

TRƯỜNG ĐẠI HỌC KHOA HỌC HUẾ
KHOA CÔNG NGHỆ THÔNG TIN

-----öö&öö-----



BÀI TIỂU LUẬN

MÔN: PHÁT TRIỂN ỨNG DỤNG IOT
NHÓM: 4

Đề tài: Điều khiển quạt dựa trên nhiệt độ với ESP32.

Giảng viên hướng dẫn: Võ Việt Dũng

Sinh viên thực hiện: Nguyễn Trần Viết Thắng

HUẾ, 04/2025

MỤC LỤC

I. TỔNG QUAN VỀ ĐỀ TÀI NGHIÊN CỨU.....	1
1.1 Đặt vấn đề.....	1
1.2 Mục tiêu đề tài.....	1
1.3 Ứng dụng thực tế.....	2
II. CƠ SỞ LÝ THUYẾT.....	3
2.1 Giới thiệu về ESP32 Node MCU.....	3
2.2. Thành phần chính của NodeMCU ESP32.....	3
2.3. Chức năng của các chân trên NodeMCU ESP32.....	4
2.3.1. Các chân nguồn và GND.....	4
2.3.2. Các chân giao tiếp Digital (GPIO).....	4
2.3.3. Các chân Analog (ADC & DAC).....	5
2.3.4. Các chân giao tiếp truyền thông.....	5
2.4. Phần mềm.....	7
2.4.1 Giới thiệu phần mềm lập trình.....	7
2.4.2. Cơ sở lý thuyết về BlynkCloud.....	8
III. NGUYÊN LÝ HOẠT ĐỘNG.....	9
3.1. Tổng quan hệ thống.....	9
3.2. Sơ đồ khối hệ thống.....	9
3.3. Nguyên lý hoạt động chi tiết.....	9
3.3.1. Đọc dữ liệu từ cảm biến nhiệt độ.....	9
3.3.2. Xử lý dữ liệu trên ESP32.....	9
3.3.3. Điều khiển relay để bật/tắt quạt.....	10
3.3.4. Gửi dữ liệu lên Blynk để giám sát từ xa.....	10
3.4 Mô tả hoạt động theo sơ đồ trạng thái.....	10
IV. THIẾT KẾ PHẦN CỨNG.....	11
4.1 Thành phần.....	11
4.2 Sơ đồ mạch.....	11
V. LẬP TRÌNH ESP32.....	12
5.1 Mô tả chương trình.....	12
5.2 Thực hiện chương trình.....	12
VI. KẾT QUẢ VÀ ĐÁNH GIÁ.....	20
6.1. Kết quả thử nghiệm.....	20
6.2. Đánh giá hiệu suất.....	20
VII. HƯỚNG PHÁT TRIỂN TƯƠNG LAI.....	22

7.1. Cải tiến phần cứng.....	22
7.1.1. Hệ thống điều khiển tốc độ quạt thông minh:	22
7.1.2. Mở rộng đa cảm biến:	22
7.1.3. Nâng cấp nguồn năng lượng:.....	22
7.1.4. Bảo vệ và độ bền:.....	22
7.2. Cải tiến phần mềm.....	23
7.2.1. Thuật toán thông minh:.....	23
7.2.2. Hệ thống thông báo nâng cao:.....	23
7.2.3. Tích hợp hệ sinh thái thông minh:.....	23
7.2.4. Phân tích dữ liệu nâng cao:	23
7.3. Mở rộng ứng dụng.....	24
7.3.1. Hệ thống điều hòa không khí tổng thể:	24
7.3.2. Ứng dụng trong nông nghiệp thông minh:.....	24
7.3.3. Giải pháp thương mại:.....	24
7.4. Dự án mã nguồn mở.....	24
VIII. TÀI LIỆU THAM KHẢO	25

LỜI CẢM ƠN

Trước tiên, em xin gửi lời cảm ơn chân thành đến quý thầy cô trong khoa Công nghệ Thông tin, đặc biệt là giảng viên bộ môn Phát triển ứng dụng IoT, người đã tận tình hướng dẫn và truyền đạt những kiến thức quý báu trong suốt quá trình học tập. Sự hỗ trợ và định hướng của thầy/cô không chỉ giúp em hoàn thành tiểu luận này mà còn tạo động lực để nhóm khám phá sâu hơn về lĩnh vực Internet of Things.

Em cũng xin bày tỏ lòng biết ơn đến các bạn học cùng lớp, những người đã đóng góp ý kiến, chia sẻ kinh nghiệm và hỗ trợ nhóm trong quá trình thực hiện đề tài. Sự động viên và hợp tác từ các bạn là nguồn cảm hứng lớn để chúng tôi nỗ lực hoàn thiện sản phẩm.

Mặc dù đã cố gắng hết sức, tiểu luận này chắc chắn vẫn còn những thiếu sót. Em rất mong nhận được sự thông cảm và những ý kiến đóng góp quý giá từ thầy cô cũng như các bạn để đề tài được hoàn thiện hơn.

Trân trọng.

Em xin chân thành cảm ơn!

I. TỔNG QUAN VỀ ĐỀ TÀI NGHIÊN CỨU

1.1 Đặt vấn đề

Trong cuộc sống hiện đại, nhu cầu sử dụng các thiết bị gia dụng thông minh ngày càng gia tăng nhằm mang lại sự tiện lợi, tiết kiệm năng lượng và nâng cao chất lượng cuộc sống. Quạt điện, một thiết bị phổ biến trong mỗi gia đình, thường được sử dụng để làm mát không gian sống. Tuy nhiên, việc điều khiển quạt thủ công không chỉ gây bất tiện mà còn dẫn đến lãng phí điện năng khi người dùng quên tắt quạt hoặc không điều chỉnh phù hợp với điều kiện môi trường. Đặc biệt, trong bối cảnh biến đổi khí hậu, nhiệt độ môi trường thay đổi thất thường càng đòi hỏi những giải pháp tự động hóa để tối ưu hóa việc sử dụng các thiết bị như quạt.

Công nghệ Internet of Things (IoT) đã mở ra một hướng đi mới, cho phép tích hợp các cảm biến và vi điều khiển vào thiết bị gia dụng, từ đó tạo ra các hệ thống thông minh có khả năng tự động điều chỉnh theo nhu cầu thực tế. ESP32, với ưu điểm là một vi điều khiển mạnh mẽ, chi phí thấp, tích hợp Wi-Fi và khả năng xử lý tín hiệu nhanh, đã trở thành lựa chọn lý tưởng cho các ứng dụng IoT. Kết hợp với cảm biến nhiệt độ và relay, ESP32 có thể được sử dụng để xây dựng một hệ thống điều khiển quạt tự động dựa trên nhiệt độ môi trường, giúp giảm thiểu sự can thiệp của con người và nâng cao hiệu quả sử dụng.

Xuất phát từ thực tế trên, đề tài "Điều khiển quạt dựa trên nhiệt độ với ESP32" được em lựa chọn nhằm nghiên cứu và phát triển một hệ thống IoT đơn giản nhưng thiết thực. Hệ thống này sử dụng cảm biến nhiệt độ để thu thập dữ liệu, sau đó thông qua ESP32 để xử lý và điều khiển relay, tự động bật hoặc tắt quạt khi nhiệt độ vượt quá hoặc giảm xuống dưới ngưỡng cài đặt. Đề tài không chỉ có ý nghĩa trong việc ứng dụng kiến thức lý thuyết vào thực tiễn mà còn góp phần hướng tới các giải pháp tiết kiệm năng lượng và thân thiện với môi trường.

1.2 Mục tiêu đề tài

- + Thiết kế và triển khai hệ thống đo nhiệt độ môi trường bằng cảm biến DHT11/DHT22 với độ chính xác cao.
- + Phát triển cơ chế điều khiển quạt tự động thông qua module relay dựa trên ngưỡng nhiệt độ đã thiết lập.
- + Tích hợp kết nối IoT với ứng dụng Blynk để giám sát và điều khiển hệ thống từ xa.
- + Tối ưu hóa việc sử dụng năng lượng thông qua hoạt động thông minh của hệ thống.

