

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



THỰC TẬP ĐỒ ÁN ĐA NGÀNH
HƯỚNG TRÍ TUỆ NHÂN TẠO (CO3107)

Đề tài:
Hệ thống nhà thông minh

Học kì 242 - Lớp L02

Instructor(s): Vương Bá Thịnh

Student(s):	Lý Nguyên Khang	2211437
	Trương Bình Minh	2212088
	Lê Đăng Khoa	2211601
	Hoa Toàn Hạc	2210917
	Nguyễn Trường Giang	2210829

TP. HỒ CHÍ MINH , 02/2025

Mục lục

1	Giới thiệu đề tài	4
2	User Requirements	5
2.1	Yêu cầu chức năng	5
2.2	Yêu cầu phi chức năng	5
3	Đặc tả Usecase	6
3.1	Use case: Điều khiển thiết bị	6
3.2	Use case: Báo cáo - hiển thị dữ liệu	6
3.3	Use case: Điều khiển bằng giọng nói	7
3.4	Use case: Nhận diện khuôn mặt	8
4	Danh sách thiết bị	11
4.1	YOLO:BIT	11
4.2	Yolo:Bit Extend Shield	11
4.3	Cảm biến DHT20	12
4.4	Mini fan	12
4.5	LED RGB	13
4.6	Servo	13
4.7	Cảm biến PIR	14
4.8	Camera	14
4.9	Micro	15
5	Framework ứng dụng	15
5.1	Speech to control	15
5.2	Face Recognition	16
6	Thiết kế kiến trúc	19
6.1	Tổng quan hệ thống	19
6.2	IoT Gateway (AI Inference Node)	19
6.3	FastAPI (Backend)	20
6.4	ReactJS (Frontend)	21
6.5	Adafruit IO (Cloud MQTT Broker)	21

6.6	PostgreSQL (Cơ sở dữ liệu)	21
7	Thực thi với Adafruit	23
7.1	Các feed chính	23
7.2	Các giao thức chính	23
7.3	Hiện thực IotGateway	24
8	Trở lý ảo giọng nói	29
8.1	Mô hình phát hiện giọng nói (Voice activity detection)	30
8.2	Mô hình chuyển giọng nói thành văn bản tiếng Việt (Speech to text)	30
8.3	Xác định hành động bằng Word-matching hoặc LLM	31
8.4	Kết nối với cơ sở dữ liệu	33
9	Hiện thực phần nhận diện gương mặt	34
9.1	Các thư viện cần sử dụng	34
9.2	Trích xuất đặc trưng gương mặt	34
9.3	Đăng ký khuôn mặt	36
9.4	Nhận diện gương mặt thời gian thực	36
9.5	Giao tiếp với hệ thống	39
10	Mockup	40
10.1	Trang đăng nhập	40
10.2	Trang chủ chính	40
10.3	Trang quản lý thành viên	41
10.4	Trang quản lý thiết bị	41
10.5	Trang xem toàn bộ thông báo	42
10.6	Trang xem lịch sử ra vào nhà	42
10.7	Trang biểu đồ	42
10.8	Trang thông tin cá nhân	43
11	Lưu trữ code - Demo	44



Phân công công việc

STT	Họ và tên	MSSV	Nội dung thực hiện	Đánh giá
1	Lý Nguyên Khang	2211437	Backend, Database	100%
2	Trương Bình Minh	2212088	Module điều khiển bằng giọng nói	100%
3	Lê Đăng Khoa	2211601	IOT Gateway	100%
4	Hoa Toàn Hạc	2210917	Module nhận diện khuôn mặt	100%
5	Nguyễn Trường Giang	2210829	Frontend web app	100%

Bảng 1: Bảng phân công công việc và kết quả thực hiện của các thành viên

1 Giới thiệu đề tài

Trong bối cảnh cuộc sống hiện đại đang ngày càng phát triển, nhu cầu nâng cao chất lượng sống, tiết kiệm năng lượng và tăng cường mức độ an toàn, tiện nghi trong không gian sống trở thành mối quan tâm hàng đầu. Một trong những xu hướng công nghệ nổi bật đang được ứng dụng rộng rãi hiện nay là tự động hóa và Internet of Things (IoT). Việc áp dụng những công nghệ này vào mô hình nhà ở thông minh (Smart Home) không chỉ là bước tiến tất yếu, mà còn là giải pháp hiệu quả để xây dựng một môi trường sống hiện đại, linh hoạt và an toàn.

Dự án Smart Home hướng tới việc xây dựng một hệ thống tích hợp giữa phần cứng và phần mềm, nhằm hiện thực hóa ý tưởng về một ngôi nhà có khả năng tự động giám sát, điều khiển và phản ứng linh hoạt trước các tình huống trong môi trường sống. Về mặt phần cứng, hệ thống sử dụng các thiết bị IoT như cảm biến nhiệt độ, độ ẩm, chuyển động, ánh sáng,... để thu thập các thông tin môi trường và trạng thái hiện tại trong ngôi nhà. Các dữ liệu này sau đó sẽ được truyền đến hệ thống xử lý trung tâm để phân tích và đưa ra các hành động thích hợp, chẳng hạn như tự động bật/tắt thiết bị điện, gửi cảnh báo đến người dùng khi phát hiện sự bất thường, hoặc điều chỉnh điều hòa nhiệt độ theo điều kiện môi trường.

Bên cạnh đó, để nâng cao trải nghiệm người dùng và tăng tính tương tác của hệ thống, dự án tích hợp các kỹ thuật Trí tuệ nhân tạo (AI) vào trong hoạt động điều khiển. Cụ thể, người dùng có thể sử dụng giọng nói để ra lệnh cho hệ thống thực hiện các thao tác như bật/tắt đèn, điều chỉnh nhiệt độ, mở cửa, v.v. Ngoài ra, hệ thống còn hỗ trợ tính năng nhận diện khuôn mặt, giúp phân quyền truy cập hoặc tự động mở cửa cho các thành viên trong gia đình mà không cần sử dụng khóa vật lý.

Một điểm mạnh khác của hệ thống là khả năng điều khiển và giám sát từ xa. Thông qua một ứng dụng web có giao diện thân thiện, người dùng có thể theo dõi tình trạng thiết bị, nhận cảnh báo thời gian thực và thao tác điều khiển từ bất kỳ đâu bằng điện thoại thông minh hoặc máy tính có kết nối Internet. Điều này giúp người dùng kiểm soát tốt hơn không gian sống của mình, ngay cả khi không có mặt tại nhà.

Tóm lại, dự án Smart Home không chỉ là một ứng dụng thực tiễn của IoT và AI, mà còn mang ý nghĩa to lớn trong việc hướng tới một cuộc sống thông minh, an toàn, tiết kiệm và thân thiện với người dùng. Với việc kết hợp giữa công nghệ hiện đại và nhu cầu thực tế, hệ thống này hứa hẹn sẽ là nền tảng cho các giải pháp nhà ở tương lai.

2 User Requirements

2.1 Yêu cầu chức năng

- Người dùng có thể điều khiển các thiết bị trong nhà như quạt, đèn từ xa thông qua ứng dụng web
- Người dùng có thể sử dụng giọng nói để điều khiển các thiết bị trong nhà.
- Hệ thống hỗ trợ nhận lệnh giọng nói để điều khiển thiết bị.
- Nhận diện khuôn mặt để mở cửa hoặc xác thực người dùng.
- Hỗ trợ danh sách khách đến nhà hoặc cảnh báo khi có người lạ.
- Ứng dụng web hiển thị trạng thái thiết bị và cảm biến (nhiệt độ, độ ẩm, chuyển động, v.v.).
- Cho phép chủ nhà nhận các cảnh báo (người lạ, sự cố,...). Đồng thời cho phép những khách nào có thể được phép vào nhà.
- Hiển thị và biểu diễn, so sánh với nhiệt độ, độ ẩm trong nhà với ngoài trời.
- Lưu lại lịch sử vào nhà của các người dùng đồng thời các thông báo.
- Hỗ trợ kết nối với các thiết bị IoT thông qua Adafruit IO và backend.

2.2 Yêu cầu phi chức năng

- **Hiệu suất:** Hệ thống phản hồi nhanh, độ trễ thấp ($<1s$ với điều khiển thiết bị).
- **Khả năng mở rộng:** Hệ thống phải có khả năng mở rộng để hỗ trợ ít nhất 100 thiết bị đồng thời
- **Tính bảo mật:** Hệ thống phải đảm bảo rằng dữ liệu người dùng được mã hóa và bảo vệ khỏi truy cập trái phép.
- **Tính sẵn sàng:** Hệ thống phải có thời gian hoạt động ít nhất 99.9% trong suốt thời gian hoạt động
- **Khả năng bảo trì:** Hệ thống phải được thiết kế để dễ dàng bảo trì và cập nhật mà không ảnh hưởng đến hoạt động của người dùng

3 Đặc tả Usecase

3.1 Use case: Điều khiển thiết bị

Use-case name	Điều khiển nhà thông qua web
Actor	Người dùng, Hệ thống, Web
Mô tả	Use-case này mô tả cách người dùng có thể điều khiển các thiết bị trong nhà (đèn, quạt, v.v.) thông qua giao diện web. Hệ thống sẽ xử lý yêu cầu từ người dùng trên web và thực hiện hành động tương ứng.
Preconditions	<ul style="list-style-type: none">• Hệ thống và giao diện web đang hoạt động.• Người dùng đã đăng nhập và có quyền điều khiển thiết bị trong nhà.
Postconditions	<ul style="list-style-type: none">• Hệ thống đã thực hiện hành động được yêu cầu.• Hệ thống phản hồi kết quả thực hiện cho người dùng.
Normal Flow	<ol style="list-style-type: none">1. Người dùng đăng nhập vào giao diện web.2. Người dùng chọn các thiết bị cần được điều khiển trong nhà (ví dụ: bật đèn, điều chỉnh nhiệt độ).3. Hệ thống xử lý yêu cầu và kiểm tra xem hành động có hợp lệ và có thể thực hiện không.4. Hệ thống gửi tín hiệu đến thiết bị và thực hiện hành động điều chỉnh.5. Hệ thống phản hồi cho người dùng (ví dụ: "Đèn phòng khách đã bật").
Alternative Flow	Ở bước 3, nếu hệ thống phát hiện lệnh không hợp lệ hoặc không nhận diện được, hệ thống sẽ hiển thị thông báo lỗi và đề xuất các lệnh hợp lệ.
Exceptions	Nếu thiết bị không được kết nối hoặc không thể điều khiển, hệ thống sẽ thông báo cho người dùng về sự cố.

3.2 Use case: Báo cáo - hiển thị dữ liệu

Use-case name	Hiển thị dữ liệu ngôi nhà
Actor	Chủ nhà
Mô tả	Hệ thống cho phép chủ nhà xem dữ liệu thời gian thực và lịch sử về các điều kiện môi trường trong nhà, bao gồm nhiệt độ, độ ẩm và mức độ ánh sáng, thông qua giao diện web.
Preconditions	Hệ thống được kết nối với cơ sở dữ liệu và đang ghi nhận dữ liệu môi trường liên tục.
Postconditions	<ul style="list-style-type: none">• Không có.
Normal Flow	<ol style="list-style-type: none">1. Chủ nhà truy cập giao diện web.2. Chủ nhà chọn chức năng "Dữ liệu môi trường".3. Hệ thống hiển thị dữ liệu thời gian thực về nhiệt độ, độ ẩm và ánh sáng.4. Chủ nhà có thể chọn khoảng thời gian cụ thể để xem dữ liệu quá khứ.5. Hệ thống truy xuất và hiển thị dữ liệu lịch sử dưới dạng biểu đồ.6. Chủ nhà nhấn "Quay lại" để trở về giao diện chính.
Alternative Flow	Tại bước 2, nếu chủ nhà chọn "Tùy chỉnh giao diện dữ liệu": <ol style="list-style-type: none">1. Hệ thống hiển thị danh sách các thông số môi trường có thể chọn.2. Chủ nhà chọn hoặc bỏ chọn các thông số muốn hiển thị.3. Hệ thống lưu cấu hình và chỉ hiển thị các thông số đã chọn trong những lần xem tiếp theo.
Exceptions	Ngoại lệ 1: Nếu không có dữ liệu lịch sử cho khoảng thời gian được chọn: <ul style="list-style-type: none">• Hệ thống hiển thị thông báo lỗi và đề xuất khoảng thời gian có dữ liệu.

3.3 Use case: Điều khiển bằng giọng nói

Tên	Điều khiển bằng giọng nói
------------	---------------------------

Actor	Người dùng, Hệ thống
Mô tả	Use-case này mô tả cách để người dùng điều khiển các thiết bị trong nhà bằng giọng nói (bật/tắt đèn, điều hoà, TV, v.v.). Hệ thống sẽ xử lý thông tin ghi âm từ giọng nói, xác định hành động cần thực hiện và thực hiện hành động tương ứng.
Preconditions	Hệ thống hoạt động bình thường và sẵn sàng để ghi âm giọng nói.
Postconditions	<ul style="list-style-type: none"> • Lệnh từ người dùng đã được hệ thống thực thi. • Hệ thống sẽ thông báo người dùng đã thực thi thành công.
Normal Flow	<ol style="list-style-type: none"> 1. Người dùng ra lệnh bằng giọng nói (ví dụ: “Bật đèn phòng ngủ.”). 2. Mô-đun nhận dạng giọng nói thu thập và xử lý đầu vào giọng nói, xác định hành động cần thực hiện tương ứng. 3. Hệ thống xác minh xem hành động được yêu cầu có hợp lệ và có thể thực hiện được hay không. 4. Hệ thống thực hiện hành động. 5. Hệ thống thông báo người dùng đã thực thi lệnh thành công.
Alternative Flow	Ở bước 2, nếu hệ thống không hiểu lệnh giọng nói (do lệnh không tồn tại hoặc lỗi trong quá trình ghi âm và xử lý giọng nói), hệ thống sẽ thông báo yêu cầu người dùng lặp lại lệnh. (“Không nhận diện được lệnh. Bạn có thể lặp lại không?”).
Exceptions	Nếu thiết bị không được kết nối hoặc không thể điều khiển, hệ thống sẽ thông báo cho người dùng về sự cố.

3.4 Use case: Nhận diện khuôn mặt

Use Case Name	Xác thực - Nhận diện khuôn mặt
----------------------	--------------------------------

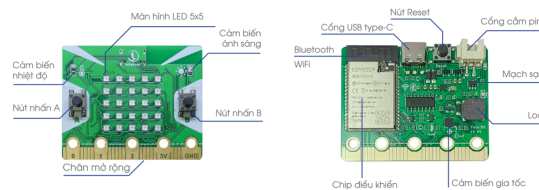
Description	Khi hệ thống không thể xác thực khuôn mặt của người đến, chủ nhà sẽ nhận được thông báo và có quyền quyết định cho phép hoặc từ chối cho người đó vào nhà. Hệ thống cung cấp thông tin kèm thời gian truy cập và tùy chọn hiển thị hoặc không hiển thị hình ảnh người đến. Ngoài ra, hệ thống cũng lưu lại lịch sử các lần truy cập để chủ nhà có thể tra cứu khi cần.
Actors	<ul style="list-style-type: none">• Chủ nhà (Primary)• Hệ thống (Secondary)
Pre-conditions	<ul style="list-style-type: none">• Hệ thống nhận diện khuôn mặt hoạt động bình thường.• Hệ thống đã không thể xác thực tự động người đến.
Normal Flow	<ol style="list-style-type: none">1. Chủ nhà nhận được thông báo có khách không xác thực được.2. Chủ nhà kiểm tra thông tin thời gian truy cập, có thể kèm hình ảnh hoặc không (tùy thiết lập).3. Nếu khách được phép vào, chủ nhà xác nhận mở cửa từ xa.4. Hệ thống thực hiện mở cửa.5. Lưu thông tin vào lịch sử ra vào nhà.
Alternative Flow	<ul style="list-style-type: none">• (3a) Nếu chủ nhà từ chối khách:<ul style="list-style-type: none">– Hệ thống giữ cửa đóng.– Có thể gửi cảnh báo hoặc lưu sự kiện vào nhật ký nếu cần.

