HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KÌ MÔN HỌC: IOT VÀ ỨNG DỤNG

Chủ đề: Dải đèn LED trang trí phản ứng âm nhạc

1. **Họ và tên: Bùi Quang Hiếu.. Mã SV: B22DCCN301 – Nhóm trưởng**
2. **Họ và tên: Dương Văn Thuận.. Mã SV: B22DCCN841**
3. **Họ và tên: Nguyễn Việt Anh.. Mã SV: B22DCCN037**

HÀ NỘI, THÁNG 10/2025

Contents

I. GIỚI THIỆU ĐỀ TÀI

1. Mục tiêu dự án

Dự án **“Dải đèn LED thông minh phản ứng âm nhạc”** nhằm tạo ra một hệ thống có khả năng:

- Tự động thay đổi màu sắc, độ sáng và hiệu ứng của dải đèn LED dựa trên tín hiệu âm thanh.
- Phản ứng theo nhịp điệu (beat) của nhạc để tạo ra trải nghiệm ánh sáng sinh động.
- Có thể **điều khiển thủ công qua web hoặc app di động**, ví dụ bật/tắt đèn, chọn hiệu ứng, thay đổi chế độ nhạc.

=> **Mục tiêu tổng quát:**
Xây dựng một hệ thống IoT nhỏ gọn, có thể ứng dụng thực tế trong phòng ngủ, sân khấu mini, quán cà phê, hoặc nhà thông minh.

2. Phạm vi hệ thống

Hệ thống bao gồm:

- **Thiết bị phần cứng:** dải LED (WS2812B hoặc NeoPixel), vi điều khiển (ESP32), microphone hoặc cảm biến âm thanh.
- **Phần mềm nhúng:** chạy trên ESP32 để nhận tín hiệu âm thanh, xử lý, điều khiển LED.
- **Giao diện điều khiển:** web app hoặc mobile app giúp người dùng chọn chế độ hoạt động, màu sắc, hiệu ứng.
- **Kết nối:** sử dụng WiFi để giao tiếp giữa thiết bị và web/app.

3. Mô tả tổng quan

Hệ thống hoạt động theo quy trình sau cho 2 chế độ:

Chế độ 1: Sử dụng microphone để thu âm thanh phản ứng

Bước 1: Thu âm thanh từ môi trường

- Microphone (cảm biến âm thanh analog) thu tín hiệu âm thanh xung quanh.
- Tín hiệu âm thanh được **chuyển đổi thành tín hiệu điện áp analog** và gửi vào **chân ADC của ESP32**.
- Tín hiệu này liên tục thay đổi theo cường độ âm thanh (to/nhỏ).

Bước 2: Xử lý tín hiệu âm thanh trên ESP32

- ESP32 đọc dữ liệu tín hiệu âm thanh bằng bộ **ADC (Analog-to-Digital Converter)**.
- Sau đó, chương trình trên ESP32 thực hiện **xử lý tín hiệu số (Digital Signal Processing)**:
    - **Lọc nhiễu (Noise Filtering):** loại bỏ tạp âm.
    - **Phân tích biên độ (Amplitude Detection):** xác định cường độ âm thanh.
    - **Phân tích tần số (Frequency Detection – nếu có):** xác định dải tần trầm/bổng để tạo hiệu ứng đa dạng.

Bước 3: Xử lý logic hiệu ứng LED

- Dựa trên giá trị cường độ và tần số thu được, ESP32 chọn và cập nhật **hiệu ứng ánh sáng tương ứng**:
    - Cường độ cao → LED sáng mạnh hoặc đổi màu nhanh.
    - Cường độ thấp → LED sáng mờ, hiệu ứng nhẹ hơn.
- Các hiệu ứng có thể bao gồm:
    - **“Music Pulse”:** LED nhấp nháy theo nhịp.
    - **“Spectrum Mode”:** LED hiển thị theo cột tần số.
    - **“Fade Mode”:** LED đổi màu mượt theo biến động âm thanh.

Bước 4: Hiển thị hiệu ứng trên dải LED

- Sau khi xử lý xong dữ liệu, **ESP32 gửi tín hiệu điều khiển tới từng LED** trong dải WS2812B.

- Mỗi LED hiển thị màu sắc và độ sáng khác nhau dựa trên **biên độ hoặc dải tần âm thanh**.

Bước 5: Điều khiển & giám sát qua web/app

- Người dùng có thể **bật/tắt hệ thống, chọn chế độ hiển thị, đổi màu nền hoặc độ sáng** thông qua **giao diện web hoặc ứng dụng điện thoại**.

- Giao diện gửi yêu cầu đến ESP32 qua Wi-Fi (HTTP/MQTT).

- ESP32 phản hồi lại trạng thái hiện tại để **đồng bộ giao diện và hệ thống LED**.

Bước 6: Cập nhật trạng thái và lưu cấu hình

- ESP32 lưu lại **chế độ hoạt động, màu nền, tốc độ hiệu ứng** vào bộ nhớ (EEPROM hoặc SPIFFS).

- Khi hệ thống khởi động lại, các thiết lập này sẽ được **tự động khôi phục** để đảm bảo trải nghiệm người dùng nhất quán.

Chế độ 2: Nhận dữ liệu âm thanh, phân tích nhạc từ thiết bị điều khiển

Bước 1: Phát hiện âm thanh và phân tích tại thiết bị điều khiển

- Khi người dùng phát nhạc trên **web/app điều khiển**, hệ thống sẽ sử dụng **thư viện xử lý âm thanh (Web Audio API hoặc tương tự)** để:
    - Lấy **dữ liệu phổ tần (frequency spectrum)** và **biên độ (amplitude)** theo thời gian thực.
    - Phân tích các **đỉnh nhịp (beat detection)** hoặc **dải tần số (bass, mid, treble)** để tạo ra dữ liệu hiệu ứng.

Bước 2: Gửi dữ liệu điều khiển đến ESP32

- Sau khi phân tích, app/web sẽ gửi **gói dữ liệu ánh sáng** (Light Data Packet) tới ESP32 qua:
    - **Wi-Fi (MQTT hoặc WebSocket)** nếu điều khiển qua Internet.
    - **LAN/Local IP** nếu thiết bị cùng mạng.

Bước 3: Xử lý dữ liệu tại ESP32

- ESP32 nhận gói dữ liệu và thực hiện:
    - Giải mã thông tin (bass/mid/treble, độ sáng, màu sắc).
    - Ánh xạ các giá trị này vào **vị trí LED cụ thể** trên dải LED.
    - Cập nhật hiển thị LED theo dữ liệu nhận được.

Bước 4: Hiển thị hiệu ứng LED

- Dải LED hiển thị hiệu ứng theo dữ liệu âm thanh gửi đến, đảm bảo:
    - Độ trễ < 200 ms để tạo cảm giác "phản ứng theo nhạc" tức thì.
    - Màu sắc và hiệu ứng đồng bộ với nhịp và giai điệu bài hát.

Bước 5: Giao diện điều khiển & chế độ hoạt động

- Người dùng có thể điều khiển các tùy chọn trên **web/app**:
    - **Bật/tắt chế độ Music Sync**.
    - **Chọn kiểu hiển thị:** Spectrum Bar, Waveform, Pulse,...
    - **Điều chỉnh tốc độ phản ứng**, độ sáng, màu nền.
- Giao diện gửi yêu cầu cập nhật đến ESP32 để thay đổi hiệu ứng tức thì.

Bước 6: Đồng bộ trạng thái hệ thống

- ESP32 gửi lại trạng thái hiện tại (chế độ, màu, độ sáng, tốc độ, v.v.) về **server hoặc web/app** để hiển thị cho người dùng.
- Đảm bảo dữ liệu hai chiều (bi-directional) để app và LED luôn đồng bộ

4. Các ràng buộc và giả định

Loại	**Mô tả**
Ràng buộc phần cứng	**Hệ thống phải dùng thiết bị chi phí thấp, dễ lắp đặt.**
Giàng buộc phần mềm	**Ứng dụng phải hoạt động ổn định trên nền WiFi, tốc độ phản hồi nhanh.**
Giả định	**Người dùng có kết nối WiFi và điện**

	thoại/máy tính có trình duyệt hoặc app
Giả định	**Môi trường âm thanh có cường độ đủ để cảm biến nhận dạng tín hiệu.**

II. PHÂN TÍCH YÊU CẦU

1. Yêu cầu chức năng (Functional Requirements)

STT	Yêu cầu chức năng	Mô tả chi tiết
1	Hệ thống thu nhận tín hiệu âm thanh	Thiết bị (ESP32) thu tín hiệu từ micro để xác định cường độ và tần số âm thanh
2	Điều khiển dải đèn LED	Đèn LED đổi màu hoặc nhấp nháy theo nhạc dựa trên tín hiệu âm thanh nhận được
3	Kết nối Wi-Fi	ESP32 kết nối mạng Wi-Fi để giao tiếp với Web App hoặc MQTT Broker
4	Điều khiển từ xa qua Web	Người dùng có thể bật/tắt đèn, đổi màu, chọn chế độ qua giao diện Web hoặc ứng dụng
5	Hiển thị trạng thái hệ thống	Giao diện Web hiển thị trạng thái LED hiện tại (màu, chế độ, kết nối)
6	Giao tiếp giữa thiết bị và Web	Dữ liệu điều khiển và phản hồi được truyền qua WebSocket hoặc MQTT
7	Cập nhật firmware từ xa (OTA)	Người quản trị có thể cập nhật chương trình cho ESP32 qua Internet mà không cần đến tận nơi
8	Lưu cấu hình người dùng	Hệ thống lưu các chế độ yêu thích hoặc cấu hình màu sắc trên server
9	Hỗ trợ nhiều thiết bị LED	Người dùng có thể thêm, xóa, điều khiển nhiều dải đèn LED khác nhau qua cùng một giao diện

