nhất trên thị trường.

-Dễ sử dụng: Không yêu cầu cấu hình phức tạp, chỉ cần thêm điện trở giới hạn

dòng cho LED và thiết kế mạch đơn giản ở phía phototransistor.

**Hạn chế:**

-Tốc độ giới hạn: Không phù hợp với các ứng dụng yêu cầu tốc độ chuyển mạch rất cao (ví dụ: trên 1MHz).

-Không có chân Base: Không thể điều chỉnh độ nhạy hoặc tốc độ của phototransistor như một số optocoupler khác (ví dụ: 4N35).

* + 1. Relay 5VDC



Hình 6: Relay

Relay 5VDC là một loại rơ-le điện cơ hoạt động với điện áp điều khiển 5V DC. Đây là linh kiện quan trọng trong các ứng dụng điện tử, cho phép cách ly và điều khiển các tải công suất lớn bằng tín hiệu điện áp thấp từ vi điều khiển hoặc các mạch logic.

**Cấu trúc và nguyên lý hoạt động**

Relay 5VDC bao gồm:

**Cuộn dây điện từ**: Được kích hoạt bởi điện áp 5V DC, tạo ra từ trường để đóng/mở tiếp điểm.

**Tiếp điểm**: Có thể là loại thường mở (NO - Normally Open) hoặc thường đóng (NC - Normally Closed).

**Vỏ cách điện**: Đảm bảo an toàn và cách ly giữa mạch điều khiển và mạch tải.

Khi dòng điện chạy qua cuộn dây (thường khoảng 70–100mA), từ trường hút tiếp điểm cơ khí, chuyển trạng thái từ mở sang đóng hoặc ngược lại, từ đó bật/tắt tải ở phía đầu ra. **Thông số kỹ thuật chính**

Điện áp điều khiển: 5V DC.

Dòng điện cuộn dây: Khoảng 70–100mA (tùy model).

Điện áp tải tối đa: Thường hỗ trợ lên đến 250V AC hoặc 30V DC.

Dòng điện tải tối đa: 10A (tùy model, ví dụ relay SRD-05VDC-SL-C).

Nhiệt độ hoạt động: −25◦C đến 70◦C.

**Ứng dụng**

Relay 5VDC được sử dụng phổ biến trong:

Điều khiển thiết bị gia dụng (đèn, quạt, máy bơm) từ vi điều khiển như Arduino hoặc Raspberry Pi.

Cách ly mạch công suất cao khỏi mạch điều khiển điện áp thấp.

Tự động hóa công nghiệp và hệ thống IoT.

**Ưu điểm và hạn chế**

**Ưu điểm**: Dễ sử dụng, chi phí thấp, khả năng cách ly tốt.

**Hạn chế**: Kích thước lớn hơn so với linh kiện bán dẫn (như transistor), tốc độ chuyển mạch chậm (vài ms), và tiêu thụ dòng điện lớn hơn.

Relay 5VDC là giải pháp lý tưởng cho các ứng dụng cần điều khiển tải lớn một cách an toàn và hiệu quả, đặc biệt trong các dự án điện tử DIY hoặc tự động hóa.

* + 1. Diode 1N4007



Hình 7: Diode 1N4007

Diode 1N4007 là một loại diode chỉnh lưu silicon phổ biến thuộc dòng 1N400x, được sử dụng rộng rãi trong các mạch điện tử để chỉnh lưu dòng điện xoay chiều (AC) thành dòng một chiều (DC) hoặc bảo vệ mạch khỏi dòng ngược.

**Cấu trúc và đặc điểm:**Diode 1N4007 có cấu trúc gồm một mối nối P-N, được đóng gói trong vỏ DO-41 (vỏ nhựa hình trụ nhỏ gọn). Nó có hai chân: cực anode (A) và cathode (K), với vòng trắng trên vỏ đánh dấu phía cathode.

**Thông số kỹ thuật chính:**

**Điện áp ngược tối đa (***VRRM***)**: 1000V – Điện áp ngược lớn nhất mà diode có thể chịu được.

**Dòng điện thuận trung bình (***IF*(*AV*)**)**: 1A – Dòng điện tối đa trong điều kiện bình thường.

**Dòng điện xung thuận tối đa (***IFSM***)**: 30A (trong thời gian ngắn).

**Điện áp thuận (***VF***)**: Khoảng 0.7V đến 1V khi dẫn dòng 1A.

**Nhiệt độ hoạt động**: Từ −55◦C đến +150◦C.

**Nguyên lý hoạt động:** Diode 1N4007 chỉ cho phép dòng điện chạy từ anode sang cathode (hướng thuận), với sụt áp khoảng 0.7V. Trong hướng ngược, nó chặn dòng điện cho đến khi điện áp vượt quá 1000V (điện áp đánh thủng).

**Ứng dụng:**

**Chỉnh lưu nguồn**: Sử dụng trong bộ nguồn AC-DC, như cầu diode trong adapter.

**Bảo vệ mạch**: Ngăn dòng ngược trong mạch pin hoặc động cơ.

**Chặn điện áp**: Xử lý điện áp ngược từ cuộn cảm hoặc biến áp.

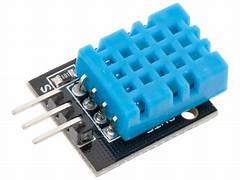
**Ưu điểm và hạn chế:**

**Ưu điểm**: Chịu điện áp ngược cao (1000V), giá rẻ, dễ sử dụng.

**Hạn chế**: Tốc độ chuyển mạch chậm, chỉ chịu dòng 1A liên tục, không phù hợp với tần số cao.

Diode 1N4007 là linh kiện đa dụng, đáng tin cậy cho các ứng dụng chỉnh lưu và bảo vệ mạch cơ bản, đặc biệt phù hợp với các dự án điện tử DIY và nguồn điện nhỏ.