- + Nâng cao sự tiện lợi trong việc quản lý nhiệt độ môi trường mà không cần sự can thiệp thường xuyên.
- + Phát triển giao diện trực quan, dễ sử dụng cho người dùng không chuyên về công nghệ.
- + Xây dựng hệ thống với khả năng mở rộng để tích hợp thêm các tính năng trong tương lai.

1.3 Ứng dụng thực tế

Hệ thống có thể được ứng dụng trong nhiều lĩnh vực khác nhau:

- **Nhà thông minh:**
 - Tự động điều chỉnh quạt dựa trên nhiệt độ phòng, tăng cường sự thoải mái cho người dùng.
 - Kết hợp với các thiết bị thông minh khác để tạo hệ sinh thái nhà thông minh hoàn chỉnh.
 - Tối ưu hóa tiêu thụ năng lượng bằng cách chỉ kích hoạt quạt khi cần thiết.
- **Quản lý nhiệt độ phòng server:**
 - Giúp duy trì nhiệt độ phù hợp cho thiết bị điện tử, tránh quá nhiệt gây hỏng hóc.
 - Giám sát từ xa tình trạng nhiệt độ phòng server, cảnh báo kịp thời khi có bất thường.
 - Tự động điều chỉnh hệ thống làm mát, giảm chi phí vận hành và bảo trì.
- **Nông nghiệp thông minh:**
 - Điều khiển quạt trong nhà kính để duy trì môi trường lý tưởng cho cây trồng.
 - Tạo điều kiện nhiệt độ phù hợp cho từng giai đoạn phát triển của cây.
 - Kết hợp với các cảm biến khác để tạo hệ thống quản lý nông nghiệp toàn diện.
- **Hệ thống nuôi trồng thủy sản:**
 - Duy trì nhiệt độ nước ổn định trong bể nuôi cá, tôm.
 - Tự động kích hoạt quạt oxy hóa khi nhiệt độ tăng cao để bảo vệ sinh vật thủy sinh.
 - Giám sát từ xa, giúp người nuôi có thể theo dõi điều kiện môi trường mọi lúc, mọi nơi.
- **Lĩnh vực y tế và dược phẩm:**

- Giám sát nhiệt độ trong kho bảo quản thuốc, vắc-xin và các sản phẩm y tế nhạy cảm với nhiệt.
- Đảm bảo điều kiện lưu trữ tối ưu cho các thiết bị y tế và mẫu xét nghiệm.
- Tự động điều chỉnh hệ thống làm mát để duy trì nhiệt độ trong khoảng an toàn.

II. CƠ SỞ LÝ THUYẾT

2.1 Giới thiệu về ESP32 Node MCU

ESP32 là một vi điều khiển (MCU) tích hợp Wi-Fi và Bluetooth, được phát triển bởi **Espressif Systems**. Đây là phiên bản nâng cấp từ ESP8266, cung cấp hiệu suất cao hơn, nhiều tính năng hơn và khả năng tiết kiệm năng lượng tốt hơn. ESP32 thường được sử dụng trong các ứng dụng IoT (Internet of Things), nhà thông minh, điều khiển tự động và các thiết bị nhúng.



NodeMCU ESP32 là một bo mạch phát triển dựa trên vi điều khiển **ESP32** của **Espressif Systems**, được thiết kế để dễ dàng lập trình và kết nối với các thiết bị ngoại vi.

2.2. Thành phần chính của NodeMCU ESP32

Thành phần	Chức năng
Vi điều khiển ESP32	Bộ xử lý trung tâm, có Wi-Fi & Bluetooth.
Cổng Micro-USB	Kết nối với máy tính để lập trình và cấp nguồn.
Chip USB-TTL (CP2102/CH340G)	Chuyển đổi tín hiệu từ USB sang UART để giao tiếp với máy tính.

Mạch nguồn (Voltage Regulator)	Chuyển đổi 5V từ USB xuống 3.3V để cấp nguồn cho ESP32.
Nút nhấn (BOOT & RESET)	Dùng để nạp chương trình và khởi động lại ESP32.
Đèn LED tích hợp	LED hiển thị trạng thái hoạt động.
Các chân GPIO (General Purpose Input/Output)	Giao tiếp với cảm biến, module, thiết bị ngoại vi.
Bộ ADC & DAC	Chuyển đổi tín hiệu tương tự sang số và ngược lại.
Module Wi-Fi & Bluetooth	Kết nối mạng và giao tiếp không dây.

2.3. Chức năng của các chân trên NodeMCU ESP32

ESP32 có nhiều chân **GPIO (General Purpose Input/Output)** hỗ trợ nhiều chức năng khác nhau.

2.3.1. Các chân nguồn và GND

Chân	Chức năng
VIN	Cấp nguồn từ ngoài (5V).
3V3	Cấp nguồn 3.3V cho module, cảm biến.
GND	Mass (chung nguồn) của mạch.

2.3.2. Các chân giao tiếp Digital (GPIO)

ESP32 có **34 chân GPIO**, có thể dùng làm Input/Output, PWM, SPI, I2C, UART,...

GPIO	Chức năng chính
GPIO 0	Boot mode, dùng khi nạp firmware.
GPIO 2	Đèn LED tích hợp trên board.
GPIO 5	SPI (CS), có thể dùng làm Output.
GPIO 12, 13, 14, 15	SPI (MISO, MOSI, SCK, CS).
GPIO 16, 17	Dùng làm UART hoặc Output.

⚠ Lưu ý:

- **GPIO 6 → GPIO 11** là các chân kết nối với bộ nhớ Flash, không nên sử dụng.

- **GPIO 34 → GPIO 39** chỉ hỗ trợ Input (không thể làm Output).

2.3.3. Các chân Analog (ADC & DAC)

ESP32 có 18 kênh ADC (độ phân giải 12-bit) và 2 kênh DAC (độ phân giải 8-bit).

Chân	Chức năng
ADC (GPIO 32 - GPIO 39)	Đọc tín hiệu Analog từ cảm biến.
DAC (GPIO 25, GPIO 26)	Xuất tín hiệu Analog.

2.3.4. Các chân giao tiếp truyền thông

UART (Giao tiếp Serial)

Chân	Chức năng
GPIO 1 (TX0), GPIO 3 (RX0)	UART0 – Giao tiếp Serial mặc định.
GPIO 9, GPIO 10	UART1 (ít dùng).
GPIO 16, GPIO 17	UART2.

I2C (Giao tiếp với cảm biến, LCD,...)