2. Yêu cầu phi chức năng (Non-functional Requirements)

STT	Yêu cầu phi chức năng	Mô tả chi tiết
1	Thời gian phản hồi ≤ 1 giây	Khi người dùng điều khiển, đèn phản hồi gần như ngay lập tức
2	Xử lý tín hiệu âm thanh real-time	Tín hiệu nhạc được xử lý và hiển thị tức thời trên LED
3	Tự động kết nối lại khi mất mạng	ESP32 tự động reconnect Wi-Fi hoặc MQTT Broker khi bị ngắt
4	Dữ liệu không bị mất khi lỗi tạm thời	Hệ thống lưu trạng thái hiện tại của LED và khôi phục lại sau khi reset
5	Xác thực khi truy cập Web	Người dùng cần đăng nhập hoặc có token để điều khiển
6	OTA an toàn	Firmware update được tải qua HTTPS và xác minh checksum
7	Hỗ trợ nhiều thiết bị cùng lúc	Server và giao thức cho phép kết nối >10 thiết bị LED cùng lúc
8	Giao diện thân thiện, dễ dùng	Giao diện Web được thiết kế trực quan, dễ điều khiển cho người mới
9	Hệ thống dễ cập nhật	Mã nguồn tách biệt giữa giao diện, server và firmware để dễ bảo trì
10	Hỗ trợ nhiều trình duyệt	Web hoạt động ổn định trên Chrome, Edge, Firefox

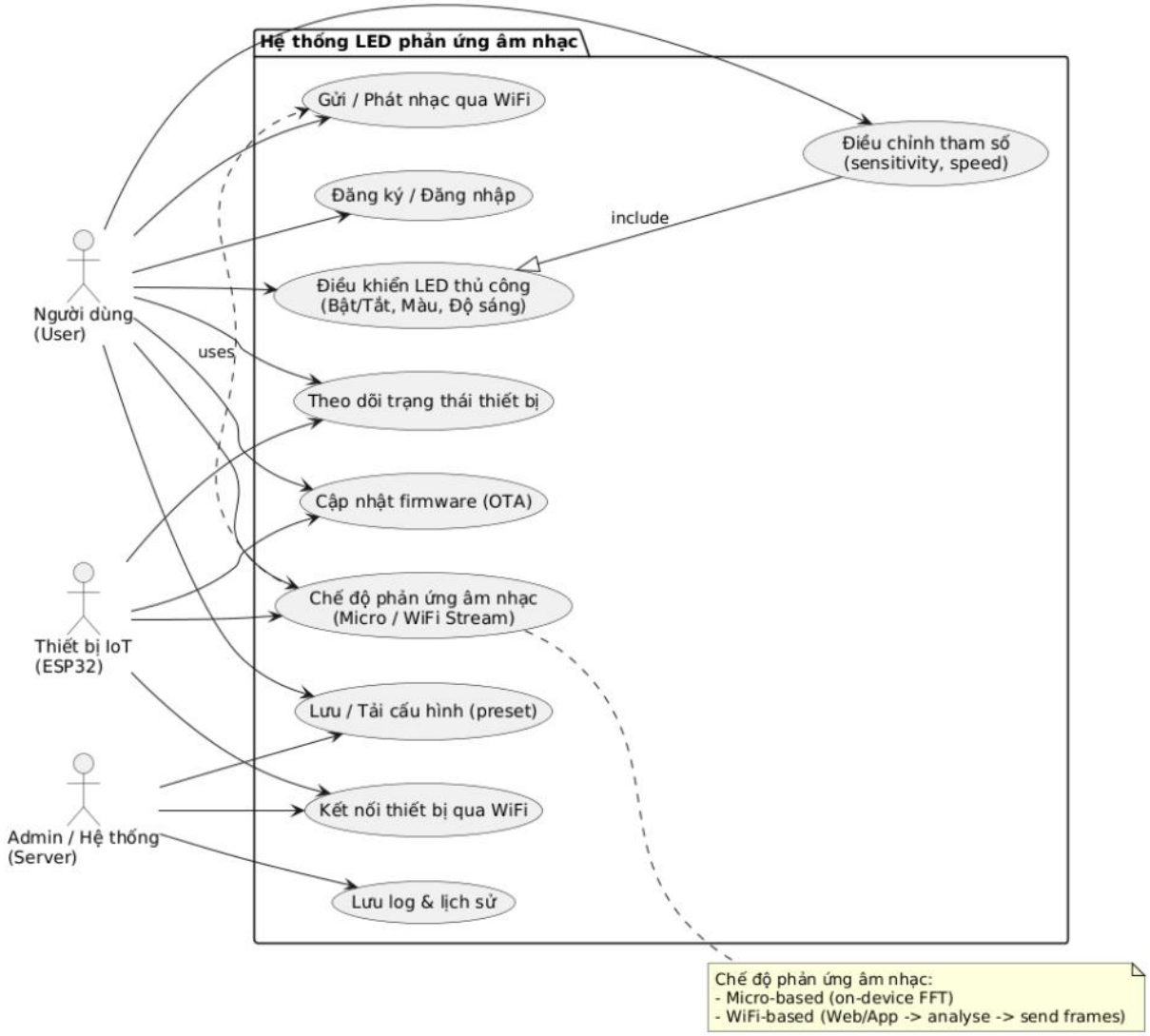
3. Thu thập yêu cầu

Nhóm liên quan	Yêu cầu chính
Người dùng cuối (chủ nhà, người dùng phòng ngủ, quán cà phê, hoặc người dùng giải trí)	- Bật/tắt đèn thủ công hoặc tự động dựa trên âm thanh. - Điều chỉnh độ sáng, màu sắc và hiệu ứng (như Music Pulse, Spectrum Mode, Fade Mode) theo nhu cầu. - Giám sát trạng thái đèn (chế độ, màu sắc, kết nối) qua web hoặc app di động. - Nhận phản ứng ánh sáng tức thì theo nhịp điệu nhạc từ microphone hoặc thiết bị phát nhạc. - Lưu và khôi phục cấu hình cá nhân (chế độ yêu thích, tốc độ hiệu ứng) khi khởi động lại. - Nhận cảnh báo khi hệ thống mất kết nối hoặc lỗi âm thanh.
Doanh nghiệp / Nhà phát triển	- Hệ thống dễ mở rộng thêm dải LED hoặc thiết bị khác. - Giao diện thân thiện, dễ bảo trì và cập nhật (tách biệt giữa firmware, server và giao diện). - Thu thập dữ liệu sử dụng (tín hiệu âm thanh, hiệu ứng) để tối ưu hóa thuật toán phản ứng nhạc. - Hỗ trợ nhiều thiết bị LED cùng lúc qua một giao diện duy nhất. - Ứng dụng thực tế trong nhà thông minh, sân khấu mini hoặc quán cà phê với chi phí thấp.
Kỹ thuật / IT	- Kết nối qua WiFi chuẩn 2.4GHz để giao tiếp giữa ESP32 và web/app. - Giao thức MQTT, WebSocket hoặc HTTP để trao đổi dữ liệu thời gian thực (độ trễ < 200ms). - Mã hóa dữ liệu và xác thực truy cập (đăng nhập hoặc token) để bảo mật. - Hỗ trợ cập nhật firmware từ xa (OTA) qua HTTPS với xác minh checksum. - Tự động kết nối lại khi mất mạng và lưu trạng thái để tránh mất dữ liệu. - Xử lý tín hiệu âm thanh real-time với lọc nhiễu, phân tích biên độ/tần số.