* + 1. Cảm biến DHT11



Hình 8: Cảm biến DHT11

Trên thị trường hiện nay có rất nhiều loại cảm biến nhiệt độ với các đặc tỉnh khác nhau để phù hợp với nhiều mục đích sử dụng như LM335, LM35, DHT11, Pt100, Với đề tài này, em lựa chọn sử dụng DHT11 vì nó tích hợp đo cá nhiệt độ và độ ẩm

Cảm biến độ ẩm và nhiệt độ DHT11 là cảm biến rất thông dụng hiện nay vì chi phí rẻ và rất dễ lấy dữ liệu thông qua giao tiếp 1 wire (giao tiếp digital 1 dây truyền dữ liệu duy nhất). Cảm biến nhiệt độ và độ ẩm DHT11 có bộ điều chỉnh nhiệt độ và độ ẩm với đầu ra tín hiệu số được hiệu chuẩn qua bộ tiền xử lý tín hiệu tích hợp trong cảm biến giúp bạn có được dữ liệu chính xác mà không phải qua bất kỳ tỉnh toán nào. Với việc sử dụng tín hiệu kỹ thuật cao nên cảm biến luôn cho độ tin cậy cao và ổn định trong thời gian dài. Cảm biến này bao gồm một thành phần đo độ ẩm kiểu điện trở và bộ phận giảm nhiệt độ NTC, và kết nổi với bộ vi điều khiển 8 bit hiệu suất cao, cung cấp chất lượng tốt, phản ứng nhanh, chống nhiều và hiệu quả về chỉ phi.

Mỗi cảm biển DHT11 đều được hiệu chuẩn trong phòng thí nghiệm để có độ chính xác cao nhất. Sự kết nối hệ thống nổi tiếp một đây nhanh chóng và dễ dàng , truyền tín hiệu lên tới 20m.

**Thông số kỹ thuật:**

Điện áp hoạt động: 3.3-5V

Dải độ ẩm hoạt động: 20

Dải nhiệt độ hoạt động: OC - 50C, sai số C2

Khoảng cách truyền tối đa: 20m

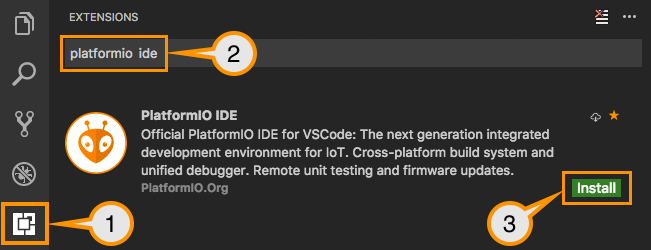
Chuẩn giao tiếp: TTL, 1-wire

Kích thước: 28x12x10mm

Dòng tối đa: 2.5mA

Tần số lấy mẫu tối đa: 1Hz

* + 1. Phần mềm PlatformIO



Hình 9: Phần mềm PlatformIO

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

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

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

Những ưu điểm của PlatformIO

PlatformIO giúp người dùng viết chương trình nhanh chóng với các tính năng như hoàn tất code C/C ++ và công cụ viết hỗ trợ viết code thông minh Linter để phát triển dự án chuyên nghiệp và nhanh chóng. Các tính năng này vốn không có trong Arduino IDE. Ngoài ra, PlatformIO cung cấp các theme với các màu tối và sáng để thay đổi bất cứ lúc nào. Nó cũng đi kèm với điều hướng code thông minh và định dạng code. Các tính năng cốt lõi bao gồm hệ thống xây dựng đa nền tảng, trình quản lý thư viện, theo dõi cổng nối tiếp, v.v.

Trình gỡ lỗi hợp nhất với sự hỗ trợ cho nhiều kiến trúc và nền tảng phát triển cho phép gỡ lỗi nhiều board nhúng với Zero-Configuration. Trình gỡ lỗi hợp nhất PlatformIO có các tính năng như điểm ngắt có điều kiện (Conditional Breakpoint), biểu thức (Expression) và điểm theo dõi (Watchpoint), trình xem bộ nhớ (Memory Viewer), khởi động lại nóng phiên gỡ lỗi đang hoạt động.

PlatformIO Core được viết bằng Python 3.6 (tại thời điểm viết bài này) và hoạt động trên Windows, macOS, Linux, FreeBSD và thậm chí cả các máy tính có kích thước nhỏ dựa trên ARM như Raspberry Pi, BeagleBone, CubieBoard, Samsung ARTIK, v.v. Ngoài ra, PlatformIO còn có File Explorer giúp tổ chức các tập tin khi dự án phát triển đến một mức nhất định và việc tổ chức trở nên cần thiết.

1. THIẾT KẾ VÀ THI CÔNG HỆ THỐNG
   1. NGUYÊN LÝ HOẠT ĐỘNG

Hệ thống sử dụng ESP32 làm vi điều khiển trung tâm, có nhiệm vụ thu thập dữ liệu từ các cảm biến, xử lý và truyền dữ liệu đến các nền tảng giám sát (web và Blynk). Nguyên lý hoạt động có thể mô tả theo các bước sau:

1. Thu thập dữ liệu cảm biến

* ESP32 đọc dữ liệu từ các cảm biến:
* Cảm biến nhiệt độ – độ ẩm (ví dụ: DHT11/DHT22).
* Cảm biến bụi (ví dụ: Sharp GP2Y10 hoặc SDS011).
* Các giá trị đo được sẽ được xử lý và chuyển đổi sang dạng số để lưu trữ, hiển thị.

1. Truyền dữ liệu lên server/web và ứng dụng Blynk

* ESP32 kết nối WiFi và gửi dữ liệu lên:
* Web server (hiển thị bằng giao diện HTML/JavaScript).
* Blynk Cloud (ứng dụng Blynk trên điện thoại).
* Người dùng có thể theo dõi nhiệt độ, độ ẩm và nồng độ bụi theo thời gian thực.