Post-conditions	
	<ul style="list-style-type: none">• Nếu chủ nhà xác nhận, cửa sẽ được mở.• Nếu chủ nhà từ chối, cửa vẫn giữ nguyên trạng thái đóng.• Lịch sử ra vào (có/không có hình ảnh) được lưu trữ lại trong hệ thống.

4 Danh sách thiết bị

4.1 YOLO:BIT

Yolo:Bit là một vi điều khiển dựa trên nền tảng ESP32, được thiết kế thân thiện với người mới bắt đầu, đặc biệt là trong việc học lập trình và phát triển các ứng dụng IoT. Thiết bị này hỗ trợ kết nối WiFi, tương thích với nhiều ngôn ngữ lập trình như MicroPython, và được tích hợp sẵn các cảm biến cơ bản như cảm biến nhiệt độ, ánh sáng, nút nhấn,...

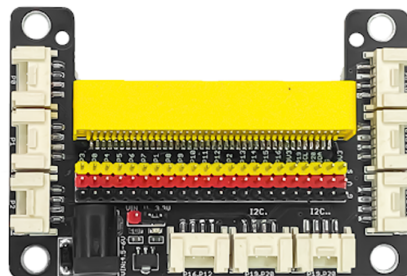


Hình 1: Vi điều khiển Yolo:Bit

4.2 Yolo:Bit Extend Shield

Mạch mở rộng Yolo:Bit Extend Shield cho phép tăng cường số lượng cổng kết nối của vi điều khiển Yolo:Bit, hỗ trợ việc tích hợp thêm nhiều cảm biến và thiết bị điều khiển như máy bơm, đèn LED, màn hình LCD,... Shield này sử dụng chuẩn kết nối Grove, giúp đơn giản hóa quá trình lắp ráp phần cứng mà không cần đến việc hàn dây hay sử dụng breadboard.

Trong hệ thống, Yolo:Bit Extend Shield đóng vai trò như một bộ trung gian, kết nối giữa vi điều khiển và các thành phần phần cứng ngoại vi, đảm bảo việc truyền nhận dữ liệu diễn ra ổn định, đồng thời giúp việc bảo trì và mở rộng hệ thống trở nên dễ dàng hơn.



Hình 2: Yolo:Bit Extend Shield

4.3 Cảm biến DHT20



Hình 3: DHT20

DHT20 là thiết bị dùng để đo độ ẩm và nhiệt độ. Được sử dụng để đo nhiệt độ hệ thống trong nhà.

- Input: Nhiệt độ và độ ẩm.
- Output: Nhiệt độ và độ ẩm được gửi đến cloud và website quản lý hệ thống SmartHome.

4.4 Mini fan



Hình 4: Mini_fan

Mini_fan được ứng dụng trong hệ thống SmartHome với khả năng nhận biết được voice command. Đồng thời cũng có thể bật tắt quạt thủ công thông qua công tắc trên website.

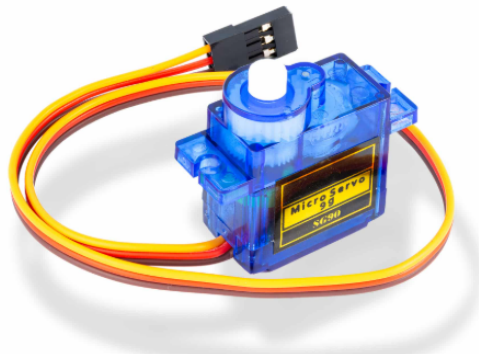
4.5 LED RGB



Hình 5: LED RGB

LED RGB được tích hợp vào hệ thống SmartHome để làm thay cho bóng đèn công suất lớn với mục đích mô tả hệ thống.

4.6 Servo



Hình 6: Servo

Servo trong hệ thống SmartHome được dùng để mô tả cho việc mở khóa căn nhà. Với tính năng xoay (0-180) độ.

4.7 Cảm biến PIR



Hình 7: PIR

PIR là một cảm biến được dùng để phát hiện chuyển động. Trong dự án SmartHome nó được tích hợp để phát hiện có người đến nhà hay không, nếu có nó sẽ trả về **False**. Mỗi lần có người đến nó sẽ như là công tắc để kích hoạt camera lấy ảnh để thông báo và nhận diện.

4.8 Camera



Hình 8: Camera

Với việc có tích hợp face recognition thì cần phải có camera để nhận diện. Do đây chỉ là đề tài nhỏ nên nhóm quyết định sẽ sử dụng camera laptop để mô phỏng lại tính năng của hệ thống.

4.9 Micro



Hình 9: Micro

Ngoài việc có nhận diện khuôn mặt nhóm còn tích hợp voice command để điều khiển nên nhóm quyết định sẽ tạo sử dụng micro laptop hoặc micro rời để có thể mô phỏng lại.

5 Framework ứng dụng

5.1 Speech to control

PhoWhisper – Hugging Face Transformers

PhoWhisper là phiên bản mô hình Whisper được tinh chỉnh (fine-tuned) cho tiếng Việt, dùng để chuyển đổi giọng nói thành văn bản (speech-to-text). Mô hình này được triển khai dựa trên thư viện **Transformers** của **Hugging Face**, một thư viện mã nguồn mở nổi tiếng hỗ trợ nhiều mô hình học sâu trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). PhoWhisper cho phép hệ thống nhận diện và phiên âm giọng nói tiếng Việt với độ chính xác cao, đặc biệt phù hợp với các ứng dụng điều khiển bằng giọng nói trong môi trường tiếng Việt.

Silero VAD – PyTorch

Silero VAD (Voice Activity Detection) là mô hình phát hiện vùng có giọng nói trong tín hiệu âm thanh. Mô hình này được xây dựng trên nền tảng **PyTorch**, giúp tách phần âm thanh có chứa lời nói ra khỏi tiếng ồn nền hoặc đoạn im lặng. Trong hệ thống này, Silero VAD được sử dụng như một bước tiền xử lý (preprocessing) quan trọng trước khi đưa dữ liệu âm thanh vào mô hình PhoWhisper. Nhờ vậy, độ chính xác khi nhận diện giọng nói được cải thiện đáng kể và hiệu suất xử lý được tối ưu.

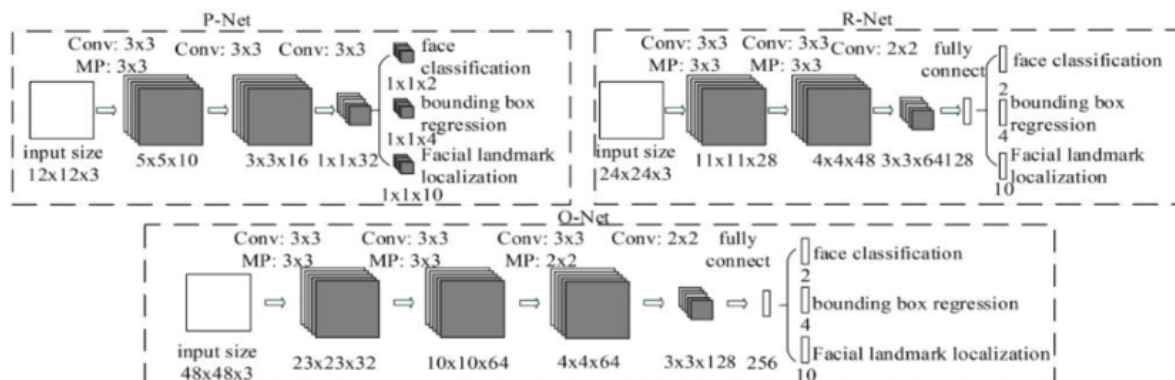
Ollama – LLM Serving

Ollama là nền tảng hỗ trợ triển khai và phục vụ các mô hình ngôn ngữ lớn (LLM – Large Language Models) một cách nhẹ nhàng và hiệu quả trên môi trường máy cục bộ. Trong dự án này, Ollama được sử dụng để chạy các mô hình như LLaMA, Mistral hoặc các phiên bản tinh chỉnh nhằm thực hiện các tác vụ xử lý ngôn ngữ như hiểu và phản hồi lệnh từ người dùng, tạo văn bản, hoặc thực hiện tương tác tự nhiên trong hệ thống nhà thông minh. Ollama giúp đơn giản hóa quy trình triển khai LLM và giảm phụ thuộc vào hạ tầng cloud.

5.2 Face Recognition

MTCNN – Multi-task Cascaded Convolutional Networks

MTCNN là một mô hình phát hiện khuôn mặt nổi bật được đề xuất bởi Zhang et al. (2016), sử dụng kiến trúc ba tầng mạng tích chập (CNN) dạng cascaded để phát hiện chính xác khuôn mặt và các điểm đặc trưng (facial landmarks) như mắt, mũi và miệng trong ảnh hoặc video. Trong dự án này, MTCNN đóng vai trò là thành phần tiền xử lý đầu vào, chịu trách nhiệm phát hiện và căn chỉnh khuôn mặt trước khi đưa vào mô hình nhận diện.



Hình 10: Multi-Task Cascaded Convolutional Neural Networks

Cấu trúc của MTCNN bao gồm 3 tầng:

- **P-Net (Proposal Network):** Quét ảnh đầu vào và tạo ra các vùng đề xuất (candidate boxes) có khả năng chứa khuôn mặt. Giai đoạn này thực hiện nhanh, song độ chính xác chưa cao.
- **R-Net (Refine Network):** Nhận vào các vùng đề xuất từ P-Net, loại bỏ các hộp sai (false positives) và tinh chỉnh vị trí các bounding boxes.
- **O-Net (Output Network):** Giai đoạn cuối, xác nhận các khuôn mặt thật sự và đồng thời trích xuất 5 điểm landmark chính gồm: mắt trái, mắt phải, mũi, mép trái và mép phải của miệng.

Đặc điểm kỹ thuật:

- Phát hiện đa khuôn mặt trong một ảnh duy nhất, không bị giới hạn số lượng người.
- Hoạt động tốt trong điều kiện ánh sáng yếu, góc nghiêng, hoặc khuôn mặt bị xoay nhẹ.
- Cung cấp toạ độ chính xác của khuôn mặt và các điểm landmark để hỗ trợ bước căn chỉnh (alignment).

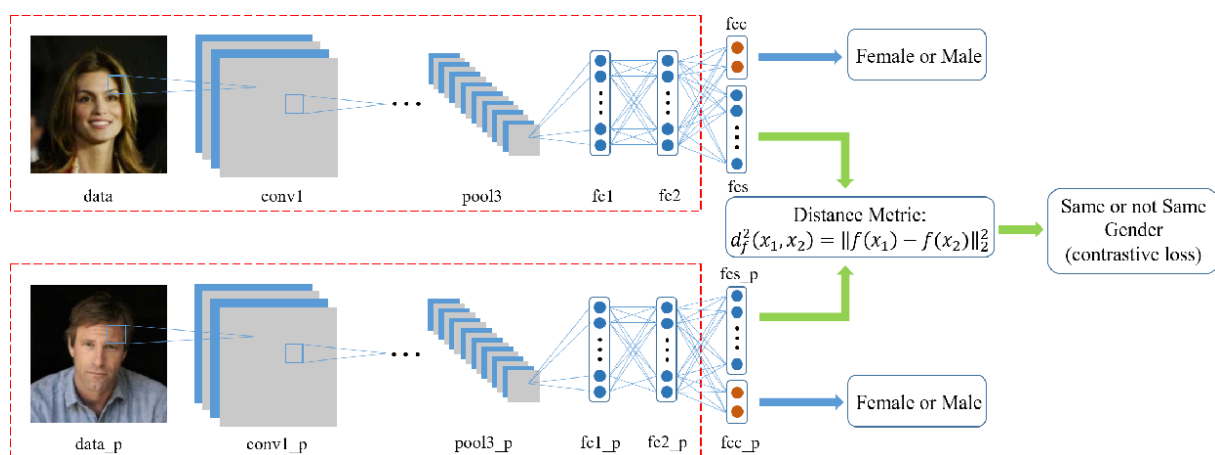
Ứng dụng trong hệ thống: MTCNN được dùng để:

- Phát hiện vùng khuôn mặt từ khung hình webcam theo thời gian thực.
- Cắt và chuẩn hóa ảnh khuôn mặt về kích thước tiêu chuẩn (160x160).
- Tăng độ chính xác cho bước trích xuất đặc trưng nhờ căn chỉnh chuẩn.

Lý do lựa chọn: So với các phương pháp truyền thống như Haar Cascade hay HOG, MTCNN cho kết quả ổn định và chính xác hơn trong điều kiện thực tế, đồng thời dễ tích hợp vào hệ thống AI hiện đại thông qua thư viện như **facenet-pytorch**.

FaceNet

FaceNet là một mô hình học sâu được phát triển bởi nhóm nghiên cứu tại Google vào năm 2015. Không giống như các mô hình nhận diện khuôn mặt truyền thống dựa trên phân loại (classification), mục tiêu của FaceNet là ánh xạ khuôn mặt thành một vector đặc trưng (embedding) trong không gian nhiều chiều, sao cho các khuôn mặt giống nhau nằm gần nhau, và các khuôn mặt khác nhau nằm xa nhau.



Hình 11: FaceNet

Nguyên lý hoạt động:

- Đầu ra của FaceNet là một vector kích thước cố định (thường là 128 hoặc 512 chiều).
- Khoảng cách giữa hai vector embedding phản ánh mức độ giống nhau giữa hai khuôn mặt.
- Huấn luyện bằng phương pháp **triplet loss**, trong đó mỗi bước huấn luyện sử dụng một bộ ba ảnh: ảnh gốc (anchor), ảnh cùng người (positive), và ảnh người khác (negative). Mục tiêu là:

$$\|f(x_a) - f(x_p)\|_2^2 + \alpha < \|f(x_a) - f(x_n)\|_2^2$$

với α là khoảng cách biên an toàn (margin).

Mô hình sử dụng trong dự án: Nhóm sử dụng phiên bản InceptionResnetV1 pretrained trên tập dữ liệu VGGFace2 thông qua thư viện **facenet-pytorch**. Mô hình này:

- Có khả năng tổng quát tốt nhờ được huấn luyện trên tập dữ liệu lớn và đa dạng.
- Không cần huấn luyện lại, có thể dùng trực tiếp như một "trích xuất đặc trưng" cho bài toán nhận diện.
- Tạo ra embedding 512 chiều cho mỗi khuôn mặt.