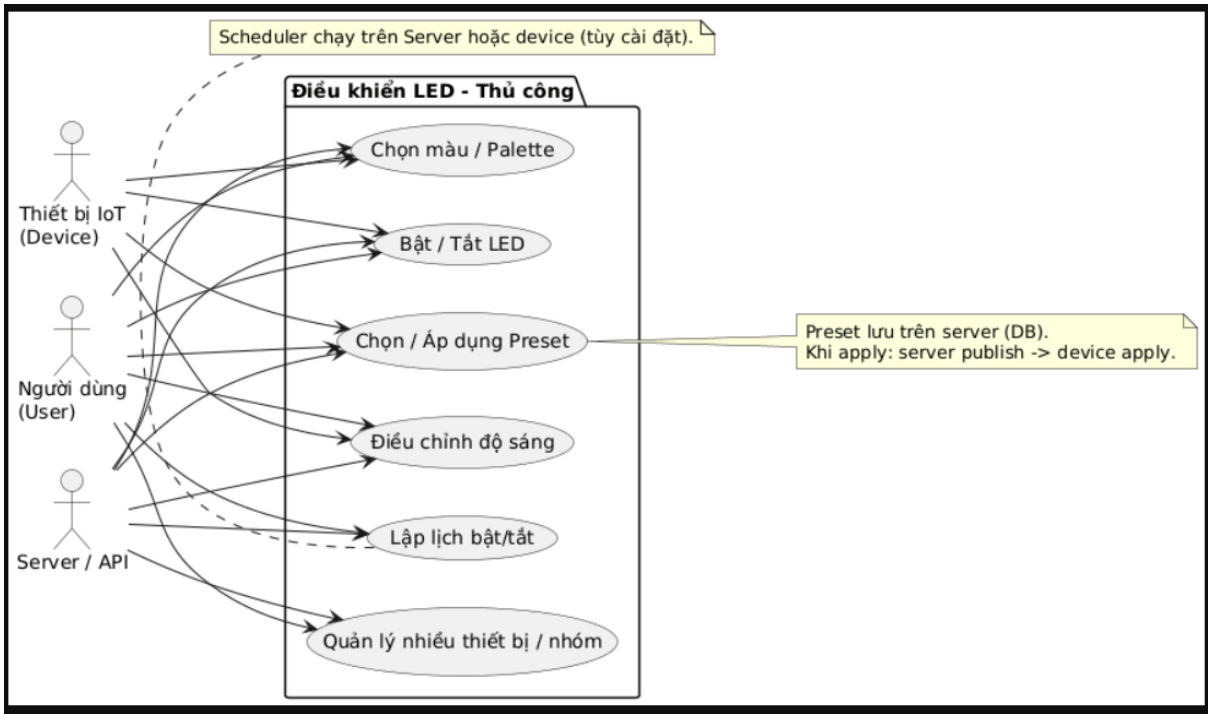
III. BIỂU DIỄN CHỨC NĂNG

1. Use Case Diagram

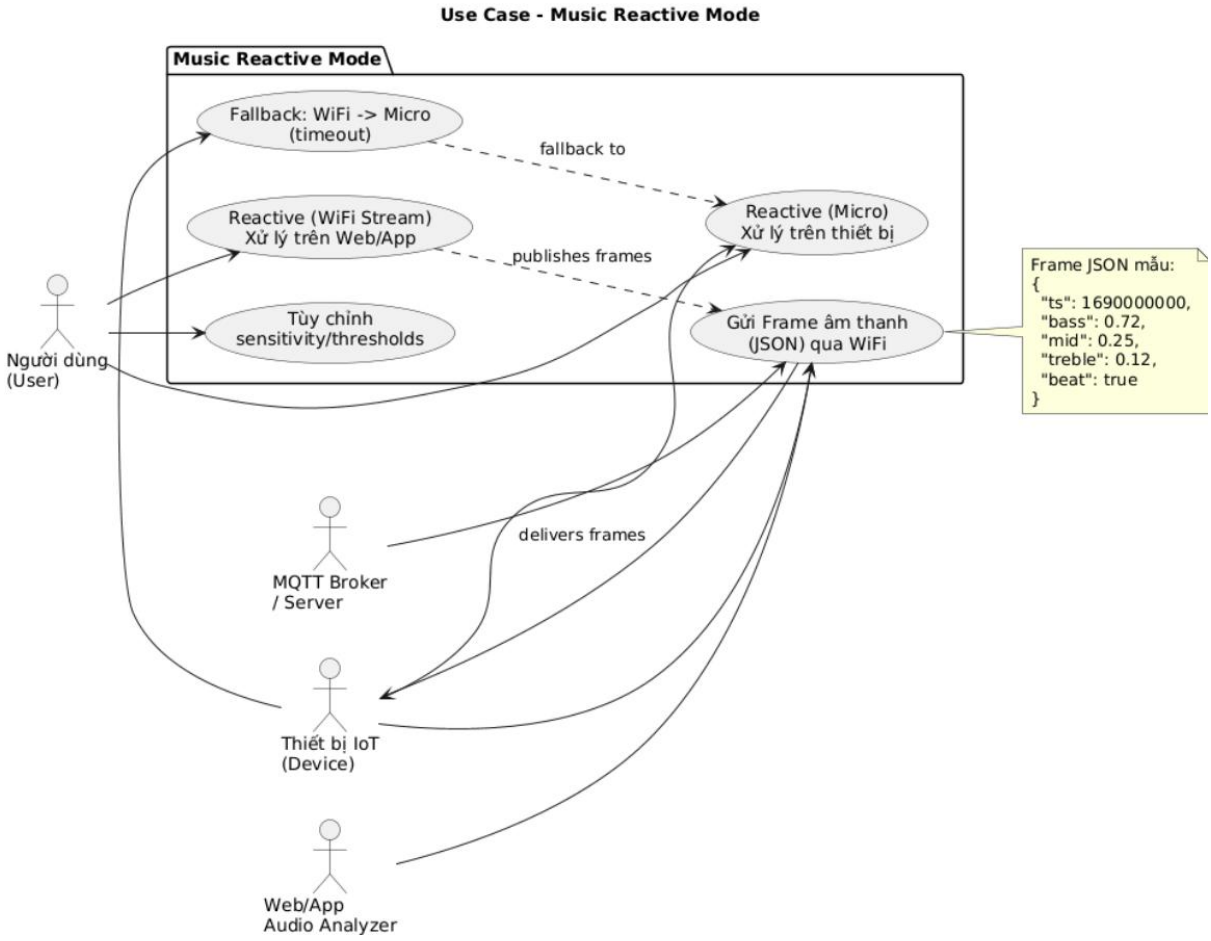
a. Usecase tổng quan



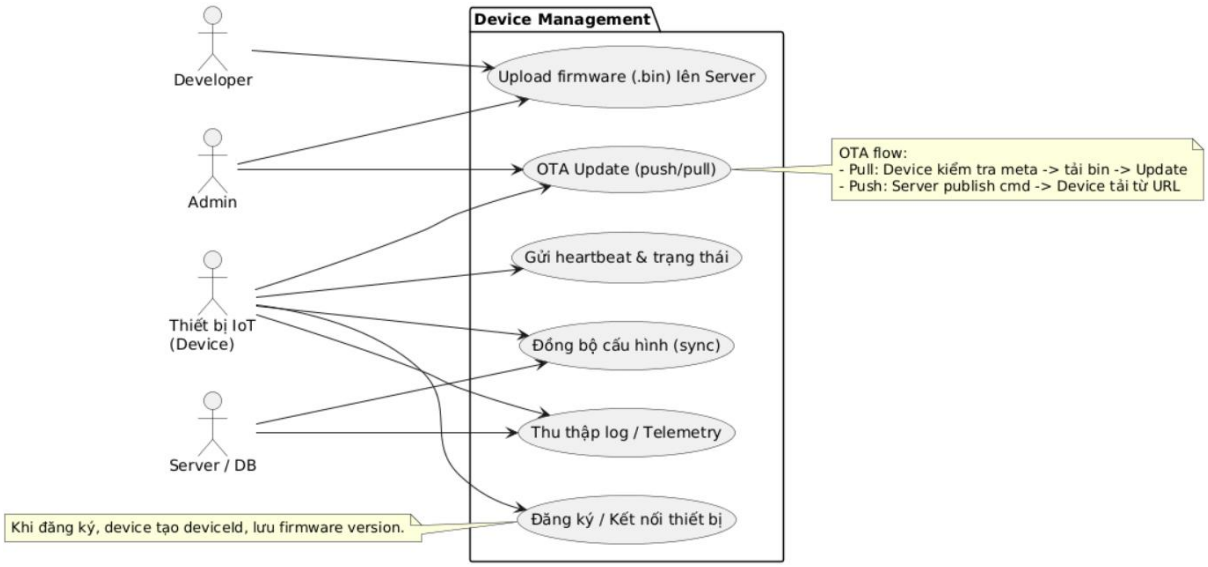
b. Usecase chi tiết điều khiển LED thủ công



