TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Nghiên cứu và xây dựng ứng dụng nhận dạng khuôn mặt qua camera tích hợp vào nhà thông minh

NGUYỄN BÁCH THẮNG
thang.nb153505@sis.hust.edu.vn

Ngành Công nghệ thông tin
Chuyên ngành Kỹ thuật máy tính

Giảng viên hướng dẫn: PGS. TS. Nguyễn Đình Thuận
Chữ ký của GVHD

Bộ môn: Kỹ thuật máy tính
Viện: Công nghệ thông tin và truyền thông

HÀ NỘI, 5/2021

ĐỀ TÀI TỐT NGHIỆP

Biểu mẫu của Đề tài/khóa luận tốt nghiệp theo qui định của viện, tuy nhiên cần đảm bảo giáo viên giao đề tài ký và ghi rõ họ và tên.

Trường hợp có 2 giáo viên hướng dẫn thì sẽ cùng ký tên.

Giáo viên hướng dẫn
Ký và ghi rõ họ tên

Lời cảm ơn

Từ những ngày đầu, Bách Khoa với tôi giống như một ly cà phê. Vị đắng chát của cà phê khi vừa đưa vào miệng chính là vị đắng khó quên nhất suốt mấy năm qua, nó chứa đựng công sức, cũng có nhiệt huyết tuổi trẻ, cũng có cả mồ hôi trên những trang giấy. Để khi cơn đắng vừa qua đi, tôi mới dần cảm thấy vị ngọt ngọt của cà phê, vị ngọt ấy dù không thể khiến con người quên đi vị đắng lúc đầu, nhưng nó xứng đáng.

Tôi xin gửi lời cảm ơn đến thầy Nguyễn Đình Thuận, một người thầy rất tận tâm trên giảng đường, một người thầy vui tính trong các tiết học nhưng cũng rất nghiêm túc và kỷ luật. Tôi hiểu và biết ơn vì thầy mong muốn chúng tôi, những sinh viên đã đồng hành cùng thầy, cố gắng để có một nền tảng trước khi bước vào đời chứ đừng để những cơn sóng gió trong đời đánh gục.

Tóm tắt nội dung đồ án

Bài toán ở đây là bài toán tích hợp sử dụng nhận dạng khuôn mặt để điều khiển các thiết bị thông minh trong nhà.

- Các vấn đề cần thực hiện bao gồm:
  - Giải quyết bài toán nhận dạng khuôn mặt trong ảnh và nhận diện khuôn mặt đó có phải là thành viên hay không
  - Viết Service xử lý ảnh có tích hợp phương pháp giải bài toán trên
  - Tìm hiểu về Home Assistant và cách giao tiếp giữa Service xử lý ảnh với Home Assistant
- Công cụ sử dụng bao gồm:
  - Một bộ điều khiển trung tâm (máy tính cấu hình không cần mạnh hoặc một Raspberry Pi hoặc Tv box)
  - Camera
  - Các thiết bị thông minh có thể kết nối với Home Assistant hoặc thay bằng các thiết bị ảo có thể tạo từ Home Assistant
- Kết quả của đồ án:
  - Đã chạy được Service để xử lý ảnh, có khả năng nhận ra những người trong dữ liệu, nhưng độ chính xác với mặt nghiêng chưa cao
  - Đã thực hiện được khi nhận diện được sẽ gọi sự kiện đến Home Assistant và điều khiển các thiết bị được tích hợp sẵn
- Tính thực tế của đồ án:
  - Một giải pháp an ninh dùng khuôn mặt với các đặc điểm như độ chính xác cao, khả năng nhận diện nhanh mà không đòi hỏi phần cứng mạnh mẽ sẽ rất được ưa chuộng.
  - Đối tượng nhắm đến chính là những cơ quan, công ty nhỏ cần kiểm soát, theo dõi nhân viên, các hộ gia đình có phòng trọ cho thuê thường rất phổ biến ở các thành phố lớn, …
  - Khả năng nhận diện khuôn mặt cũng sẽ tiện lợi hơn dùng máy chấm vân tay bởi nó có thể nhận diện nhiều người cùng một lúc trong khi máy chấm vân tay chỉ có thể nhận diện một lúc.
- Định hướng phát triển mở rộng của đồ án:
  - Tích hợp service nhận dạng khuôn mặt thành Add-on trên Home Assistant sử dụng Docker
  - Cải thiện khả năng nhận diện để nâng cao độ chính xác hơn với những mặt nghiêng, giảm bớt sự ảnh hưởng bởi các yếu tố về môi trường như ánh sáng

Sinh viên thực hiện
Ký và ghi rõ họ tên

MỤC LỤC

DANH MỤC HÌNH VẼ

CHƯƠNG 1. **GIỚI THIỆU**

1.1 Giới thiệu chung

1.1.1 Vấn đề đặt ra

Theo xu hướng phát triển để Việt Nam trở thành một đất nước công nghiệp, các công ty, xí nghiệp, cơ quan đều đã và đang áp dụng ngày càng nhiều khoa học công nghệ vào phục vụ sản xuất nhằm đẩy mạnh và tạo ra nhiều giá trị sản xuất hơn. Mà trong đó, chấm công bằng vân tay cũng thường được thấy ở các công ty, cơ quan, xí nghiệp để đảm bảo quyền lợi của các thành viên trong công ty, cơ quan cũng như tạo điều kiện để giám đốc, ban nhân sự có thể dễ dàng quản lý hơn.