Ứng dụng trong hệ thống:

- Tạo embedding từ các ảnh đăng ký ban đầu và lưu trữ trong định dạng JSON.
- Trong quá trình nhận diện, trích xuất embedding từ ảnh mới và so sánh với các embedding đã lưu bằng độ tương đồng cosine:

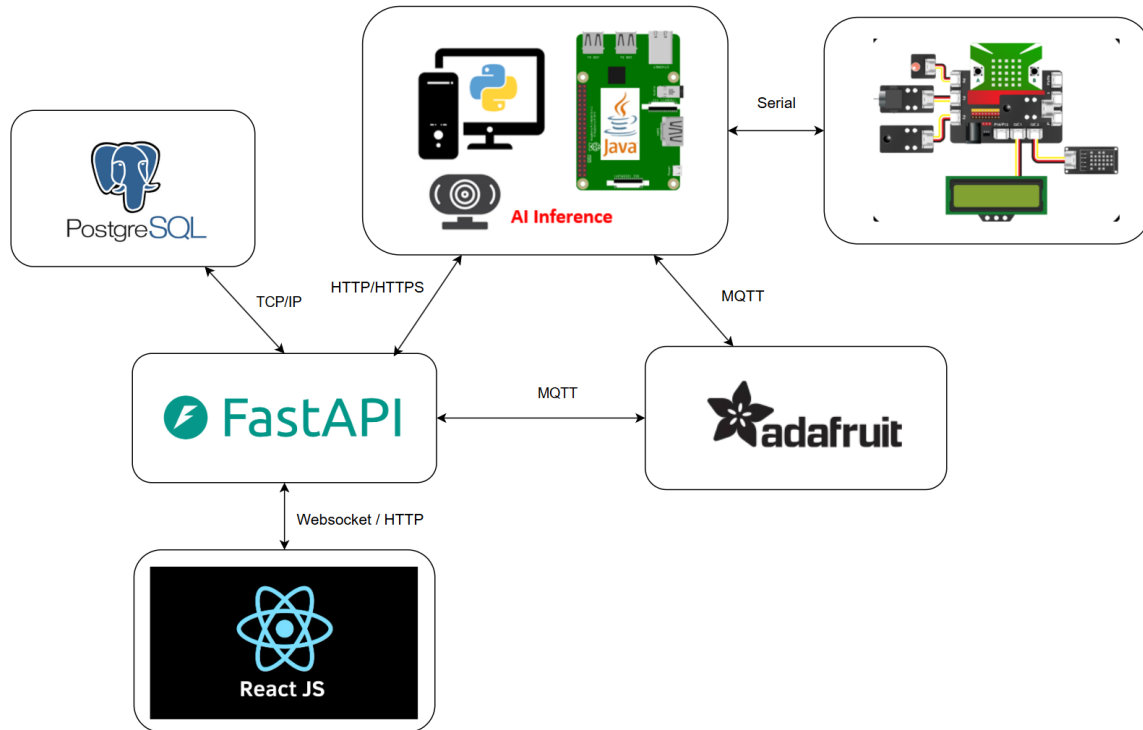
$$\text{cosine_similarity}(a, b) = \frac{a \cdot b}{\|a\| \cdot \|b\|}$$

- Nếu độ tương đồng vượt một ngưỡng định trước (ví dụ: 0.88), gán nhãn người dùng tương ứng.

Ưu điểm:

- Hiệu quả với cả tập dữ liệu nhỏ vì dùng mô hình pretrained.
- Có thể mở rộng để thêm người dùng mới mà không cần huấn luyện lại.
- Hoạt động ổn định và nhanh trong môi trường thực tế.

6 Thiết kế kiến trúc



Hình 12: Hệ thống đề xuất

6.1 Tổng quan hệ thống

Hệ thống bao gồm nhiều thành phần hoạt động phối hợp nhằm cung cấp khả năng giám sát và điều khiển thiết bị IoT thông qua trí tuệ nhân tạo và nền tảng web. Các thành phần chính:

- **IoT Gateway:** thiết bị nhúng chạy các mô hình AI.
- **FastAPI:** backend xử lý logic, giao tiếp và kết nối dữ liệu.
- **ReactJS:** frontend cung cấp giao diện điều khiển, theo dõi.
- **PostgreSQL:** cơ sở dữ liệu lưu trữ thông tin và lịch sử hệ thống.
- **Adafruit IO:** nền tảng MQTT hỗ trợ trao đổi dữ liệu giữa các thiết bị.

6.2 IoT Gateway (AI Inference Node)

Trong hệ thống này, chúng tôi sử dụng một **laptop** để thay thế cho thiết bị IoT nhúng truyền thống. Laptop đảm nhận vai trò suy luận AI và giao tiếp với các thiết bị phần cứng cũng như hệ thống backend:

- Chạy mô hình nhận diện khuôn mặt để xác thực người dùng (ví dụ: mở cửa).
- Chạy mô hình điều khiển bằng giọng nói để nhận lệnh từ người dùng (bật/tắt thiết bị,...).
- Giao tiếp:
 - **Serial** với các thiết bị phần cứng như cảm biến, module relay, LCD,...
 - **MQTT** với Adafruit IO và FastAPI để truyền thông tin điều khiển và trạng thái thiết bị.
 - **HTTP/HTTPS** để gửi dữ liệu suy luận và nhật ký về server.
- Do có cấu hình tốt hơn thiết bị nhúng, laptop giúp việc suy luận mô hình AI nhanh chóng và chính xác hơn.

6.3 FastAPI (Backend)

Là trung tâm điều phối logic và giao tiếp của toàn hệ thống, **FastAPI** đảm nhiệm các chức năng chính sau:

- **Nhận dữ liệu từ IoT Gateway và Adafruit IO:** Hệ thống sử dụng giao thức **MQTT (publish/subscribe)** để giao tiếp với Adafruit IO.
 - Một **client MQTT duy nhất** được triển khai theo **Singleton Pattern** để đảm bảo chỉ có một kết nối duy nhất quản lý việc đăng ký (subscribe) và gửi dữ liệu (publish) đến các kênh (feed) trên Adafruit IO.
 - Dữ liệu từ các feed được xử lý linh hoạt bằng **Factory Pattern**, cho phép tạo tự động các *handler* phù hợp theo từng loại dữ liệu (trạng thái cửa, nhiệt độ, độ ẩm, giọng nói,...), giúp hệ thống dễ mở rộng và bảo trì.
- **Lưu trữ dữ liệu vào PostgreSQL:** Tất cả dữ liệu thu thập được từ Adafruit IO hoặc IoT Gateway sẽ được ghi nhận vào cơ sở dữ liệu **PostgreSQL**, bao gồm thông tin trạng thái thiết bị, sự kiện khuôn mặt, lệnh giọng nói, v.v.
- **Cung cấp API cho frontend React thông qua HTTP và WebSocket:**
 - Các API **HTTP (RESTful)** dùng để truy vấn dữ liệu (ví dụ: lịch sử nhận diện, danh sách thiết bị,...) và gửi lệnh điều khiển từ frontend hoặc cấu hình hệ thống.
 - **WebSocket** được dùng để truyền dữ liệu thời gian thực từ FastAPI đến frontend. Khi có dữ liệu mới từ IoT Gateway (trạng thái thiết bị, nhận diện khuôn mặt, cảm biến,...), FastAPI lập tức đẩy thông tin qua WebSocket, giúp giao diện React cập nhật tức thì mà không cần polling.
- **Xác thực và phân quyền bằng JWT:** Hệ thống sử dụng **JSON Web Token (JWT)** để xác thực và phân quyền người dùng. Mỗi người dùng khi đăng nhập sẽ nhận được một access token có thời gian sống (TTL) nhất định.

- **Quản lý phiên đăng nhập bằng Redis:** Khi người dùng đăng xuất, token sẽ được đưa vào danh sách **blacklist** lưu trữ trong **Redis**. Với Redis hỗ trợ TTL, token sẽ tự động bị xóa khỏi blacklist sau khi hết hạn, đảm bảo an toàn và tiết kiệm bộ nhớ.

6.4 ReactJS (Frontend)

Là ứng dụng web cung cấp giao diện tương tác cho người dùng, với các chức năng chính:

- **Hiển thị trạng thái thiết bị theo thời gian thực:** React kết nối với FastAPI thông qua WebSocket để nhận dữ liệu cập nhật liên tục từ các thiết bị IoT. Trạng thái thiết bị, cảm biến, sự kiện giọng nói, và nhận diện khuôn mặt đều được hiển thị tức thời.
- **Gửi lệnh điều khiển đến FastAPI:** Các thao tác điều khiển của người dùng (như bật/tắt thiết bị, mở cửa,...) sẽ được gửi về backend thông qua HTTP hoặc WebSocket, từ đó backend sẽ xử lý và gửi lệnh điều khiển qua MQTT đến Adafruit IO.

6.5 Adafruit IO (Cloud MQTT Broker)

- **Adafruit IO** là nền tảng điện toán đám mây hỗ trợ giao tiếp theo giao thức **MQTT**, đóng vai trò như một **Cloud MQTT Broker** trung gian giúp truyền và nhận dữ liệu giữa các thiết bị IoT và hệ thống backend từ khoảng cách xa.
- Nền tảng này cho phép các thiết bị nhúng hoặc gateway tại nhà xuất bản dữ liệu cảm biến (nhiệt độ, độ ẩm, trạng thái thiết bị,...) lên các kênh dữ liệu gọi là **feed**, đồng thời backend có thể đăng ký (subscribe) để nhận dữ liệu mới một cách chủ động và thời gian thực.
- Một ưu điểm nổi bật của Adafruit IO là khả năng tích hợp sẵn giao diện **dashboard trực quan**, cho phép người dùng dễ dàng giám sát, hiển thị trạng thái thiết bị, biểu đồ cảm biến,... thông qua trình duyệt mà không cần lập trình thêm.
- Việc sử dụng Adafruit IO giúp giảm tải cho hệ thống backend trong việc xử lý các kết nối trực tiếp từ thiết bị, đồng thời đảm bảo tính ổn định khi vận hành ở môi trường Internet công cộng hoặc từ xa.

6.6 PostgreSQL (Cơ sở dữ liệu)

- **PostgreSQL** là hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, được sử dụng để lưu trữ toàn bộ dữ liệu có cấu trúc của hệ thống. Nhờ khả năng mở rộng, hỗ trợ ACID và truy vấn mạnh mẽ, PostgreSQL rất phù hợp cho các hệ thống IoT có tính chất thời gian thực và yêu cầu bảo mật dữ liệu.
- Dữ liệu được lưu trữ trong PostgreSQL bao gồm:

- **Thông tin người dùng:** tên đăng nhập, mật khẩu đã mã hóa, vai trò (admin, thành viên), token xác thực, cài đặt cá nhân,...
- **Quyền truy cập:** quản lý phân quyền truy cập từng chức năng trong hệ thống, đảm bảo tính riêng tư và bảo mật giữa các nhóm người dùng khác nhau.
- **Nhật ký hệ thống:** lưu lại toàn bộ hoạt động của người dùng và thiết bị như:
 - * Dữ liệu nhận diện khuôn mặt, bao gồm thời điểm, ảnh khuôn mặt, và thiết bị nhận diện.
 - * Các lệnh điều khiển từ giọng nói hoặc giao diện người dùng.
 - * Trạng thái thiết bị được cập nhật theo thời gian (mở/đóng cửa, đèn bật/tắt,...).
 - * Thông tin môi trường như nhiệt độ, độ ẩm từ cảm biến gửi về.
- Ngoài việc lưu trữ, PostgreSQL còn được sử dụng để truy vấn phân tích dữ liệu, hỗ trợ frontend hiển thị biểu đồ, lịch sử hoạt động hoặc báo cáo sự kiện bất thường.

User	Notification
+ id Int (unique, required)	+ id Int (unique, required)
+ username String (unique, required)	+ member_id Int (unique, required)
+ password String (required, hash)	+ message String (unique, required)
+ DOB Date	+ created_at DateTime (required)
+ level Enum(admin, member) (required)	+ is_read bool (required)

Device	AccessRecord
+ id Int (unique, required)	+ id Int (unique, required)
+ name String (unique, required)	+ userid Int (unique, required)
+ type String (required)	+ timestamp DateTime (required)
+ ada_feed String (required)	+ dangerous bool (required)

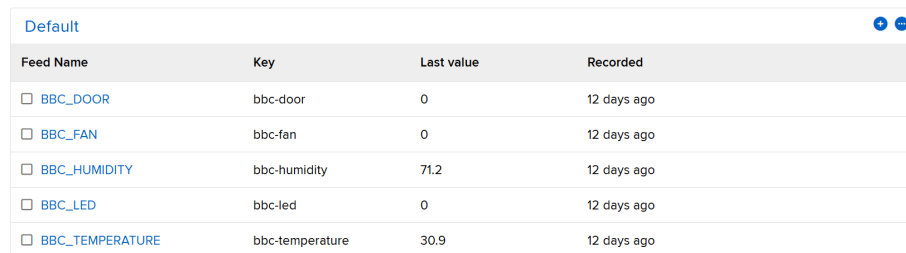
Hình 13: Database Design

7 Thực thi với Adafruit

7.1 Các feed chính

Với project lần này nhóm chúng em quyết định sẽ tạo các feeds sau:

- BBC_DOOR: lưu và giữ các tín hiệu mở cửa hệ thống.
- BBC_FAN: lưu và giữ các tín hiệu điều khiển quạt
- BBC_HUMIDITY: lưu và giữ các tín hiệu độ ẩm của hệ thống.
- BBC_LED: lưu và giữ các tín hiệu tín độ đèn của hệ thống.
- BBC_TEMPERATURE: lưu và giữ các tín hiệu nhiệt độ của hệ



Feed Name	Key	Last value	Recorded
<input type="checkbox"/> BBC_DOOR	bbc-door	0	12 days ago
<input type="checkbox"/> BBC_FAN	bbc-fan	0	12 days ago
<input type="checkbox"/> BBC_HUMIDITY	bbc-humidity	71.2	12 days ago
<input type="checkbox"/> BBC_LED	bbc-led	0	12 days ago
<input type="checkbox"/> BBC_TEMPERATURE	bbc-temperature	30.9	12 days ago

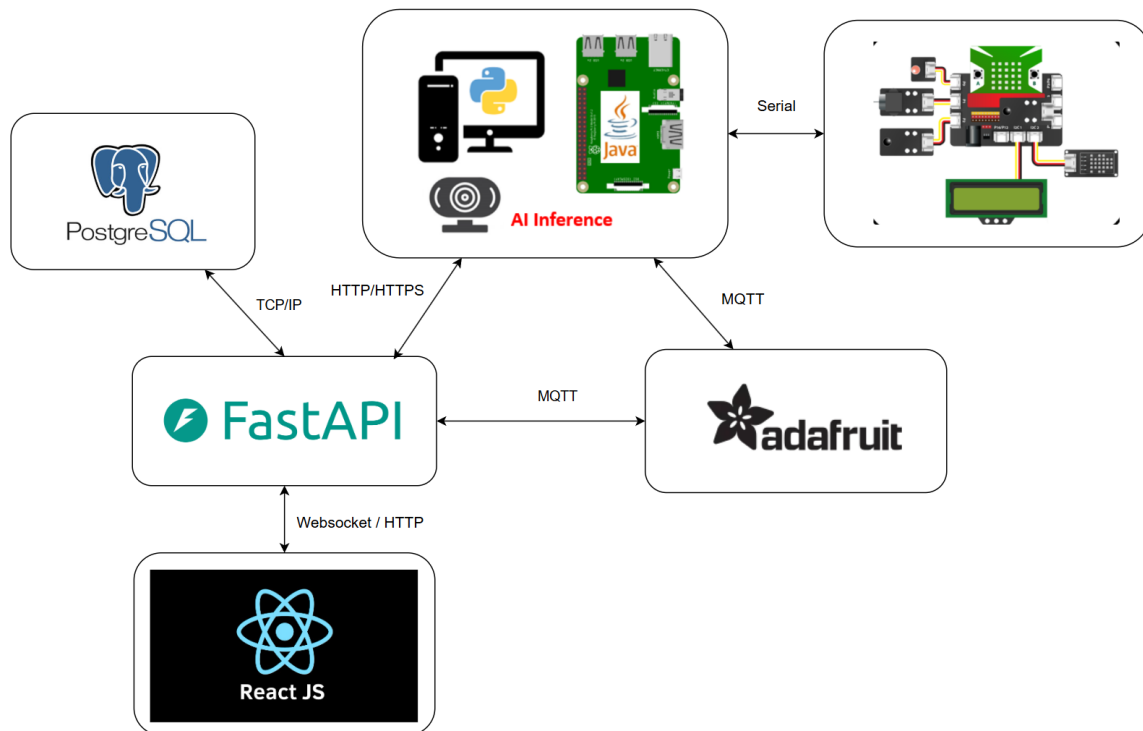
Hình 14: Các feed của project

Các feed như BBC_DOOR, BBC_FAN và BBC_LED được tạo ra như vai trò trung gian để điều khiển các thiết bị tương ứng thông qua web và IoTGateway có tích hợp AI, có thể điều khiển thủ công cũng như tự động thông qua các models AI thông minh.

