HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I

_____o0o_____

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: IOT VÀ ỨNG DỤNG

Nhóm lớp: 06

Giảng viên hướng dẫn: TS. Kim Ngọc Bách

Sinh viên thực hiện:

Họ và tên	**Mã sinh viên**
Vũ Ngọc Hùng	B22DCCN373
Nguyễn Đức Huy	B22DCCN385
Đặng Hữu Nghĩa	B22DCCN601
Kim Duy Hưng	B22DCCN409

Hà Nội, Tháng 10/2025

DANH MỤC HÌNH ẢNH

MỤC LỤC

I. Giới thiệu đề tài

1. Mô tả hệ thống

Trong bối cảnh hiện nay, với sự phát triển vượt bậc của công nghệ trí tuệ nhân tạo (AI) và Internet of Things (IoT), nhu cầu sử dụng các hệ thống giao tiếp thông minh ngày càng trở nên quan trọng hơn bao giờ hết. Một trong những xu hướng nổi bật trong thế giới công nghệ hiện nay là việc tích hợp AI vào các thiết bị và ứng dụng trong đời sống hàng ngày. Đặc biệt, việc sử dụng giọng nói làm phương tiện giao tiếp không còn là điều mới mẻ mà đang trở thành một phần không thể thiếu trong các hệ thống điều khiển, hỗ trợ người dùng trong các tác vụ từ cơ bản đến phức tạp. Dự án này đề xuất xây dựng Thiết bị AIOT điều khiển bằng giọng nói giúp người dùng tương tác với hệ thống thông qua câu lệnh nói và nhận phản hồi bằng giọng nói.

Mục đích của dự án:

- Tạo ra một hệ thống giao tiếp thông minh qua giọng nói, giúp người dùng dễ dàng điều khiển và tương tác với thiết bị IoT.
- Cải thiện trải nghiệm người dùng với hệ thống điều khiển bằng giọng nói, giúp tiết kiệm thời gian và dễ sử dụng.
- Xây dựng nền tảng cho các sản phẩm IoT thông minh, có thể mở rộng và tích hợp thêm nhiều thiết bị và cảm biến trong tương lai.
- Nâng cao tính tiện lợi và ứng dụng của công nghệ AI vào các thiết bị đời sống hàng ngày.
- Học hỏi và ứng dụng các công nghệ hiện đại như AI, IoT, API và điều khiển thiết bị qua internet.

Phương hướng phát triển của dự án:

- Tích hợp trí tuệ nhân tạo (AI): Dùng AI để cải thiện khả năng nhận diện và hiểu lệnh người dùng, giúp hệ thống hiểu được ngữ cảnh và trả lời một cách chính xác.
- Tăng cường tính năng IoT: Tích hợp tính năng mở rộng hệ thống để hỗ trợ nhiều loại thiết bị Slave, cho phép người dùng điều khiển và thu thập dữ liệu từ nhiều loại cảm biến và thiết bị khác nhau.
- Tích hợp vào hệ sinh thái nhà thông minh: Hệ thống có thể được kết hợp với các nền tảng nhà thông minh, giúp tự động hóa các tác vụ trong gia đình như bật đèn, điều hòa, kiểm soát các thông số trong gia đình như nhiệt độ, độ ẩm phòng…

2. Phạm vi hệ thống

- ***Phạm vi chức năng***:
    - Nhận diện giọng nói: Hệ thống sẽ chỉ hỗ trợ nhận diện các lệnh giọng nói từ người dùng trong một môi trường tương đối yên tĩnh, nơi không có nhiều tạp âm. Các lệnh này sẽ được chuyển đổi thành văn bản và gửi đến server để xử lý.
    - Phản hồi giọng nói: Sau khi server nhận được yêu cầu, hệ thống sẽ gửi phản hồi dưới dạng giọng nói thông qua API Text-to-Speech, giúp người dùng nhận được kết quả trả lời từ hệ thống một cách tự nhiên.
    - Điều khiển thiết bị IoT: Hệ thống sẽ có khả năng điều khiển các thiết bị thông qua các lệnh giọng nói, như bật/tắt đèn, quạt, điều hòa, v.v. Các thiết bị này phải được đăng kí và cấu hình từ trước trên nền tảng hệ thống.
    - Thu thập dữ liệu cảm biến: Các thiết bị Slave sẽ có khả năng thu thập dữ liệu từ các cảm biến (như nhiệt độ, độ ẩm) và gửi dữ liệu này lên server để quản lý, theo dõi, hoặc hiển thị trên giao diện người dùng.
    - Quản lý tài khoản và phân quyền: Hệ thống sẽ hỗ trợ việc đăng ký, đăng nhập, và phân quyền người dùng với các loại tài khoản khác nhau.
- **Giới hạn về khả năng mở rộng:**
    - Số lượng thiết bị Slave: Mặc dù hệ thống có thể mở rộng để hỗ trợ nhiều thiết bị Slave, nhưng hiện tại phạm vi của hệ thống sẽ chỉ tập trung vào một số lượng thiết bị nhất định. Việc mở rộng số lượng thiết bị Slave sẽ yêu cầu tối ưu hóa và điều chỉnh hệ thống để đảm bảo hiệu suất và độ ổn định.
    - Giới hạn về cảm biến và thiết bị IoT: Các thiết bị và cảm biến được kết nối với hệ thống sẽ có giới hạn về số lượng và loại thiết bị mà hệ thống có thể hỗ trợ. Ví dụ, chỉ các cảm biến độ ẩm, nhiệt độ, và các thiết bị điều khiển cơ bản sẽ được hỗ trợ.
    - Khả năng xử lý giọng nói: Hệ thống hiện tại sẽ chỉ hỗ trợ các lệnh giọng nói cơ bản và không hỗ trợ các lệnh giọng nói phức tạp hoặc đối thoại dài, xử lí đa ngôn ngữ. Các yêu cầu giọng nói sẽ phải được chuẩn hóa và đơn giản để đảm bảo nhận diện chính xác.
- **Giới hạn về bảo mật và quyền riêng tư:**
    - Quản lý quyền truy cập cơ bản: Hệ thống sẽ cung cấp các mức quyền hạn cơ bản cho người dùng, bao gồm quyền truy cập vào các thiết bị Master, Slave và quản lý tài khoản. Tuy nhiên, hệ thống sẽ không triển

khai các biện pháp bảo mật phức tạp như mã hóa dữ liệu người dùng ở cấp độ cao hoặc bảo vệ dữ liệu toàn diện ngay từ đầu.

- Quản lý thông tin người dùng: Hệ thống sẽ lưu trữ thông tin tài khoản và trạng thái thiết bị.

3. Công nghệ sử dụng

3.1. Phần cứng

3.1.1. ESP32

Giới thiệu: ESP32 là một dòng vi điều khiển hệ thống trên chip (SoC) chi phí thấp, tiêu thụ điện năng thấp, được tích hợp sẵn Wi-Fi và Bluetooth. Với bộ xử lý lõi kép, hiệu năng mạnh mẽ và số lượng chân GPIO phong phú, ESP32 là một lựa chọn phổ biến cho nhiều ứng dụng IoT, từ các nút cảm biến đơn giản đến các gateway phức tạp.

Chi tiết ứng dụng:

- **Thiết bị Master:** ESP32 được chọn làm vi điều khiển trung tâm cho thiết bị Master do yêu cầu xử lý cao hơn.
- **Kết nối Wi-Fi:** Để giao tiếp với server qua MQTT và WebSocket.
- **Xử lý âm thanh:** ESP32 đủ mạnh để điều khiển codec âm thanh, thu dữ liệu từ microphone (qua giao thức I2S) và gửi luồng âm thanh đến server. Nó cũng có khả năng nhận dữ liệu âm thanh từ server và phát ra loa.
- **Xử lý đa nhiệm:** Bộ xử lý lõi kép cho phép ESP32 xử lý đồng thời nhiều tác vụ: một lõi có thể chuyên trách việc thu và gửi âm thanh, trong khi lõi còn lại xử lý kết nối mạng và các logic điều khiển khác.