Hình 1.1 Máy chấm công bằng vân tay

Hiện nay, máy chấm công bằng vân tay cũng được ứng dụng rất rộng rãi. Nhưng trong thời điểm dịch Covid-19 đang diễn biến hết sức phức tạp và là điểm nóng được quan tâm trọng điểm của nhà nước và cộng đồng, máy chấm vân tay vẫn lại trở thành vấn đề nhạy cảm, hạn chế do tiếp xúc chung với máy chấm. Nếu như có một hệ thống chấm công bằng khuôn mặt với những ưu điểm như về yêu cầu về phần cứng không cần mạnh, hoạt động độc lập tốt dù trong khi không có mạng, nhận dạng tốt và chính xác với thời gian nhận diện nhanh chóng ra đời sẽ trở thành một giải pháp rất hay và thực dụng hoàn toàn có thể hỗ trợ và thay thế cho máy chấm công bằng vân tay truyền thống.

1.1.2 Đối tượng nghiên cứu

Việc quản lý lịch làm việc của các thành viên trong một tập thể sẽ giúp việc quản lý trở nên dễ dàng hơn và giảm tải công việc cho các nhà quản lý để kịp thời đánh giá, nhắc nhở cũng như đảm bảo hiệu suất của công việc và quyền lợi của các thành viên.

Do vậy, một hệ thống chấm công bằng khuôn mặt sẽ có thể ứng dụng trong những công ty, xí nghiệp, hay trong các khu vực, toà nhà hạn chế ra vào.

Hệ thống chấm công cũng giống như một người bảo vệ có thể làm việc 24/24, góp phần tiết kiệm ngân sách cũng như gia tăng thêm tính hiệu quả và an toàn.

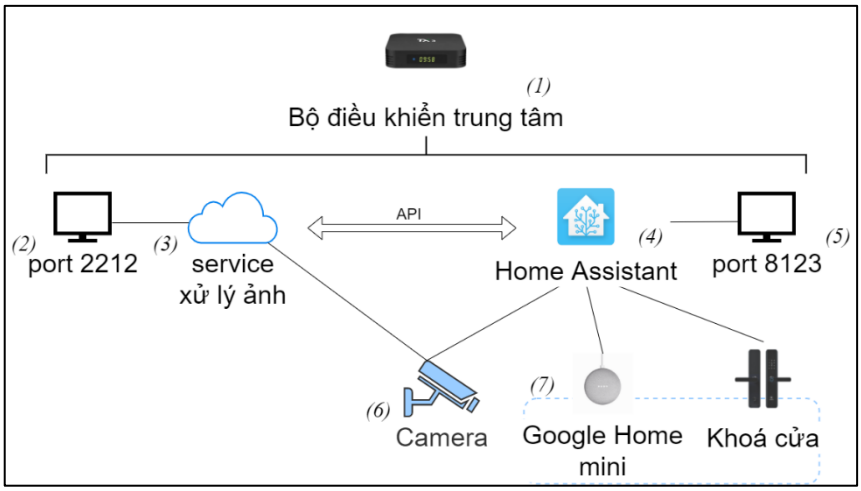
1.2 Tổng quan các vấn đề nghiên cứu liên quan

- Tìm hiểu hiểu về nền tảng nhà thông minh Home Assistant: các giao thức kết nối, các loại thiết bị, các ứng dụng thường dùng trong nhà thông minh, …
- Tìm hiểu yêu cầu phần cứng để có thể đảm nhiệm làm hệ thống trung tâm xử lý nhận dạng
- Tìm hiểu các giao thức và mô hình kết nối, truyền dữ liệu hình ảnh vào hệ thống trung tâm
- Tìm hiểu về camera an ninh, các giao thức và mô hình kết nối, gửi nhận hình ảnh
- Tìm hiểu về các mô hình thuật toán và thư viện dùng để xử lý nhận dạng khuôn mặt
- Tìm hiểu sử dụng FLASK cách viết WEB Services
- Xây dựng mô hình web services để gửi, nhận API và xử lý hình ảnh từ camera trên hệ điều hành Ubuntu
- Tìm hiểu và xây dựng module chụp ảnh từ camera, gửi đến service để xử lý và nhận dạng
- Tìm hiểu và xây dựng các kịch bản tự động hoá trên nền tảng nhà thông minh Home Assistant dựa trên kết quả nhận dạng được: tự động mở khoá cửa và thông báo nhận dạng bằng giọng nói, gửi cảnh báo người lạ đột nhập trong khung giờ nhất định

CHƯƠNG 2. **THIẾT KẾ BÀI TOÁN**

2.1 Tổng quan bài toán

2.1.1 Mô hình tổng quát



Hình 2.1 Mô hình tổng quát của hệ thống nhận dạng khuôn mặt

2.1.1.1. Các thành phần của hệ thống nhận dạng khuôn mặt bao gồm:

(1) **Bộ điều khiển trung tâm**: là một máy tính bình thường, cấu hình không cần quá mạnh, hoặc Raspberry Pi hoặc Tv box dùng để chứa Service xử lý ảnh và Home Assistant.