1. Điều khiển chế độ hoạt động (Tự động/Thủ công)

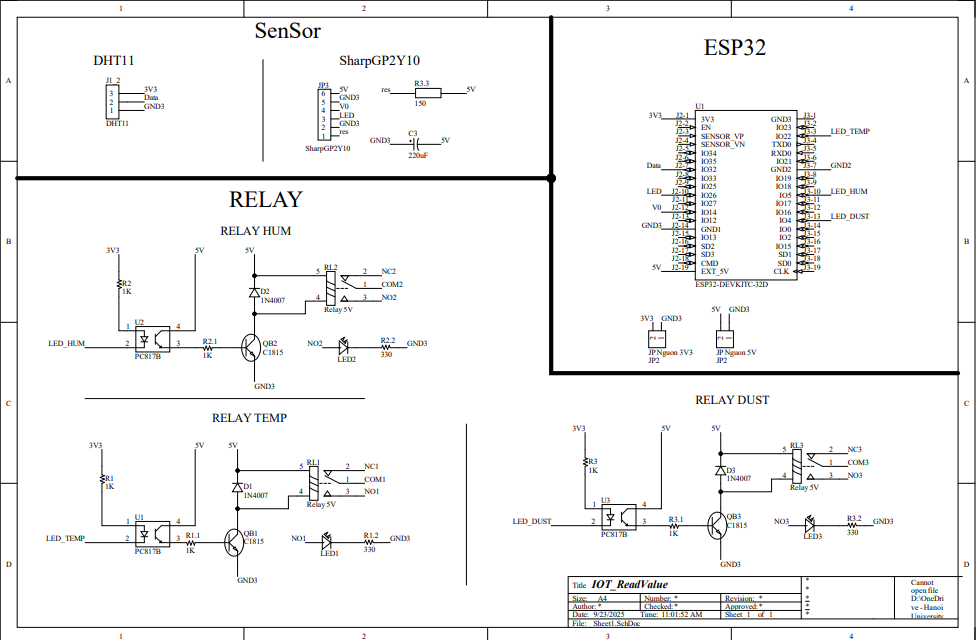
* Chế độ tự động:
* ESP32 so sánh giá trị cảm biến với ngưỡng cài đặt.
* Nếu vượt ngưỡng → ESP32 kích hoạt Relay để bật/tắt thiết bị (quạt lọc, máy bơm, đèn cảnh báo…).
* Chế độ thủ công:
* Người dùng có thể bật/tắt Relay trực tiếp trên Blynk mà không phụ thuộc vào giá trị cảm biến.

1. Cập nhật ngưỡng điều khiển từ xa

* Người dùng nhập giá trị ngưỡng mong muốn trên Web hoặc Blynk.
* ESP32 nhận dữ liệu và lưu lại trong bộ nhớ (Preferences/SPIFFS).
* Từ đó hệ thống sẽ so sánh cảm biến với ngưỡng mới để đưa ra quyết định điều khiển.