3.1.2. ESP8266

Giới thiệu: ESP8266 là tiền thân của ESP32, một vi điều khiển Wi-Fi SoC chi phí thấp, rất phổ biến trong cộng đồng DIY và các ứng dụng IoT cơ bản. Mặc dù không mạnh mẽ bằng ESP32, ESP8266 vẫn là một lựa chọn tuyệt vời cho các tác vụ không đòi hỏi xử lý phức tạp.

Chi tiết ứng dụng:

- **Vi điều khiển thay thế cho các Slave:** Đối với các thiết bị Slave chỉ có chức năng đơn giản như đọc dữ liệu từ một hoặc hai cảm biến và điều khiển một rơ-le, việc sử dụng ESP8266 là một giải pháp tối ưu về chi phí.

3.1.3. Các cảm biến

Giới thiệu: Cảm biến là các thiết bị điện tử có chức năng đo lường các đại lượng vật lý từ môi trường và chuyển đổi chúng thành tín hiệu điện mà vi điều khiển có thể đọc và xử lý.

Chi tiết ứng dụng: Các thiết bị Slave được trang bị các loại cảm biến phù hợp với chức năng của chúng. Trong giai đoạn đầu, các cảm biến chính được sử dụng bao gồm:

- **Cảm biến nhiệt độ và độ ẩm (DHT11):** Các cảm biến này được kết nối với các vi điều khiển Slave đặt tại các phòng khác nhau. Slave sẽ định kỳ đọc giá trị nhiệt độ và độ ẩm, sau đó gửi dữ liệu này lên server thông qua MQTT. Dữ liệu này có thể được khai thác tùy nhu cầu người dùng.
- **Đèn :** Để đơn giản hóa và đảm bảo an toàn, sử dụng đèn LED công suất thấp thay cho đèn chiếu sáng 220V thực tế, dùng để mô phỏng trực quan trạng thái bật/tắt của một thiết bị.

3.2. Backend

3.2.1. Database : SupaBase

Chi tiết ứng dụng: Cơ sở dữ liệu là nơi lưu trữ toàn bộ thông tin của hệ thống:

- **Thông tin tài khoản người dùng:** Lưu trữ tên đăng nhập, mật khẩu đã được mã hóa, thông tin cá nhân, và vai trò của người dùng (Admin, user thường).
- **Thông tin thiết bị:** Lưu trữ danh sách tất cả các thiết bị (Master, Slaves) trong hệ thống, bao gồm ID duy nhất, tên do người dùng đặt, loại thiết bị, phòng được gán, và trạng thái hiện tại.
- **Dữ liệu lịch sử từ cảm biến:** Lưu trữ các giá trị nhiệt độ, độ ẩm theo thời gian, kèm theo ID của thiết bị và dấu thời gian (timestamp).

3.2.2. RESTful API

Giới thiệu: REST (Representational State Transfer) là một kiểu kiến trúc để thiết kế các ứng dụng mạng. RESTful API là một giao diện lập trình ứng dụng tuân thủ các ràng buộc của kiến trúc REST, thường sử dụng các phương thức HTTP (GET, POST, PUT, DELETE) để thực hiện các hoạt động tạo, đọc, cập nhật và xóa (CRUD) tài nguyên.

Chi tiết ứng dụng: RESTful API đóng vai trò là cổng giao tiếp chính giữa frontend (giao diện web) và hệ thống backend.

- Cung cấp dữ liệu cho frontend.
- Nhận yêu cầu điều khiển từ frontend.

3.2.3. Hệ thống quản lý tài khoản

Xác thực:

- Triển khai các chức năng đăng ký, đăng nhập cho người dùng. Mật khẩu sẽ được băm (hashing) trước khi lưu vào cơ sở dữ liệu để đảm bảo an toàn.

- Sử dụng các cơ chế JSON Web Tokens (JWT). Sau khi đăng nhập thành công, người dùng sẽ được cấp một token. Token này phải được đính kèm trong mỗi yêu cầu API tiếp theo để xác thực danh tính.

Phân quyền:

- **Admin:** Có toàn quyền quản lý hệ thống.
- **User:** Chỉ có quyền quản lý và xem các thiết bị mà họ đã đăng ký hoặc được chia sẻ quyền.

3.2.4. Firmware Over-the-Air - FOTA

Giới thiệu: Firmware Over-the-Air (FOTA) là một cơ chế cho phép cập nhật phần mềm (firmware) trên các thiết bị nhúng từ xa thông qua kết nối không dây (Wi-Fi), mà không cần kết nối vật lý với máy tính. Tính năng này cực kỳ quan trọng đối với các hệ thống IoT đã được triển khai, giúp việc sửa lỗi, vá lỗ hổng bảo mật và bổ sung tính năng mới trở nên dễ dàng, nhanh chóng và tiết kiệm chi phí.

Chi tiết ứng dụng: Hệ thống FOTA được tích hợp vào cả backend và firmware của thiết bị để tạo thành một quy trình cập nhật hoàn chỉnh.

1. **Quản lý phiên bản tại Backend:** Quản trị viên (Admin) sẽ sử dụng giao diện web để tải lên tệp firmware mới cho một loại thiết bị. Backend sẽ lưu trữ tệp này cùng với thông tin phiên bản và bản mô tả các thay đổi (changelog).
2. **Kiểm tra cập nhật:** Các thiết bị ESP32/ESP8266 sẽ được lập trình để định kỳ gửi một yêu cầu đến server để kiểm tra xem có phiên bản firmware mới nào không.
3. **Thông báo và Tải xuống:** Khi server nhận được yêu cầu và xác định có phiên bản mới hơn, nó sẽ phản hồi lại với một URL để tải xuống tệp firmware mới.
4. **Quá trình cập nhật trên thiết bị:** Firmware mới sẽ được tải xuống và ghi vào một phân vùng không hoạt động (inactive partition) trong khi thiết bị vẫn đang chạy firmware cũ từ phân vùng hoạt động (active partition). Điều này đảm bảo rằng nếu quá trình tải xuống bị lỗi, thiết bị vẫn có thể hoạt động bình thường.
5. **Xác minh và Khởi động lại:** Sau khi tải xuống hoàn tất, thiết bị sẽ kiểm tra tính toàn vẹn của tệp firmware mới. Nếu hợp lệ, nó sẽ cấu hình bộ nạp khởi động (bootloader) để khởi động từ phân vùng chứa firmware mới trong lần khởi động tiếp theo và sau đó tự khởi động lại.
6. **Xác nhận cập nhật:** Sau khi khởi động thành công với firmware mới, thiết bị sẽ gửi một thông báo đến server để xác nhận rằng quá trình cập nhật đã thành công.

3.3. Frontend

Giao diện web cung cấp cho người dùng một trung tâm điều khiển trực quan để quản lý và tương tác với toàn bộ hệ thống IoT của họ. Các chức năng chính bao gồm:
đăng nhập

- **Đăng ký và quản lý thiết bị:** Cung cấp một quy trình đơn giản để người dùng có thể thêm thiết bị mới vào tài khoản của mình. Người dùng có thể đặt tên cho thiết bị, gán nó vào một phòng cụ thể, và cấu hình các thông số liên quan.
- **Bảng điều khiển (Dashboard):** Hiển thị tổng quan trạng thái của tất cả các thiết bị.
- **Xem dữ liệu lịch sử:** Lấy dữ liệu từ RESTful API và hiển thị dưới dạng biểu đồ, cho phép người dùng theo dõi sự thay đổi của nhiệt độ, độ ẩm theo thời gian (giờ, ngày, tuần).
- **Quản lý tài khoản:** Cho phép người dùng thay đổi thông tin cá nhân, mật khẩu.
- **Giao diện quản trị (Admin Panel):** Một khu vực riêng dành cho quản trị viên để quản lý người dùng và giám sát toàn bộ hệ thống.