(2) **Port 2212**: Cổng giao tiếp của Service xử lý ảnh.

(3) **Service xử lý ảnh**: là Web Service với các tính năng dùng để chụp ảnh các thành viên trong tổ chức, mã hoá và lưu trữ lại ảnh được chụp, nhận sự kiện chụp ảnh thông qua API, phân tích và trả về kết quả nhận diện.

(4) **Home Assistant**: là một nền tảng tự động hoá ngôi nhà có mã nguồn mở, chạy trên Python 3.x, được thiết kế để dễ dàng triển khai trên bất cứ máy tính nào từ Raspberry Pi, các thiết bị lưu trữ trên mạng như NAS, thậm chí một container Docker dùng để thao dõi tình trạng thiết bị, điều khiển tất cả các thiết bị đã được tích hợp trên một giao diện duy nhất.

(5) **Port 8123**: Cổng giao tiếp của Home Assistant dùng để theo dõi và diều khiển các thiết bị được hỗ trợ

(6) **Camera**: Camera được tích hợp vào trong Home Assistant thông qua giao thức RTSP (Real Time Streaming Protocol)

(7) **Các thiết bị được điều khiển**: các thiết bị được tích hợp vào Home Assistant cho phép theo dõi và điều khiển, cụ thể ở đây là các thiết bị như Google Home Mini, khoá cửa có vân tay

2.1.1.2. Kịch bản hoạt động của các thành phần trong mô hình

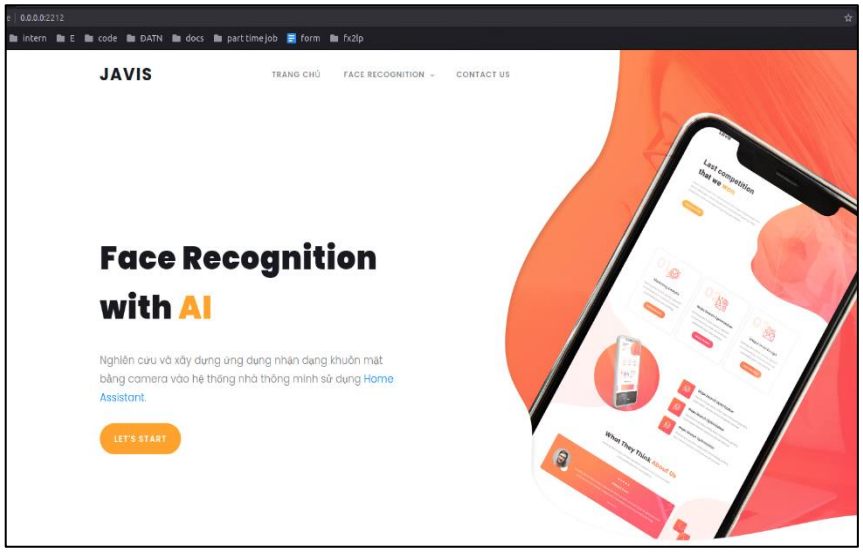
- Đứng trước cửa và nhìn vào camera.
- Khi đó, hệ thống phát hiện đã có khuôn mặt thì nó sẽ tự động gửi sự kiện yêu cầu chụp ảnh tới Service xử lý ảnh.
- Service sẽ phân tích ảnh được chụp và đưa ra kết quả dự đoán. Kết quả dự đoán sẽ được gửi về Home Assistant.
- Home Assistant sẽ dựa trên kết quả trả về để thực hiện kịch bản mở cửa nếu kết quả nhận dạng là người trong tổ chức, công ty để mở khoá cửa hoặc thực hiện cảnh báo nếu phát hiện ra người lạ

2.1.2 Tìm hiểu về Service xử lý ảnh

Serive xử lý ảnh ở đây được viết với Flask, một Framework được sử dụng để phát triển ứng dụng Web với ngôn ngữ Python phổ biến. Flask là một framework được đóng gói thành thư viện có cú pháp sử dụng đơn giản, nhanh và tiện lợi.

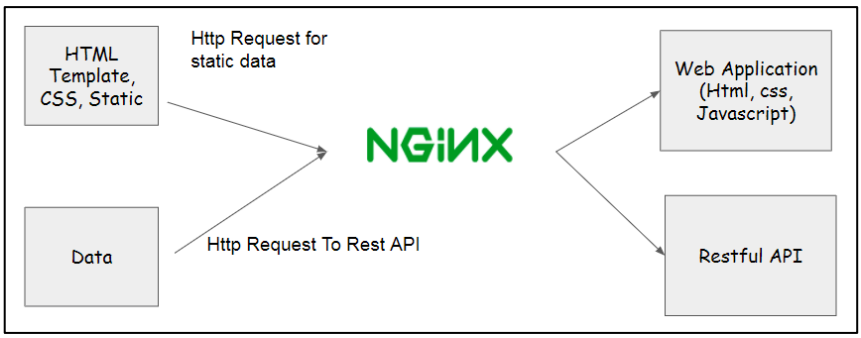
Service bao gồm các chức năng như sau:

- Chụp ảnh và lưu ảnh lại để nhận diện thành viên
- Demo nhận diện khuôn mặt realtime
- Đánh nhãn các ảnh đã chụp
- Upload ảnh đã chụp lên Database
- Cung cấp các API để điều khiển chụp ảnh, thông báo kết quả, …



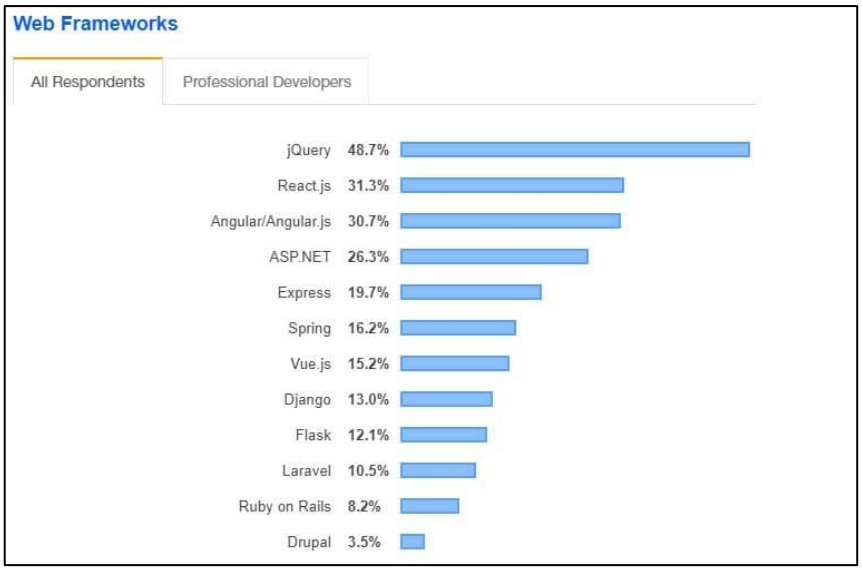
Hình 2.2 Giao diện trang chủ của Service xử lý ảnh thông qua cổng 2212

2.1.2.1. Tìm hiểu về ứng dụng Web viết bằng Flask



Hình 2.3 REST (Representational State Transfer)

Flask là một Micro Web framework viết bằng Python được phát triển bởi Armin Ronacher, có nền tảng là Werkzeug WSGI toolkit và Jinja2 template, và là một trong những framework phổ biến nhất trên Python.



Hình 2.4 Thống kê theo độ phổ biến của các Web Framework theo khảo sát của Stack Overflow trên toàn thế giới

Web Server Gateway Interface (WSGI) là một tiêu chuẩn để phát triển ứng dụng trên Python. WSGI là một kỹ thuật đặc biệt với giao diện chung cho máy chủ Web và các ứng dụng Web.

Werkzeug là bộ công cụ WSGI hỗ trợ thực hiện các request, đối tượng response và các chức năng tiện ích khác. Werkzeug là cơ sở đồng thời cho phép xây dựng Web Framework trên nó.

Jinja2 là một công cụ tạo template (khuôn mẫu) phổ biến cho Python. Một hệ thống Web Template kết hợp với cơ sở dữ liệu nhất định để tạo ra các trang Web động. [1]

Một framework bất kỳ nào cũng mang những đặc tính riêng biệt với các thế mạnh và nhược điểm khác nhau, Flask cũng như vậy.

Thế mạnh của Flask [2] là:

- Tốc độ
- Hỗ trợ cho NoQuery
- Chủ nghĩa tối giản tuyệt đối kể cả về độ phức tạp
- Không có ORM, dễ dàng kết nối với các tiện ích mở rộng
- Trình gỡ lỗi được nhúng trong trình duyệt
- Mã ngắn và đơn giản
- Ít phụ thuộc vào bên thứ 3 để tăng tính bảo mật
- Dễ học bởi sự tối giản của Flask, làm tiền đề để phát triển về sau

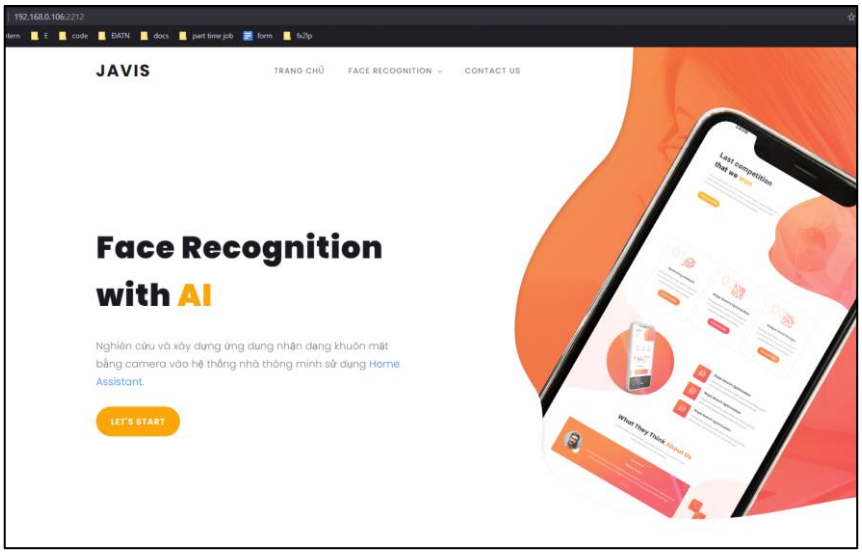
Đi kèm với thế mạnh nổi bật của Flask, các tiện ích mở rộng đôi khi yêu cầu phải tự thao tác thủ công.