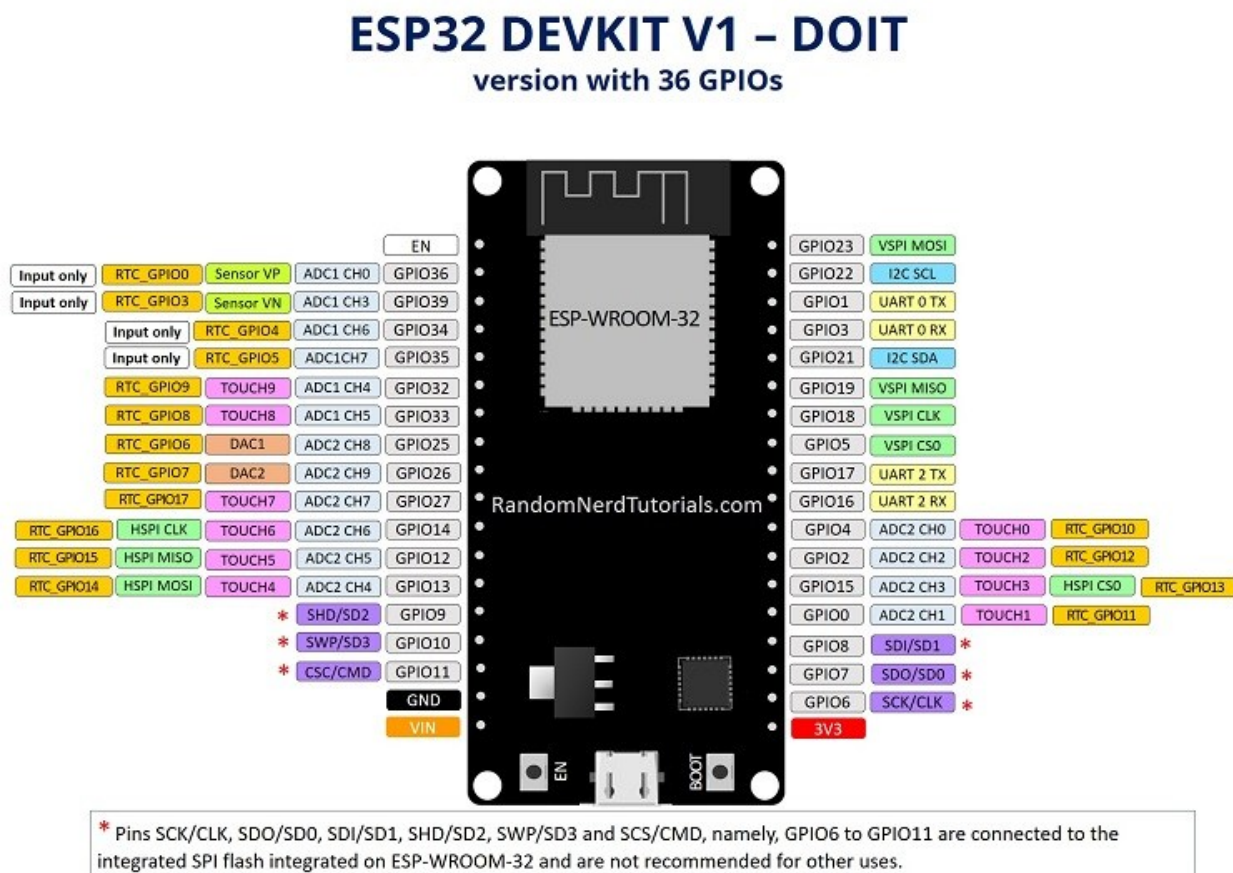
Chân	Chức năng
GPIO 21	SDA (Data).
GPIO 22	SCL (Clock).

SPI (Giao tiếp với module RF, thẻ nhớ,...)

Chân	Chức năng
GPIO 23	MOSI (Master Out Slave In).
GPIO 19	MISO (Master In Slave Out).
GPIO 18	SCK (Clock).
GPIO 5	CS (Chip Select).

PWM (Điều chế độ rộng xung, dùng để điều khiển quạt, LED, động cơ servo,...)

Tất cả các chân **GPIO** trên ESP32 đều có thể xuất tín hiệu **PWM** với tần số và độ phân giải tùy chỉnh.



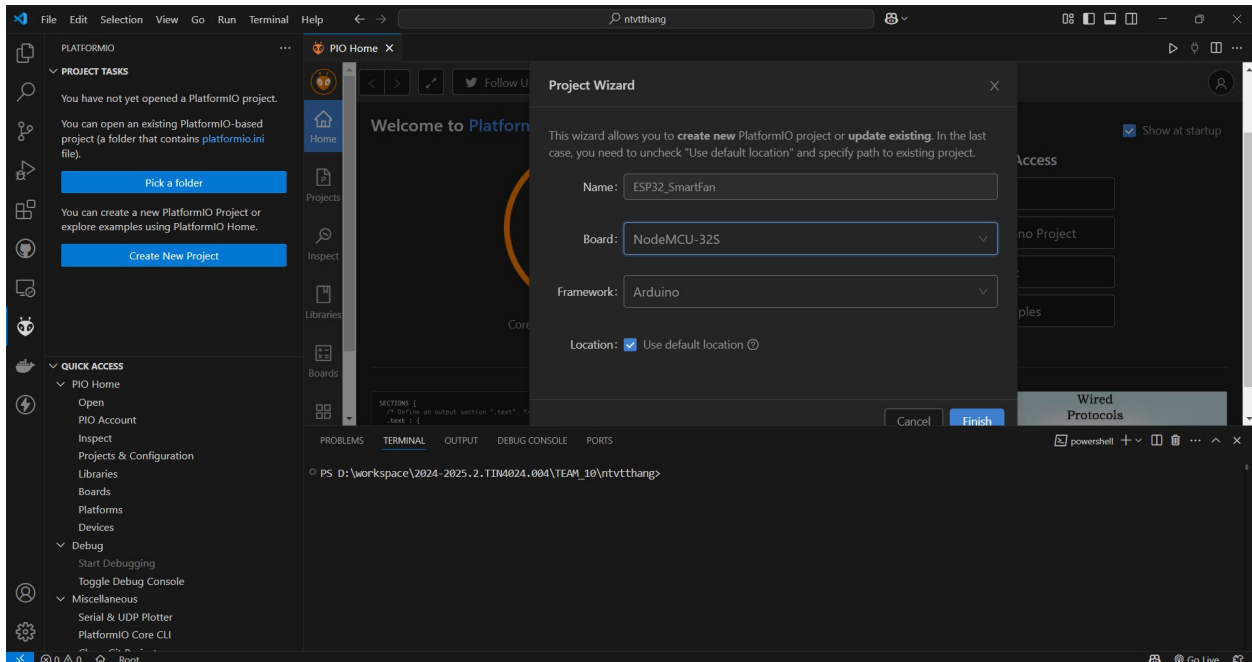
Hình ảnh sơ đồ chân kết nối ESP32.

2.4. Phần mềm

2.4.1 Giới thiệu phần mềm lập trình

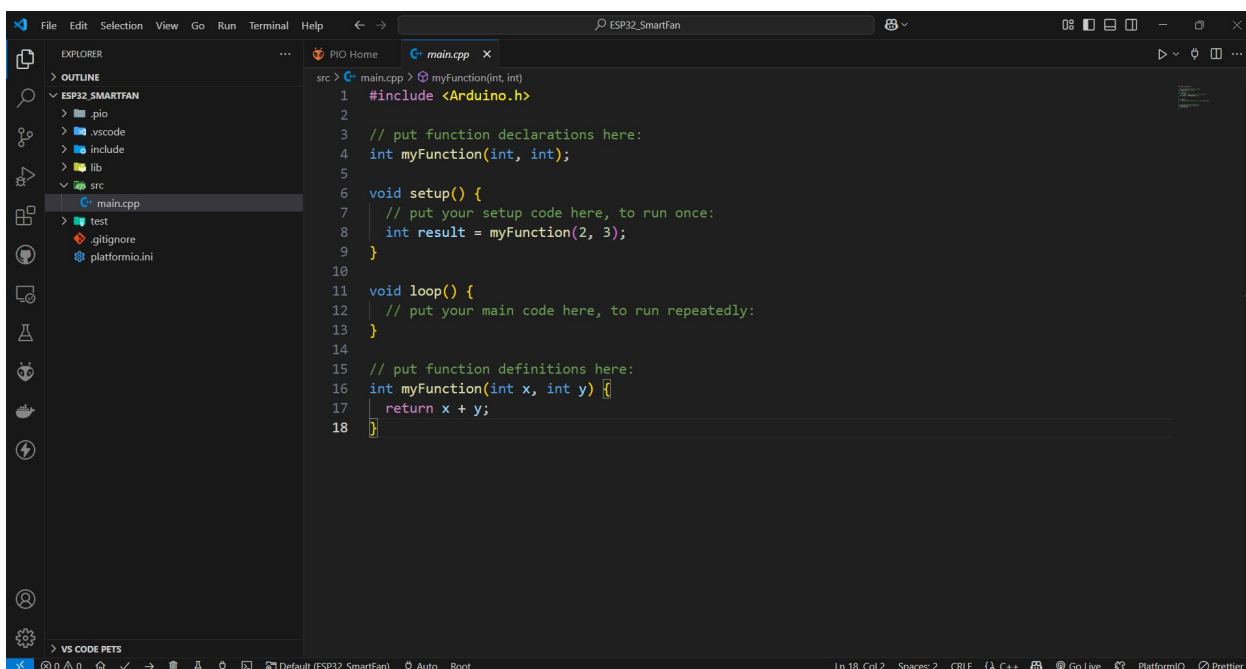
PlatformIO là một nền tảng phát triển nhúng (IoT) hỗ trợ nhiều loại vi điều khiển (ESP32, Arduino, STM32, AVR,...). PlatformIO hoạt động như một extension trong VSCode, giúp lập trình dễ dàng hơn so với Arduino IDE truyền thống.