3.4. Giao thức truyền thông

3.4.1. MQTT

Giới thiệu: MQTT là một giao thức truyền tin nhắn hoạt động theo mô hình publish/subscribe (xuất bản/đăng ký), được thiết kế đặc biệt cho các thiết bị có tài nguyên hạn chế và hoạt động trên các mạng không ổn định, băng thông thấp. Đặc tính nhẹ, hiệu quả và đáng tin cậy của MQTT làm cho nó trở thành một tiêu chuẩn phổ biến trong các ứng dụng Internet of Things (IoT). Giao thức này hoạt động dựa trên một máy chủ trung tâm gọi là MQTT Broker, có nhiệm vụ nhận tin nhắn từ các client xuất bản (publisher) và chuyển tiếp chúng đến các client đã đăng ký (subscriber) vào cùng một chủ đề (topic) cụ thể.
Chi tiết ứng dụng: Trong kiến trúc của hệ thống, MQTT được sử dụng làm giao thức chính để đảm bảo luồng giao tiếp hiệu quả và ổn định giữa các thiết bị IoT (Master và Slaves) và Server.

- **Giao tiếp từ Slave đến Server:** Mỗi thiết bị Slave, sau khi thu thập dữ liệu từ cảm biến (ví dụ: nhiệt độ, độ ẩm), sẽ đóng gói dữ liệu này vào một tin nhắn MQTT. Tin nhắn này được xuất bản (publish) lên một topic cụ thể, được định danh duy nhất cho thiết bị đó. MQTT Broker trên server sẽ nhận tin nhắn này và chuyển tiếp đến các dịch vụ backend đã đăng ký (subscribe) vào topic tương ứng để xử lý và lưu trữ dữ liệu vào cơ sở dữ liệu.

- **Giao tiếp từ Master đến Server:** Thiết bị Master, sau khi nhận lệnh từ người dùng qua giọng nói và xử lý thành văn bản, sẽ gửi các yêu cầu hoặc dữ liệu này đến server thông qua MQTT.
- **Giao tiếp từ Server đến các thiết bị (Master/Slave):** Khi có một yêu cầu điều khiển từ giao diện web hoặc do một logic tự động trên server, server sẽ publish một tin nhắn lệnh đến topic của thiết bị đích

3.4.2. WebSocket

Giới thiệu: WebSocket là một giao thức truyền thông máy tính, cung cấp các kênh giao tiếp song công (full-duplex) qua một kết nối TCP duy nhất. Không giống như mô hình yêu cầu-phản hồi của HTTP, WebSocket cho phép giao tiếp hai chiều và real-time, nơi cả client và server đều có thể gửi dữ liệu cho nhau bất cứ lúc nào mà không cần yêu cầu trước. Điều này làm giảm đáng kể độ trễ và lượng dữ liệu thừa so với các kỹ thuật polling HTTP.

Chi tiết ứng dụng: WebSocket được ứng dụng trong hai kịch bản chính của dự án, nơi yêu cầu độ trễ thấp và truyền dữ liệu liên tục.

- **Giao diện web real-time:** Để hiển thị trạng thái và dữ liệu cảm biến từ các thiết bị Slave trên giao diện web một cách tức thời, WebSocket được sử dụng. Khi có dữ liệu mới từ cảm biến được gửi đến server qua MQTT, backend sẽ ngay lập tức đẩy (push) dữ liệu này đến giao diện web của người dùng thông qua kết nối WebSocket đã được thiết lập. Người dùng sẽ thấy sự thay đổi nhiệt độ, độ ẩm hoặc trạng thái thiết bị ngay lập tức mà không cần phải làm mới trang.
- **Streaming Audio từ Master đến Server:** Để xử lý giọng nói, luồng âm thanh (audio stream) thô thu được từ microphone trên thiết bị Master cần được gửi đến server để xử lý Speech-to-Text. Một kết nối WebSocket sẽ được thiết lập giữa Master và server. Master sẽ liên tục gửi các gói dữ liệu âm thanh qua kết nối này. Phía server sẽ nhận các gói này theo thời gian thực, ghép chúng lại và đưa vào mô hình STT để chuyển đổi. Việc sử dụng WebSocket đảm bảo luồng dữ liệu âm thanh được truyền tải liên tục, có độ trễ thấp, là yếu tố quan trọng để đảm bảo trải nghiệm tương tác giọng nói mượt mà và tự nhiên.

3.5. AI và xử lý ngôn ngữ

3.5.1. Function Calling

Giới thiệu: Function Calling là một kỹ thuật tiên tiến trong các mô hình ngôn ngữ lớn (LLM), cho phép mô hình AI không chỉ tạo ra văn bản thuần túy mà còn có thể tạo ra một đối tượng JSON chứa thông tin chi tiết để gọi một hàm cụ thể trong mã nguồn.

Khi nhận một yêu cầu từ người dùng, mô hình có thể phân tích, xác định ý định và trích xuất các tham số cần thiết, sau đó định dạng chúng theo một cấu trúc được xác định trước để kích hoạt một hành động trong thế giới thực.

Chi tiết ứng dụng: Trong hệ thống này, Function Calling là cầu nối thông minh giữa ngôn ngữ tự nhiên của người dùng và các hành động điều khiển thiết bị vật lý. Khi người dùng nói một câu lệnh, quy trình sẽ diễn ra như sau:

1. Giọng nói được chuyển thành văn bản.
2. Văn bản này được gửi đến Chat API (LLM).
3. Mô hình sẽ phân tích văn bản đầu vào và xác định rằng người dùng muốn gọi hàm nào. Nó sẽ trích xuất các tham số và sẽ trả về một đối tượng JSON
4. Backend của hệ thống nhận đối tượng JSON này, phân tích, sau đó thực thi logic nghiệp vụ tương ứng.

3.5.2. Speech-to-Text (STT)

Giới thiệu: Speech-to-Text (STT), hay còn gọi là nhận dạng giọng nói tự động (Automatic Speech Recognition - ASR), là công nghệ cho phép chuyển đổi tín hiệu âm thanh giọng nói của con người thành dạng văn bản có thể đọc được bằng máy. Các hệ thống STT hiện đại sử dụng các mô hình học sâu để đạt được độ chính xác cao trong nhiều ngôn ngữ và môi trường khác nhau.

Chi tiết ứng dụng: Thiết bị Master thu âm thanh từ môi trường. Khi người dùng bắt đầu nói một câu lệnh, Master sẽ:

1. Kích hoạt microphone và bắt đầu ghi lại luồng âm thanh.
2. Luồng âm thanh này được truyền tải liên tục đến server thông qua kết nối WebSocket đã được thiết lập.
3. Trên server,STT sẽ lắng nghe luồng âm thanh này.
4. Dịch vụ STT xử lý âm thanh trong thời gian thực và trả về kết quả là một chuỗi văn bản tương ứng.
5. Chuỗi văn bản này sau đó được chuyển đến Chat API để tiếp tục xử lý và phân tích ý định.

3.5.3. Text-to-Speech

Giới thiệu: Text-to-Speech (TTS) là công nghệ tổng hợp giọng nói, thực hiện quá trình ngược lại với STT: chuyển đổi văn bản thành âm thanh giọng nói nhân tạo. Các hệ thống TTS hiện đại có khả năng tạo ra giọng nói tự nhiên, có ngữ điệu và cảm xúc, gần giống với giọng người thật.

Chi tiết ứng dụng: Sau khi hệ thống đã xử lý xong yêu cầu của người dùng và thực hiện hành động, việc cung cấp phản hồi bằng âm thanh là cần thiết để hoàn thiện trải nghiệm tương tác. TTS được sử dụng để tạo ra các phản hồi này.

1. Chat API sau khi xử lý yêu cầu sẽ tạo ra một câu trả lời bằng văn bản
2. Văn bản này được gửi đến dịch vụ TTS API.
3. Dịch vụ TTS sẽ chuyển đổi chuỗi văn bản thành một tệp hoặc luồng dữ liệu âm thanh.
4. Server sẽ gửi tệp/luồng âm thanh này đến thiết bị Master.
5. Master nhận dữ liệu âm thanh và phát ra loa ngoài, cung cấp phản hồi cho người dùng.