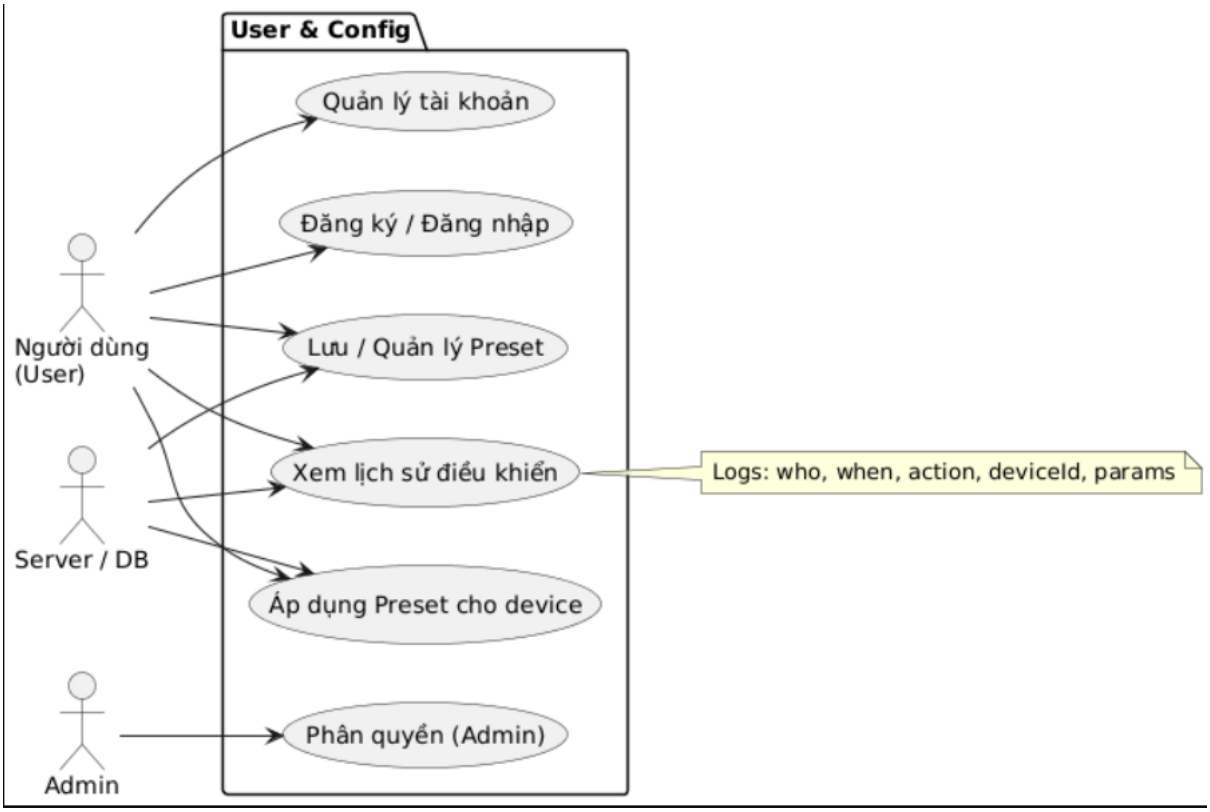
c. Usecase chi tiết phản ứng âm nhạc



d. Usecase chi tiết quản lý thiết bị và OTA

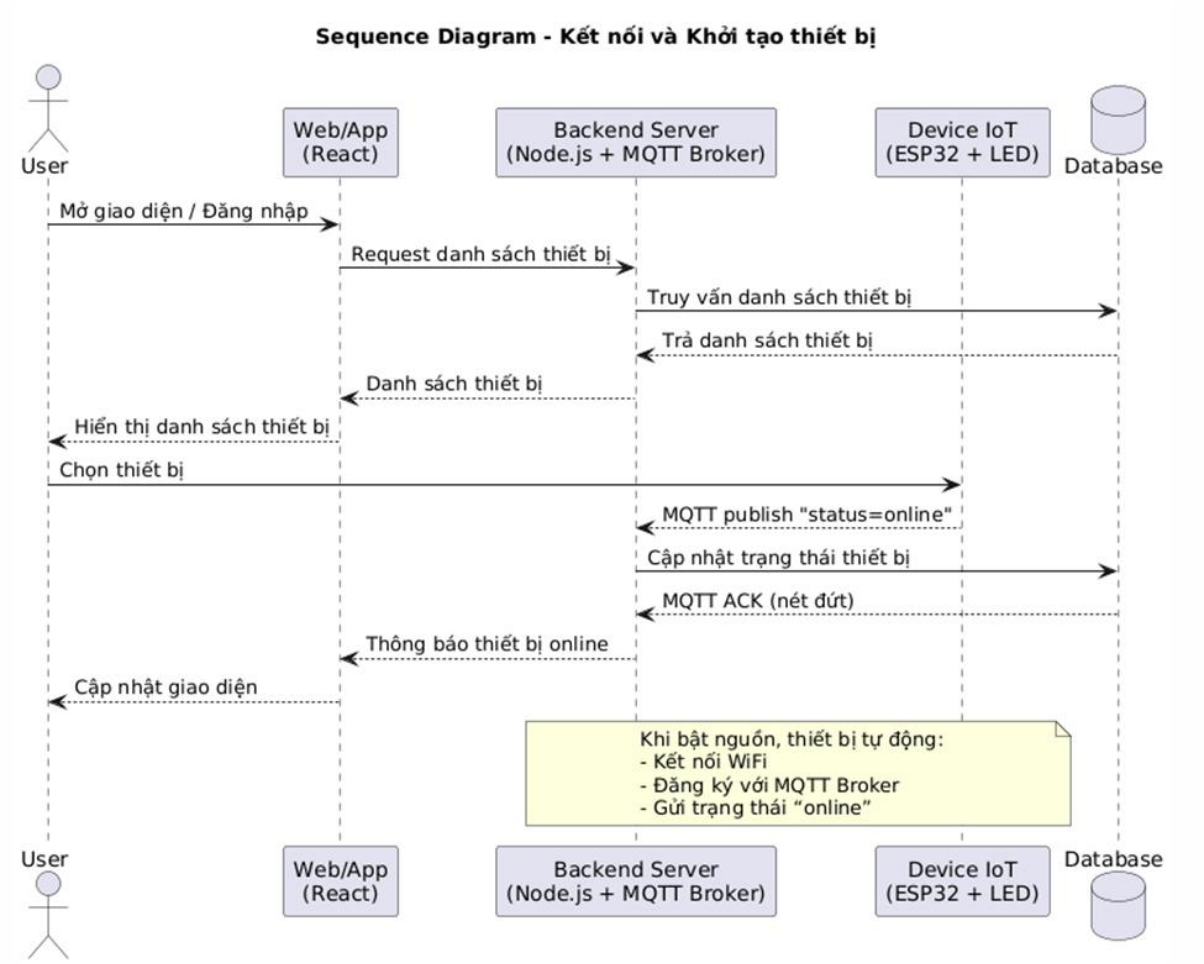


e. Usecase chi tiết quản lý người dùng, cấu hình và lịch sử

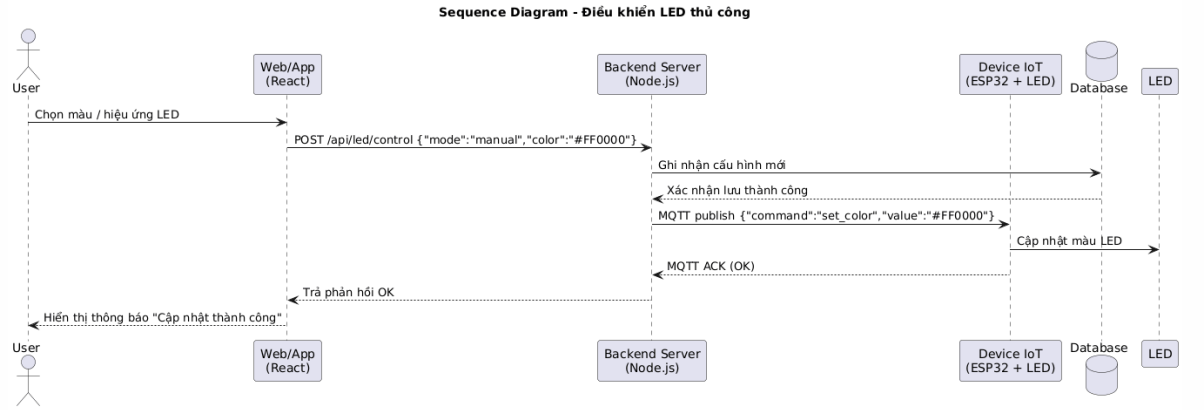


2. Sequence Diagram

2.1 Usecase Kết nối và khởi tạo thiết bị

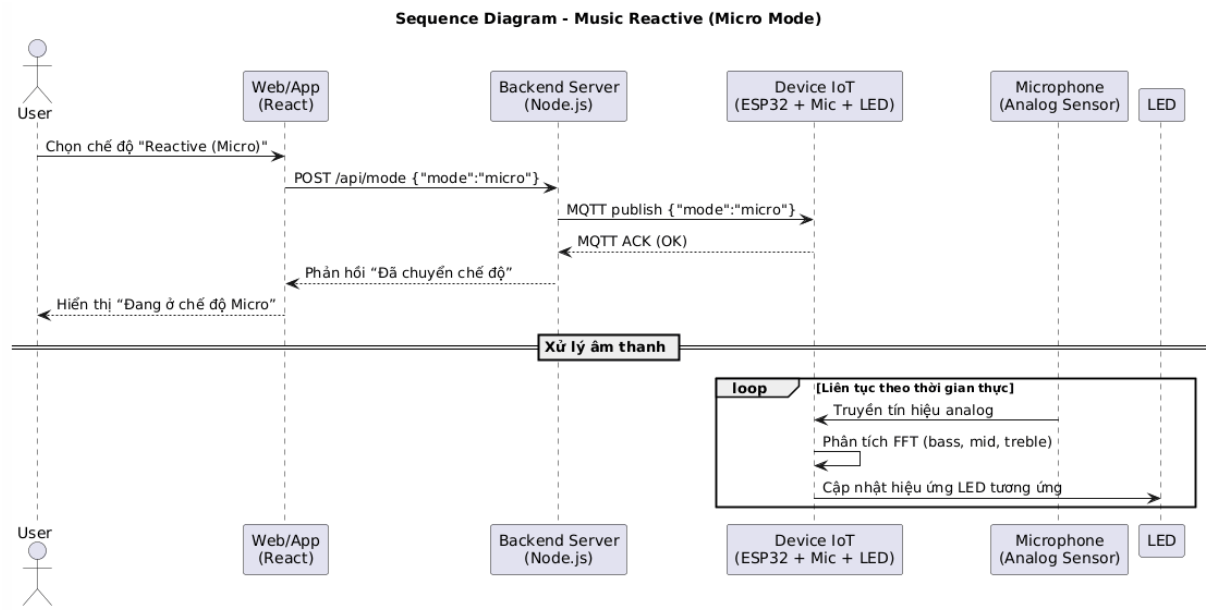


2.2 Usecase điều khiển led thủ công