- + Mở **VSCode** → Click vào biểu tượng **PlatformIO**.
- + Chọn **New Project**.
- + Đặt tên dự án, chọn **Board: ESP32**, Framework: **Arduino**.
- + Click **Finish** để tạo dự án.



Giao diện của phần mềm

Tiếp theo vào src/main.cpp và nhập code:



2.4.2. Cơ sở lý thuyết về BlynkCloud

Blynk Cloud hoạt động dựa trên ba thành phần chính:

- Blynk App: Ứng dụng trên điện thoại hoặc giao diện web cho phép người dùng tạo giao diện điều khiển (widget) như nút bấm, biểu đồ, hoặc thanh trượt để tương tác với phần cứng.
- Blynk Server (Cloud): Máy chủ trung tâm xử lý dữ liệu, lưu trữ thông tin và truyền thông tin giữa ứng dụng và phần cứng. Blynk Cloud sử dụng giao thức TCP/IP và hỗ trợ kết nối an toàn qua SSL/TLS (trên các thiết bị hỗ trợ).
- Blynk Library: Thư viện mã nguồn mở tích hợp vào phần cứng (như ESP32), giúp thiết bị giao tiếp với Blynk Cloud thông qua Wi-Fi hoặc các phương thức kết nối khác.

Trong dự án **"Điều khiển quạt dựa trên nhiệt độ với ESP32"**, Blynk Cloud nhận dữ liệu nhiệt độ từ ESP32, xử lý và gửi lệnh bật/tắt quạt dựa trên ngưỡng nhiệt độ cài đặt trong ứng dụng.

III. NGUYÊN LÝ HOẠT ĐỘNG

3.1. Tổng quan hệ thống

Hệ thống này sử dụng vi điều khiển ESP32 để đọc dữ liệu từ cảm biến nhiệt độ DHT11/DHT22. Khi nhiệt độ vượt quá một ngưỡng nhất định (ví dụ: 30°C), ESP32 sẽ kích hoạt relay để bật quạt. Nếu nhiệt độ giảm xuống dưới ngưỡng cài đặt, ESP32 sẽ tắt relay, ngắt nguồn điện của quạt.

Ngoài ra, dữ liệu nhiệt độ sẽ được gửi lên Blynk, một nền tảng IoT giúp giám sát từ xa thông qua ứng dụng di động.

3.2. Sơ đồ khối hệ thống

Hệ thống bao gồm các thành phần chính sau:

1. Cảm biến nhiệt độ DHT11/DHT22: Đo nhiệt độ môi trường và gửi tín hiệu về ESP32.
2. ESP32: Xử lý dữ liệu nhiệt độ, quyết định bật/tắt quạt dựa vào giá trị ngưỡng đặt trước.
3. Relay 5V: Đóng/ngắt nguồn điện quạt khi nhận tín hiệu từ ESP32.
4. Quạt điện: Được điều khiển bật/tắt thông qua relay.
5. Blynk: Giao diện điều khiển và giám sát từ xa trên điện thoại và web.

3.3. Nguyên lý hoạt động chi tiết

3.3.1. Đọc dữ liệu từ cảm biến nhiệt độ

- Cảm biến DHT11/DHT22 đo nhiệt độ môi trường.
- ESP32 đọc dữ liệu từ cảm biến thông qua giao tiếp 1-Wire.

3.3.2. Xử lý dữ liệu trên ESP32

- ESP32 so sánh giá trị nhiệt độ đo được với ngưỡng cài đặt (ví dụ: 30°C).
- Nếu nhiệt độ $\geq 30^{\circ}\text{C}$ \rightarrow ESP32 kích hoạt relay để bật quạt.
- Nếu nhiệt độ $\leq 28^{\circ}\text{C}$ \rightarrow ESP32 ngắt relay để tắt quạt.
- Hysteresis (ngưỡng trễ) giúp tránh bật/tắt quạt liên tục khi nhiệt độ dao động gần 30°C.

3.3.3. Điều khiển relay để bật/tắt quạt

- ESP32 điều khiển relay thông qua tín hiệu Digital Output.
- Khi relay ON, quạt được cấp nguồn và quay.
- Khi relay OFF, quạt mất nguồn và dừng hoạt động.

3.3.4. Gửi dữ liệu lên Blynk để giám sát từ xa

- ESP32 gửi dữ liệu nhiệt độ lên Blynk Cloud qua Wi-Fi.
- Ứng dụng Blynk hiển thị nhiệt độ theo thời gian thực.
- Người dùng có thể thay đổi ngưỡng nhiệt độ hoặc bật/tắt quạt thủ công trên Blynk.

3.4 Mô tả hoạt động theo sơ đồ trạng thái

Trạng thái	Điều kiện	Hành động của ESP32	Trạng thái quạt
Chờ nhiệt độ đo	ESP32 đọc dữ liệu từ DHT	Lưu giá trị vào biến nhiệt độ	-
Nhiệt độ thấp	$T < 28^{\circ}\text{C}$	Tắt relay	Quạt OFF
Nhiệt độ cao	$T \geq 30^{\circ}\text{C}$	Bật relay	Quạt ON
Nhiệt độ trung gian	$28^{\circ}\text{C} \leq T < 30^{\circ}\text{C}$	Giữ trạng thái trước đó	Không thay đổi

IV. THIẾT KẾ PHẦN CỨNG

4.1 Thành phần

- ESP32: Vi điều khiển chính.
- Cảm biến nhiệt độ DHT11/DHT22: Đo nhiệt độ môi trường.
- Relay 5V: Điều khiển bật/tắt quạt.
- Quạt 220V hoặc 12V: Là thiết bị làm mát.

4.2 Sơ đồ mạch

- DHT11/DHT22:
 - VCC -> 3.3V ESP32
 - GND -> GND ESP32
 - DATA -> GPIO4 ESP32
- Relay 5V:
 - VCC -> 5V ESP32
 - GND -> GND ESP32
 - IN -> GPIO5 ESP32

Chi tiết kết nối phần cứng:

- ESP32 cấp nguồn 3.3V cho cảm biến DHT11/DHT22, giúp cảm biến hoạt động ổn định.
- ESP32 cấp nguồn 5V cho relay, đảm bảo relay hoạt động đúng mức điện áp.
- Chân DATA của DHT11/DHT22 nối với GPIO4 của ESP32 để truyền dữ liệu nhiệt độ.
- Chân IN của relay kết nối với GPIO5 để điều khiển bật/tắt quạt.
- Nguồn quạt 220V hoặc 12V sẽ được đóng/ngắt qua relay, đảm bảo an toàn.

V. LẬP TRÌNH ESP32

5.1 Mô tả chương trình

Chương trình được viết trên Visual Studio Code với thư viện hỗ trợ DHT, Blynk và relay.

Hệ thống:

- Kết nối WiFi và Blynk.
- Đọc nhiệt độ từ DHT.
- So sánh nhiệt độ với ngưỡng cài đặt.
- Bật/tắt relay tương ứng.
- Cập nhật dữ liệu lên Blynk để giám sát.