3.5.4. Chat API

Giới thiệu: Chat API là một giao diện lập trình ứng dụng cho phép tương tác với một mô hình ngôn ngữ lớn (LLM) hoặc một chatbot. Nó nhận đầu vào là văn bản (thường là một câu hỏi hoặc một lệnh) và trả về một phản hồi cũng bằng văn bản, được tạo ra dựa trên kiến thức và khả năng suy luận của mô hình.

Chi tiết ứng dụng: Sau khi văn bản được tạo ra từ dịch vụ STT, nó sẽ được gửi đến Chat API. Vai trò của Chat API bao gồm:

- **Phân tích ý định người dùng (Intent Recognition):** Xác định mục đích chính của câu lệnh.
- **Trích xuất thực thể (Entity Extraction):** Rút ra các thông tin quan trọng từ câu lệnh.
- **Kích hoạt Function Calling:** Dựa trên ý định và các thực thể đã trích xuất, Chat API sẽ định dạng đầu ra thành một cấu trúc JSON để gọi hàm tương ứng như đã mô tả ở phần Function Calling.
- **Tạo phản hồi tự nhiên:** Trong trường hợp người dùng hỏi thông tin (ví dụ: "Nhiệt độ phòng khách bây giờ là bao nhiêu?"), sau khi backend truy vấn cơ sở dữ liệu gửi ngược lại cho Chat API. Mô hình sẽ tạo ra một câu trả lời hoàn chỉnh và tự nhiên như: "Nhiệt độ tại phòng khách hiện tại là 28 độ C." thay vì chỉ trả về con số khô khan.

II. Phân tích yêu cầu

1. Yêu cầu chức năng của hệ thống

- **FR1: Quản lý tài khoản người dùng**
    - FR1.1: Hệ thống phải cho phép người dùng mới đăng ký tài khoản bằng email và mật khẩu.
    - FR1.2: Hệ thống phải cho phép người dùng đăng nhập bằng tài khoản đã đăng ký.
    - FR1.3: Hệ thống phải cho phép người dùng đăng xuất khỏi tài khoản.
    - FR1.4: Hệ thống phải cho phép người dùng thay đổi mật khẩu của mình.
- **FR2: Quản lý thiết bị**
    - FR2.1: Người dùng phải có thể thêm một thiết bị mới (Master hoặc Slave) vào tài khoản của mình thông qua giao diện web.
    - FR2.2: Người dùng phải có thể xem danh sách tất cả các thiết bị đã được liên kết với tài khoản.
    - FR2.3: Người dùng phải có thể đặt tên tùy chỉnh và gán thiết bị vào một phòng cụ thể (ví dụ: "Đèn phòng khách").
    - FR2.4: Người dùng phải có thể xóa một thiết bị khỏi tài khoản của mình.
- **FR3: Tương tác bằng giọng nói**
    - FR3.1: Hệ thống phải có khả năng thu âm giọng nói của người dùng thông qua thiết bị Master.
    - FR3.2: Hệ thống phải chuyển đổi giọng nói của người dùng thành văn bản (Speech-to-Text).
    - FR3.3: Hệ thống phải phân tích văn bản để hiểu ý định và các tham số trong câu lệnh của người dùng (ví dụ: hành động "bật", thiết bị "đèn", vị trí "phòng khách").
    - FR3.4: Hệ thống phải có khả năng tạo ra phản hồi bằng văn bản và chuyển đổi nó thành âm thanh (Text-to-Speech).
    - FR3.5: Thiết bị Master phải phát lại phản hồi bằng âm thanh cho người dùng.
- **FR4: Điều khiển thiết bị IoT**
    - FR4.1: Hệ thống phải cho phép người dùng bật/tắt các thiết bị (đèn, quạt) bằng lệnh giọng nói.
    - FR4.2: Hệ thống phải cho phép người dùng bật/tắt các thiết bị thông qua giao diện trên bảng điều khiển web.
    - FR4.3: Trạng thái của thiết bị (bật/tắt) phải được cập nhật đồng bộ và hiển thị chính xác trên giao diện web theo thời gian thực.
- **FR5: Thu thập và hiển thị dữ liệu cảm biến**

- FR5.1: Các thiết bị Slave được trang bị cảm biến phải có khả năng tự động đọc dữ liệu (nhiệt độ, độ ẩm) theo một chu kỳ thời gian định sẵn.
- FR5.2: Thiết bị Slave phải gửi dữ liệu cảm biến lên server.
- FR5.3: Hệ thống phải lưu trữ dữ liệu cảm biến theo thời gian.
- FR5.4: Hệ thống phải hiển thị dữ liệu cảm biến hiện tại trên bảng điều khiển web.
- FR5.5: Hệ thống phải cung cấp biểu đồ để người dùng xem lại lịch sử dữ liệu cảm biến theo các khoảng thời gian (ngày, tuần).

2. Yêu cầu phi chức năng của hệ thống

- **NFR1: Hiệu năng**
  - **Độ trễ phản hồi giọng nói:** Thời gian từ khi người dùng nói xong câu lệnh đến khi hệ thống phát ra phản hồi âm thanh không được vượt quá 3 giây trong điều kiện mạng ổn định.
  - **Cập nhật trạng thái thời gian thực:** Sự thay đổi trạng thái của thiết bị trên giao diện web phải được cập nhật trong vòng dưới 1 giây sau khi hành động thực tế diễn ra.
  - **Tần suất cập nhật cảm biến:** Dữ liệu cảm biến phải được cập nhật trên server và giao diện người dùng ít nhất 5 phút một lần.
- **NFR2: Độ tin cậy**
  - **Thời gian hoạt động:** Hệ thống backend (server) phải đạt thời gian hoạt động (uptime) 99%.
  - **Khả năng phục hồi kết nối:** Các thiết bị ESP32/ESP8266 phải có khả năng tự động kết nối lại với mạng Wi-Fi và server MQTT nếu kết nối bị mất.
  - **Tính toàn vẹn cập nhật FOTA:** Quá trình FOTA phải đảm bảo an toàn. Nếu cập nhật thất bại (mất điện, mất mạng), thiết bị phải có khả năng quay trở lại phiên bản firmware cũ đang hoạt động ổn định.
- **NFR3: Tính khả dụng**
  - **Giao diện người dùng:** Giao diện web phải trực quan, dễ sử dụng, và tương thích với các trình duyệt web.
  - **Phản hồi hệ thống:** Hệ thống phải cung cấp phản hồi rõ ràng cho người dùng trong mọi trường hợp, ví dụ: "Đã bật đèn" khi thành công, hoặc "Tôi không hiểu lệnh của bạn" khi thất bại.
  - **Độ chính xác nhận dạng:** Tỷ lệ nhận dạng chính xác các câu lệnh được định sẵn trong môi trường yên tĩnh phải đạt trên 90%.
- **NFR4: Bảo mật**

    - **Xác thực:** Mọi truy cập đến API và MQTT broker đều phải được xác thực bằng token (JWT) hoặc cơ chế tên người dùng/mật khẩu.
    - **Mã hóa mật khẩu:** Mật khẩu người dùng phải được băm (hashing) trước khi lưu trữ trong cơ sở dữ liệu.
    - **Phân quyền:** Người dùng thông thường không thể truy cập vào dữ liệu hoặc điều khiển thiết bị của người dùng khác.
- **NFR5: Khả năng mở rộng**
    - **Kiến trúc backend:** Backend phải được thiết kế theo dạng module để dễ dàng tích hợp các loại cảm biến hoặc thiết bị mới trong tương lai mà không cần thay đổi lớn về kiến trúc.
- **NFR6: Khả năng bảo trì**
    - **Tài liệu hóa:** Mã nguồn (code) phải được bình luận (comment) rõ ràng. Các API phải có tài liệu hướng dẫn sử dụng.
    - **Ghi log:** Hệ thống server phải ghi lại các sự kiện quan trọng và lỗi để giúp quản trị viên dễ dàng gỡ lỗi và giám sát hoạt động.