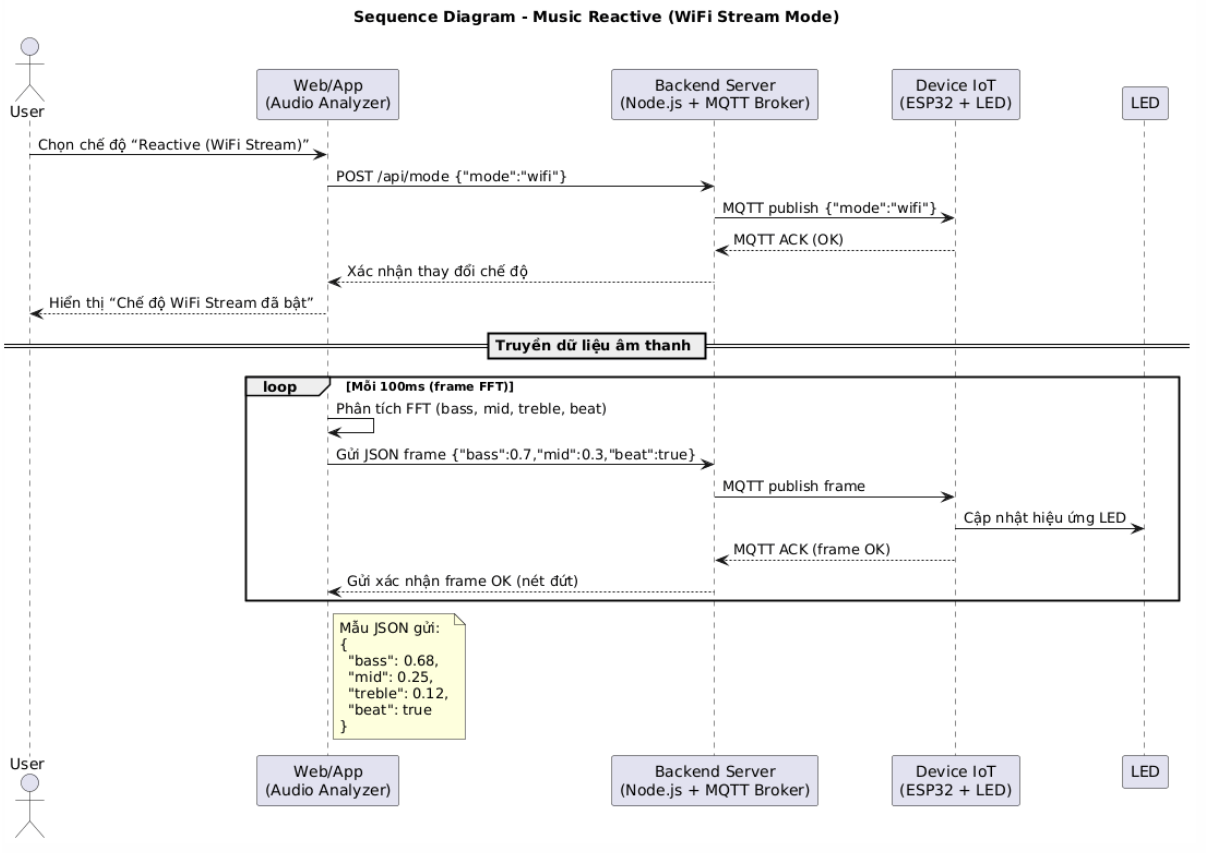


2.3 Usecase chi tiết phản ứng âm nhạc

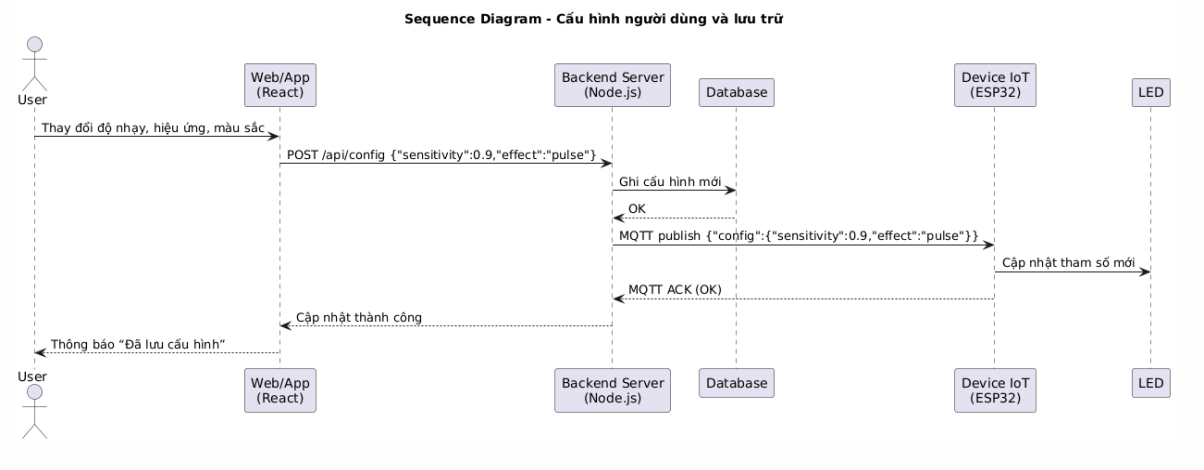
2.3.1 Phương án 1 : Micro Mode



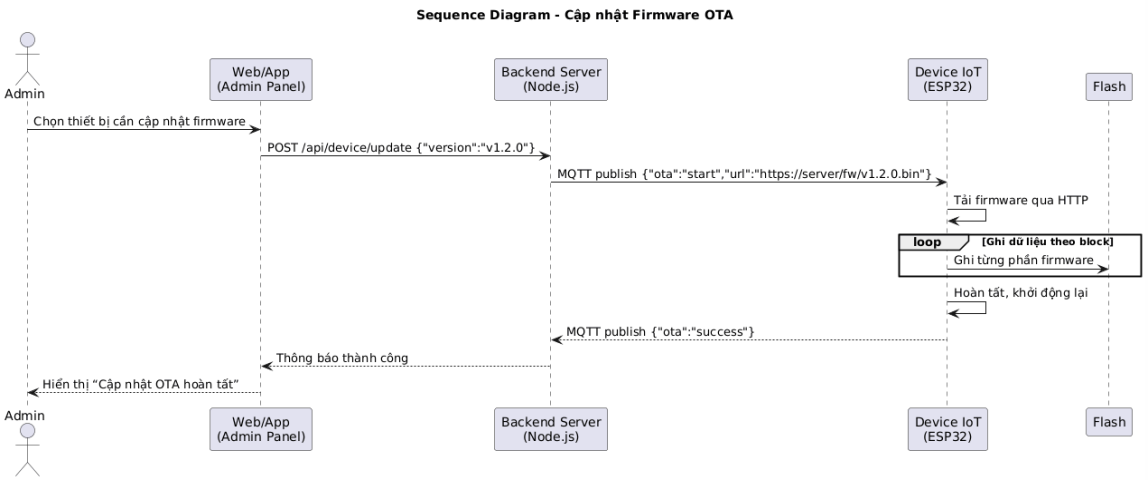
2.3.2 Phương án 2: Wifi Stream Mode



2.4 Usecase quản lý người dùng, cấu hình và lịch sử



2.5 Usecase quản lý thiết bị và OTA



3. Kiến trúc 3 lớp

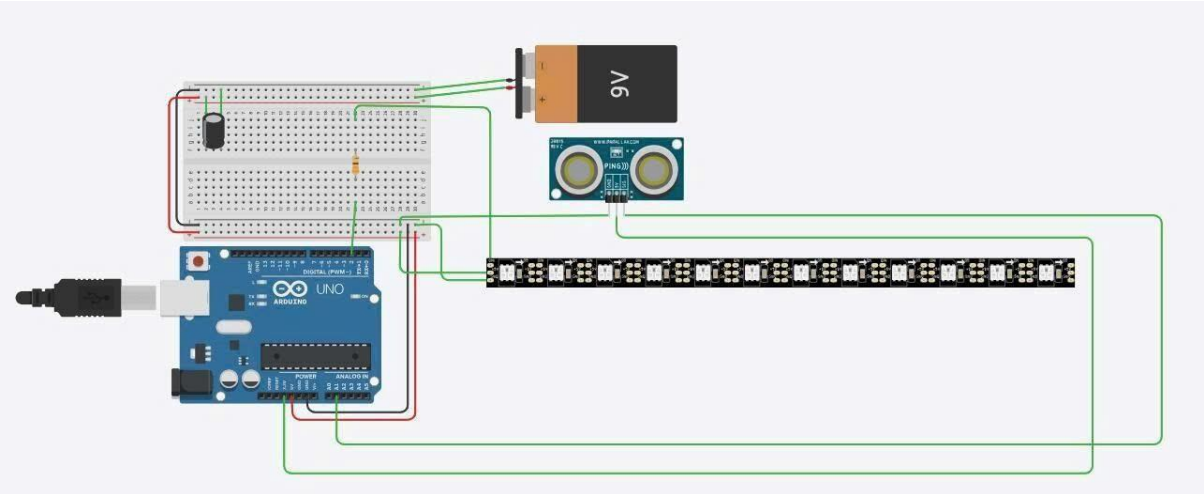
Ba lớp chính:

- **Presentation Layer:** Giao diện web/app (React).
- **Application Layer:** Server xử lý logic, API, kết nối MQTT hoặc WebSocket.
- **Hardware Layer:** ESP32 + LED + Microphone, nhận tín hiệu âm thanh và điều khiển đèn

IV. PHÂN TÍCH CÔNG NGHỆ & THIẾT BỊ

1. Phần cứng

Thành phần	Chức năng chính
1. Vi điều khiển ESP32	- Trái tim của hệ thống IoT. - Nhận tín hiệu từ microphone (analog). - Xử lý dữ liệu âm thanh. - Điều khiển dải LED qua giao tiếp dữ liệu số (PWM hoặc one-wire). - Kết nối Wi-Fi để nhận lệnh điều khiển từ web/app.
2. Microphone Module (MAX4466 / KY-037)	- Thu âm thanh môi trường. - Xuất tín hiệu analog tỷ lệ với cường độ âm thanh. - Được đọc bởi ADC của ESP32.
3. Dải LED RGB (WS2812B / Neopixel)	- Hiển thị màu sắc và hiệu ứng ánh sáng. - Mỗi LED có thể điều khiển riêng biệt (địa chỉ hóa). - Giao tiếp theo chuẩn **one-wire digital** với ESP32.
4. Nguồn điện 5V – 10A	- Cung cấp điện cho ESP32 và dải LED. - Dải LED công suất lớn → cần nguồn ổn định.
5. Module Wi-Fi (tích hợp trên ESP32)	- Dùng để giao tiếp với web/app qua mạng nội bộ hoặc Internet.
6. Breadboard, dây nối, điện trở, tụ điện phụ trợ	- Kết nối mạch thử nghiệm. - Giúp ổn định tín hiệu và đảm bảo an toàn mạch điện.

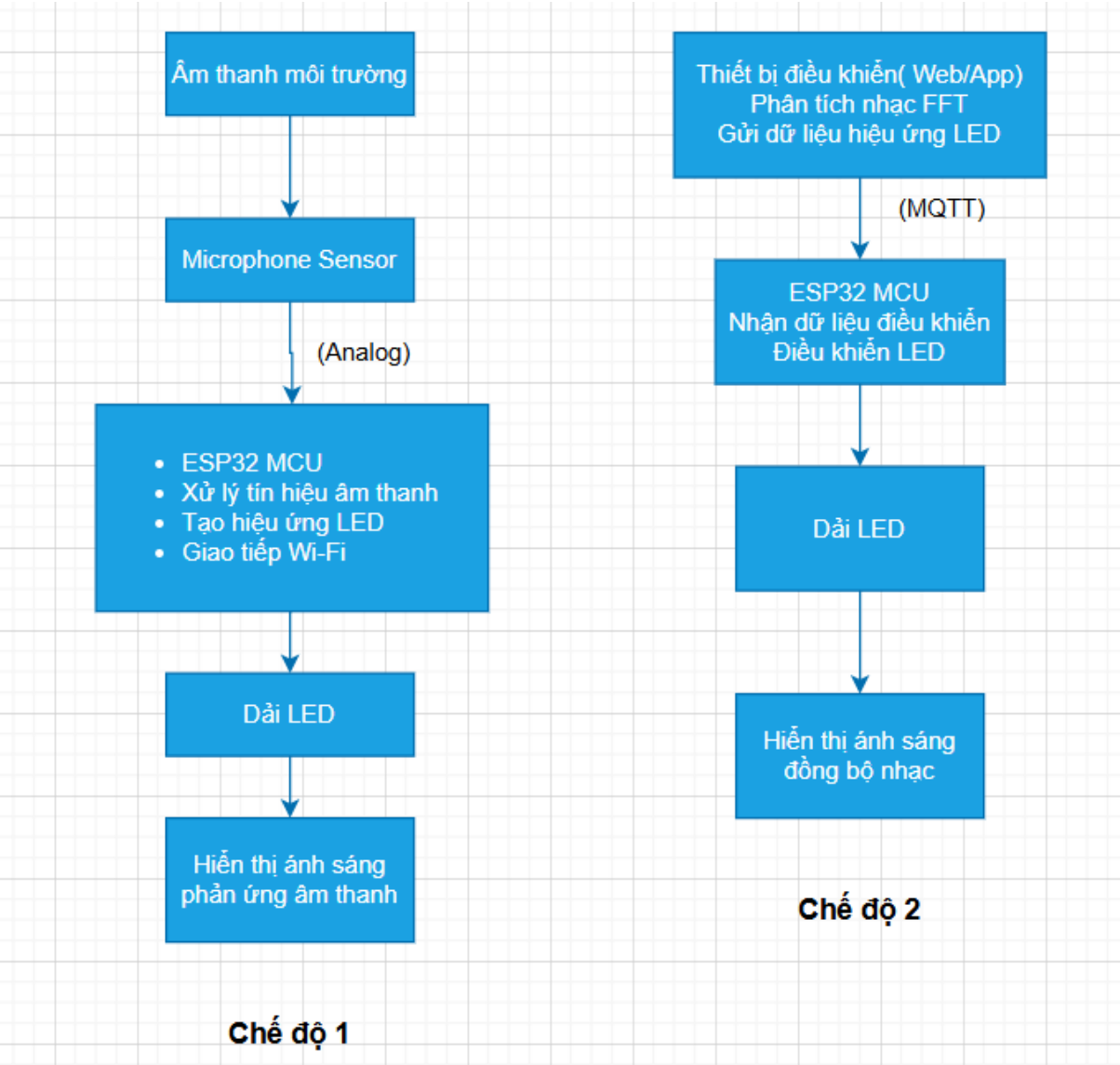


Hình 1 Sơ đồ kết nối

2. Phần mềm & công nghệ

Thành phần	Công nghệ sử dụng	Chức năng / Vai trò
Firmware (ESP32 Code)	Arduino IDE / PlatformIO – C/C++	- Đọc tín hiệu analog từ microphone. - Xử lý biên độ hoặc tần số âm thanh (FFT). - Gửi tín hiệu điều khiển dải LED. - Giao tiếp với web/app qua MQTT hoặc HTTP.
Giao diện người dùng (Web/App)	ReactJS / HTML5 / CSS / JavaScript	- Cung cấp giao diện điều khiển (bật/tắt, đổi màu, chọn hiệu ứng, đổi mode). - Hiển thị trạng thái hệ thống.
Máy chủ IoT (tuỳ chọn)	Node.js + Express / Firebase / MQTT Broker	- Nhận và phân phối dữ liệu điều khiển. - Quản lý kết nối nhiều thiết bị cùng lúc. - Cung cấp API cho app/web.
Giao thức truyền thông	MQTT / WebSocket / HTTP	- Truyền dữ liệu giữa ESP32 ↔ App/Web. - Hỗ trợ điều khiển thời gian thực.
Thư viện LED	FastLED / Adafruit_NeoPixel	- Tạo hiệu ứng LED đa dạng, đồng bộ với tín hiệu âm thanh.
Thư viện xử lý tín hiệu âm thanh	ArduinoFFT / AnalogRead()	- Phân tích tín hiệu âm thanh (biên độ, tần số).

3. Sơ đồ tổng quan hệ thống (System Architecture)



4. Mối quan hệ giữa phần cứng và phần mềm

Phần cứng	Phần mềm điều khiển tương ứng	Mô tả tương tác
ESP32	Arduino Firmware	Đọc tín hiệu, điều khiển LED, xử lý dữ liệu.
Microphone	ADC Module	Cung cấp tín hiệu âm thanh analog cho ESP32.

LED WS2812B	FastLED Library	Nhận tín hiệu điều khiển để đổi màu, hiển thị hiệu ứng.
Web/App	ReactJS + MQTT	Gửi lệnh điều khiển và nhận phản hồi từ ESP32.
Cloud Server	Node.js / Firebase	Lưu trữ cấu hình và dữ liệu người dùng (tùy chọn).

5. Xây dựng và thiết kế hệ thống

Dự án này sử dụng kiến trúc **Web (React) <->Socket.IO <-> Node.js Server <-> MQTT Broker (Mosquitto) <-> ESP32**.