Ứng dụng Web chạy bằng Flask mở trên cổng 2212:

```
 * Serving Flask app 'main' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it
   Use a production WSGI server instead.
 * Debug mode: on
INFO:werkzeug: * Restarting with stat
WARNING:root:Checking condition:
WARNING:root:- Task: Checking configration.yaml...
WARNING:root:Error: File configuration.yaml is missing
WARNING:root:- Task: Checking rest_command.yaml exist...
WARNING:root:File rest_command.yaml existed.
WARNING:root:-> File rest_command.yaml is ready to use.
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 110-577-831
WARNING:werkzeug: * Running on all addresses.
   WARNING: This is a development server. Do not use it
INFO:werkzeug: * Running on http://192.168.0.106:2212/
```

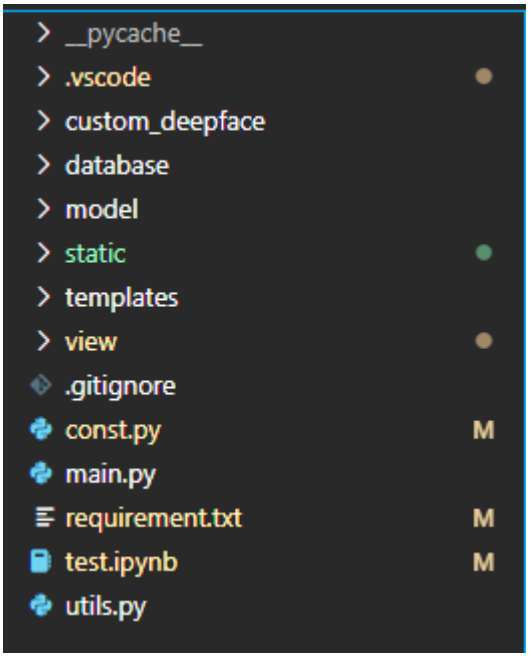
Hình 2.5 Ứng dụng chạy bằng Flask trên cổng 2212

Truy cập vào cổng 2212, trang chủ của Service xử lý ảnh tương ứng sẽ xuất hiện như sau:



Hình 2.6 Trang chủ của Service xử lý ảnh trên cổng 2212 tương ứng

2.1.2.2. Cấu trúc một chương trình Web chạy bằng Flask



Hình 2.7 Cấu trúc của một chương trình chạy bằng Flask

- Template: Thư mục mặc định dùng để chưa các giao diện web viết bằng HTML
- Static: Thư mục dùng để chứa các file static như file javascript (.js) hoặc CSS, … dùng để hỗ trợ hiển thị trang Web. Một lưu ý là Flask bắt buộc các file hình ảnh phải được đặt trong thư mục static để có thể hiện thị được.
- Khi một ứng dụng Flask được chạy, ứng dụng sẽ bắt đầu bằng cách gọi đến app.run() trong file main.py:

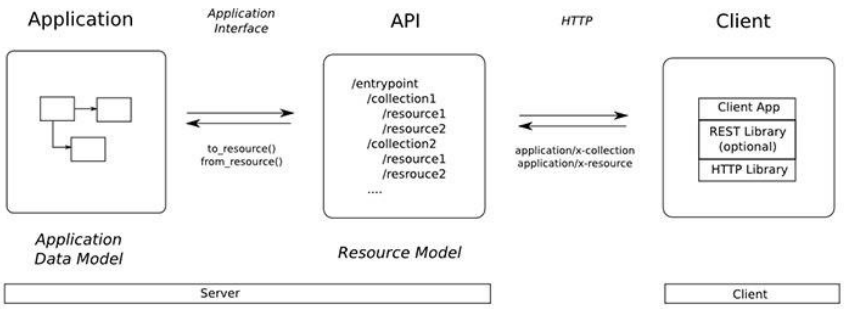
```
if __name__ == "__main__":
    check_running_condition()
    app.run(host='0.0.0.0', port=2212, debug=True)
```

Hình 2.8 Kịch bản chương trình Flask được thực thi trong Python Shell

2.1.2.3. RESTful API [3]

REST hay đầy đủ là **RE**presentational **S**tate **T**ransfer, phong cách thiết kế hệ thống phân tán siêu phương tiện (Distributed Hypermedia Systems) được giới thiệu bằng Ray Fielding năm 2000.