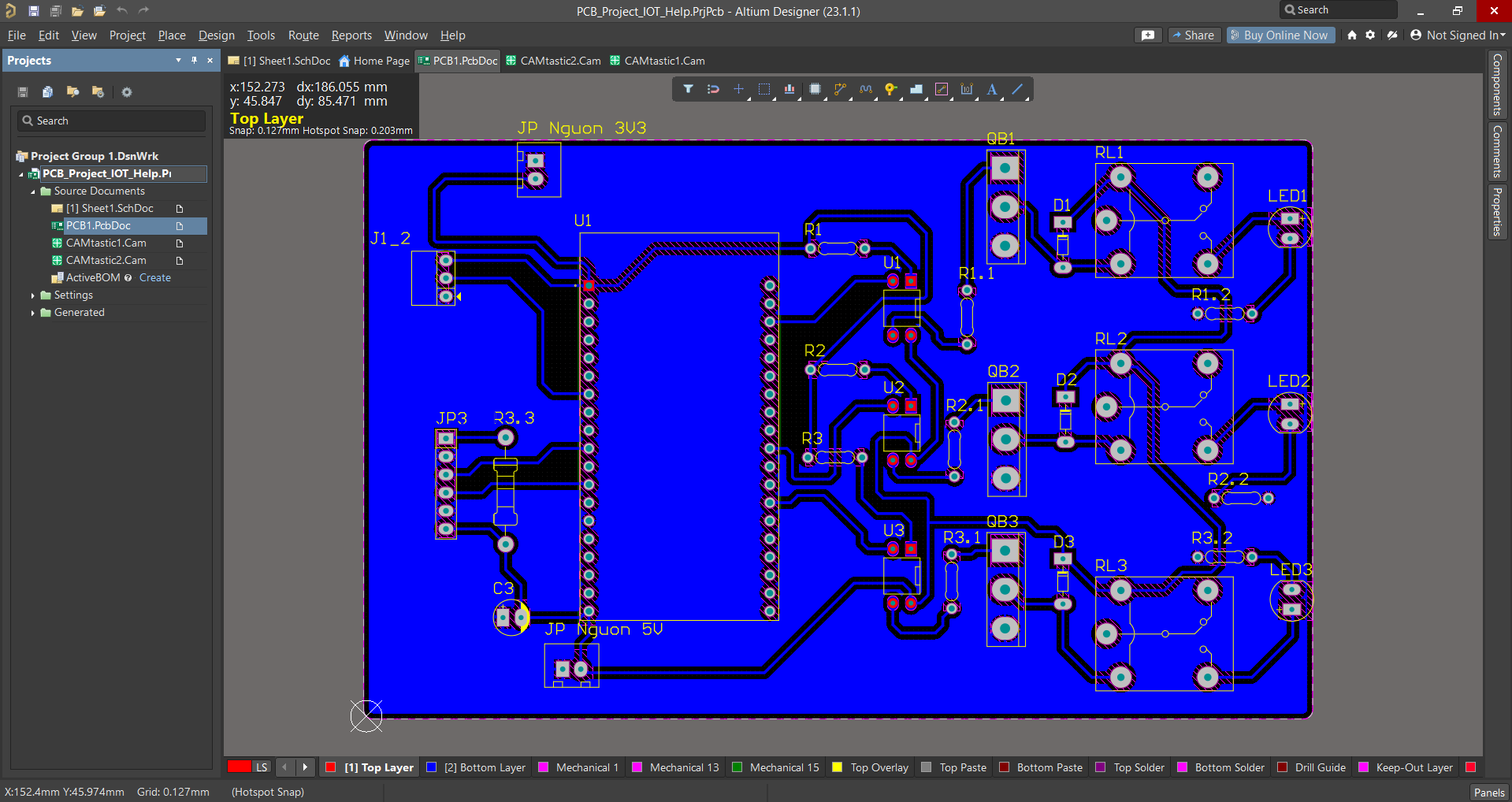
1. Hiển thị và giám sát

* Trên Web: hiển thị dữ liệu cảm biến theo thời gian thực.
* Trên Blynk: hiển thị giá trị đo, nút điều khiển Relay, lựa chọn chế độ Tự động/Thủ công và chỉnh sửa ngưỡng.
  1. SƠ ĐỒ MẠCH



Hình 10 : Sơ đồ mạch nguyên lí

* 1. MẠCH IN



Hình 11: Sơ đồ mạch in

* 1. CODE PlatformIO

1. Folder data:

File index.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

<link rel="stylesheet" href="Style.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.8/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-sRIl4kxILFvY47J16cr9ZwB07vP4J8+LH7qKQnuqkuIAvNWLzeN8tE5YBujZqJLB" crossorigin="anonymous">

</head>

<body>

<div class="Header">

<h1 style="margin: 0;"> ESP32 Server </h1>

</div>

<!-- Hiển thị giá trị cảm biến -->

<div class="sensorValue">

<h3>Values Sensor</h3>

<div class="grid\_tem">

<i class="fa-solid fa-temperature-low" style="color:chocolate ;"></i>

<span class="dht-labels">Temperature:</span>

<span id="temperature">%TEMPERATURE%</span>

<sup class="units">&deg;C</sup>

</div>

<div class="grid\_hum">

<i class="fas fa-tint" style="color:#00add6;"></i>

<span class="dht-labels">Humidity:</span>

<span id="humidity">%HUMIDITY%</span>

<sup class="units">&percnt;</sup>

</div>

<div class="grid\_dus">

<i class="fa-solid fa-wind" style="color:cadetblue;"></i>

<span class="dht-labels">Dust Concentration:</span>

<span id="dust">%DUST%</span>

<sup class="units">&micro;g/m&sup3;</sup>

</div>

</div>

<!-- Điều chỉnh giá trị ngưỡng -->

<div class="customeThreshold">

<div class="topnav">

<h3>Edit Threshold SenSor </h3>

</div>

<div class="content">

<div class="card-grid">

<div class="card">

<p class="card-title">Sensor 1</p>

<p class="switch">

<input type="range" onchange="updateSliderPWM(this)" id="slider1" min="0" max="100" step="1"

value="0" class="slider">

</p>

<p class="state">Temperature: <span id="sensor1"></span>&deg;C</p>

</div>

<div class="card">

<p class="card-title"> Sensor 2</p>

<p class="switch">

<input type="range" onchange="updateSliderPWM(this)" id="slider2" min="0" max="100" step="1"

value="0" class="slider">

</p>

<p class="state">Humidity: <span id="sensor2"></span> &percnt;</p>

</div>

<div class="card">

<p class="card-title"> Sensor 3</p>

<p class="switch">

<input type="range" onchange="updateSliderPWM(this)" id="slider3" min="0" max="250" step="1"

value="0" class="slider">

</p>

<p class="state">Dust Concentration: <span id="sensor3"></span> &micro;g/m&sup3;</p>

</div>

</div>

</div>

</div>

<script src="main.js"></script>

</body>

</html>

File Style.css:

html {

     font-family: Arial;

     display: inline-block;

     margin: 0px auto;

     text-align: center;

}

.Header {

    background-color: aqua;

    height: 100px;

    display: flex;

    align-items: center;

    justify-content: center;

}

.sensorValue {

    margin-top: 3rem;

}

h2 {

    font-size: 3.0rem;

}

.sensorValue{

    font-size: 2rem;

    text-align: center;

    align-items: center;

}

.units {

    font-size: 1.2rem;

}

.dht-labels{

      font-size: 1.5rem;

      vertical-align:middle;

      padding-bottom: 15px;

}

/\* Phần điều chỉnh giá trị ngưỡng \*/

.customeThreshold {

    margin-top: 10rem;

}

p {

    font-size: 1.4rem;

}

.topnav {

    overflow: hidden;

}

body {

    margin: 0;

}

.content {

    padding: 30px;

}

.card-grid {

    max-width: 900px;

    margin: 0 auto;

    display: grid;

    grid-gap: 2rem;

    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

}

.card {

    background-color: white;

    box-shadow: 2px 2px 12px 1px rgba(140, 140, 140, .5);

}

.card-title {

    font-size: 1.2rem;

    font-weight: bold;

    color: #034078

}

.state {

    font-size: 1.2rem;

    color: #1282A2;

}

.slider {

    -webkit-appearance: none;

    margin: 0 auto;

    width: 100%;

    height: 15px;

    border-radius: 10px;

    background: #FFD65C;

    outline: none;

}

.slider::-webkit-slider-thumb {

    -webkit-appearance: none;

    appearance: none;

    width: 30px;

    height: 30px;

    border-radius: 50%;

    background: #034078;

    cursor: pointer;

}

.slider::-moz-range-thumb {

    width: 30px;

    height: 30px;

    border-radius: 50%;

    background: #034078;

    cursor: pointer;