3. Các yêu cầu ràng buộc

3.1 Ràng buộc kỹ thuật

- **Ràng buộc về phần cứng:**
    - **Hiệu năng ESP32/ESP8266:** Vi điều khiển ESP32 (Master) có bộ xử lý lõi kép và RAM tương đối lớn, nhưng vẫn bị giới hạn trong việc xử lý các tác vụ phức tạp tại chỗ. Toàn bộ các tác vụ nặng về AI (STT, TTS, LLM) phải được chuyển lên server xử lý. ESP8266 (Slave) có hiệu năng và bộ nhớ thấp hơn, chỉ phù hợp cho các tác vụ đơn giản như đọc cảm biến và điều khiển relay, không thể thực hiện các logic phức tạp.
    - **Chất lượng Micro và Loa:** Chất lượng của microphone tích hợp trên thiết bị Master ảnh hưởng trực tiếp đến độ chính xác của việc nhận dạng giọng nói. Microphone cần có khả năng lọc nhiễu cơ bản. Tương tự, loa ngoài cần có công suất đủ lớn để người dùng nghe rõ phản hồi trong môi trường sinh hoạt thông thường.
    - **Bộ nhớ Flash:** Dung lượng bộ nhớ flash của ESP32/ESP8266 bị giới hạn, ảnh hưởng đến kích thước của firmware. Việc triển khai FOTA (cập nhật qua mạng) yêu cầu phải phân chia bộ nhớ hợp lý (tối thiểu hai phân vùng OTA), làm giảm không gian lưu trữ cho chương trình chính.
- **Ràng buộc về phần mềm và công nghệ:**
    - **Phụ thuộc vào dịch vụ bên thứ ba:** Hệ thống phụ thuộc hoàn toàn vào các API của bên thứ ba cho các chức năng cốt lõi như Speech-to-Text,

Text-to-Speech và Chat API (LLM). Bất kỳ sự thay đổi nào về chính sách, chi phí, hoặc sự cố kỹ thuật từ các nhà cung cấp này đều sẽ ảnh hưởng trực tiếp đến hoạt động của hệ thống.

    - **Độ trễ mạng (Network Latency):** Quá trình tương tác bằng giọng nói yêu cầu độ trễ thấp. Luồng xử lý bao gồm: Master -> Server (STT) -> Server (Chat API) -> Server (TTS) -> Master. Độ trễ ở bất kỳ khâu nào, đặc biệt là do tốc độ mạng hoặc thời gian xử lý của các API, đều sẽ làm giảm trải nghiệm người dùng, khiến cuộc trò chuyện thiếu tự nhiên.
    - **Giao thức truyền thông:** Việc lựa chọn MQTT và WebSocket là cố định. MQTT phù hợp cho việc truyền tin nhắn điều khiển và dữ liệu cảm biến một cách tin cậy, trong khi WebSocket là bắt buộc để streaming audio thời gian thực. Kiến trúc hệ thống phải được xây dựng xoay quanh đặc điểm của hai giao thức này.

- **Ràng buộc về cơ sở dữ liệu:**
    - **Giới hạn của Supabase (Free Tier):** Nếu sử dụng gói miễn phí của Supabase, hệ thống sẽ bị giới hạn về dung lượng lưu trữ, số lượng lượt truy cập API mỗi tháng, và số kết nối đồng thời. Việc lưu trữ dữ liệu cảm biến theo thời gian có thể nhanh chóng làm đầy dung lượng cho phép, đòi hỏi phải có cơ chế nén hoặc xóa dữ liệu cũ.

3.2 Ràng buộc môi trường

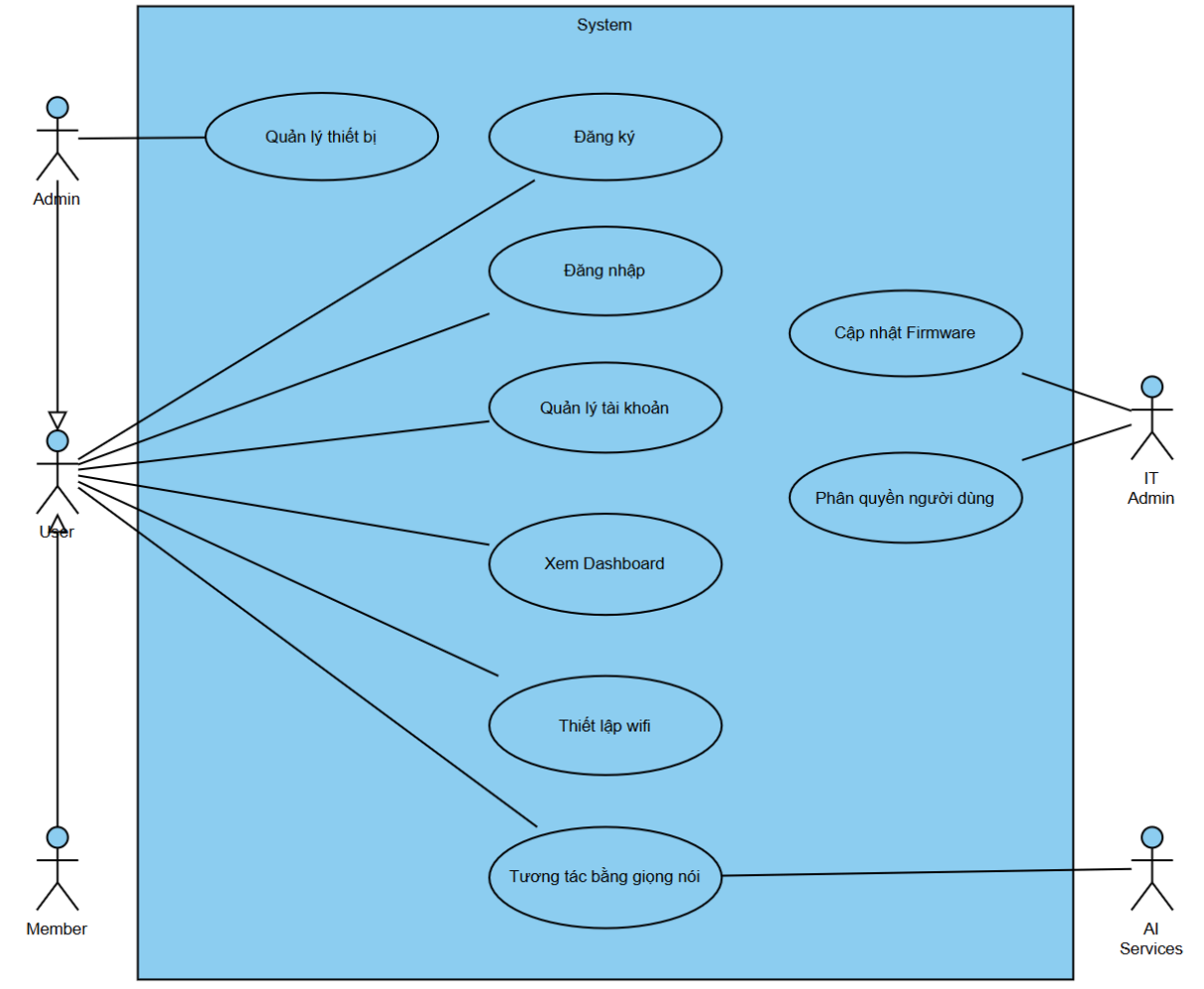
- **Môi trường vật lý:**
    - **Nhiễu âm thanh (Acoustic Noise):** Độ chính xác của hệ thống nhận dạng giọng nói giảm đáng kể trong môi trường có nhiều tiếng ồn (ví dụ: tiếng TV, quạt lớn, nhiều người nói chuyện). Hệ thống được thiết kế để hoạt động tối ưu trong môi trường tương đối yên tĩnh.
    - **Khoảng cách đến thiết bị Master:** Người dùng cần phải ở trong một khoảng cách hợp lý (ví dụ: dưới 3-5 mét) so với thiết bị Master để microphone có thể thu được giọng nói rõ ràng.
    - **Nhiệt độ và độ ẩm hoạt động:** Các vi điều khiển ESP32/ESP8266 và cảm biến có dải nhiệt độ hoạt động nhất định (thường từ -40°C đến 85°C). Mặc dù phù hợp với môi trường trong nhà, chúng có thể hoạt động không ổn định nếu đặt ở những nơi có nhiệt độ quá cao (gần nguồn nhiệt) hoặc độ ẩm quá lớn.
- **Môi trường mạng:**
    - **Độ ổn định của mạng Wi-Fi:** Toàn bộ hệ thống phụ thuộc vào kết nối Wi-Fi. Tín hiệu Wi-Fi yếu, chập chờn hoặc mất kết nối sẽ làm gián đoạn hoàn toàn các chức năng, từ điều khiển thiết bị đến tương tác giọng nói.