RESTful API là tiêu chuẩn dùng trong thiết kế các API cho các ứng dụng Web quản lý các resources. RESTful cũng là một trong những thiết kế API rất phổ biến được sử dụng nhằm thực hiện giao tiếp qua lại giữa các ứng dụng web, mobile, …, trong đó, Home Assistant cũng có hỗ trợ RESTful API.



Hình 2.9 Mô hình giao tiếp thông qua API

REST hoạt động dựa trên giao thức HTTP với các phương thức:

- GET (SELECT): Trả về một Resource hoặc một danh sách Resource.
- POST (CREATE): Tạo mới một Resource.
- PUT (UPDATE): Cập nhật thông tin cho Resource.
- DELETE (DELETE): Xoá một Resource.

RESTful API không sử dụng session và cookie mà sử dụng một Access_token với mỗi Request, dữ liệu trả về thường có dạng JSON.

Trong Flask, các API được định tuyến bằng phương thức mặc định là GET.

```
@mod.route("/take_shots")
def take_shots():
    return render_template('./take_shot.html')
```

Hình 2.10 Phương thức GET trong Flask

Khi sử dụng các phương thức khác, Flask cần được chỉ định rõ ràng trong một danh sách các phương thức được cho phép trong dòng định nghĩa API.

```
@mod.route("/readdress",  methods=['POST'])
def readdress():
    rename_list = request.args.get("rename_list")
    data = literal_eval(rename_list)
    for key, value in data.items():
        rename(key, value)
    return jsonify()
```

Hình 2.11 Phương thức POST trong Flask

2.1.3 Tìm hiểu về Home Assistant

2.1.3.4. Giới thiệu về nền tảng nhà thông minh

Ngoài AI, công nghệ blockchain, điện toán lượng tử, … thì IOT (Internet Of Things) là khái niệm được nhắc đến rất nhiều ở Việt Nam những năm gần đây. IOT cũng được xem như là xu hướng xông nghệ trong năm 2021 và những năm tiếp theo. Smarthome hay nhà thông minh cũng là một điển hình trong IOT.

Vậy nhà thông minh là gì?

Smarthome hay nhà thông thông minh là một khái niệm dùng để nhắc đến những ngôi nhà với các thiết bị như thiết bị gia dụng, hệ thông âm thanh, hình ảnh, hệ thống camera, chuông báo động an ninh, hệ thống cửa ra vào, đèn, quạt, điều hoà, … có khả năng kết nối với nhau và có thể điều khiển từ xa theo lịch trình, thời gian không bị giới hạn bởi vị trí không gian, thời gian mà chỉ cần thông qua thiết bị có thể kết nối mạng internet hoặc dữ liệu di động.

Hình 2.12 Nhà thông minh Smarthome

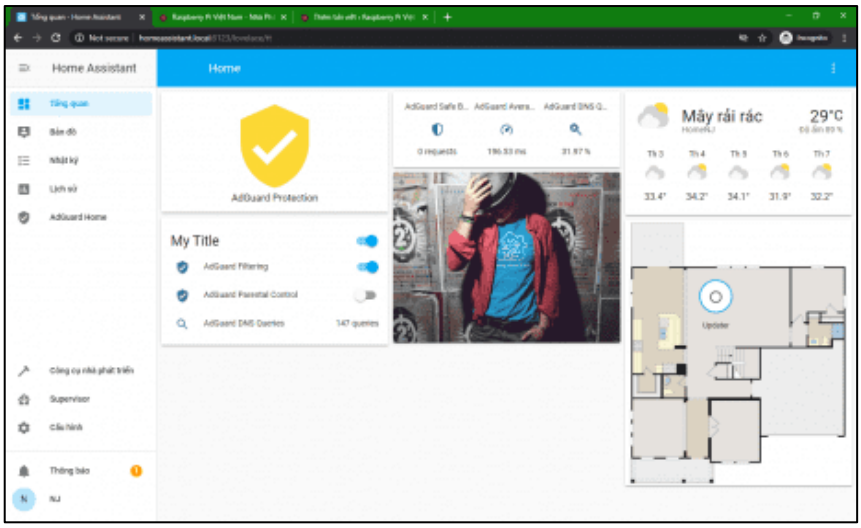
Những lợi ích của nhà thông minh: [4]

1. Cho phép chủ nhà và các thành viên điều khiển cũng như có khả năng giám sát, theo dõi ngôi nhà từ xa, chỉ cần sử dụng thiết bị duy nhất để điều khiển toàn bộ căn nhà chứ không cần phải đến khu vực đặt thiết bị
2. Nâng cao và hoàn thiện an ninh của ngôi nhà cũng như nâng cao chất lượng đời sống cho các thành viên trong gia đình với các tính năng như báo động khi phát hiện đột nhập căn nhà bất thường, đảm bảo tất cả các thiết bị an ninh như cửa đã được đóng, chuông báo chống trộm sẵn sàng hoạt động trong trường hợp nhà không có người, cảnh báo nguy cơ cháy nổ, dự báo về nhiệt độ thời tiết, rỏ rỉ điện, …
3. Tiết kiệm năng lượng tiêu thụ bằng cách tắt những thiết bị không cần thiết (ví dụ như đêm khuya thì sẽ tắt bớt đèn, chỉ bật đèn khi trời đã tối và có người trong nhà, tự động tắt điều hoà khi phòng đã lạnh về đêm, …).
4. Tăng khả năng hỗ trợ trong các trường hợp đặc biệt như người già, trẻ nhỏ, vật nuôi, cây trồng.