}

.switch {

    padding-left: 5%;

    padding-right: 5%;

}

/\* Responsive \*/

@media (max-width: 768px) {

  h2 {

    font-size: 2rem;

  }

  .sensorValue {

    font-size: 1.2rem;

  }

  .dht-labels {

    font-size: 1rem;

  }

  .grid\_tem, .grid\_hum, .grid\_dus {

      font-size: 1rem;

  }

  .units {

      font-size: 1rem;

      top: 0;  /\* hạ bớt để cân \*/

  }

}

File js:

var gateway = `ws://${window.location.hostname}/ws`;

var webSocket;

window.addEventListener('load' , onload);

function onload() {

    initWebSocket();

}

function initWebSocket() {

    console.log('Trying to open a WebSocket connection...');

    webSocket = new WebSocket(gateway);

    webSocket.onopen = onOpen;

    webSocket.onclose = onClose;

    webSocket.onmessage = onMessage;

}

function onOpen() {

    webSocket.send("getValues")

    console.log('Connection opened');

}

function onClose() {

    console.log('Connection Closed');

    setTimeout(initWebSocket , 2000);

}

function onMessage(event) {

    let sensorRes = JSON.parse(event.data);

    // Nếu gói tin là sensor values (temperature..dust)

    if (sensorRes.temperature !== undefined) {

        console.log("Nhiet do: " , sensorRes.temperature);

        console.log("Do Am: " , sensorRes.humidity);

        console.log("Nong Do Bui: " , sensorRes.dust)

        document.getElementById("temperature").innerHTML = sensorRes.temperature;

        document.getElementById("humidity").innerHTML = sensorRes.humidity;

        document.getElementById("dust").innerHTML = sensorRes.dust;

    }

    // Nếu gói tin là slider values (sensor1..3)

    if (sensorRes.sensor1 !== undefined) {

        document.getElementById("sensor1").innerHTML = sensorRes.sensor1;

        document.getElementById("slider1").value = sensorRes.sensor1;

        document.getElementById("sensor2").innerHTML = sensorRes.sensor2;

        document.getElementById("slider2").value = sensorRes.sensor2;

        document.getElementById("sensor3").innerHTML = sensorRes.sensor3;

        document.getElementById("slider3").value = sensorRes.sensor3;

    }

}

function updateSliderPWM(element) {

    var sliderNumber = element.id.charAt(element.id.length - 1);

    var sliderValue = document.getElementById(element.id).value;

    document.getElementById("sensor" + sliderNumber).innerHTML = sliderValue;

    console.log(sliderValue);

    webSocket.send(sliderNumber + "s" + sliderValue);

}

File cpp:

#include <Arduino.h>

#include <WiFi.h>

#include <SPIFFS.h>

#include <AsyncTCP.h>

#include <ESPAsyncWebServer.h>

#include <Arduino\_JSON.h>

#include <DHT.h>

#include "SharpGP2Y10.h"

#include <Preferences.h>

Preferences prefs;

// =========== Blynk =============

#define BLYNK\_TEMPLATE\_ID "TMPL6Jz6zCaGZ"

#define BLYNK\_TEMPLATE\_NAME "IOT Measure Sensor"

#define BLYNK\_AUTH\_TOKEN "kNDkR1p17mJJ6lcBspS4ivnyDhQnqiA9"

#include <BlynkSimpleEsp32.h>

// #include <TimeLib.h>

// #include <WidgetRTC.h>

// ========== End Blynk =============

// ========= SenSor ============

unsigned long lastSend = 0;

const unsigned long SEND\_INTERVAL = 2000; // ms

double t = -999;

double h = -999;

double dust = -999;

// Ngưỡng cài đặt

int thresold\_NHIETDO = 35;   // độ C

int thresold\_DOAM = 80;      // %

int thresold\_BUI = 100;    // µg/m3

// LED

#define LED\_NhietDo 22

#define LED\_DoAm 19

#define LED\_Bui 4

// DHT11

#define DHTtype DHT11

#define DHTpin 14

DHT dht(DHTpin , DHTtype);

// SharpGP2Y10

#define v0Pin 32

#define ledPin 26

SharpGP2Y10 dustSensor(v0Pin , ledPin);

void readSensor() {

  t = dht.readTemperature();

  h = dht.readHumidity();

  dust = dustSensor.readDustDensity();

  if(isnan(t)) t = -999;

  if(isnan(h)) h = -999;

  if (dust < 0 || dust > 1000) dust = -999;

}

// =============== End Sensor ==================

// =============== Blynk ===============

BlynkTimer timer;

bool manualMode = false;  // false = Auto, true = Manual

void sendSensor() {

  readSensor();

  if(isnan(t) && isnan(h))

  {

    // Serial.println("Failed to read from DHT and SharpGP2Y10 sensor !");

    // return;

    t = 50;

    h = 50;

    dust = 50;

  }

  Serial.println("Gia tri doc tu cam bien gui len Blynk: ");

  Serial.println("Temp: " + String(t));

  Serial.println("Hum: " + String(h));

  Serial.println("Dust: " + String(dust));

  Blynk.virtualWrite(V0 , t);

  Blynk.virtualWrite(V1 , h);

  Blynk.virtualWrite(V2 , dust);

}

// =========== End Blynk =================

// =========== Wifi ====================

const char\* ssid = "Wifi Chua";

const char\* password = "7780990204@Ok";

void initWifi() {

  WiFi.mode(WIFI\_STA);

  WiFi.begin(ssid , password);

  Serial.println("Connecting to Wifi ...");

  unsigned long start = millis();

  while (WiFi.status() != WL\_CONNECTED && millis() - start < 10000) {

    delay(1000);

    Serial.print(".");

  }

  if (WiFi.status() == WL\_CONNECTED) {

    Serial.println("WiFi Connected!");