7.2 Các giao thức chính

Trong đề tài này, hai giao thức truyền thông quan trọng được sử dụng là HTTP và MQTT, đóng vai trò hỗ trợ cho hệ thống AIoT. HTTP được ứng dụng để xây dựng giao diện web, giúp người dùng tương tác trực tiếp với hệ thống thông qua trình duyệt. Với tính phổ biến và dễ triển khai, HTTP là lựa chọn phù hợp cho việc truyền tải dữ liệu phi thời gian thực và truy cập thông tin từ server. Ngược lại, MQTT được sử dụng để giao tiếp giữa các thiết bị IoT nhờ khả năng truyền dữ liệu nhẹ, ổn định và hiệu quả trong môi trường mạng không ổn định. Việc kết hợp cả hai giao thức cho phép hệ thống vừa đảm bảo tính trực quan cho người dùng cuối (qua giao diện web), vừa đảm bảo hiệu năng cao trong thu thập và truyền nhận dữ liệu thời gian thực từ các thiết bị AIoT.

7.3 Hiện thực IotGateway



Hình 15: Hệ thống đề xuất

Việc lựa chọn kiến trúc hệ thống IoT có gateway mang lại nhiều lợi ích vượt trội so với mô hình không sử dụng gateway. Trước hết, gateway đóng vai trò trung gian, giúp chuyển đổi giao thức (ví dụ từ MQTT sang HTTP), hỗ trợ tích hợp nhiều loại thiết bị IoT sử dụng các giao thức khác nhau. Ngoài ra, gateway còn giúp xử lý sơ bộ dữ liệu tại biên (edge computing), giảm tải cho hệ thống trung tâm và cải thiện độ trễ trong truyền thông. Đặc biệt trong môi trường mạng không ổn định hoặc băng thông hạn chế, gateway có thể lưu trữ tạm thời dữ liệu và đồng bộ khi kết nối được phục hồi, đảm bảo tính liên tục của hệ thống. So với kiến trúc không có gateway, mô hình này tăng tính bảo mật và khả năng mở rộng, từ đó phù hợp hơn cho các ứng dụng AIoT yêu cầu thu thập và xử lý dữ liệu theo thời gian thực với độ tin cậy cao.

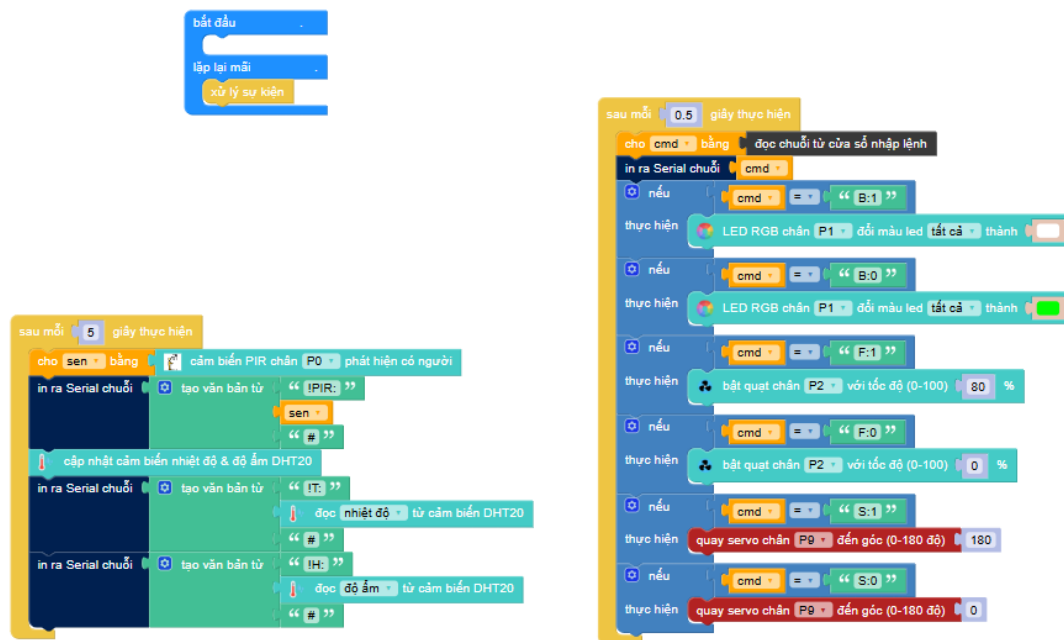
```

1 import serial.tools.list_ports
2 import random
3 import time
4 import sys
5 from Adafruit_IO import MQTTClient
6
7 AIO_FEED_ID = ["BBC_TEMPERATURE", "BBC_HUMIDITY", "BBC_LED", "BBC_FAN",
8               "BBC_DOOR"]
9
10 AIO_USERNAME = ""
  
```

11 AIO_KEY = ""

Listing 1: Kết nối với Adafruit IO và định nghĩa các feed

Việc quan trọng đầu tiên trong việc xây dựng *gateway* là việc xây dựng các kênh *feed* dữ liệu cùng với quyền truy cập (*access*) vào nền tảng Adafruit. Để xác định được AIO_USERNAME và AIO_KEY, hãy xem lại phần ??



Hình 16: Set up thiết bị bằng Ohstem App

```

1 def connected(client):
2     print("Ket noi thanh cong...")
3     for v in AIO_FEED_ID:
4         client.subscribe(v)
5
6 def subscribe(client, userdata, mid, granted_qos):
7     print("Subscribe thanh cong...")
8
9 def disconnected(client):
10    print("Ngat ket noi...")
11    sys.exit(1)
12
13 def message(client, feed_id, payload):
14     if feed_id == "BBC_FAN":
15         print("Nhan data tu BBC_FAN: ", payload)
16         if payload == "1":
17             sendSerial('F:1')
18         else:
19             sendSerial('F:0')
20
21     if feed_id == "BBC_LED":
22         print("Nhan data tu BBC_LED: ", payload)

```

```
23     if payload == "1":
24         sendSerial("B:1")
25     else:
26         sendSerial("B:0")
27
28     if feed_id == 'BBC_DOOR':
29         print('Nhan data tu BBC_DOOR: ', payload)
30         if payload == '1':
31             sendSerial('S:1')
32         else:
33             sendSerial('S:0')
34
35 client = MQTTClient(AIO_USERNAME, AIO_KEY)
36 client.on_connect = connected
37 client.on_disconnect = disconnected
38 client.on_message = message
39 client.on_subscribe = subscribe
40 client.connect()
41 client.loop_background()
```

Listing 2: Chương trình xử lý MQTT và điều khiển thiết bị qua UART

Bước đầu tiên chúng ta cần phải subscribe vào các kênh feed đã được đề cập ở trên thông qua hàm `connected` đồng thời để có thể gửi dữ liệu nếu có bất cứ tín hiệu nào từ web thì hệ thống sẽ lấy từ kênh feed trên Adafruit thông qua `def message` và gửi tín hiệu qua `sendSerial` mà chúng ta sẽ định nghĩa sau.

Để không có sai sót khi gửi dữ liệu đến thiết bị chúng ta phải xác định feed nào đang gửi yêu cầu bật thiết bị. Chẳng hạn như nếu feed `BBC_FAN` đang gửi yêu cầu bật quạt với tín hiệu bật sẽ là 1 và tắt là 0, để thiết bị có thể hiểu được ta sẽ định nghĩa lệnh gửi đến thiết bị như `F:1` hay `F:0` tương ứng với yêu cầu bật và tắt.

Quan sát lại ảnh thiết bị 17 ta có thể thấy cứ 0.5s thiết bị sẽ bắt lệnh qua `serial` mà ta sẽ cài đặt sau và sẽ thực hiện lệnh tương ứng với command mà ta sẽ gửi thông qua `sendSerial`. Ví dụ nếu ta gửi `F:1` thì serial sẽ bắt lệnh và mở quạt.

```
1 def processData(data, client):
2     global anyOneThere
3     data = data.replace("!", "")
4     data = data.replace("#", "")
5     splitData = data.split(":")
6
7     if splitData[0] == "T":
8         client.publish(AIO_FEED_ID[0], splitData[1])
9     if splitData[0] == "H":
10        client.publish(AIO_FEED_ID[1], splitData[1])
11
12 mess = ""
13 def readSerial(client):
14     bytesToRead = ser.inWaiting()
15     if (bytesToRead > 0):
16         global mess
17         mess = mess + ser.read(bytesToRead).decode("UTF-8")
18         while ("#" in mess) and ("!" in mess):
19             start = mess.find("!")
20             end = mess.find("#")
```

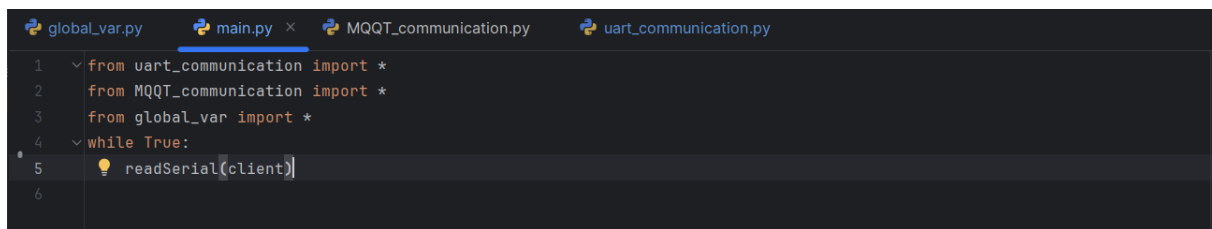
```

21         print(mess)
22         processData(mess[start:end + 1], client)
23         if (end == len(mess)):
24             mess = ""
25         else:
26             mess = mess[end+1:]
27
28 def sendSerial(payload):
29     global ser
30     if isYoloBitConnect and ser.is_open:
31         ser.write(str(payload).encode())
32         print("success")
33     else:
34         print(isYoloBitConnect)
35         print("fail to send data: Serial not ready")

```

Listing 3: Hiện thực UART

Chúng ta định nghĩa các tín hiệu được gửi từ thiết bị đến gateway với format sau (!T:value#) cứ mỗi 5s thiết bị sẽ gửi tín hiệu một lần. Tương ứng T (nhiệt độ), H (độ ẩm), PIR (sensor phát hiện người). Ở gateway ta bắt tín hiệu bằng hàm `readSerial`, sau khi bắt tín hiệu ta sẽ xử lý tín hiệu tương ứng với `processData` và sẽ tín hiệu đến Adafruit.



Hình 17: Danh sách các files gateway

Ở việc hiện thực gateway ta sẽ chia code theo từng files riêng lẻ để dễ dàng quản lý cũng như debug.

- `global_var`: file chứa các biến toàn cục như danh sách các feed.
- `MQQT_communication`: file chứa cách thức kết nối Adafruit cũng như lấy dữ liệu từ Adafruit.
- `uart_communication`: file định nghĩa serial và xử lý dữ liệu từ thiết bị và publish lên Adafruit.
- `main`: file chứa vòng loop liên tục đọc dữ liệu từ serial. Nếu mỗi lần có tín hiệu từ thiết bị gateway sẽ xử lý.

```

1 !T:30.9# !H:70# !PIR:True#
2 Nhan data tu BBC_FAN: 1
3 Dang gui: F:1

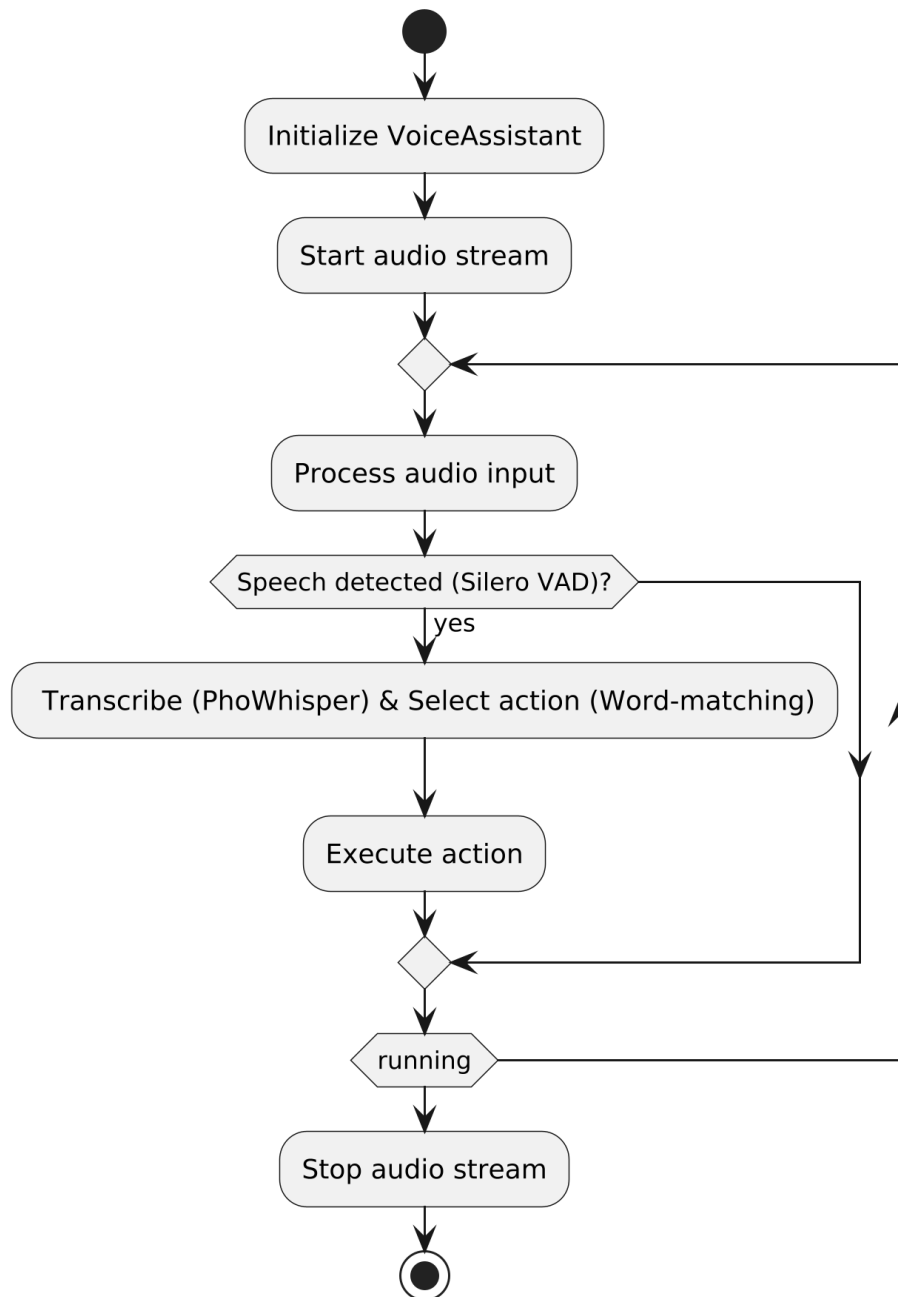
```