5.1 Luồng Xử Lý Điều Khiển & Giám Sát (Manual Control & Monitoring)

5.1.1 Điều khiển Thủ công (Web -> Server -> ESP32)

Bước	Mô tả	Thành phần	Topic MQTT (Gửi)	Payload
B1: Gửi lệnh	Người dùng nhấn nút ON trên giao diện React (App.jsx).	React Frontend	control (Socket.IO)	{ topic: "led/control/power", payload: "on" }
B2: Chuyển tiếp	Node.js Server (index.js) nhận sự kiện control qua Socket.IO.	Node.js Server	led/control/power (MQTT)	"on"
B3: Nhận & Xử lý	ESP32 (Music_Reactive_LED.ino) đã subscribe topic led/control/#. Hàm callback xử lý tin nhắn.	ESP32	(Nhận)	"on"
B4: Thực thi	ESP32 bật biến power = true, gọi FastLED.show() để bật đèn.	ESP32	(Nội bộ)	

5.1.2 Giám sát Trạng thái (ESP32 -> Server -> Web)

Bước	Mô tả	Thành phần	Topic MQTT (Gửi)	Payload (Ví dụ)
B5: Gửi trạng thái	Sau khi xử lý xong lệnh (B3), ESP32 gửi trạng thái hoạt động về Broker bằng hàm publishStatus("OK").	ESP32	led/status	"OK"
B6: Nhận & Phát sóng	Node.js Server đã subscribe topic led/status. Nó nhận tin nhắn và phát sóng (io.emit) cho tất cả các client Web đang kết nối.	Node.js Server	mqtt (Socket.IO)	{ topic: "led/status", payload: "OK" }
B7: Cập	React Frontend (App.jsx) lắng nghe sự kiện mqtt. Nếu topic === "led/status", nó cập nhật	React Frontend	(Nhận)	"OK"

nhật UI	state status để hiển thị **Trạng thái ESP32**.			

5.2 Luồng Xử Lý Chế độ Phản Ứng Âm Nhạc

Dự án này có hai chế độ phản ứng âm nhạc: **Mic (Microphone)** và **WiFi Sync**.

5.2.1 Chế độ Micro (Mic Mode)

Bước	Mô tả	Thành phần
B1: Thu thập	ESP32 đọc dữ liệu từ Mic Analog (PIN 34) trong hàm micReact().	ESP32
B2: Xử lý	Dữ liệu sample (0-4095) được ánh xạ (map) thành level (0-255).	ESP32
B3: Thực thi hiệu ứng	Nếu effect là "pulse" , ESP32 sẽ điều chỉnh độ sáng (FastLED.setBrightness(level)) của toàn bộ dải LED dựa trên biên độ âm thanh.	ESP32
B4: Hiển thị	ESP32 gọi FastLED.show() để cập nhật dải LED.	ESP32

5.2.2 Chế độ WiFi Sync (Web -> Server -> ESP32)

Đây là luồng xử lý phức tạp, nơi giao diện Web **phân tích âm thanh** và **gửi dữ liệu tần số** qua MQTT để ESP32 đồng bộ.

Bước	Mô tả	Thành phần	Topic MQTT (Gửi)	Payload (Ví dụ)
B1: Tải & Phân tích	Người dùng tải file nhạc lên Web. Hàm handleMusicUpload dùng **Web Audio API** để phân tích real-time.	React Frontend	**(Nội bộ)**	
B2: Tính toán tần số	Hàm drawVisualizer tính toán giá trị **Bass** (0-10), **Mid** (10-60), **Treble** (60+) từ dataArray[11].	React Frontend	**(Nội bộ)**	
B3: Gửi dữ liệu	Giao diện Web gửi giá trị Bass/Mid/Treble đã	React Frontend	control (Socket.IO)	{ topic: "led/control/audio/bass", payload: "450" }

	tính được tới Node.js Server qua Socket.IO.			
B4: Chuyển tiếp	Node.js Server chuyển tiếp dữ liệu này qua MQTT Broker.	Node.js Server	led/audio/bass, led/audio/mid, led/audio/treble	"450", "300", "150"
B5: Cập nhật Level	ESP32 nhận tin nhắn. Hàm callback kiểm tra t.startsWith("led/audio/") , sau đó cập nhật các biến bassLevel, midLevel, trebleLevel.	ESP32	(Nhận)	
B6: Thực thi hiệu ứng	Trong loop(), nếu mode == "wifi", ESP32 sẽ chạy hiệu ứng ánh sáng dựa trên bassLevel, midLevel, trebleLevel (ví dụ: chia dải LED thành 3 phần, mỗi phần hiển thị cường độ của Bass/Mid/Treble).	ESP32	(Nội bộ)	

5.3 Thiết kế giao diện UI/UX

5.3.1. Tổng quan

Thiết kế giao diện người dùng (UI) tập trung vào việc tạo ra một bảng điều khiển hiện đại, trực quan và dễ điều khiển (minimalist control panel) cho thiết bị IoT (LED Strip). Trải nghiệm người dùng (UX) được tối ưu hóa cho việc truy cập nhanh các chức năng chính (bật/tắt, thay đổi màu, chế độ âm nhạc).

Giao diện hỗ trợ **Dark Mode** và **Light Mode**, giúp người dùng tùy chỉnh môi trường xem phù hợp.

5.3.2. Điều Khiển Cơ Bản (Cột Trái)

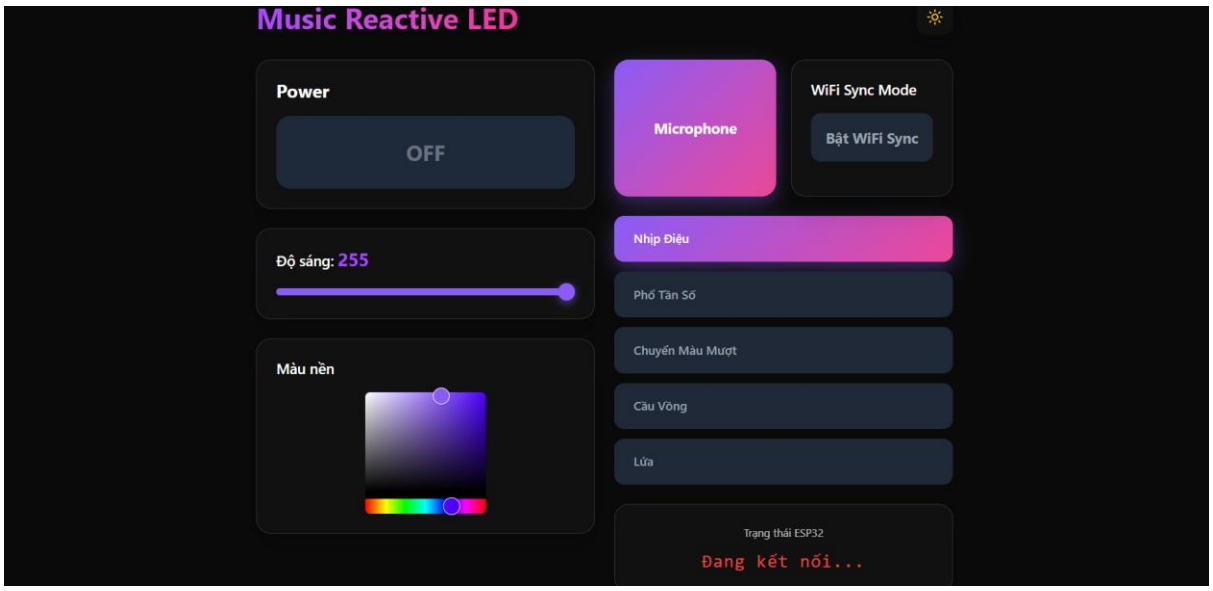
Các điều khiển cơ bản được đặt ở cột trái, dễ thấy:

- **Power (Nút Bật/Tắt):** Một nút lớn, chiếm toàn bộ chiều rộng, thay đổi màu sắc rõ rệt giữa **MÀU XANH LÁ CÂY** (ON) và **XÁM ĐEN** (OFF) để cung cấp phản hồi trực quan ngay lập tức.
- **Độ Sáng (Brightness):** Sử dụng **thanh trượt (range input)** từ 0 đến 255. Giá trị số hiện tại được hiển thị bằng chữ lớn, giúp người dùng biết chính xác mức độ sáng.
- **Chọn Màu (Color Picker):** Sử dụng thư viện HexColorPicker của react-colorful. Giao diện hiển thị trực quan cho phép chọn màu Hex và ngay lập tức gửi lệnh qua MQTT.

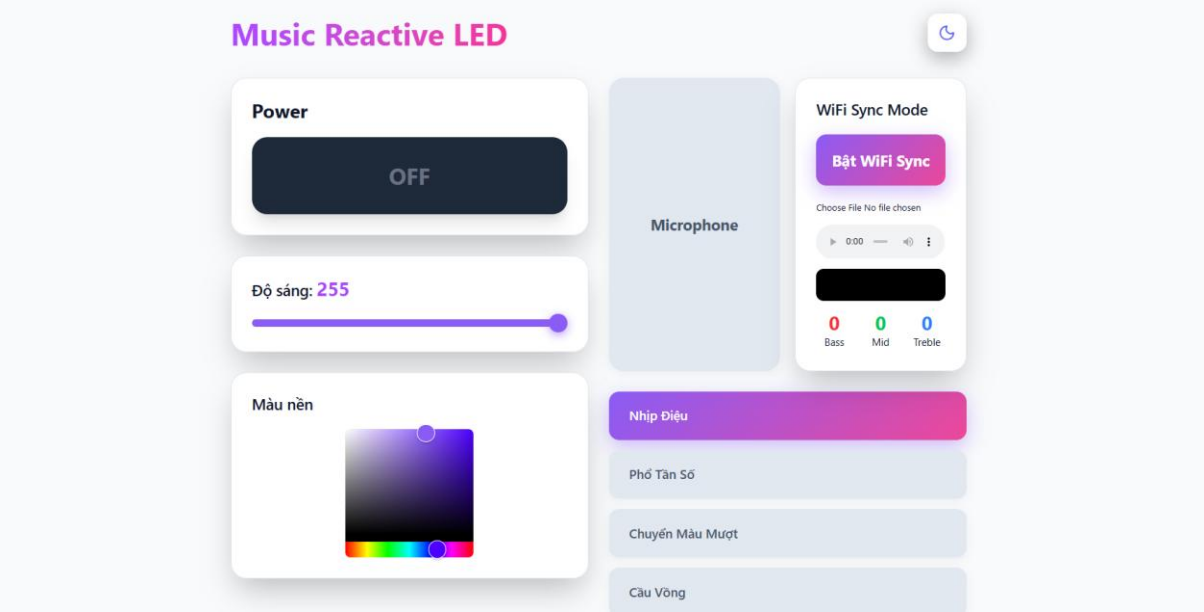
5.3.3. Điều Khiển Chế Độ & Hiệu Ứng (Cột Phải)