5.2 Thực hiện chương trình

Bước 1: Cấu hình Blynk

Tạo tài khoản và dự án trên Blynk

1. Tải ứng dụng Blynk:

- Đăng nhập Blynk web hoặc tải ứng dụng Blynk từ Google Play Store (Android) hoặc App Store (iOS).
- Tạo tài khoản bằng email của cá nhân.

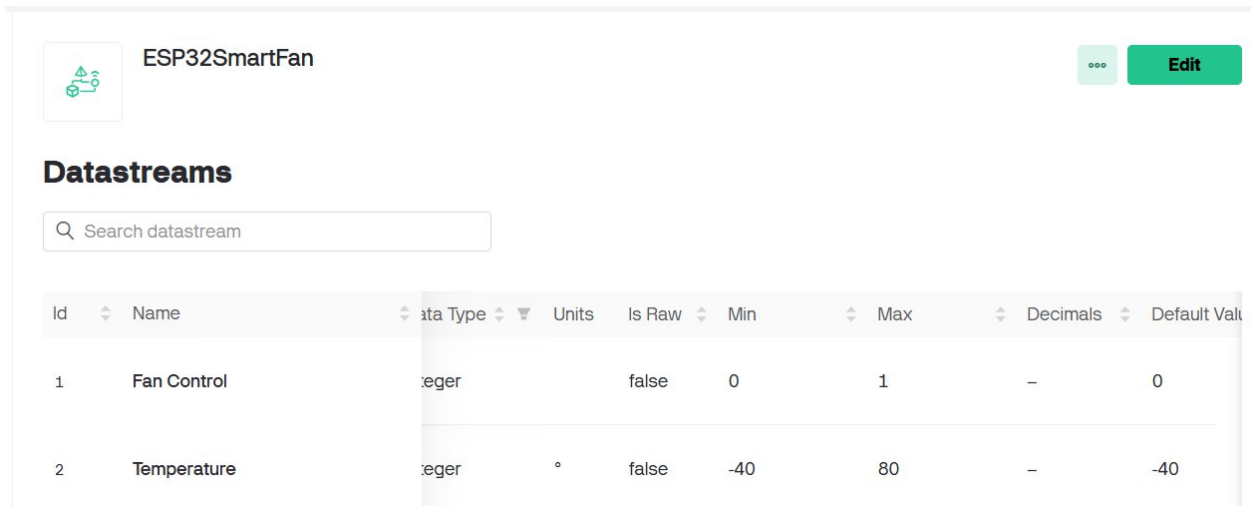
2. Tạo dự án mới:

- Mở ứng dụng Blynk, đăng nhập và nhấn vào "New Device".
- Chọn Hardware: ESP32.
- Chọn Connection Type: WiFi.
- Đặt tên dự án, ví dụ: "ESP32SmartFan".
- Nhấn "Create". Blynk sẽ gửi một Auth Token qua email của bạn. Lưu lại mã này (ví dụ: C26g*****%^^%&*****).

3. Tạo Datastreams:

- Trong Blynk Console (truy cập qua web tại blynk.cloud hoặc qua ứng dụng), vào phần "Datastreams":
 - Datastream 1 (V0): Điều khiển quạt (LED).
 - Name: "Fan Control"
 - Data Type: Integer

- Min: 0, Max: 1 (0 = Tắt, 1 = Bật)
- Pin: Virtual Pin V0
- Datastream 2 (V1): Hiển thị nhiệt độ.
 - Name: "Temperature"
 - Data Type: Float
 - Unit: °C
 - Pin: Virtual Pin V1



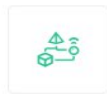
The screenshot shows the Blynk project interface for 'ESP32SmartFan'. Under the 'Datastreams' section, there is a search bar and a table listing the configured datastreams.

Id	Name	Data Type	Units	Is Raw	Min	Max	Decimals	Default Value
1	Fan Control	Integer		false	0	1	-	0
2	Temperature	Integer	°	false	-40	80	-	-40

Hình ảnh sau khi tạo Datastream.

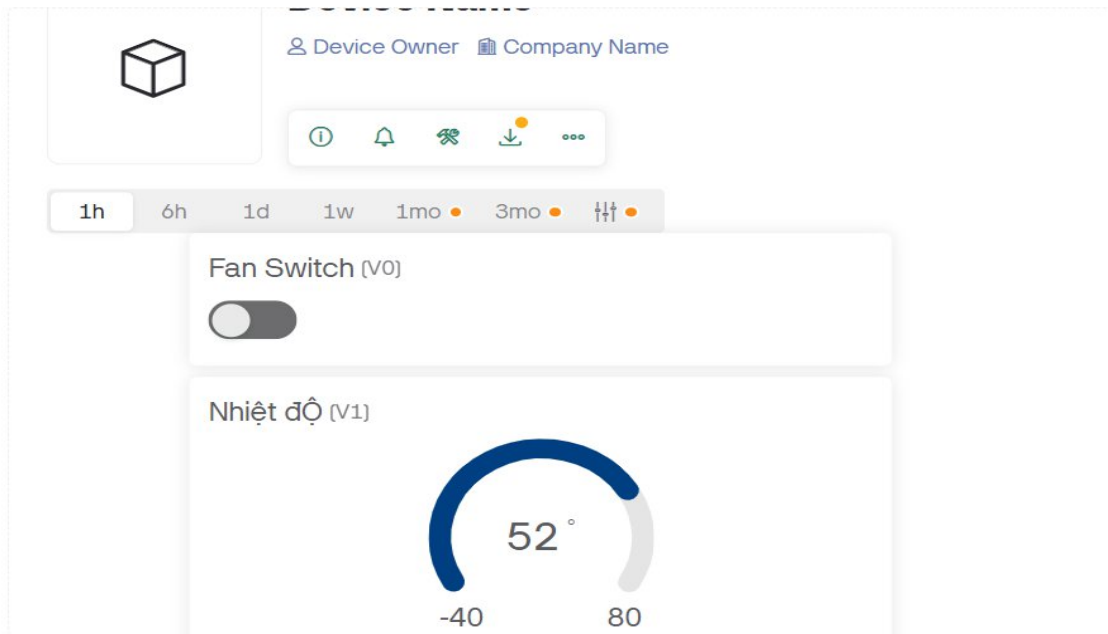
4. Thiết kế giao diện (Dashboard):

- Trong ứng dụng Blynk, vào phần "Web Dashboard" hoặc "Mobile Dashboard":
 - Thêm Switch Widget:
 - Kéo thả một Switch widget, gắn nó với Datastream V0 (Fan Control).
 - Đặt tên: "Fan Switch".
 - Thêm Gauge Widget:
 - Kéo thả một Gauge widget, gắn nó với Datastream V1 (Temperature).
 - Đặt tên: "Temperature Gauge".
 - Đặt phạm vi: 0°C đến 50°C.
- Nhấn "Save" để lưu giao diện.



ESP32SmartFan

Web Dashboard



Giao diện Dashboard sau khi đã được tạo.

Bước 2: Cấu hình dự án

- PlatformIO sẽ tạo một thư mục dự án với cấu trúc như sau:
SmartFan/
 - include/
 - lib/
 - src/
 - main.cpp
 - test/
 - platformio.ini
 - diagram.json
- File main.cpp trong thư mục src là nơi bạn sẽ viết mã lập trình.
- File platformio.ini là file cấu hình dự án.
- File diagram.json là file bổ sung .