Listing 4: Ví dụ về cách giao tiếp

Với dòng đầu tiên trong terminal ta có thể thấy gateway đã đọc tín hiệu từ thiết bị với định dạng mà ta define lúc trước. Đồng thời tín hiệu bật quạt từ Adafruit được gửi đến và sẽ được chuyển tiếp đến thiết bị thông qua gateway.

8 Trở lý ảo giọng nói

Trợ lý ảo giọng nói (Voice Assistant) đã trở thành một công nghệ phổ biến, giúp người dùng tương tác với thiết bị thông minh một cách tự nhiên thông qua giọng nói. Phần này sẽ trình bày về các phần chính trong mô đun trợ lý ảo giọng nói, bao gồm mô hình phát hiện giọng nói, chuyển giọng nói thành văn bản, xác định hành động (bằng Word-matching và mô hình ngôn ngữ lớn), và kết nối với cơ sở dữ liệu.



Hình 18: Luồng xử lý của mô đun trợ lý ảo giọng nói

8.1 Mô hình phát hiện giọng nói (Voice activity detection)

Mô hình phát hiện giọng nói (Voice Activity Detection - VAD) chịu trách nhiệm xác định liệu trong luồng âm thanh đầu vào có chứa giọng nói hay không. Quá trình này giúp hệ thống chỉ bắt đầu ghi âm và xử lý khi có giọng nói, tiết kiệm tài nguyên và tăng độ chính xác của việc nhận diện.

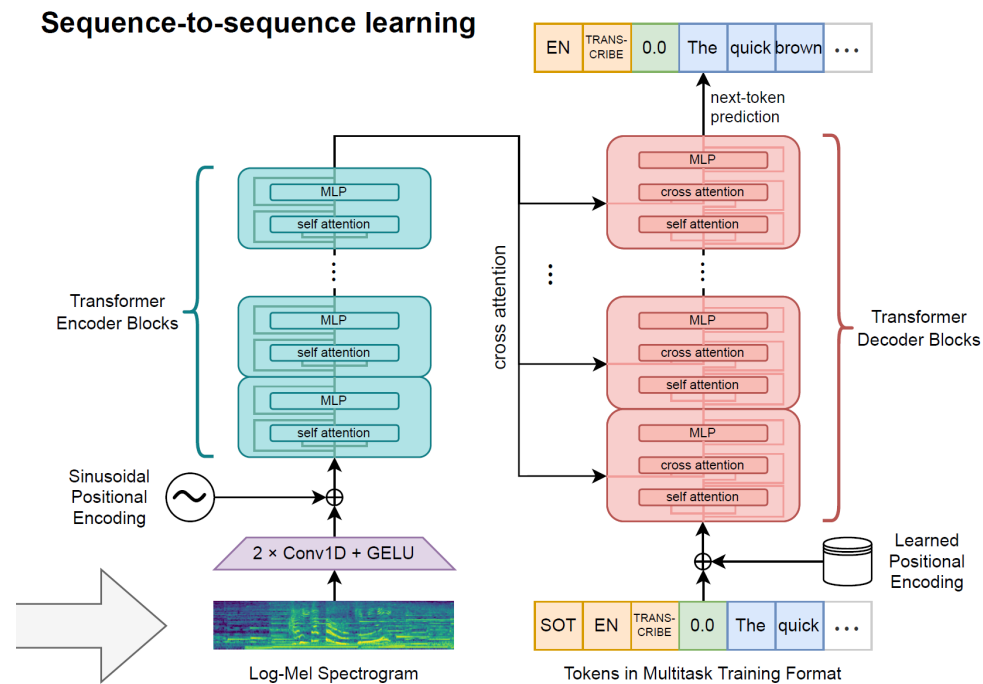
Mô hình được sử dụng cho hệ thống là Silero VAD [1]. Mô hình là một mạng neural network dựa trên cơ chế multi head attention [2], với đầu vào là các feature từ short-time Fourier transform.

Mô hình này được chọn vì nó đáp ứng các tiêu chí:

- Chất lượng cao: Precision và Recall cao
- Độ trễ thấp: Mô hình có thể xử lý với các đoạn âm thanh chỉ dài 30ms với thời gian xử lý chỉ 1ms trên CPU.
- Mô hình đã được huấn luyện trên 13 ngàn giờ dữ liệu âm thanh, hỗ trợ hơn 100 ngôn ngữ.

8.2 Mô hình chuyển giọng nói thành văn bản tiếng Việt (Speech to text)

PhoWhisper [3] là một mô hình nhận dạng giọng nói tự động (Automatic speech recognition - ASR) cho tiếng Việt, được phát triển bởi VinAI Research. Mô hình này được Fine-tuning từ Whisper[4], một mô hình ASR đa ngôn ngữ mạnh mẽ của OpenAI, trên một tập dữ liệu lớn tiếng Việt với tổng cộng 844 giờ thu âm. PhoWhisper có khả năng xử lý tốt nhiều giọng địa phương khác nhau của tiếng Việt đến từ 63 tỉnh thành trên cả nước.



Hình 19: Kiến trúc của Whisper

Whisper được xây dựng theo kiến trúc Transformers[2] với đầu vào là âm thanh đã được chuyển thành biểu đồ phổ Log-Mel. Whisper gồm 2 khối chính là Encoder và Decoder, với Encoder cung cấp thông tin trích xuất đặc trưng cho Decoder thông qua cross attention, và Decoder kết hợp thông tin từ Encoder và các Token tín hiệu đầu vào và các Token trước đó để dự đoán ra các Token tiếp theo trong văn bản.

PhoWhisper có năm phiên bản với kích thước khác nhau (tiny, base, small, medium, large), cho phép người dùng lựa chọn mô hình phù hợp với yêu cầu về hiệu suất và tài nguyên tính toán. Mô hình này đã được công khai và có thể dễ dàng tích hợp vào các ứng dụng thực tế thông qua thư viện Transformers của Hugging Face.

Nhóm sử dụng PhoWhisper-medium để nhận diện giọng nói vì trong lúc thí nghiệm PhoWhisper-medium cho ra kết quả tương đối chính xác với độ trễ chấp nhận được.

8.3 Xác định hành động bằng Word-matching hoặc LLM

Phương pháp so khớp từ khóa (Word-matching) xác định hành động bằng cách kiểm tra sự xuất hiện của các từ khóa đặc trưng trong câu lệnh sau khi đã được chuyển đổi thành văn bản. Dựa trên tập hợp các từ khóa được định nghĩa trước, hệ thống tiến hành đối chiếu với nội dung câu lệnh để suy luận hành động tương ứng.

Ưu điểm:

- Đơn giản và dễ triển khai: Phương pháp này không yêu cầu nhiều tài nguyên tính toán hay dữ liệu huấn luyện.
- Thời gian xử lý nhanh: Vì không cần các bước phân tích ngôn ngữ phức tạp, việc

xác định hành động diễn ra gần như tức thì.

Nhược điểm:

- Độ linh hoạt thấp: Phương pháp này khó xử lý các biến thể ngôn ngữ, đồng nghĩa với việc người dùng phải nói đúng với các từ khóa được định nghĩa trước.
- Dễ nhầm lẫn trong ngữ cảnh phức tạp: Nếu một từ khóa xuất hiện trong câu không nhằm mục đích thể hiện hành động, hệ thống vẫn có thể nhận diện sai.
- Không hiểu được ngữ nghĩa: Phương pháp chỉ dựa trên sự xuất hiện của từ, không đánh giá được ý nghĩa thực sự của câu lệnh.

Sự bùng nổ của các mô hình ngôn ngữ lớn (Large Language Models – LLM) như ChatGPT đã cho thấy LLM có thể hiểu ngôn ngữ gần như con người. Do đó, LLM trở thành công cụ lý tưởng trong các bài toán yêu cầu hiểu và phân loại ngôn ngữ tự nhiên. Tuy nhiên, việc sử dụng các mô hình lớn như ChatGPT qua nền tảng đám mây thường gặp các hạn chế về độ trễ, chi phí và yêu cầu bảo mật dữ liệu. Do đó, nhóm đã tiến hành thử nghiệm ứng dụng LLM chạy local thông qua nền tảng Ollama [5], sử dụng các mô hình LLM mã nguồn mở nhằm nâng cao độ chính xác và khả năng mở rộng trong phân loại lệnh điều khiển thiết bị.

Phương pháp được triển khai gồm việc xây dựng prompt ứng với các hành động, liệt kê danh sách hành động hợp lệ và các ví dụ minh họa. Mô hình sẽ nhận đầu vào là câu lệnh từ người dùng, phân tích nội dung và trả về một hành động tương ứng hoặc "unknown" nếu không khớp với bất kỳ hành động nào đã định nghĩa. Ví dụ, prompt có thể như sau:

```
You are an AI model tasked with classifying user's command into one of the following actions:  
- "turn_on_light": Turn on the light  
- "turn_off_light": Turn off the light  
- "turn_on_fan": Turn on the fan  
- "turn_off_fan": Turn off the fan  
- "unknown": If the user's command does not match any of the defined actions.
```

Classification Rules:

1. If the user's command clearly refers to an action, return the appropriate action.
2. If user's command refers to any other device or topic, return only "unknown". Do not attempt to generalize.
3. You must not guess or infer new actions beyond the listed above.
4. Return only one of the predefined keywords without explanation.

User's command input and desired output examples:

User's command -> Desired output:

"bật đèn đi" -> "turn_on_light"

"hãy tắt đèn đi" -> "turn_off_light"

"bật quạt ngay" -> "turn_on_fan"
"hãy tắt quạt ngay" -> "turn_off_fan"
"bạn khoẻ không?" -> "unknown"
"vui lòng bật đèn đi" -> "turn_on_light"
"làm ơn tắt đèn ngay" -> "turn_off_light"
"bật quạt đi" -> "turn_on_fan"
"hãy tắt quạt đi" -> "turn_off_fan"
"hôm nay thời tiết thế nào?" -> "unknown"

User's command: "bật sáng đèn"

Desired output:

Nhóm cũng tiến hành khảo sát độ chính xác của phương pháp sử dụng LLM so với phương pháp so từ khoá trên bằng việc phân loại 10 hành động khác nhau với 50 câu lệnh của người dùng.

Model	Độ chính xác	Thời gian chạy (s)
Word-matching	78.00%	0.01
smollm2 (1.7B) [6]	96.00%	2.73
llama3.2 (3B) [7]	98.00%	2.87
phi4-mini (3.8B) [8]	96.00%	4.35
qwen2.5 (7B) [9]	100.00%	4.48
llama3.1 (8B) [7]	100.00%	5.47
gemma3 (4B) [10]	100.00%	6.55

Bảng 6: Kết quả thí nghiệm của các mô hình mã nguồn mở khác nhau, với các kí hiệu nB là n tỉ tham số, thí nghiệm được chạy trên máy tính có GPU RTX 4070

Ưu điểm: Linh hoạt, dễ mở rộng, có thể hiểu các biến thể ngôn ngữ tự nhiên, ví dụ như người dùng có thể nói không đúng từ khoá

Nhược điểm: Yêu cầu phần cứng để có thể chạy mô hình LLM là khá cao, máy tính cần có GPU để có thể chạy LLM với độ trễ thấp.

8.4 Kết nối với cơ sở dữ liệu

MQTT với Adafruit IO: Sử dụng MQTTClient từ Adafruit_IO để publish giá trị điều khiển tới các feed.

9 Hiện thực phần nhận diện gương mặt

9.1 Các thư viện cần sử dụng

Để xây dựng hệ thống nhận diện gương mặt sử dụng trí tuệ nhân tạo, dự án sử dụng một số thư viện quan trọng trong Python, phục vụ cho việc xử lý ảnh, huấn luyện mô hình học sâu, và trực quan hóa kết quả. Dưới đây là danh sách các thư viện chính cùng phiên bản được sử dụng:

- **torch==2.0.1+cu118**: Thư viện nền tảng cho việc xây dựng và huấn luyện các mô hình học sâu.
- **torchvision==0.15.2+cu118**: Thư viện mở rộng của PyTorch, hỗ trợ các hàm xử lý ảnh và mô hình tiền huấn luyện phục vụ cho thị giác máy tính.
- **torchaudio==2.0.2+cu118**: Dù không sử dụng trực tiếp trong xử lý ảnh, thư viện này đi kèm để đảm bảo tính tương thích đầy đủ với hệ sinh thái PyTorch.
- **facenet-pytorch==2.5.2**: Thư viện quan trọng sử dụng mô hình FaceNet để trích xuất đặc trưng khuôn mặt và thực hiện so sánh khoảng cách embedding.
- **opencv-python==4.8.1.78**: Thư viện xử lý ảnh mạnh mẽ, hỗ trợ đọc, hiển thị và thao tác trực tiếp trên ảnh và video.
- **numpy==1.24.4**: Thư viện toán học cơ bản, hỗ trợ các thao tác tính toán ma trận và xử lý dữ liệu số hiệu quả.
- **Pillow==9.5.0**: Thư viện thao tác ảnh đơn giản, hỗ trợ đọc, ghi và chuyển đổi định dạng ảnh.
- **matplotlib==3.7.1**: Thư viện trực quan hóa dữ liệu, được dùng để vẽ biểu đồ kết quả, loss, và accuracy trong quá trình huấn luyện.
- **scikit-learn==1.3.2**: Thư viện hỗ trợ các thuật toán học máy truyền thống và công cụ đánh giá mô hình (như confusion matrix, accuracy, v.v.).
- **tqdm==4.66.1**: Thư viện tạo thanh tiến trình trong vòng lặp huấn luyện, giúp theo dõi trạng thái mô hình một cách trực quan.