- **Băng thông Internet:** Việc streaming audio từ thiết bị Master lên server đòi hỏi một băng thông tải lên (upload) đủ lớn và ổn định. Nếu băng thông thấp, âm thanh sẽ bị ngắt quãng, dẫn đến việc nhận dạng giọng nói thất bại.
- **Cấu hình tường lửa (Firewall):** Mạng nội bộ có thể có các thiết lập tường lửa chặn các cổng (port) hoặc giao thức mà hệ thống sử dụng (MQTT, WebSocket), gây ra sự cố kết nối.

III. Phân tích thiết kế hệ thống

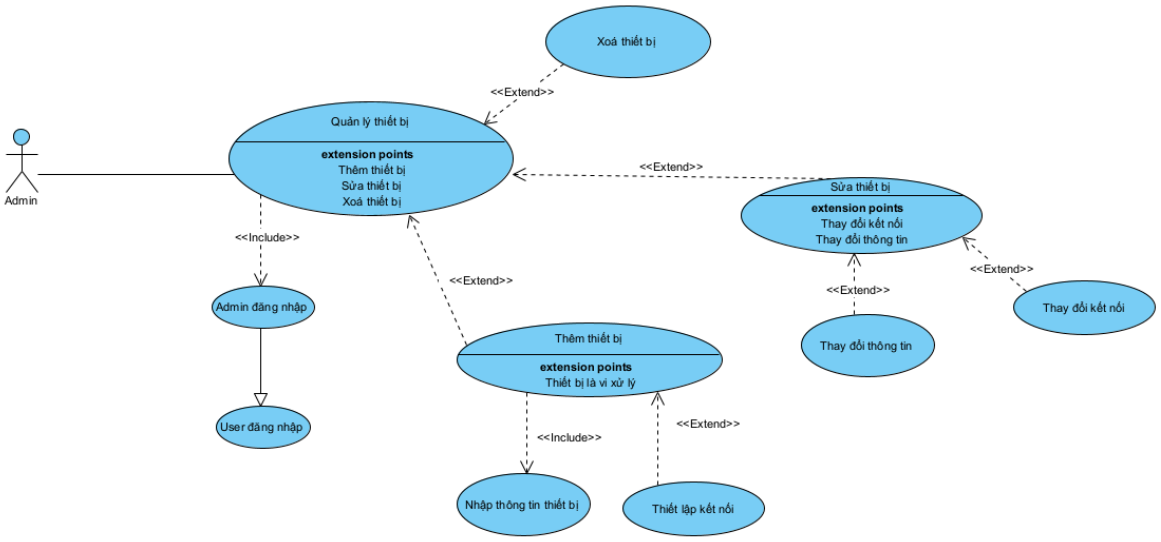
1. Biểu đồ Usecase

1.1 Biểu đồ Usecase tổng quan



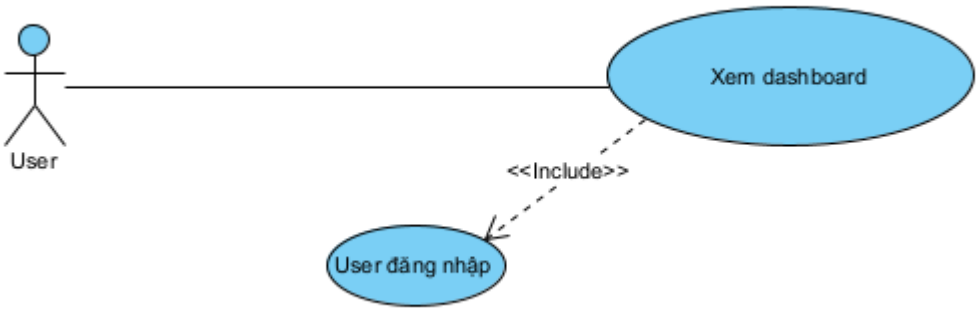
Hình 1: Biểu đồ usecase tổng quan của hệ thống

1.2 Biểu đồ Usecase chi tiết Quản lý thiết bị (Admin)



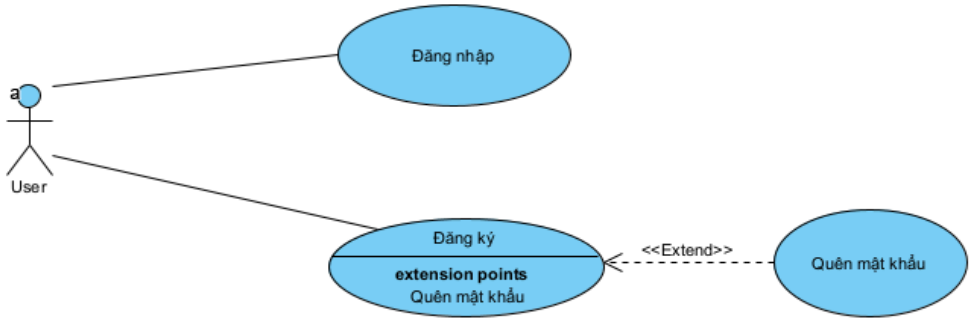
Hình 2: Biểu đồ Usecase quản lý thiết bị

1.3 Biểu đồ Usecase chi tiết Xem Dashboard (User)



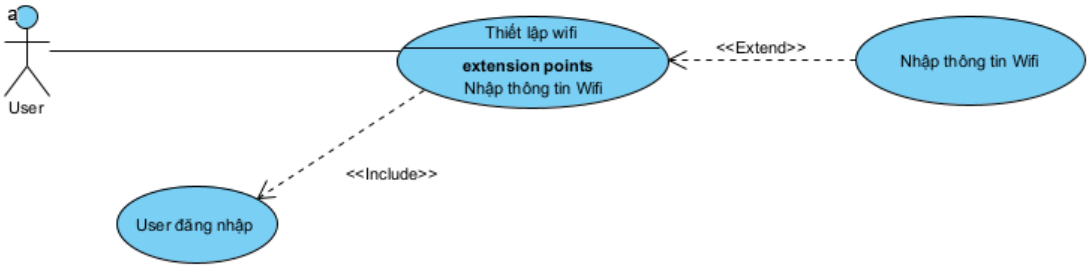
Hình 3: Biểu đồ Usecase xem Dashboard

1.4. Biểu đồ Usecase chi tiết Đăng nhập, Đăng ký



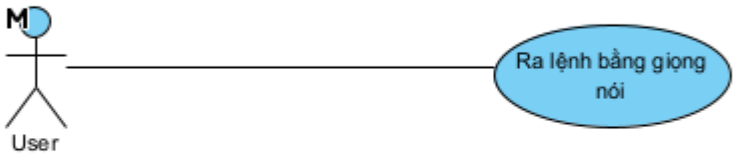
Hình 4: Biểu đồ UseCase Đăng nhập, Đăng ký