    Serial.println(WiFi.localIP());

  } else {

    Serial.println("WiFi Connect Failed!");

  }

}

// ================== End Wifi =======================

// ================= SPIFFS ==================

void initSPIFFS() {

  if(!SPIFFS.begin()) {

    Serial.println("An error has occurred while mounting SPIFFS!");

  }

  else {

    Serial.println("SPIFFS mounted successfully.");

  }

}

// ================== End SPIFFS ===================

// ========= Server , WebSocket ==========

AsyncWebServer server(80);

AsyncWebSocket ws("/ws");

// ============= WebSocket ===============

JSONVar responeSensor;                           // Nếu gửi JSON dạng thủ công thì k cần dùng

String message = "";

String sensor1 = "0";

String sensor2 = "0";

String sensor3 = "0";

String getSliderValues() {

  responeSensor["sensor1"] = String(sensor1);

  responeSensor["sensor2"] = String(sensor2);

  responeSensor["sensor3"] = String(sensor3);

  String jsonString = JSON.stringify(responeSensor);

  return jsonString;

}

void notifyClients(String respone) {

  ws.textAll(respone);

}

void handleWebSocketRequest(void \* arg, uint8\_t \*data, size\_t len) {

  AwsFrameInfo \*info = (AwsFrameInfo\*) arg;

  if(info->final && info->index == 0 && info->len == len && info->opcode == WS\_TEXT) {

    // Xử lí khi nhận được request

    char msg[len + 1];

    memcpy(msg, data, len);   // Copy dữ liệu từ data sang msg

    msg[len] = '\0';

    message = String(msg);

    if(message.indexOf("1s") >= 0) {

      sensor1 = message.substring(2);

      thresold\_NHIETDO = sensor1.toInt();

      Serial.printf("Giá trị 1: %d" , thresold\_NHIETDO);

      Blynk.virtualWrite(V10 , thresold\_NHIETDO);

      prefs.putInt("NhietDo" , thresold\_NHIETDO);

      Serial.println(getSliderValues());

      notifyClients(getSliderValues());

    }

    if(message.indexOf("2s") >= 0) {

      sensor2 = message.substring(2);

      thresold\_DOAM = sensor2.toInt();

      Serial.printf("Giá trị 2: %d" , thresold\_DOAM);

      Blynk.virtualWrite(V11 , thresold\_DOAM);

      prefs.putInt("DoAm" , thresold\_DOAM);

      Serial.println(getSliderValues());

      notifyClients(getSliderValues());

    }

    if(message.indexOf("3s") >= 0) {

      sensor3 = message.substring(2);

      thresold\_BUI = sensor3.toInt();

      Serial.printf("Giá trị 3: %d" , thresold\_BUI);

      Blynk.virtualWrite(V12 , thresold\_BUI);

      prefs.putInt("DoBui" , thresold\_BUI);

      Serial.println(getSliderValues());

      notifyClients(getSliderValues());

    }

    // Ban đầu trình duyệt khi chưa có dữ liệu

    if(strcmp((char \*)data , "getValues") == 0) {

      Serial.printf("Gia tri ban dau %s\n" , getSliderValues());

      notifyClients(getSliderValues());

    }

  }

}

void onEvent(AsyncWebSocket \* server, AsyncWebSocketClient \* client, AwsEventType type, void \* arg, uint8\_t \*data, size\_t len) {

  switch (type)

  {

  case WS\_EVT\_CONNECT:

    Serial.printf("WebSocket client #%u connected from %s" , client->id() , client->remoteIP().toString().c\_str());

    sensor1 = String(thresold\_NHIETDO);

    sensor2 = String(thresold\_DOAM);

    sensor3 = String(thresold\_BUI);

    notifyClients(getSliderValues());

    break;

  case WS\_EVT\_DISCONNECT:

    Serial.printf("WebSocket client #%u disconnected\n", client->id());

    break;

  case WS\_EVT\_DATA:

    handleWebSocketRequest(arg , data , len);

    break;

  default:

    break;

  }

}

void initWebSocket() {

  ws.onEvent(onEvent);

  server.addHandler(&ws);

}

void setup() {

  Serial.begin(9600);

  delay(1000);

  // Khởi tạo LED

  pinMode(LED\_NhietDo, OUTPUT);

  pinMode(LED\_DoAm, OUTPUT);

  pinMode(LED\_Bui, OUTPUT);

  dht.begin();

  prefs.begin("Gia tri nguong" ,false);

  thresold\_NHIETDO = prefs.getInt("NhietDo", 35);

  thresold\_DOAM = prefs.getInt("DoAm", 80);

  thresold\_BUI = prefs.getInt("Bui", 100);

  initWifi();

  initSPIFFS();

  initWebSocket();

  // ========== render web ===========

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

    if(SPIFFS.exists("/index.html")) {

      request->send(SPIFFS , "/index.html" , "text/html");

    } else {

      request->send(200 , "text/html" , "<h1>No index.html found</h1>");

    }

  });

  server.serveStatic("/" , SPIFFS , "/");

  server.begin();

  // ======== Blynk ==========

  Blynk.begin(BLYNK\_AUTH\_TOKEN , ssid , password);

  timer.setInterval(3000L , sendSensor);

}

void loop() {

  // ======== Blynk ==========

  Blynk.run();

  timer.run();

  // ======= end Blynk =========

  unsigned long now = millis();

  readSensor();

  // send for clients

  if(now - lastSend >= SEND\_INTERVAL) {

    lastSend = now;

    JSONVar data;

    data["temperature"] = t;

    data["humidity"] = h;

    data["dust"] = dust;

    String payload = JSON.stringify(data);

    ws.textAll(payload);

    Serial.println(payload);

    ws.cleanupClients();

  }

  //// Relay on/off (chỉ chạy khi ở chế độ Auto)

  if (!manualMode && t!=-999 && h!=-999 && dust!=-999) {

    if(t > thresold\_NHIETDO) {

      digitalWrite(LED\_NhietDo , HIGH);