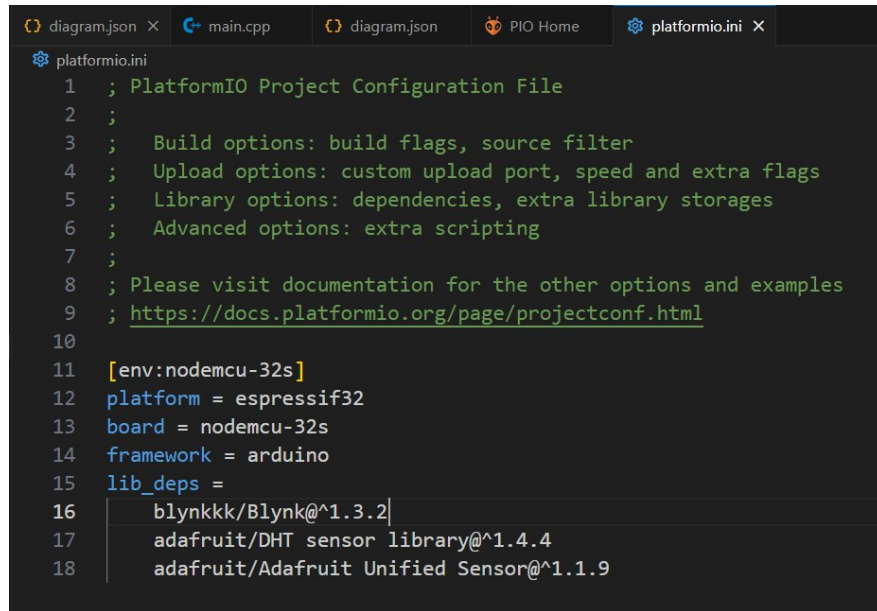
- File wokwi.toml là file cấu hình cho Wokwi simulator.

* Cài đặt thư viện

PlatformIO cho phép bạn cài đặt thư viện trực tiếp qua file platformio.ini.

1. Mở file platformio.ini:

- Thêm các thư viện cần thiết bằng cách chỉnh sửa file platformio.ini như sau



```

platformio.ini
1 ; PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, source filter
4 ; Upload options: custom upload port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ; Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:nodemcu-32s]
12 platform = espressif32
13 board = nodemcu-32s
14 framework = arduino
15 lib_deps =
16     blynkkk/Blynk@^1.3.2|
17     adafruit/DHT sensor library@^1.4.4
18     adafruit/Adafruit Unified Sensor@^1.1.9
  
```

- **Giải thích:**
 - blynk/Blynk: Thư viện Blynk để kết nối với ứng dụng Blynk.
 - adafruit/DHT sensor library: Thư viện để đọc dữ liệu từ cảm biến DHT22.
 - adafruit/Adafruit Unified Sensor: Thư viện phụ trợ cho DHT.

2. Lưu file:

- PlatformIO sẽ tự động tải các thư viện khi bạn lưu file platformio.ini.

3. Kết nối ESP32 với máy tính

- Kết nối bo mạch ESP32 với máy tính qua cáp USB.
- Trong VSCode, vào PlatformIO, chọn **Devices** (biểu tượng USB) để kiểm tra xem ESP32 đã được nhận diện chưa. Bạn sẽ thấy một cổng, ví dụ: COM3 (Windows)

Bước 3: Cập nhật mã lập trình trên VSCode

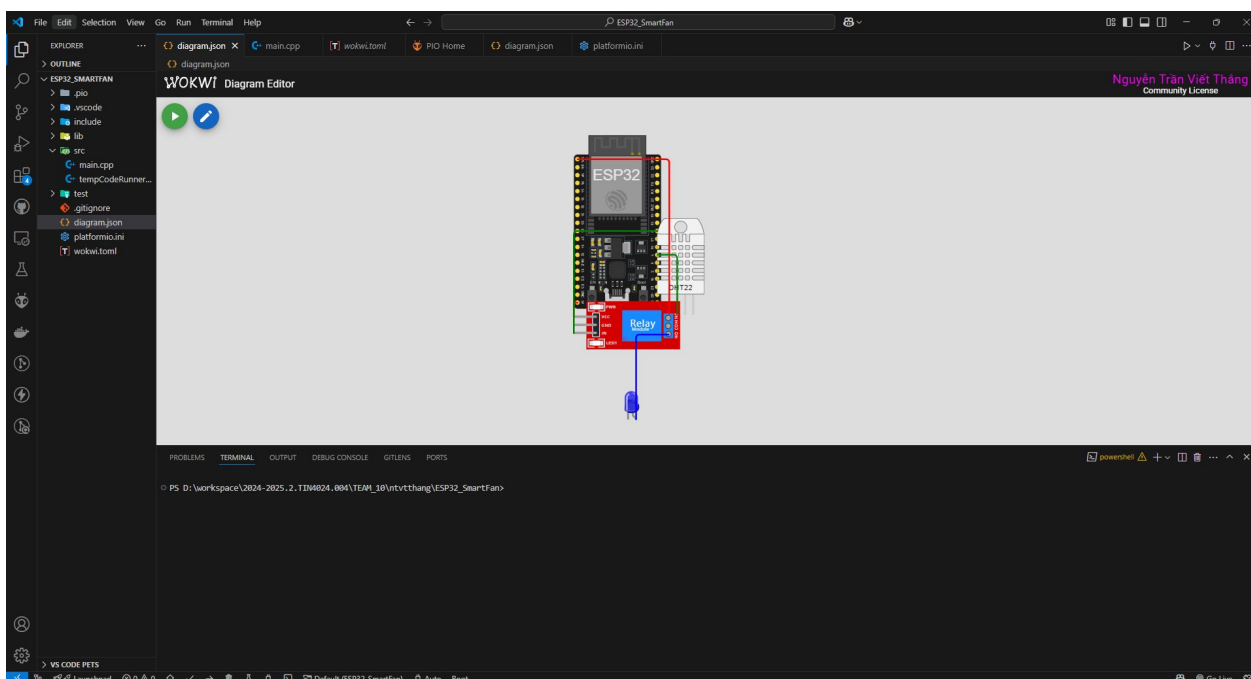
Mở file src/main.cpp và diagram.json trong dự án PlatformIO và dán mã sau:

```

1 // Thêm các thư viện cần thiết
2 #define BLYNK_TEMPLATE_ID "TMPL6V13Q9ywk"
3 #define BLYNK_TEMPLATE_NAME "ESP32SmartFan"
4 #define BLYNK_AUTH_TOKEN "CST80Y2RQx1F3kD8MkFuIFrHTGLqIZRz" // Thay bằng Auth Token từ email
5
6 #include <WiFi.h>
7 #include <WiFiClient.h>
8 #include <BlynkSimpleEsp32.h>
9 #include <DHT.h>
10
11 // Thông tin WiFi
12 char ssid[] = "Wokwi-GUEST"; // Tên mạng WiFi
13 char pass[] = ""; // Mật khẩu mạng WiFi
14
15 // Định nghĩa chân và loại cảm biến
16 #define DHTPIN 4 // Chân GPIO 4 kết nối với DHT22
17 #define DHTTYPE DHT22 // Loại cảm biến
18 #define RELAY_PIN 5 // Chân GPIO 5 kết nối với Relay
19
20 DHT dht(DHTPIN, DHTTYPE);
21
22 // Biến lưu trạng thái quạt
23 int fanState = 0;
24
25 void setup() {
26 // Khởi tạo Serial để debug
27 Serial.begin(115200);
28
29 // Khởi tạo cảm biến DHT
30 dht.begin();
31
32 // Cấu hình chân Relay
33 pinMode(RELAY_PIN, OUTPUT);
34 digitalWrite(RELAY_PIN, LOW); // Tắt Relay ban đầu
35
36 // Kết nối với Blynk
37 Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
38 Serial.println("Connecting to WiFi and Blynk...");
39 }
40
41 // Hàm nhận lệnh từ Blynk để điều khiển quạt
42 BLYNK_WRITE(V0) {
43 fanState = param.asInt(); // Nhận giá trị từ Switch (0 hoặc 1)
44 digitalWrite(RELAY_PIN, fanState); // Bật/tắt Relay
45 Serial.print("Fan State: ");
46 Serial.println(fanState ? "ON" : "OFF");
47 }
48
49 void loop() {
50 // Chạy Blynk
51 Blynk.run();
52
53 // Đọc nhiệt độ từ DHT22
54 float temp = dht.readTemperature();
55 if (isnan(temp)) {
56 Serial.println("Failed to read from DHT sensor!");
57 return;
58 }
59
60 // Gửi nhiệt độ lên Blynk
61 Blynk.virtualWrite(V1, temp);
62 Serial.print("Temperature: ");
63 Serial.print(temp);
64 Serial.println(" °C");
65
66 // Điều khiển quạt tự động dựa trên nhiệt độ (nếu cần)
67 if (temp > 30 && fanState == 0) {
68 fanState = 1;
69 digitalWrite(RELAY_PIN, HIGH);
70 Blynk.virtualWrite(V0, 1); // Cập nhật trạng thái trên Blynk
71 Serial.println("Fan ON (Auto)");
72 } else if (temp < 25 && fanState == 1) {
73 fanState = 0;
74 digitalWrite(RELAY_PIN, LOW);
75 Blynk.virtualWrite(V0, 0); // Cập nhật trạng thái trên Blynk
76 Serial.println("Fan OFF (Auto)");
77 }
78
79 delay(2000); // Đọc mỗi 2 giây
80 }