Có rất nhiều nền tảng nhà thông mình như Apple Homekit, Google Smart Home, Amazon Alexa, … Mà trong đó, Home Assistant là một nền tảng khá nổi tiếng và được yêu thích.

Hình 2.13 Logo nền tảng nhà thông minh Home Assistant

2.1.3.5. Home Assistant là gì?



Hình 2.14 Giao diện điều khiển của Home Assistant

Home Assistant là một mã nguồn mở chạy trên Python 3.x, một nền tảng tự động hoá dành cho nhà thông minh được thiết kế để có thể dễ dàng triển khai trên bất kỳ thiết bị máy tính nào từ Raspberry Pi đến các thiết bị lưu trữ trên mạng (NAS) thậm chí là một container Docker để triển khai trên một hệ thống khác dễ dàng.

Home Assistant có khả năng tích hợp một số lượng lớn các sản phẩm mã nguồn mở cũng như thương mại. cho phép kết nối giữa các thiết bị hoặc dữ liệu với nhau, ví dụ như IFTT (if this then that) – công cụ tự động hoá các thao tác, Google home mini, công tắc thông minh, thông tin về dự báo thời tiết, cho phép kiểm soát từ các thiết bị trong nhà.

Được thiết kế trên Python 3.x, Home Assistant có thể chạy trên bất cứ thiết bị, dịch vụ nào có thể cài Python. Dự án Home Assistant do Paulus Schoutsen khơi xướng được ra mắt vào năm 2013. Dự án này thu hút được 20 người hoạt động tích cực và phát hành cập nhật liên tục 2 lần 1 tuần (theo số liệu năm 2019).

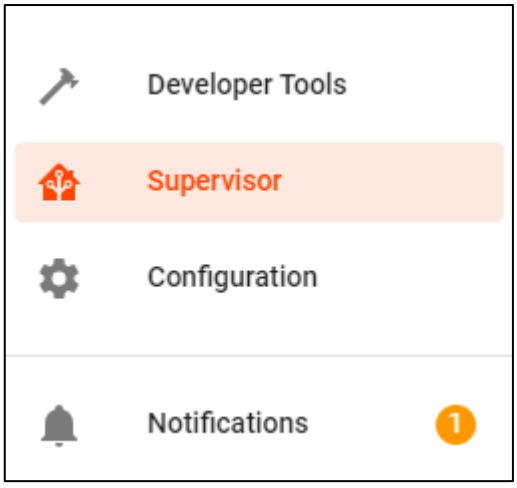
2.1.3.6. Thành phần của Home Assistant [5]

Home Assistant bao gồm Home Assistant Core, Supervisor và các công cụ khác giúp hỗ trợ và quản lý server Home Assistant dễ dàng hơn. Các công cụ này thường được gọi là Add-ons nhưng ngoài ra, Home Assistant còn có các công cụ khác như DNS, Audio, observer, multicast, v.v…

```
root@javis-hc:~# docker ps
CONTAINER ID        IMAGE
47574aad7023        homeassistant/raspberrypi4-64-homeassistant:2021.6.4
ff215685c10f        homeassistant/aarch64-hassio-supervisor
a28f3da00320        javishome/zigbee2mqtt:1.18.1-9
f34ddb1f02be        javishome/javis_hc_tool:1.1.5
4c71dccdc7ee        homeassistant/aarch64-addon-configurator:5.3.1
26cb820ddff1        homeassistant/aarch64-addon-mosquitto:5.1
b9f837cec06d        homeassistant/aarch64-hassio-multicast:2021.04.0
15439e4b9d3c        homeassistant/aarch64-hassio-cli:2021.05.1
6336cd70f834        homeassistant/aarch64-hassio-audio:2021.04.0
4791ab27c9d0        homeassistant/aarch64-hassio-dns:2021.04.0
aeb4874e6f32        homeassistant/aarch64-hassio-observer:2020.10.1
```

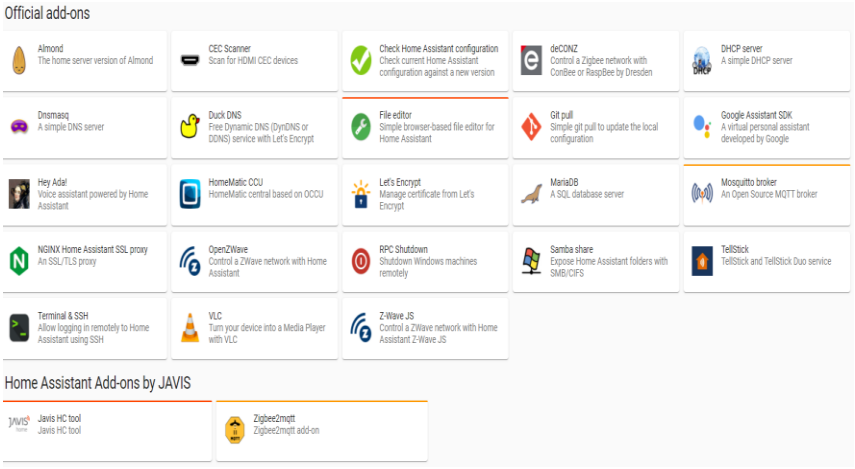
Hình 2.15 Các thành phần của Home Assistant khi cài bằng Docker