Việc sử dụng đúng phiên bản thư viện giúp đảm bảo tính tương thích và ổn định trong suốt quá trình phát triển và triển khai hệ thống.

9.2 Trích xuất đặc trưng gương mặt

Sau khi người dùng đã đăng ký ảnh gương mặt, hệ thống sẽ tiến hành trích xuất đặc trưng (embedding) từ các ảnh này để phục vụ cho việc so sánh, nhận diện sau này. Việc trích xuất được thực hiện bằng mô hình **InceptionResnetV1** từ thư viện **facenet-pytorch**, với trọng số được huấn luyện sẵn trên tập dữ liệu **VGGFace2**.

Tăng cường dữ liệu (Data Augmentation)

Trước khi trích xuất embedding, mỗi ảnh đầu vào sẽ được áp dụng một số phép biến đổi nhằm tạo ra các phiên bản dữ liệu phong phú hơn, giúp mô hình học được các đặc trưng ổn định và giảm hiện tượng overfitting. Các phép biến đổi bao gồm:

- Lật ngang ảnh (horizontal flip)
- Thay đổi độ sáng, tương phản (color jitter)
- Xoay ảnh ngẫu nhiên (rotation)
- Làm mờ Gaussian
- Cắt ngẫu nhiên và thay đổi kích thước (resized crop)

Mỗi ảnh đầu vào sẽ được tăng cường thành 5 phiên bản ngẫu nhiên.

Quy trình trích xuất

Với mỗi người dùng, chương trình sẽ duyệt qua các ảnh trong thư mục cá nhân, thực hiện tăng cường và trích xuất embedding cho từng ảnh. Sau đó, tất cả các vector embedding sẽ được trung bình hoá để tạo thành một vector đại diện duy nhất cho người đó.

```
1 def extract(name="empty", dataset_path="data"):  
2     model = InceptionResnetV1(pretrained='vggface2').eval()  
3  
4     transform = transforms.Compose([  
5         transforms.Resize((160, 160)),  
6         transforms.ToTensor()  
7     ])  
8  
9     output_path = "embeddings"  
10    os.makedirs(output_path, exist_ok=True)  
11  
12    for person in os.listdir(dataset_path):  
13        if (person == name):  
14            person_dir = os.path.join(dataset_path, person)  
15            if not os.path.isdir(person_dir):  
16                continue  
17  
18            all_embeddings = []  
19  
20            for img_file in os.listdir(person_dir):  
21                img_path = os.path.join(person_dir, img_file)  
22                try:  
23                    img = Image.open(img_path).convert('RGB')  
24                    versions = augment(img, num_augments=5)  
25
```

```
26         for version_tensor in versions:
27             if isinstance(version_tensor, Image.Image)
28             :
29                 version_tensor = transform(
30                     version_tensor)
31                 emb = model(version_tensor.unsqueeze(0)).
32                     detach().squeeze()
33                 all_embeddings.append(emb)
34
35             except Exception as e:
36                 print(f"[ERROR] Error with {img_path}: {e}")
37
38         if all_embeddings:
39             avg_embedding = torch.stack(all_embeddings).mean(
40                 dim=0)
41             data = {
42                 "name": person,
43                 "embedding": avg_embedding.tolist()
44             }
45
46             json_path = os.path.join(output_path, f"{person}.
47             json")
48             with open(json_path, 'w') as f:
49                 json.dump(data, f)
50
51             print(f"[INFO] Saved embedding for {person} (from
52                 {len(all_embeddings)} augments)")
```

Vector đặc trưng được lưu ở định dạng JSON, gồm tên người và mảng số thực 512 chiều. Các embedding này sẽ được sử dụng trong bước nhận diện để tính khoảng cách giữa gương mặt cần nhận và các vector đã lưu trữ.

9.3 Đăng ký khuôn mặt

Trong quá trình vận hành hệ thống, người dùng thực hiện thao tác đăng ký khuôn mặt thông qua giao diện web. Hình ảnh từ máy khách (client) sẽ được gửi lên máy chủ thông qua API, sau đó module trích xuất đặc trưng sẽ lấy các hình ảnh để tiến hành xử lý và lưu vào cơ sở dữ liệu.

Quy trình này giúp xây dựng cơ sở dữ liệu khuôn mặt ban đầu một cách linh hoạt, hỗ trợ triển khai hệ thống nhận diện trong các ứng dụng thực tế như điểm danh, kiểm soát ra vào, hoặc xác thực danh tính người dùng.

9.4 Nhận diện gương mặt thời gian thực

Sau khi đã xây dựng được cơ sở dữ liệu vector đặc trưng cho từng người, hệ thống tiến hành nhận diện khuôn mặt thời gian thực từ camera. Quy trình này bao gồm các bước chính sau:

1. Phát hiện khuôn mặt từ khung hình webcam sử dụng MTCNN.
2. Trích xuất embedding từ khuôn mặt mới bằng InceptionResnetV1.
3. So sánh embedding này với tất cả các embedding đã lưu bằng độ tương đồng cosine.
4. Xác định danh tính nếu độ tương đồng vượt ngưỡng cho trước, ngược lại gán nhãn "Unknown".

Mã nguồn thực hiện nhận diện

```
1 def faceReg():
2     device = 'cuda' if torch.cuda.is_available() else 'cpu'
3     mtcnn = MTCNN(image_size=160, margin=20, keep_all=True, device
4     =device, post_process=True)
5     resnet = InceptionResnetV1(pretrained='vggface2').eval().to(
6     device)
7
8     embedding_data = load_embeddings("embeddings")
9     for name in embedding_data:
10         embedding_data[name] = embedding_data[name].to(device) /
11         embedding_data[name].norm(p=2)
12
13     confidence_threshold = 0.88
14     cap = cv2.VideoCapture(0)
15     unknown_count = 0
16
17     while True:
18         ret, frame = cap.read()
19         if not ret:
20             break
21
22         img = Image.fromarray(cv2.cvtColor(frame, cv2.
23         COLOR_BGR2RGB))
24         faces = mtcnn(img)
25
26         if faces is not None:
27             for i, face in enumerate(faces):
28                 emb = resnet(face.unsqueeze(0).to(device)).detach
29                 ().squeeze(0)
30                 emb = emb / emb.norm(p=2)
31
32                 best_match = "Unknown"
33                 best_confidence = 0.0
34
35                 for name, saved_emb in embedding_data.items():
36                     similarity = torch.nn.functional.
37                     cosine_similarity(
38                         emb.unsqueeze(0), saved_emb.unsqueeze(0)
39                     ).item()
```

```
34         confidence = (similarity + 1) / 2
35
36         if confidence > best_confidence:
37             best_confidence = confidence
38             best_match = name
39
40         if best_confidence >= confidence_threshold:
41             print(f'[INFO] {best_match}')
42             return best_match
43         else:
44             unknown_count += 1
45
46     if unknown_count > 2:
47         print('[INFO] Unknown face detected and saved.')
48         return 'Unknown'
49
50 cap.release()
51 cv2.destroyAllWindows()
```

Chi tiết kỹ thuật

- **MTCNN** được sử dụng để phát hiện và cắt chính xác vùng khuôn mặt từ ảnh đầu vào.
- **InceptionResnetV1** thực hiện trích xuất vector đặc trưng 512 chiều từ ảnh khuôn mặt.
- **So sánh cosine similarity**: tính độ tương đồng giữa hai vector theo công thức:

$$\text{cosine_similarity}(a, b) = \frac{a \cdot b}{\|a\| \cdot \|b\|}$$

sau đó chuẩn hóa về thang điểm từ 0 đến 1:

$$\text{confidence} = \frac{\text{similarity} + 1}{2}$$

- Nếu độ tin cậy (**confidence**) vượt ngưỡng 0.88, hệ thống xác định được người dùng; ngược lại gán nhãn **Unknown**.

Hàm này hoạt động thời gian thực, có thể tích hợp vào các hệ thống điểm danh, kiểm soát truy cập, hoặc xác thực người dùng bằng khuôn mặt.

Trường hợp Unknown và nhiều gương mặt trong webcam

Xử lý gương mặt không xác định (Unknown): Khi hệ thống không thể tìm thấy bất kỳ embedding nào có độ tương đồng cao hơn ngưỡng đã định (ví dụ 0.88), khuôn mặt sẽ được gán nhãn **Unknown**. Trong trường hợp này, hệ thống sẽ thực hiện hai hành động:

- Lưu ảnh hiện tại từ webcam vào thư mục riêng biệt (ví dụ: `Unknown/Unknown_n.jpg`) để phục vụ kiểm tra sau này.
- Gửi thông tin ảnh và cảnh báo lên nền tảng web nhằm thông báo cho người quản lý hoặc chủ nhà về sự xuất hiện của người lạ.

Xử lý nhiều gương mặt cùng lúc: Khi webcam phát hiện đồng thời nhiều gương mặt trong một khung hình, hệ thống sẽ tiến hành:

1. Trích xuất embedding cho từng khuôn mặt.
2. So sánh tất cả embedding với cơ sở dữ liệu.
3. Nếu **ít nhất một người** có độ tương đồng cao hơn ngưỡng cho phép, hệ thống sẽ nhận diện người đó và trả về kết quả nhận diện **thay vì Unknown**.

Cách xử lý này đảm bảo hệ thống có thể nhận diện đúng trong các tình huống phổ biến như: nhiều người xuất hiện cùng lúc, hoặc có người lạ đi cùng người quen.

9.5 Giao tiếp với hệ thống

Khi có tính hiệu "PIR" được trả về từ cảm biến người của mạch hệ thống sẽ bắt đầu chạy để mở webcam (như trong video demo). Sau đó hệ thống sẽ về tên của người đang đứng trước webcam (hoặc Unknown) thông qua hàm `send_name`.

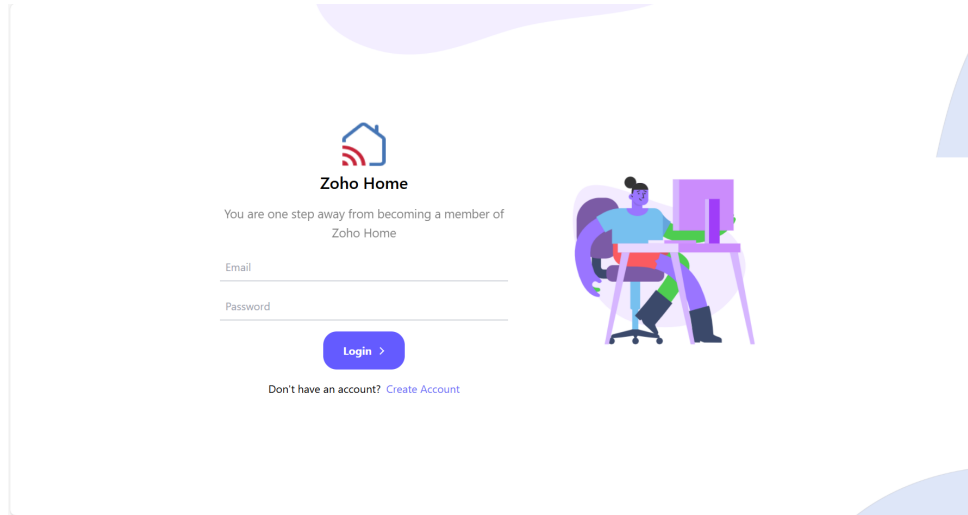
```
1 def send_name(name: str, server_url: str = "http
  ://192.168.206.158:8000/record"):
2     data = {"username": name}
3     response = requests.post(server_url, json=data)
4     print("Status code:", response.status_code)
5     print("Data:", data)
```

Bên cạnh việc gửi thông tin về website hệ thống cũng sẽ điều khiển mạch để mở cửa (hoặc không) thông qua tính hiệu serial:

```
1 if splitData[0] == "PIR":
2     if splitData[1] == "True":
3         ok = faceReg()
4         if ok != 'Unknown':
5             print('[INFO] OK')
6             sendSerial("S:1")
7             send_name(ok)
8         else:
9             print('[INFO] Not OK')
10            sendSerial("S:0")
11            send_name("Unknown")
```


10 Mockup

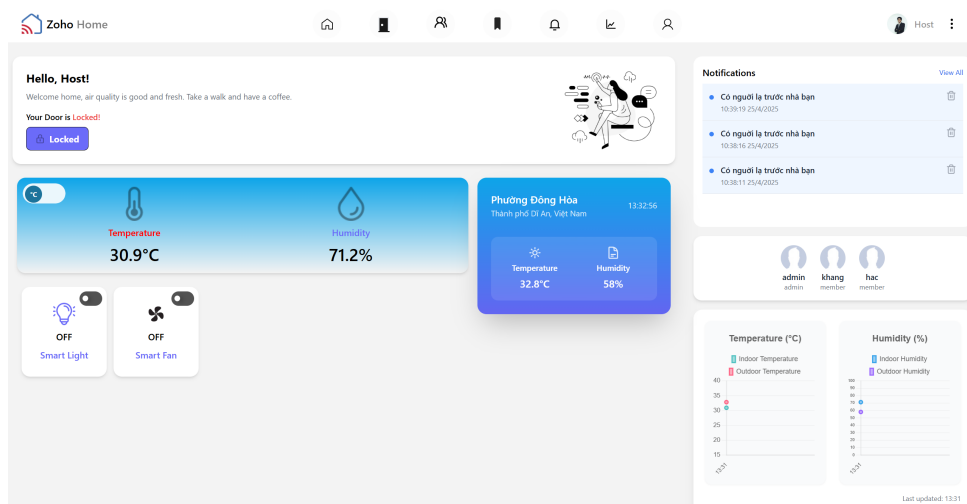
10.1 Trang đăng nhập



Hình 20: Giao diện trang đăng nhập

Trang đăng nhập của ZOHO Home gồm 2 trường (email và password) cho người dùng nhập thông tin vào để đăng nhập tới ứng dụng. Sau khi nhấn nút "Login", hệ thống sẽ xác thực với dữ liệu trong database, nếu đúng thì sẽ chuyển tới trang chủ chính, nếu nhập sai thông tin sẽ có thông báo và yêu cầu nhập lại thông tin

10.2 Trang chủ chính

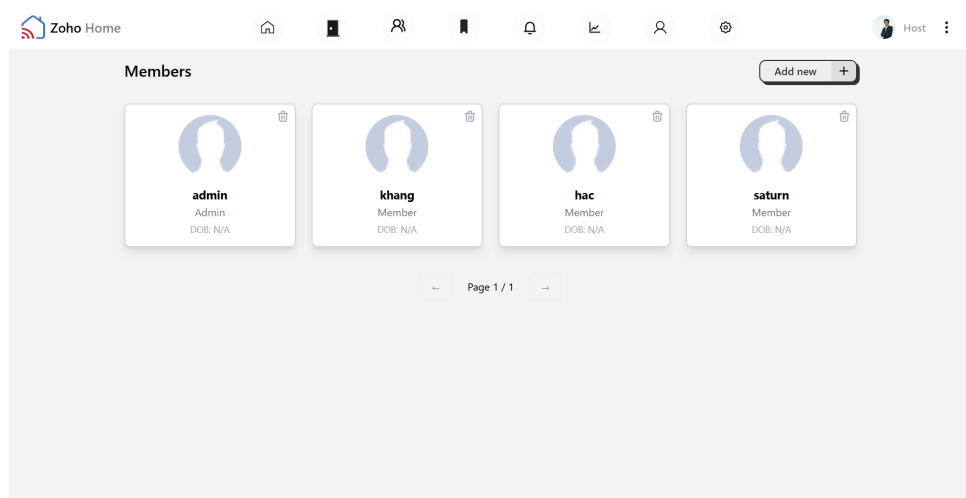


Hình 21: Giao diện trang chủ chính

Trang chủ chính của ZOHO Home hiển thị trạng thái cửa hiện tại, và nút điều khiển cho phép chủ nhà đóng hoặc mở cửa thông qua ứng dụng. Bên cạnh đó còn có thông tin

về nhiệt độ và độ ẩm được nhận từ cảm biến và hiển thị lên trang web, có thể chuyển đổi qua lại giữa độ F và độ C tùy ý. Ngoài ra trên trang chủ chính còn có thông tin vị trí, thời gian, nhiệt độ và độ ẩm ngoài trời hiện tại của nhà để chủ nhà có thể tham khảo. Trên trang chủ chính cũng có hiện thông tin của các thành viên trong nhà và còn có thanh điều hướng để đi tới các trang chức năng khác của web. Cuối cùng sẽ có biểu đồ tương quan nhiệt độ và độ ẩm hiện tại theo thời gian hiển thị ở góc màn hình được cập nhật liên tục theo từng phút.