```

Hình ảnh code main.cpp.



Hình ảnh file *diagram.json*.

Bước 4: Upload mã lên ESP32

1. Kiểm tra kết nối:

- Đảm bảo ESP32 đã được kết nối với máy tính qua cáp USB.
- Trong PlatformIO, nhấn vào biểu tượng **PlatformIO: Serial Monitor** (hình kính lúp) để kiểm tra cổng COM.

2. Build mã:

- Nhấn vào biểu tượng **PlatformIO: Build** (hình dấu tích) để biên dịch mã.
- Nếu không có lỗi, bạn sẽ thấy thông báo "SUCCESS".

3. Upload mã:

- Nhấn vào biểu tượng **PlatformIO: Upload** (hình mũi tên phải) để tải mã lên ESP32.
- Chờ quá trình upload hoàn tất, bạn sẽ thấy thông báo "SUCCESS".

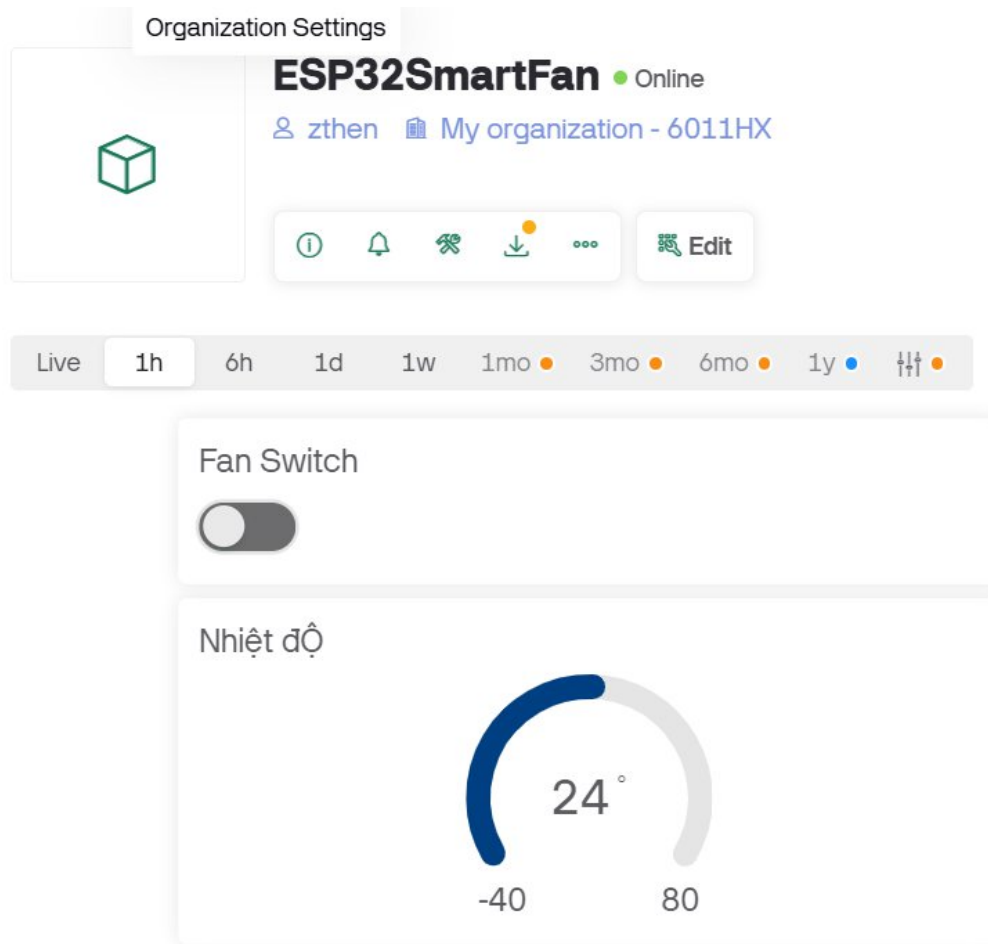
4. Mở Serial Monitor:

- Nhấn vào biểu tượng **PlatformIO: Serial Monitor** (hình kính lúp) để mở Serial Monitor.
- Đặt tốc độ baud rate là 115200 (khớp với `Serial.begin(115200)` trong mã).
- Bạn sẽ thấy các thông báo như:

```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE  PORTS

Temperature: 24.00 °C
Temperature: 24.00 °C
Fan State: ON
Temperature: 24.00 °C
Fan OFF (Auto)
Temperature: 24.00 °C
Temperature: 24.00 °C
```

Hình ảnh chạy ở file JSON.



Hình ảnh hoạt động ở Blynk.

Bước 5: Kiểm tra và khắc phục sự cố

1. Không kết nối được WiFi:

- Kiểm tra lại ssid và pass trong mã.
- Đảm bảo WiFi là băng tần 2.4GHz (ESP32 không hỗ trợ 5GHz).

2. Không kết nối được Blynk:

- Kiểm tra “BLYNK_AUTH_TOKEN” có đúng không.
- Đảm bảo ESP32 đã kết nối WiFi thành công trước khi kết nối Blynk.

3. Cảm biến không đọc được:

- Kiểm tra kết nối vật lý của DHT22 (GND, VCC, SDA) theo sơ đồ.
- Đảm bảo bạn đã cài đặt đúng thư viện DHT.

4. Relay/LED không hoạt động:

- Kiểm tra kết nối của Relay (GND, VCC, IN) và LED (NO của Relay → Anode của LED, Cathode → GND).

VI. KẾT QUẢ VÀ ĐÁNH GIÁ

6.1. Kết quả thử nghiệm

Hệ thống đã được triển khai và kiểm tra toàn diện, cho kết quả đáp ứng đúng theo yêu cầu thiết kế:

- Khi nhiệt độ vượt ngưỡng 30°C , quạt tự động kích hoạt để làm mát môi trường.
- Khi nhiệt độ giảm xuống dưới 30°C , quạt tự động tắt để tiết kiệm năng lượng.
- Dữ liệu nhiệt độ được thu thập liên tục và hiển thị chính xác trên nền tảng Blynk.
- Thời gian phản hồi của hệ thống đạt dưới 5 giây kể từ khi phát hiện thay đổi nhiệt độ.