1.5. Biểu đồ Usecase chi tiết Thiết lập wifi (User)



Hình 5: Biểu đồ Usecase Thiết lập wifi

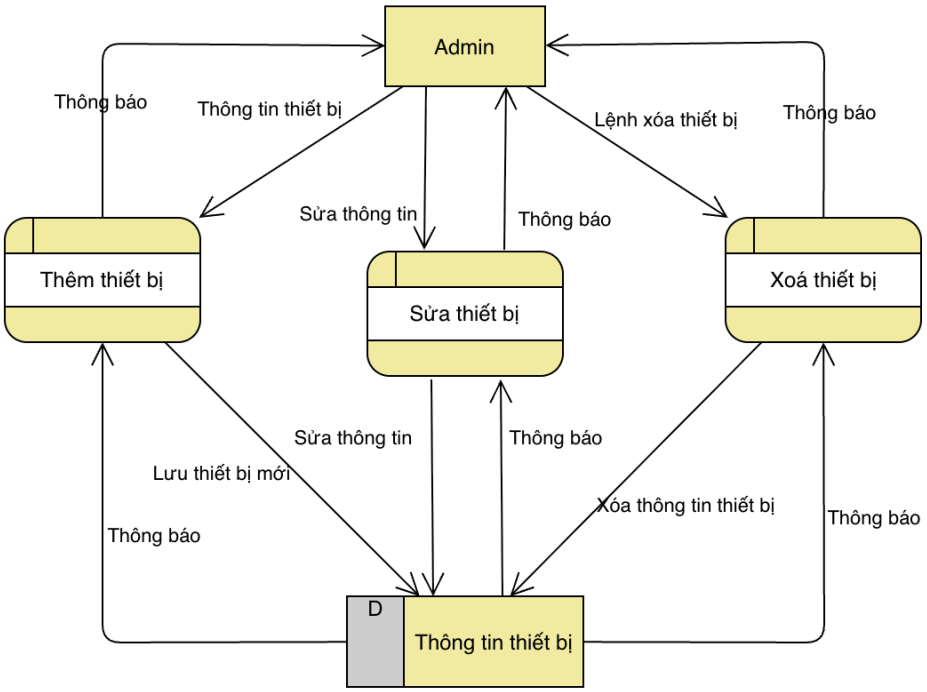
1.6. Biểu đồ Usecase tương tác với thiết bị master (User)



Hình 6: Biểu đồ Usecase tương tác với thiết bị master.

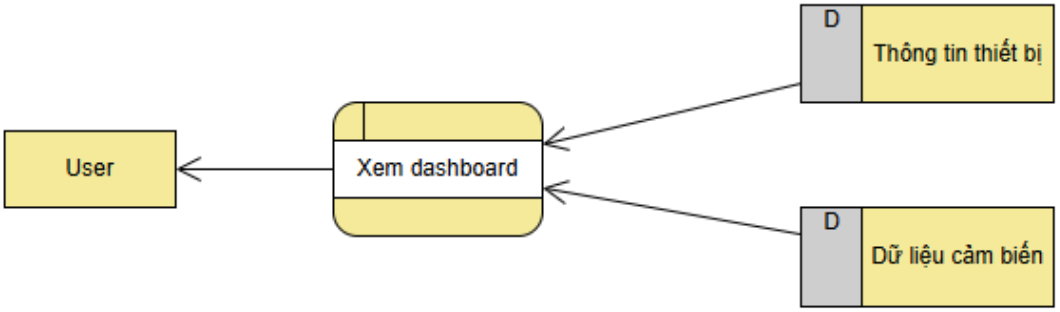
2. Biểu đồ luồng dữ liệu (Data Flow Diagram - DFD)

2.1 Biểu đồ luồng dữ liệu Quản lý thiết bị



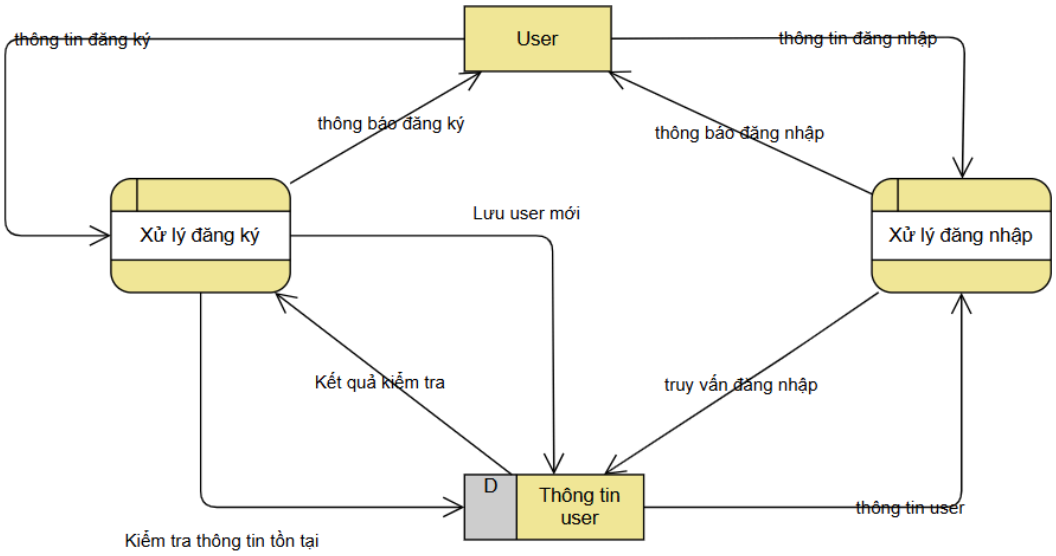
Hình 7: Biểu đồ luồng dữ liệu Quản lý thiết bị

2.2. Biểu đồ luồng dữ liệu Xem Dashboard



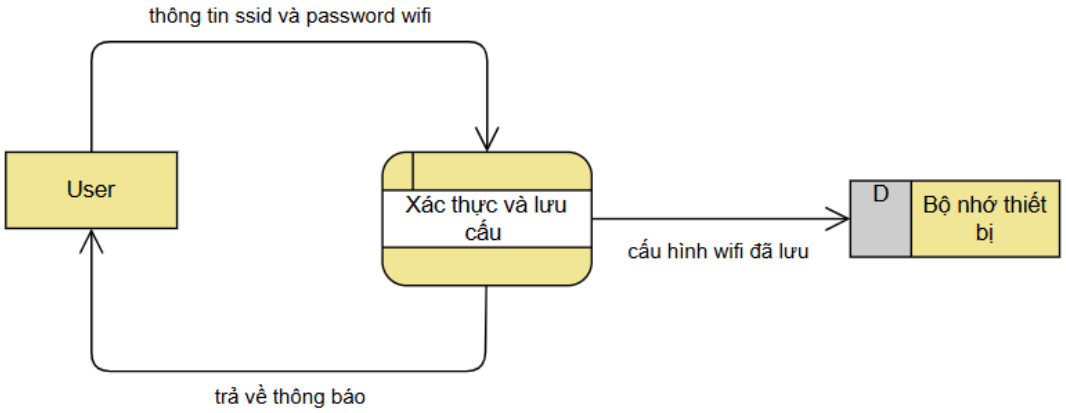
Hình 8: Biểu đồ luồng dữ liệu Xem Dashboard