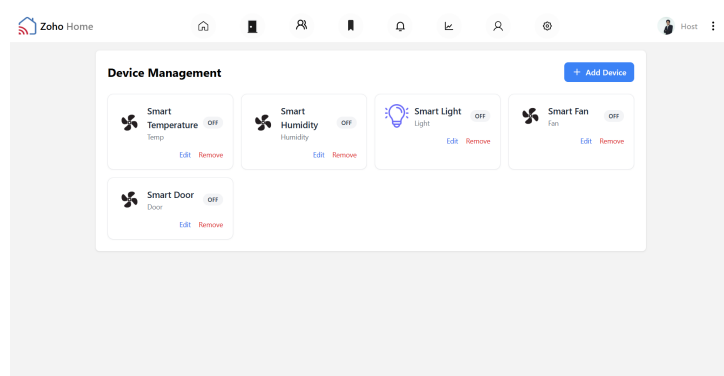
10.3 Trang quản lý thành viên



Hình 22: Giao diện trang quản lý thành viên

Trang thêm thành viên của ZOHO Home hiển thị danh sách thành viên với ảnh, tên, và vai trò. Mỗi trang sẽ hiển thị tối đa 8 thành viên và có thể qua trang tiếp theo để xem các thành viên khác thông qua nút bấm. Trên trang này chủ nhà còn có thể thêm thành viên vào trong danh sách để sử dụng cho việc nhận diện khuôn mặt

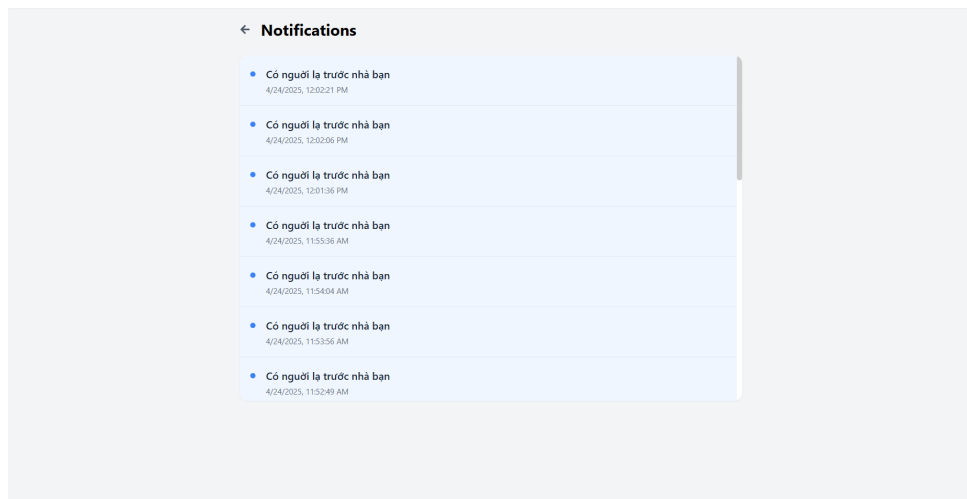
10.4 Trang quản lý thiết bị



Hình 23: Giao diện trang quản lý thiết bị

Giao diện quản lý thiết bị liệt kê các thiết bị thông minh trong hệ thống Zoho Home, bao gồm cảm biến nhiệt, cảm biến độ ẩm, đèn, và quạt thông minh, tất cả đều ở trạng thái "OFF". Mỗi thiết bị có các tùy chọn "Edit" và "Remove", cùng nút "Add Device" để mở rộng hệ thống. Hình ảnh phản ánh khả năng quản lý thiết bị linh hoạt.

10.5 Trang xem toàn bộ thông báo



Hình 24: Giao diện trang toàn bộ thông báo

Giao diện trang xem toàn bộ thông báo sẽ hiển thị danh sách thông báo từ hệ thống Zoho Home, với các thông điệp khi có người lạ hoặc người quen vào nhà kèm thời gian cụ thể khi nhận được. Các thông báo được hiển thị theo thứ tự thời gian, hỗ trợ người dùng theo dõi các sự kiện quan trọng.

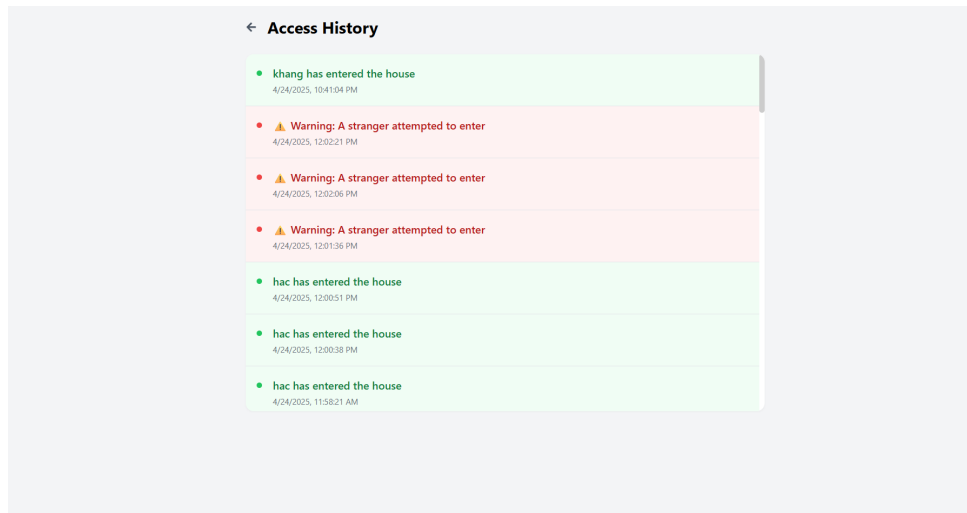
10.6 Trang xem lịch sử ra vào nhà

Trang "Access History" ghi lại lịch sử truy cập vào nhà, với các sự kiện được phân loại thành thành công (màu xanh lá) và cảnh báo (màu đỏ). Các sự kiện như khi có người lạ vào nhà và cảnh báo được ghi lại với thời gian cụ thể. Dữ liệu này cung cấp thông tin bảo mật chi tiết, hữu ích để phân tích hành vi ra vào và phát hiện các mối đe dọa.

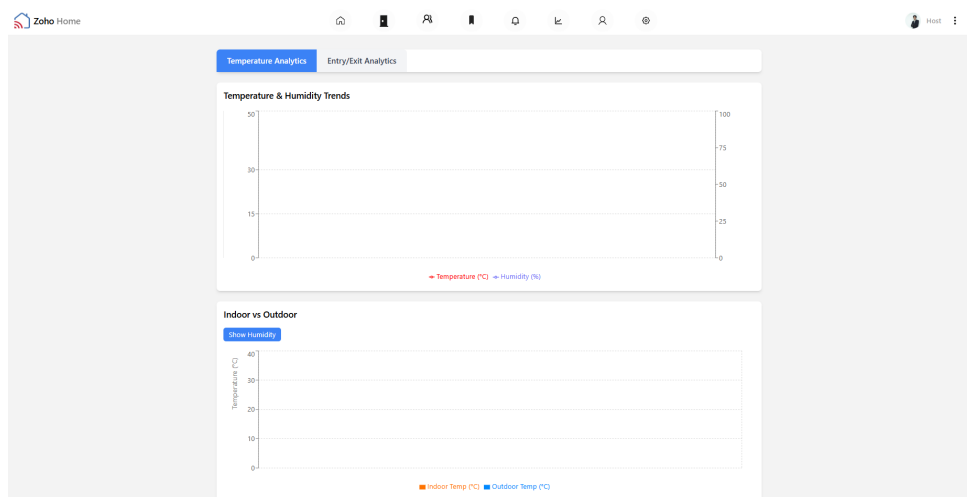
10.7 Trang biểu đồ

Trang cung cấp giao diện của biểu đồ xu hướng nhiệt độ và độ ẩm từ hệ thống Zoho Home, nơi các biểu đồ thể hiện xu hướng nhiệt độ (đơn vị °C) và độ ẩm theo thời gian. Tuy nhiên, biểu đồ hiện tại không hiển thị dữ liệu cụ thể, cho thấy có thể không có dữ liệu được ghi nhận hoặc cần cấu hình thêm cảm biến để thu thập thông tin. Giao diện này cho phép người dùng theo dõi môi trường trong nhà, hỗ trợ phân tích và tối ưu hóa điều kiện sống.

Biểu đồ trong nhà và ngoài nhà từ Zoho Home so sánh nhiệt độ trong nhà (Indoor Temp)



Hình 25: Giao diện trang lịch sử ra vào nhà



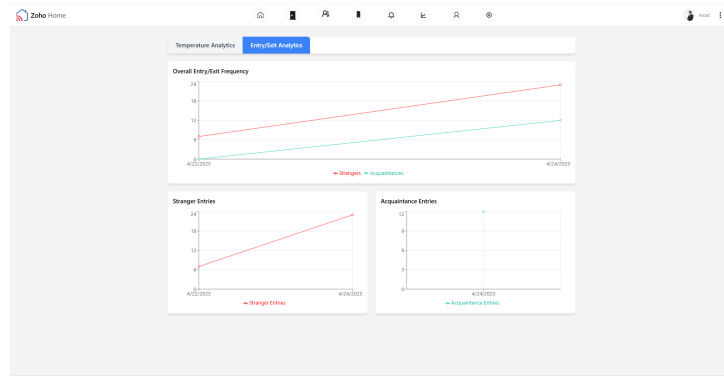
Hình 26: Biểu đồ tương quan nhiệt độ và độ ẩm trong và ngoài nhà

và ngoài trời (Outdoor Temp) theo thời gian. Với tùy chọn "Show Humidity" để chuyển sang hiển thị độ ẩm. Hệ thống tích hợp dữ liệu từ các cảm biến nhiệt độ trong và ngoài nhà để cung cấp thông tin hữu ích cho người dùng.

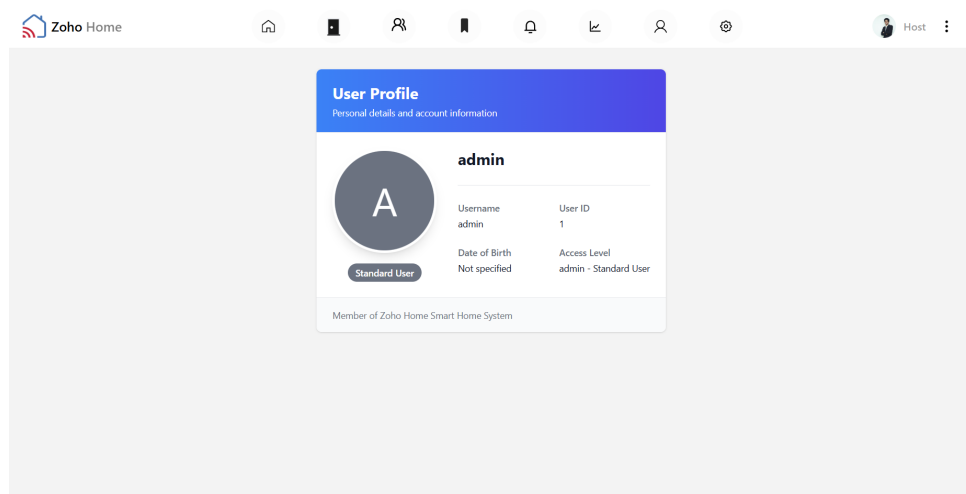
Biểu đồ biến động số lượng người lạ/ quen ra vào nhà từ Zoho Home phân tích tần suất ra vào nhà, chia thành hai nhóm: "Strangers" (màu đỏ) và "Acquaintances" (màu xanh lá), trong khoảng thời gian chi tietes. Biểu đồ cho thấy số lần ra vào của người lạ và người quen, cung cấp dữ liệu hữu ích để đánh giá an ninh và lưu lượng ra vào.

10.8 Trang thông tin cá nhân

Giao diện "User Profile" hiển thị thông tin cá nhân và tài khoản của người dùng với vai trò , bao gồm tên người dùng, ngày sinh, và cấp độ truy cập . Hình ảnh thể hiện khả năng quản lý người dùng trong hệ thống, hỗ trợ phân quyền và cá nhân hóa trải nghiệm.