6.2. Đánh giá hiệu suất

Ưu điểm:

- Tự động hóa hoàn toàn: Hệ thống hoạt động độc lập, không cần sự can thiệp thường xuyên từ người dùng.
- Kết nối IoT hiệu quả: Giao diện Blynk thân thiện với người dùng, cho phép điều khiển và giám sát từ xa thông qua điện thoại thông minh hoặc máy tính.
- Tiết kiệm năng lượng: Quạt chỉ hoạt động khi điều kiện nhiệt độ yêu cầu, giảm thiểu lãng phí điện năng.
- Độ chính xác cao: Cảm biến DHT22 cung cấp phép đo nhiệt độ với độ chính xác $\pm 0.5^{\circ}\text{C}$, đảm bảo kích hoạt quạt tại ngưỡng nhiệt độ phù hợp.
- Chi phí triển khai hợp lý: Sử dụng các thành phần phần cứng phổ biến, dễ tiếp cận và chi phí thấp.
- Khả năng mở rộng: Hệ thống có thể dễ dàng tích hợp thêm các chức năng khác như đo độ ẩm, ánh sáng hoặc điều khiển nhiều thiết bị.

Nhược điểm:

- Phụ thuộc vào kết nối mạng: Chức năng giám sát và điều khiển từ xa phụ thuộc hoàn toàn vào tính ổn định của mạng WiFi. Trong trường hợp mất kết nối, chỉ chức năng tự động cơ bản được duy trì.

- Hạn chế trong điều khiển quạt: Relay hiện tại chỉ có chế độ bật/tắt đơn giản, chưa hỗ trợ điều chỉnh tốc độ quạt theo nhiều cấp độ khác nhau.
- Khả năng chống chịu môi trường: Hệ thống chưa được trang bị vỏ bảo vệ chống bụi và nước, giới hạn khả năng triển khai trong các môi trường khắc nghiệt.
- Hạn chế về nguồn điện: Phụ thuộc vào nguồn điện cố định, chưa có giải pháp dự phòng khi mất điện.
- Thiếu cơ chế cảnh báo: Hệ thống chưa tích hợp chức năng cảnh báo qua thông báo đầy hoặc email khi nhiệt độ vượt ngưỡng nguy hiểm.

Đề xuất cải tiến:

- Tích hợp module PWM để điều khiển tốc độ quạt theo nhiều cấp độ dựa trên mức chênh lệch nhiệt độ.
- Bổ sung pin dự phòng hoặc UPS mini để đảm bảo hoạt động liên tục trong trường hợp mất điện.
- Phát triển chế độ offline cho phép lưu trữ dữ liệu cục bộ khi mất kết nối internet.
- Thiết kế vỏ bảo vệ chống bụi và nước để nâng cao độ bền của hệ thống.
- Tích hợp thêm cảm biến độ ẩm để điều khiển quạt dựa trên chỉ số nhiệt độ cảm nhận thực tế.

VII. HƯỚNG PHÁT TRIỂN TƯƠNG LAI

7.1. Cải tiến phần cứng

7.1.1. Hệ thống điều khiển tốc độ quạt thông minh:

- Thay thế relay bật/tắt đơn giản bằng mạch điều khiển TRIAC hoặc MOSFET PWM để điều chỉnh tốc độ quạt vô cấp dựa trên chênh lệch giữa nhiệt độ hiện tại và nhiệt độ mong muốn.
- Tích hợp cảm biến nhiệt hồng ngoại MLX90614 để đo nhiệt độ không tiếp xúc, cho phép đo trực tiếp nhiệt độ người dùng để điều chỉnh quạt thông minh hơn.

7.1.2. Mở rộng đa cảm biến:

- Tích hợp cảm biến CO2 MH-Z19 để đo nồng độ CO2, điều chỉnh quạt dựa trên chất lượng không khí.
- Bổ sung cảm biến PM2.5/PM10 để giám sát bụi mịn, tự động tăng tốc quạt khi phát hiện không khí ô nhiễm.
- Thêm cảm biến chuyển động PIR để chỉ bật quạt khi phát hiện có người trong phòng, tiết kiệm điện năng.

7.1.3. Nâng cấp nguồn năng lượng:

- Thiết kế mạch thu năng lượng mặt trời kết hợp với pin Li-ion 18650 có bộ quản lý sạc (BMS) để tạo hệ thống tự cung tự cấp.
- Tích hợp UPS mini với chuyển đổi nguồn tự động giữa điện lưới và pin dự phòng, đảm bảo hoạt động liên tục.

7.1.4. Bảo vệ và độ bền:

- Thiết kế vỏ bảo vệ in 3D với chuẩn IP65 (chống bụi và chống nước) cho toàn bộ hệ thống.
- Tích hợp mạch bảo vệ quá áp và quá dòng để bảo vệ các thành phần điện tử khỏi đột biến điện.

7.2. Cải tiến phần mềm

7.2.1. Thuật toán thông minh:

- Phát triển thuật toán học máy đơn giản chạy trên ESP32 để dự đoán xu hướng nhiệt độ và điều chỉnh quạt trước khi nhiệt độ thay đổi đáng kể.
- Tích hợp thuật toán PID (Proportional-Integral-Derivative) để điều khiển tốc độ quạt với độ chính xác cao, tránh dao động nhiệt độ quá lớn.

7.2.2. Hệ thống thông báo nâng cao:

- Tích hợp Telegram Bot API hoặc MQTT broker để gửi thông báo và cảnh báo tới người dùng qua nhiều kênh.
- Phát triển hệ thống cảnh báo đa cấp dựa trên mức độ nghiêm trọng của nhiệt độ (cảnh báo, khẩn cấp, nguy hiểm).

7.2.3. Tích hợp hệ sinh thái thông minh:

- Kết nối với API thời tiết để điều chỉnh hoạt động quạt dựa trên dự báo nhiệt độ sắp tới.
- Tích hợp với Google Assistant hoặc Amazon Alexa để điều khiển bằng giọng nói.
- Phát triển gateway Z-Wave/Zigbee để kết nối với các hệ thống nhà thông minh hiện có.

7.2.4. Phân tích dữ liệu nâng cao:

- Triển khai cơ sở dữ liệu thời gian thực (InfluxDB) để lưu trữ lịch sử nhiệt độ và hoạt động quạt.
- Phát triển dashboard trực quan (Grafana) hiển thị phân tích nhiệt độ theo thời gian, mức tiêu thụ điện năng và hiệu suất hệ thống.
- Tích hợp thuật toán phát hiện bất thường để cảnh báo khi nhiệt độ biến động bất thường hoặc cảm biến/quạt gặp sự cố.

7.3. Mở rộng ứng dụng

7.3.1. Hệ thống điều hòa không khí tổng thể:

- Phát triển thành hệ thống điều khiển đa vùng cho phép điều khiển nhiều quạt/điều hòa trong các phòng khác nhau.
- Tích hợp với máy phun sương để tạo giải pháp làm mát toàn diện, tiết kiệm năng lượng hơn điều hòa truyền thống.

7.3.2. Ứng dụng trong nông nghiệp thông minh:

- Mở rộng thành hệ thống quản lý nhà kính tích hợp điều khiển nhiệt độ, độ ẩm, ánh sáng và CO₂.
- Phát triển mô hình dự báo sản lượng dựa trên dữ liệu nhiệt độ thu thập được.

7.3.3. Giải pháp thương mại:

- Thiết kế bộ công cụ DIY cho phép người dùng tự lắp đặt hệ thống điều khiển quạt thông minh.
- Phát triển dịch vụ đám mây chuyên biệt cho việc giám sát và quản lý nhiệt độ trong không gian văn phòng và công nghiệp.

7.4. Dự án mã nguồn mở

- Phát triển thư viện ESP32-FanControl mã nguồn mở, hỗ trợ nhiều loại cảm biến và phương thức điều khiển quạt.
- Tạo tài liệu chi tiết và video hướng dẫn để cộng đồng dễ dàng triển khai và cải tiến dự án.
- Xây dựng cộng đồng người dùng trên GitHub để chia sẻ cải tiến và giải quyết vấn đề.

VIII. TÀI LIỆU THAM KHẢO

- [1]. https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.
- [2]. <https://learn.adafruit.com/dht/overview>.
- [3]. Blynk Inc. (2024). *Blynk IoT Platform Documentation*.
- [4]. <https://cdn.sparkfun.com/assets/f/7/d/9/c/DHT22.pdf>.