      Blynk.virtualWrite(V3 , HIGH);

    } else {

      digitalWrite(LED\_NhietDo , LOW);

      Blynk.virtualWrite(V3 , LOW);

    }

    if(h < thresold\_DOAM) {

      digitalWrite(LED\_DoAm , HIGH);

      Blynk.virtualWrite(V4 , HIGH);

    } else {

      digitalWrite(LED\_DoAm , LOW);

      Blynk.virtualWrite(V4 , LOW);

    }

    if(dust > thresold\_BUI) {

      digitalWrite(LED\_Bui , HIGH);

      Blynk.virtualWrite(V5 , HIGH);

    } else {

      digitalWrite(LED\_Bui , LOW);

      Blynk.virtualWrite(V5 , LOW);

    }

  }

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

    Serial.println("Wifi lost , reconnecting ...");

    WiFi.disconnect();

    WiFi.reconnect();

    delay(1000);

  }

}

BLYNK\_WRITE(V6) {

  manualMode = true;

  int pinValue = param.asInt();

  if(pinValue == 1) {

    digitalWrite(LED\_NhietDo , HIGH);

    Blynk.virtualWrite(V3 , HIGH);

  } else {

    digitalWrite(LED\_NhietDo , LOW);

    Blynk.virtualWrite(V3 , LOW);

  }

}

BLYNK\_WRITE(V7) {

  manualMode = true;

  int pinValue = param.asInt();

  if(pinValue == 1) {

    digitalWrite(LED\_DoAm , HIGH);

    Blynk.virtualWrite(V4 , HIGH);

  } else {

    digitalWrite(LED\_DoAm , LOW);

    Blynk.virtualWrite(V4 , LOW);

  }

}

BLYNK\_WRITE(V8) {

  manualMode = true;

  int pinValue = param.asInt();

  if(pinValue == 1) {

    digitalWrite(LED\_Bui , HIGH);

    Blynk.virtualWrite(V5 , HIGH);

  } else {

    digitalWrite(LED\_Bui , LOW);

    Blynk.virtualWrite(V5 , LOW);

  }

}

BLYNK\_WRITE(V9) {

  int mode = param.asInt();

  if (mode == 1) {

    manualMode = false;

    Serial.println("Mode to AUTO mode");

  } else {

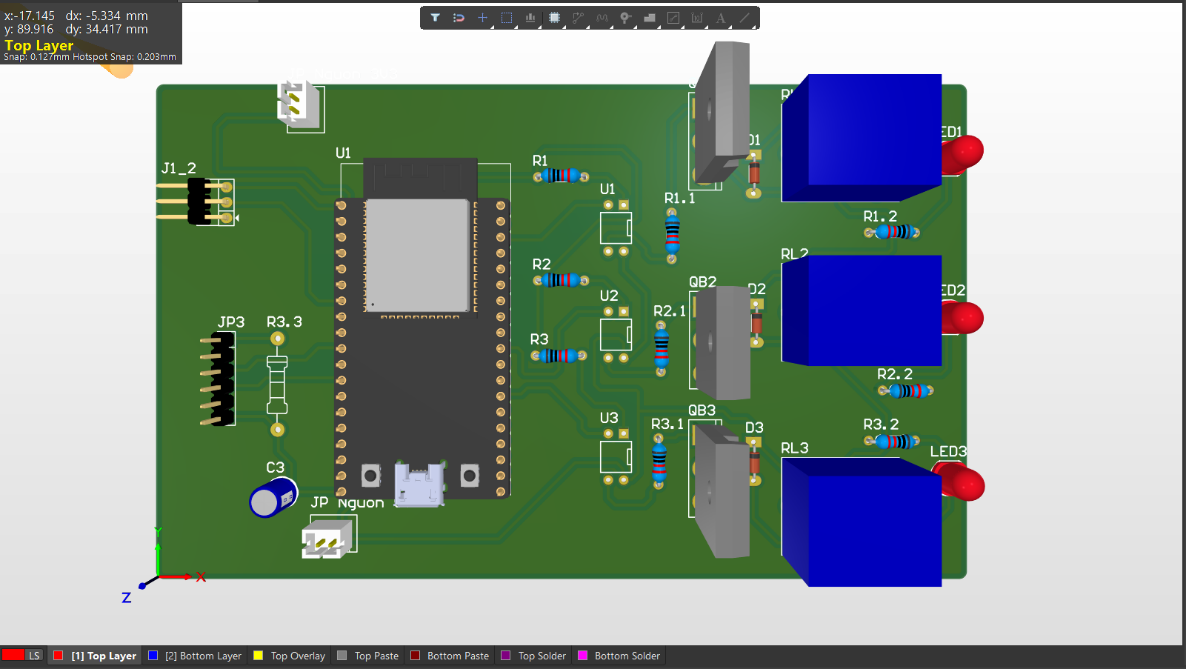
    manualMode = true;

    Serial.println("Mode to MANUAL mode");

  }

}

1. KẾT QUẢ



Hình 12 : Sơ đồ mạch

1. KẾT LUẬN

Sau quá trình nghiên cứu, thiết kế và triển khai, hệ thống giám sát và điều khiển môi trường dựa trên vi điều khiển ESP32, tích hợp các cảm biến nhiệt độ, độ ẩm và nồng độ bụi, đã được hoàn thiện và chứng minh tính khả thi trong thực tế. Hệ thống không chỉ đáp ứng nhu cầu theo dõi chất lượng không khí mà còn mang lại hiệu quả trong việc cảnh báo sớm, điều khiển thiết bị ngoại vi và hỗ trợ người dùng giám sát từ xa thông qua nền tảng Web và ứng dụng Blynk.