Hình 27: Biểu đồ biến động số lượng người quen / lạ vào nhà theo



Hình 28: Giao diện trang hồ sơ người dùng

11 Lưu trữ code - Demo

- Đường dẫn tới source code của ứng dụng web: <https://github.com/kan3103/smart-home-web.git>
- Đường dẫn tới source code của IOT Gateway: <https://github.com/dangkhoale11/IoTGateway>
- Đường dẫn tới source code của module Điều khiển giọng nói: <https://github.com/minhharry/VoiceAssistant/blob/main/VoiceAssistant.py>
- Đường dẫn tới source code của module Nhận diện khuôn mặt: <https://github.com/toanhac/FaceRegDADN>
- Đường dẫn tới video demo: <https://drive.google.com/file/d/1GWDkoRTw4L81ytHneeW8oS4XB/view?usp=sharing>

Tài liệu tham khảo

- [1] S. Team, “Silero vad: pre-trained enterprise-grade voice activity detector (vad), number detector and language classifier.” <https://github.com/snakers4/silero-vad>, 2024.
- [2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2023.
- [3] T.-T. Le, L. T. Nguyen, and D. Q. Nguyen, “Phowhisper: Automatic speech recognition for vietnamese,” 2024.
- [4] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” 2022.
- [5] Ollama, “Ollama: Run large language models locally,” 2025. Accessed: 2025-05-10.
- [6] L. B. Allal, A. Lozhkov, E. Bakouch, G. M. Blázquez, G. Penedo, L. Tunstall, A. Marafioti, H. Kydlicek, A. P. Lajarín, V. Srivastav, J. Lochner, C. Fahlgren, X.-S. Nguyen, C. Fourrier, B. Burtenshaw, H. Larcher, H. Zhao, C. Zakka, M. Morlon, C. Raffel, L. von Werra, and T. Wolf, “Smollm2: When smol goes big – data-centric training of a small language model,” 2025.
- [7] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, C. Keller, C. Touret, C. Wu, C. Wong, C. C. Ferrer, C. Nikolaidis, D. Allonsius, D. Song, D. Pintz, D. Livshits, D. Wyatt, D. Esiobu, D. Choudhary, D. Mahajan, D. Garcia-Olano, D. Perino, D. Hupkes, E. Lakomkin, E. AlBadawy, E. Lobanova, E. Dinan, E. M. Smith, F. Radenovic, F. Guzmán, F. Zhang, G. Synnaeve, G. Lee, G. L. Anderson, G. Thattai, G. Nail, G. Mialon, G. Pang, G. Cucurell, H. Nguyen, H. Korevaar, H. Xu, H. Touvron, I. Zarov, I. A. Ibarra, I. Kloumann, I. Misra, I. Evtimov, J. Zhang, J. Copet, J. Lee, J. Geffert, J. Vranes, J. Park, J. Mahadeokar, J. Shah, J. van der Linde, J. Billock, J. Hong, J. Lee, J. Fu, J. Chi, J. Huang, J. Liu, J. Wang, J. Yu, J. Bitton, J. Spisak, J. Park, J. Rocca, J. Johnstun, J. Saxe, J. Jia, K. V. Alwala, K. Prasad, K. Upasani, K. Plawiak, K. Li, K. Heafield, K. Stone, K. El-Arini, K. Iyer, K. Malik, K. Chiu, K. Bhalla, K. Lakhotia, L. Rantala-Yeary, L. van der Maaten, L. Chen, L. Tan, L. Jenkins, L. Martin, L. Madaan, L. Malo, L. Blecher, L. Landzaat, L. de Oliveira, M. Muzzi, M. Pasupuleti, M. Singh, M. Paluri, M. Kardas, M. Tsimpoukelli, M. Oldham, M. Rita, M. Pavlova, M. Kambadur, M. Lewis, M. Si, M. K. Singh, M. Hassan, N. Goyal, N. Torabi, N. Bashlykov, N. Bogoychev, N. Chatterji, N. Zhang, O. Duchenne, O. Çelebi, P. Alrassy, P. Zhang, P. Li, P. Vasic, P. Weng, P. Bhargava, P. Dubal, P. Krishnan, P. S. Koura, P. Xu, Q. He, Q. Dong, R. Srinivasan, R. Ganapathy, R. Calderer, R. S. Cabral, R. Stojnic, R. Raileanu, R. Maheswari, R. Girdhar, R. Patel, R. Sauvestre, R. Polidoro, R. Sumbaly, R. Taylor, R. Silva, R. Hou, R. Wang, S. Hosseini, S. Chennabasappa, S. Singh, S. Bell, S. S. Kim, S. Edunov, S. Nie, S. Narang,

S. Raparthy, S. Shen, S. Wan, S. Bhosale, S. Zhang, S. Vandenhende, S. Batra, S. Whitman, S. Sootla, S. Collot, S. Gururangan, S. Borodinsky, T. Herman, T. Fowler, T. Sheasha, T. Georgiou, T. Scialom, T. Speckbacher, T. Mihaylov, T. Xiao, U. Karn, V. Goswami, V. Gupta, V. Ramanathan, V. Kerkez, V. Gonguet, V. Do, V. Vogeti, V. Albiero, V. Petrovic, W. Chu, W. Xiong, W. Fu, W. Meers, X. Martinet, X. Wang, X. Wang, X. E. Tan, X. Xia, X. Xie, X. Jia, X. Wang, Y. Goldschlag, Y. Gaur, Y. Babaei, Y. Wen, Y. Song, Y. Zhang, Y. Li, Y. Mao, Z. D. Coudert, Z. Yan, Z. Chen, Z. Papakipos, A. Singh, A. Srivastava, A. Jain, A. Kelsey, A. Shajnfeld, A. Gangidi, A. Victoria, A. Goldstand, A. Menon, A. Sharma, A. Boesenberg, A. Baevski, A. Feinstein, A. Kallet, A. Sangani, A. Teo, A. Yunus, A. Lupu, A. Alvarado, A. Caples, A. Gu, A. Ho, A. Poulton, A. Ryan, A. Ramchandani, A. Dong, A. Franco, A. Goyal, A. Saraf, A. Chowdhury, A. Gabriel, A. Bharambe, A. Eisenman, A. Yazdan, B. James, B. Maurer, B. Leonhardt, B. Huang, B. Loyd, B. D. Paola, B. Paranjape, B. Liu, B. Wu, B. Ni, B. Hancock, B. Wasti, B. Spence, B. Stojkovic, B. Gamido, B. Montalvo, C. Parker, C. Burton, C. Mejia, C. Liu, C. Wang, C. Kim, C. Zhou, C. Hu, C.-H. Chu, C. Cai, C. Tindal, C. Feichtenhofer, C. Gao, D. Civin, D. Beaty, D. Kreymer, D. Li, D. Adkins, D. Xu, D. Testuggine, D. David, D. Parikh, D. Liskovich, D. Foss, D. Wang, D. Le, D. Holland, E. Dowling, E. Jamil, E. Montgomery, E. Presani, E. Hahn, E. Wood, E.-T. Le, E. Brinkman, E. Arcaute, E. Dunbar, E. Smothers, F. Sun, F. Kreuk, F. Tian, F. Kokkinos, F. Ozgenel, F. Caggioni, F. Kanayet, F. Seide, G. M. Florez, G. Schwarz, G. Badeer, G. Swee, G. Halpern, G. Herman, G. Sizov, Guangyi, Zhang, G. Lakshminarayanan, H. Inan, H. Shojanazeri, H. Zou, H. Wang, H. Zha, H. Habeeb, H. Rudolph, H. Suk, H. Aspegren, H. Goldman, H. Zhan, I. Damlaj, I. Molybog, I. Tufanov, I. Leontiadis, I.-E. Veliche, I. Gat, J. Weissman, J. Geboski, J. Kohli, J. Lam, J. Asher, J.-B. Gaya, J. Marcus, J. Tang, J. Chan, J. Zhen, J. Reizenstein, J. Teboul, J. Zhong, J. Jin, J. Yang, J. Cummings, J. Carvill, J. Shepard, J. McPhie, J. Torres, J. Ginsburg, J. Wang, K. Wu, K. H. U, K. Saxena, K. Khandelwal, K. Zand, K. Matosich, K. Veeraraghavan, K. Michelena, K. Li, K. Jagadeesh, K. Huang, K. Chawla, K. Huang, L. Chen, L. Garg, L. A, L. Silva, L. Bell, L. Zhang, L. Guo, L. Yu, L. Moshkovich, L. Wehrstedt, M. Khabsa, M. Avalani, M. Bhatt, M. Mankus, M. Hasson, M. Lennie, M. Reso, M. Groshev, M. Naumov, M. Lathi, M. Keneally, M. Liu, M. L. Seltzer, M. Valko, M. Restrepo, M. Patel, M. Vyatskov, M. Samvelyan, M. Clark, M. Macey, M. Wang, M. J. Hermoso, M. Metanat, M. Rastegari, M. Bansal, N. Santhanam, N. Parks, N. White, N. Bawa, N. Singhal, N. Egebo, N. Usunier, N. Mehta, N. P. Laptev, N. Dong, N. Cheng, O. Chernoguz, O. Hart, O. Salpekar, O. Kalinli, P. Kent, P. Parekh, P. Saab, P. Balaji, P. Rittner, P. Bontrager, P. Roux, P. Dollar, P. Zvyagina, P. Ratanchandani, P. Yuvraj, Q. Liang, R. Alao, R. Rodriguez, R. Ayub, R. Murthy, R. Nayani, R. Mitra, R. Parthasarathy, R. Li, R. Hogan, R. Battey, R. Wang, R. Howes, R. Rinott, S. Mehta, S. Siby, S. J. Bondu, S. Datta, S. Chugh, S. Hunt, S. Dhillon, S. Sidorov, S. Pan, S. Mahajan, S. Verma, S. Yamamoto, S. Ramaswamy, S. Lindsay, S. Lindsay, S. Feng, S. Lin, S. C. Zha, S. Patil, S. Shankar, S. Zhang, S. Zhang, S. Wang, S. Agarwal, S. Sajuyigbe, S. Chintala, S. Max, S. Chen, S. Kehoe, S. Satterfield, S. Govindaprasad, S. Gupta, S. Deng, S. Cho, S. Virk, S. Subramanian, S. Choudhury, S. Goldman, T. Remez, T. Glaser, T. Best, T. Koehler, T. Robinson, T. Li, T. Zhang, T. Matthews, T. Chou, T. Shaked, V. Vontimitta, V. Ajayi, V. Montanez, V. Mohan, V. S. Kumar,

- V. Mangla, V. Ionescu, V. Poenaru, V. T. Mihailescu, V. Ivanov, W. Li, W. Wang, W. Jiang, W. Bouaziz, W. Constable, X. Tang, X. Wu, X. Wang, X. Wu, X. Gao, Y. Kleinman, Y. Chen, Y. Hu, Y. Jia, Y. Qi, Y. Li, Y. Zhang, Y. Zhang, Y. Adi, Y. Nam, Yu, Wang, Y. Zhao, Y. Hao, Y. Qian, Y. Li, Y. He, Z. Rait, Z. DeVito, Z. Rosnbrick, Z. Wen, Z. Yang, Z. Zhao, and Z. Ma, “The llama 3 herd of models,” 2024.
- [8] Microsoft, :, A. Abouelenin, A. Ashfaq, A. Atkinson, H. Awadalla, N. Bach, J. Bao, A. Benhaim, M. Cai, V. Chaudhary, C. Chen, D. Chen, D. Chen, J. Chen, W. Chen, Y.-C. Chen, Y. ling Chen, Q. Dai, X. Dai, R. Fan, M. Gao, M. Gao, A. Garg, A. Goswami, J. Hao, A. Hendy, Y. Hu, X. Jin, M. Khademi, D. Kim, Y. J. Kim, G. Lee, J. Li, Y. Li, C. Liang, X. Lin, Z. Lin, M. Liu, Y. Liu, G. Lopez, C. Luo, P. Madan, V. Mazalov, A. Mitra, A. Mousavi, A. Nguyen, J. Pan, D. Perez-Becker, J. Platin, T. Portet, K. Qiu, B. Ren, L. Ren, S. Roy, N. Shang, Y. Shen, S. Singhal, S. Som, X. Song, T. Sych, P. Vaddamanu, S. Wang, Y. Wang, Z. Wang, H. Wu, H. Xu, W. Xu, Y. Yang, Z. Yang, D. Yu, I. Zabir, J. Zhang, L. L. Zhang, Y. Zhang, and X. Zhou, “Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras,” 2025.
- [9] Qwen, :, A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu, “Qwen2.5 technical report,” 2025.
- [10] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, L. Rouillard, T. Mesnard, G. Cideron, J. bastien Grill, S. Ramos, E. Yvinec, M. Casbon, E. Pot, I. Penchev, G. Liu, F. Visin, K. Kenealy, L. Beyer, X. Zhai, A. Tsitsulin, R. Busa-Fekete, A. Feng, N. Sachdeva, B. Coleman, Y. Gao, B. Mustafa, I. Barr, E. Parisotto, D. Tian, M. Eyal, C. Cherry, J.-T. Peter, D. Sinopalnikov, S. Bhupatiraju, R. Agarwal, M. Kazemi, D. Malkin, R. Kumar, D. Vilar, I. Brusilovsky, J. Luo, A. Steiner, A. Friesen, A. Sharma, A. Sharma, A. M. Gilady, A. Goedeckemeyer, A. Saade, A. Feng, A. Kolesnikov, A. Bendebury, A. Abdagic, A. Vadi, A. György, A. S. Pinto, A. Das, A. Bapna, A. Miech, A. Yang, A. Paterson, A. Shenoy, A. Chakrabarti, B. Piot, B. Wu, B. Shahriari, B. Petrini, C. Chen, C. L. Lan, C. A. Choquette-Choo, C. Carey, C. Brick, D. Deutsch, D. Eisenbud, D. Cattle, D. Cheng, D. Paparas, D. S. Sreepathihalli, D. Reid, D. Tran, D. Zelle, E. Noland, E. Huizenga, E. Kharitonov, F. Liu, G. Amirkhanyan, G. Cameron, H. Hashemi, H. Klimczak-Plucińska, H. Singh, H. Mehta, H. T. Lehari, H. Hazimeh, I. Ballantyne, I. Szpektor, I. Nardini, J. Pouget-Abadie, J. Chan, J. Stanton, J. Wieting, J. Lai, J. Orbay, J. Fernandez, J. Newlan, J. yeong Ji, J. Singh, K. Black, K. Yu, K. Hui, K. Vodrahalli, K. Greff, L. Qiu, M. Valentine, M. Coelho, M. Ritter, M. Hoffman, M. Watson, M. Chaturvedi, M. Moynihan, M. Ma, N. Babar, N. Noy, N. Byrd, N. Roy, N. Momchev, N. Chauhan, N. Sachdeva, O. Bunyan, P. Botarda, P. Caron, P. K. Rubenstein, P. Culliton, P. Schmid, P. G. Sessa, P. Xu, P. Stanczyk, P. Tafti, R. Shivanna, R. Wu, R. Pan, R. Rokni, R. Willoughby, R. Vallu, R. Mullins, S. Jerome, S. Smoot, S. Girgin, S. Iqbal, S. Reddy, S. Sheth, S. Pöder, S. Bhatnagar, S. R. Panyam, S. Eiger,

S. Zhang, T. Liu, T. Yacovone, T. Liechty, U. Kalra, U. Evcı, V. Misra, V. Roseberry, V. Feinberg, V. Kolesnikov, W. Han, W. Kwon, X. Chen, Y. Chow, Y. Zhu, Z. Wei, Z. Egyed, V. Cotruta, M. Giang, P. Kirk, A. Rao, K. Black, N. Babar, J. Lo, E. Moreira, L. G. Martins, O. Sanseviero, L. Gonzalez, Z. Gleicher, T. Warkentin, V. Mirrokni, E. Senter, E. Collins, J. Barral, Z. Ghahramani, R. Hadsell, Y. Matias, D. Sculley, S. Petrov, N. Fiedel, N. Shazeer, O. Vinyals, J. Dean, D. Hassabis, K. Kavukcuoglu, C. Farabet, E. Buchatskaya, J.-B. Alayrac, R. Anil, Dmitry, Lepikhin, S. Borgeaud, O. Bachem, A. Joulin, A. Andreev, C. Hardin, R. Dadashi, and L. Hussenot, “Gemma 3 technical report,” 2025.