2.3 Biểu đồ luồng dữ liệu Đăng ký, Đăng nhập



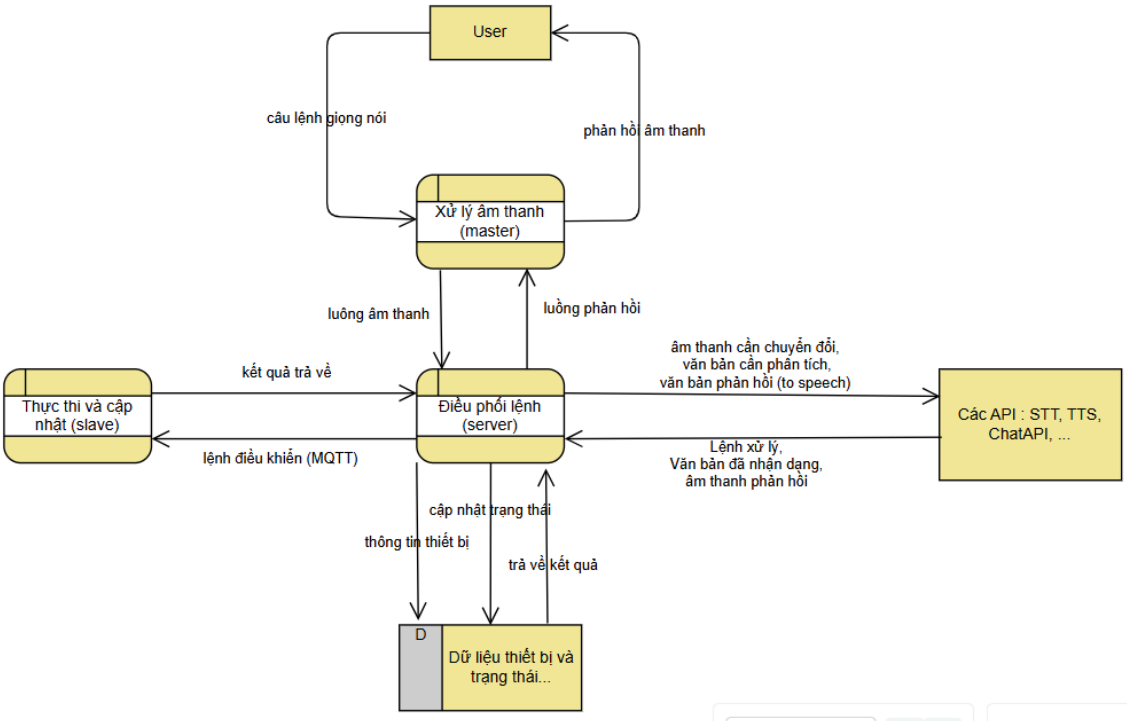
Hình 9: Biểu đồ luồng dữ liệu Đăng ký, Đăng nhập

2.4 Biểu đồ luồng dữ liệu Thiết lập wifi



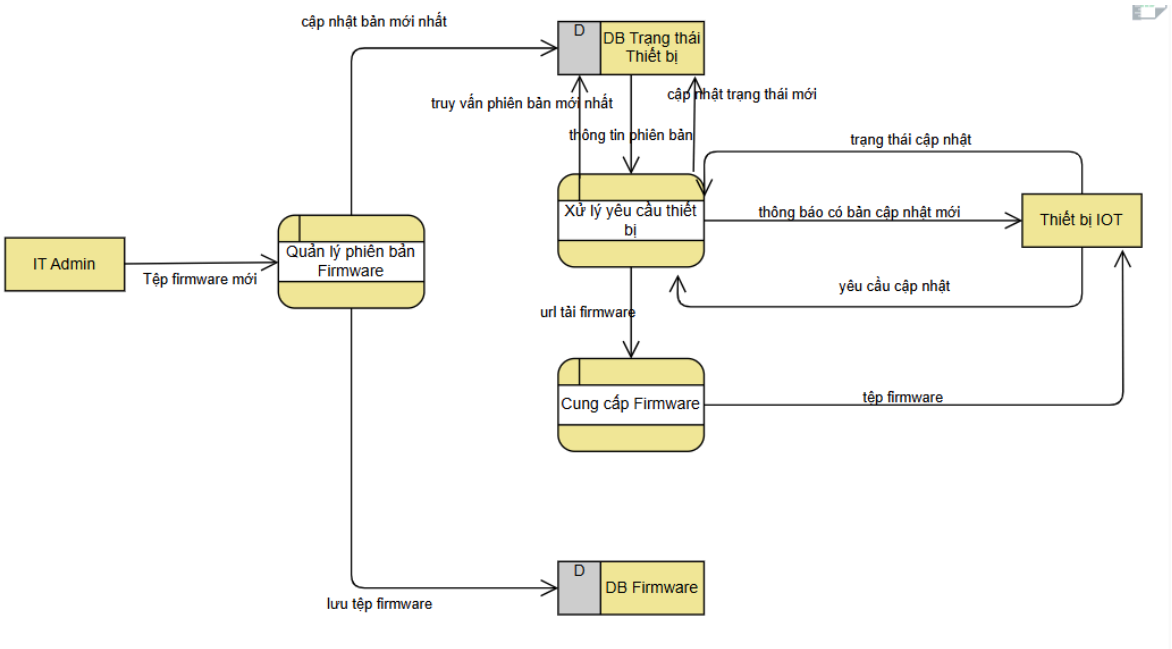
Hình 10: Biểu đồ luồng dữ liệu Thiết lập wifi

2.5 Biểu đồ luồng dữ liệu Tương tác với thiết bị master



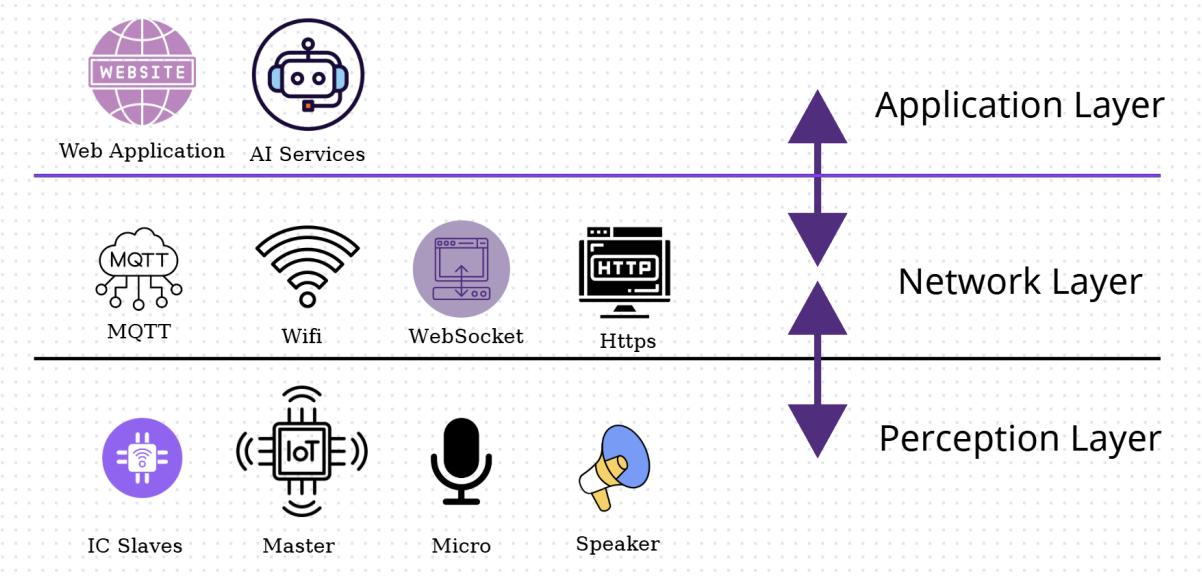
Hình 11: Biểu đồ luồng dữ liệu Tương tác với thiết bị master

2.6 Biểu đồ luồng dữ liệu FOTA



Hình 12: Biểu đồ luồng dữ liệu FOTA

3. Biểu đồ kiến trúc IOT 3 lớp (Three-Layer IoT Architecture).



Hình 13: Biểu đồ kiến trúc IOT 3 lớp

IV. TÀI LIỆU THAM KHẢO

- Espressif Systems. (n.d.). ESP32 Technical Reference Manual.
- *Software Requirement Specification (SRS) Document Checklist*. (2025, July 23). GeeksforGeeks. Retrieved October 15, 2025, from https://www.geeksforgeeks.org/software-engineering/software-requirement-specification-srs-document-checklist/
- *3 layer IoT architecture*. (2025, July 23). GeeksforGeeks. Retrieved October 15, 2025, from https://www.geeksforgeeks.org/computer-networks/3-layer-iot-architecture/
- Vũ, H. T., & Hoàng, T. T. (n.d.). *Bài giảng Phát triển ứng dụng Internet of Things*.

Bảng phân công

Họ và tên	Mã sinh viên	Công việc
Vũ Ngọc Hùng	B22DCCN373	Nhóm trưởng, Backend server
Nguyễn Đức Huy	B22DCCN385	Frontend, AI calling
Đặng Hữu Nghĩa	B22DCCN601	Backend server, Hardware setup
Kim Duy Hưng	B22DCCN409	Frontend, Hardware setup