Trong quá trình thiết kế, ESP32 đã được khai thác tối đa nhờ khả năng xử lý mạnh mẽ và kết nối WiFi tích hợp, giúp thu thập và truyền tải dữ liệu cảm biến một cách chính xác, nhanh chóng. Cảm biến DHT cung cấp thông tin về nhiệt độ và độ ẩm môi trường, trong khi cảm biến bụi (Sharp GP2Y10 hoặc tương tự) đo nồng độ bụi trong không khí. Dữ liệu được hiển thị trực quan trên Web và ứng dụng Blynk, đồng thời người dùng có thể thiết lập ngưỡng giới hạn và lựa chọn chế độ điều khiển tự động hoặc thủ công. Ở chế độ tự động, ESP32 sẽ tự động kích hoạt relay khi giá trị vượt ngưỡng; ở chế độ thủ công, người dùng có thể trực tiếp bật/tắt relay từ ứng dụng Blynk.

Kết quả thử nghiệm cho thấy hệ thống hoạt động ổn định, phản hồi nhanh với sự thay đổi của môi trường, đảm bảo các chức năng giám sát – cảnh báo – điều khiển hoạt động chính xác. Việc lưu trữ ngưỡng trong bộ nhớ giúp dữ liệu được duy trì ngay cả khi mất nguồn, nâng cao tính ổn định và độ tin cậy. Hệ thống đồng thời mang tính linh hoạt, có thể dễ dàng mở rộng thêm các cảm biến khác hoặc tích hợp với các nền tảng IoT phổ biến.

Về mặt thực tiễn, hệ thống mang lại lợi ích rõ rệt trong giám sát chất lượng không khí, bảo vệ sức khỏe, tự động hóa nông nghiệp và quản lý môi trường đô thị. Với chi phí hợp lý, linh kiện phổ biến và phần mềm mã nguồn mở, hệ thống dễ tiếp cận và có thể triển khai ở nhiều quy mô khác nhau.

Trong tương lai, hệ thống có thể được nâng cấp bằng cách tích hợp các cảm biến mở rộng (ví dụ: khí độc hại, pH đất, ánh sáng), bổ sung thuật toán thông minh hoặc AI/ML để dự đoán và điều khiển tốt hơn, cũng như phát triển giao diện người dùng trực quan hơn. Những cải tiến này sẽ giúp hệ thống trở thành một giải pháp IoT toàn diện, đóng góp vào sự phát triển của nông nghiệp thông minh, giám sát môi trường và đô thị thông minh.

Tóm lại, đồ án sử dụng ESP32 và các cảm biến môi trường đã chứng minh được tính hiệu quả, ổn định và khả năng ứng dụng cao, là một bước tiến quan trọng trong việc kết hợp công nghệ IoT với đời sống thực tiễn

1. ỨNG DỤNG

Hệ thống giám sát và điều khiển môi trường dựa trên vi điều khiển ESP32, kết hợp với các cảm biến nhiệt độ, độ ẩm và nồng độ bụi, không chỉ là một giải pháp kỹ thuật tiên tiến mà còn mang lại nhiều ứng dụng thực tiễn trong các lĩnh vực khác nhau. Dưới đây là một số ứng dụng nổi bật:

1. Giám sát chất lượng không khí trong gia đình và văn phòng:

Hệ thống có thể triển khai trong nhà ở, trường học, văn phòng nhằm giám sát các thông số môi trường cơ bản. Việc theo dõi nồng độ bụi, nhiệt độ và độ ẩm giúp người dùng chủ động điều chỉnh máy lọc không khí, quạt hay điều hòa, góp phần nâng cao chất lượng cuộc sống và bảo vệ sức khỏe.

1. Ứng dụng trong nông nghiệp thông minh:

Trong các mô hình trồng trọt, đặc biệt là nhà kính hoặc nông nghiệp đô thị, hệ thống có thể được dùng để giám sát môi trường xung quanh cây trồng. Khi nồng độ bụi hoặc nhiệt độ vượt ngưỡng, hệ thống sẽ tự động điều khiển quạt thông gió hoặc cảnh báo người dùng, từ đó giúp duy trì điều kiện tối ưu cho sự phát triển của cây.

1. Mô hình nghiên cứu và đào tạo:

Đây là một công cụ lý tưởng cho sinh viên, học sinh và các viện nghiên cứu. Hệ thống hỗ trợ thực hành các kiến thức về IoT, vi điều khiển, cảm biến và lập trình nhúng, đồng thời có thể dùng trong các thí nghiệm khoa học liên quan đến môi trường, sức khỏe hoặc kỹ thuật điều khiển.

1. Giám sát môi trường đô thị:

Hệ thống có thể được triển khai tại các khu vực công cộng như công viên, nhà ga, trường học để giám sát nồng độ bụi mịn và các thông số môi trường. Dữ liệu được thu thập từ nhiều điểm có thể kết hợp lại thành bản đồ chất lượng không khí, hỗ trợ chính quyền và người dân theo dõi tình trạng ô nhiễm.

1. Ứng dụng trong các dự án DIY và IoT cá nhân:

Với chi phí hợp lý và tính linh hoạt cao, hệ thống phù hợp cho các dự án "tự làm" (DIY) của những người yêu công nghệ. Người dùng có thể tùy biến hệ thống để giám sát môi trường trong nhà, ban công hoặc không gian làm việc, đồng thời tích hợp thêm các thiết bị như máy phun sương, đèn cảnh báo hay máy lọc không khí.

Nhờ tính đa dụng và khả năng mở rộng, hệ thống giám sát môi trường dựa trên ESP32 không chỉ giới hạn trong một lĩnh vực cụ thể mà còn có thể được điều chỉnh để đáp ứng nhu cầu đa dạng trong thực tế. Sự kết hợp giữa công nghệ vi điều khiển ESP32, các cảm biến hiện đại và nền tảng IoT như Blynk đã tạo nên một nền tảng vững chắc cho các ứng dụng sáng tạo trong tương lai.