Cột phải tập trung vào các chế độ hoạt động và tùy chọn hiệu ứng:

- **Chọn Chế Độ (Mode):**
  - **Microphone:** Chế độ mặc định, sử dụng Mic vật lý của ESP32 (MIC_PIN 34).
  - **WiFi Sync Mode:** Chế độ phức tạp hơn, cho phép **tải file nhạc lên trực tiếp** trên Web.
    - Khi chế độ này được kích hoạt, một khu vực điều khiển âm nhạc sẽ xuất hiện, bao gồm: input type="file", thẻ <audio controls>, và một **Canvas Visualizer** để hiển thị phổ âm thanh real-time.
    - Giá trị **Bass, Mid, Treble** được tính toán và hiển thị bằng số lớn, màu sắc khác nhau (Đỏ, Xanh lá, Xanh dương).
- **Chọn Hiệu Ứng (Effect):** Danh sách các nút bấm rõ ràng cho các hiệu ứng ánh sáng như pulse, spectrum, fade, rainbow, fire.
- **Trạng thái ESP32:** Một bảng hiển thị trạng thái kết nối và hoạt động của ESP32. Sử dụng màu **XANH LÁ** cho trạng thái OK và **ĐỎ** cho lỗi/kết nối.



Hình 2: Giao diện chế độ Micro – dark mode



Hình 3: Giao diện chế độ wifi sync – light mode

5.4 Thiết kế Web server

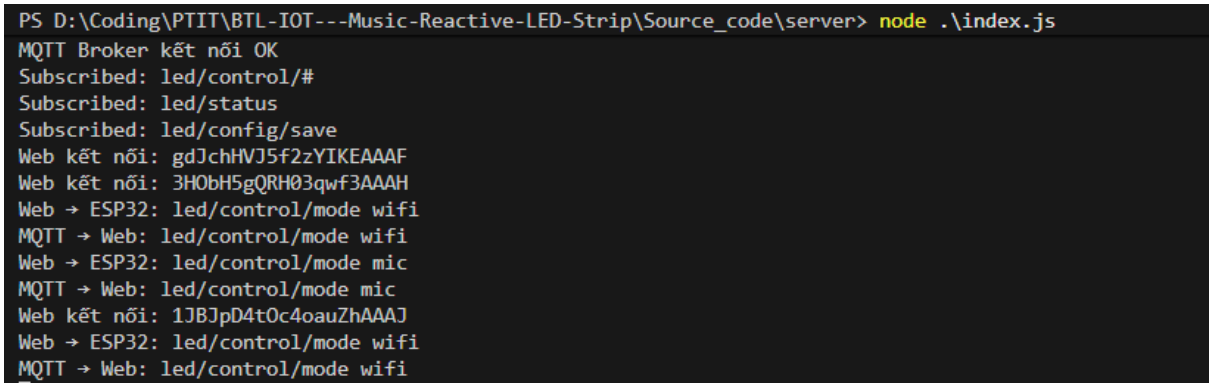
Node.js Server đóng vai trò là **cầu nối trung tâm (bridge)**, kết nối giao diện web (dùng Socket.IO) với thiết bị IoT (dùng MQTT).

Chức năng cốt lõi

- HTTP Server: Cung cấp điểm truy cập cơ bản (http://localhost:3000) và xác nhận server đang chạy.
- Socket.IO Server: Thiết lập giao tiếp real-time hai chiều với các trình duyệt web (Frontend React).
    - Phản hồi Web: Lắng nghe sự kiện control từ Web/App để nhận lệnh điều khiển (ví dụ: bật/tắt).
    - Đẩy dữ liệu Real-time: Dùng io.emit('mqtt', { topic, payload }) để gửi dữ liệu trạng thái nhận được từ MQTT (ví dụ: trạng thái led/status của ESP32) tới tất cả các client web đang mở.
- MQTT Client: Kết nối đến Mosquitto Broker và thực hiện các hành động:
    - Subscribe: Theo dõi các topic cần thiết như led/control/#, led/status, và led/config/save để nhận trạng thái từ ESP32.
    - Publish: Gửi lệnh điều khiển nhận được từ Web/App tới ESP32 thông qua topic MQTT.

Tóm tắt luồng dữ liệu chính

- Web -> ESP32 (Điều khiển): Socket.IO (Web) -> Server (lắng nghe socket.on('control')) -> MQTT Client (thực hiện mqttClient.publish).
- ESP32 -> Web (Giám sát): MQTT Client (lắng nghe mqttClient.on('message')) -> Server -> Socket.IO (thực hiện io.emit('mqtt')) -> Web.



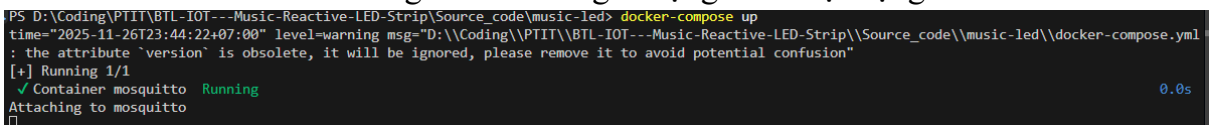
Hình 4: Chạy server node js tại máy local

5.5 Mosquitto broker

Mosquitto (chạy bằng Docker image eclipse-mosquitto:latest) đóng vai trò là **cơ chế nhắn tin (Messaging Queue)**, cho phép các thiết bị và ứng dụng giao tiếp một cách không đồng bộ (asynchronously).

Vai trò trong dự án

- **Trung gian (Hub):** Nhận tin nhắn từ bên gửi (Publisher - có thể là Node.js Server hoặc ESP32) và chuyển tiếp tới tất cả các bên quan tâm (Subscriber - có thể là Node.js Server hoặc ESP32).
- **Giao tiếp giữa Server và ESP32:**
  - **Lệnh (Server -> ESP32):** Node.js Server Publish lệnh lên topic led/control/power, và ESP32 Subscribe topic này để nhận.
  - **Trạng thái (ESP32 -> Server):** ESP32 Publish trạng thái lên topic led/status, và Node.js Server Subscribe topic này để cập nhật lên Web.
- **Topic Hierarchy (Cấu trúc Topic):** Các topic được tổ chức rõ ràng theo phân cấp:
  - led/control/#: Dùng cho các lệnh điều khiển từ Web (power, brightness, color, effect, mode, audio/...).
  - led/status: Dùng cho ESP32 gửi trạng thái hoạt động về Server.



Hình 5: Chạy broker mosquitto trên local bằng docker

V. PHÂN CÔNG CÔNG VIỆC & KẾ HOẠCH

1. **Bùi Quang Hiếu – Trưởng nhóm / Quản lý dự án**

- Quản lý tiến độ, phân công công việc.
- Viết phần Giới thiệu đề tài và Công nghệ áp dụng trong báo cáo.
- Chuẩn bị slide thuyết trình.
- Quản lý repo GitHub, tổ chức thư mục.
- Đảm bảo báo cáo và sản phẩm đúng hạn.

2. **Nguyễn Việt Anh – Phát triển phần cứng & kết nối**

- Biểu diễn chức năng
- Lắp mạch ESP32 + micro + LED strip.
- Viết firmware điều khiển LED dựa trên tín hiệu âm thanh.
- Hiệu chỉnh tín hiệu âm thanh, kiểm thử thực tế.
- Push code phần cứng vào repo (Source Code/device).

3. **Dương Văn Thuận – Phát triển phần mềm & tích hợp**

- Phân tích yêu cầu, biểu diễn chức năng, phân tích công nghệ và thiết bị
- Xây dựng ứng dụng/web nhỏ để chọn chế độ LED.
- Viết code kết nối ESP32 với Wi-Fi.
- Thiết kế giao diện dashboard đơn giản.
- Push code phần mềm vào repo (Source Code/server).

VI. KẾT LUẬN DỰ KIẾN

Dự án "Music Reactive LED" giúp ứng dụng kiến thức IoT, lập trình nhúng, xử lý tín hiệu và giao diện người dùng.

Sản phẩm hướng đến trải nghiệm thực tế và có thể mở rộng cho hệ thống **nhà thông minh hoặc giải trí âm thanh ánh sáng.**

Thời gian phản hồi ≤ 1 giây
Xử lý tín hiệu âm thanh real-time
Tự động kết nối lại khi mất mạng
Dữ liệu không bị mất khi lỗi tạm thời
Xác thực khi truy cập Web
OTA an toàn
Hỗ trợ nhiều thiết bị cùng lúc
Giao diện thân thiện, dễ dùng
Hệ thống dễ cập nhật
Hỗ trợ nhiều trình duyệt