- Home Assistant Core: Thành phần lõi của Home Assistant chứa tất cả các tính năng cơ bản và cần thiết để máy chủ có thể hoạt động bình thường. Home Assistant Core được viết bằng ngôn ngữ Python (phiên bản Python 3.7), có khả năng hoạt động tốt trên hầu hết các máy chủ Linux/Windows được hỗ trợ bởi ngôn ngữ này. Để hoạt động được Home Assistant, thành phần Home Assistant Core là thành phần cần thiết và không thể thiếu.
- Home Assistant Supervisor: Là một công cụ nhằm mục đích phục vụ quản lý như: cập nhật, cài đặt, xoá Home Assistant Core, Add-ons.



Hình 2.16 Công cụ Home Assistant trên thanh menu điều khiển

- ADD-ONS: Là các ứng dụng chạy như một service độc lập cùng với Home Assistant Core để cung cấp các tính năng hỗ trợ người dùng sử dụng Home Assistant. Các ứng dụng này đều được quản lý bằng Supervisor của Home Assistant. Các ứng dụng này được Home Assistant cung cấp hoặc được tạo bởi bởi cá nhân hoặc tổ chức riêng biệt khác bởi Home Assistant tự tạo addons riêng.



Hình 2.17 Các công cụ được tích hợp vào Home Assistant

2.1.3.7. Các khái niệm trong Home Assistant

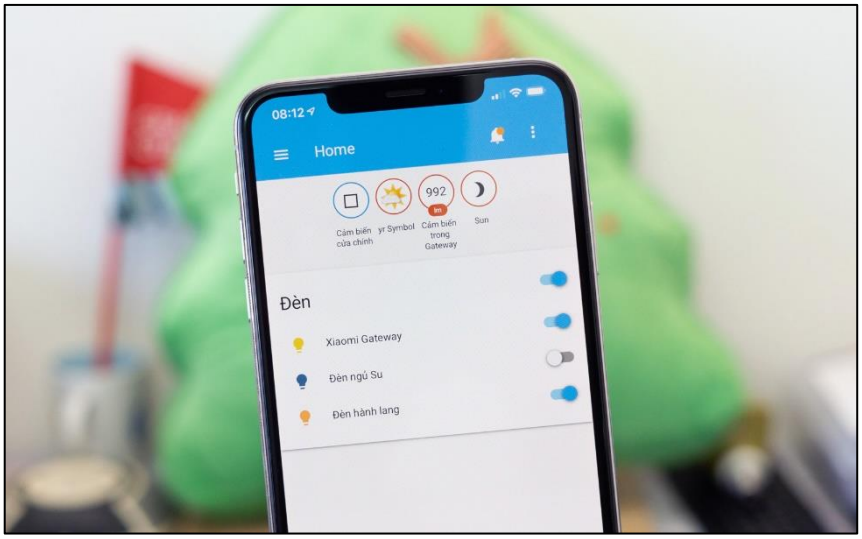
- **AUTOMATION**: tự động hoá, một thế mạnh của Home Assistant, cho phép người dùng thực hiện một hoặc một chuỗi ACTION khi có một hoặc một nhóm sự kiện kích hoạt xảy ra. AUTOMATION cho phép kiểm tra các điều kiện trước khi thực hiện ACTION phù hợp.
- **ACTION**: hành động, một sự kiện dành riêng cho AUTOMATION, **ACTION** được thực hiện khi có một sự kiện kích hoạt xảy ra mà đảm bảo thoả mãn các điều kiện cho trước.
- **COMPONENT**: thành phần tích hợp
- **CONDITION**: điều kiện, thường được sử dụng trong các AUTOMATION hoặc SCRIPT. Các kịch bản này chỉ được thực hiện khi đã thoả mãn CONDITION. Các điều kiện này trả về kết quả là True/False tương đương với 1/0. Ngoài ra, nó có thể kết hợp với nhau theo kiểu And, Or hoặc kết hợp nhiều cặp thành các nhóm điều kiện một cách linh hoạt.
- **CUSTOMIZE / CUSTOMIZATION**: tuỳ biến, Home Assistant cho phép bạn thay đổi theo ý muốn một vài thông số mặc định của ENTITY ví dụ như tên hiển thị, biểu tượng, đơn vị đo, v.v… để thân thiện và phù hợp hơn với từng mục đích sử dụng.
- **DEVICE**: thiết bị, là các thiết bị phần cứng, các thiết bị vật lý có khả năng tích hợp vào Home Assistant
- **DISCOVERY**: tự động nhận dạng, là một chức năng được tích hợp vào Home Assistant để tự động phát hiện các thiết bị và thiết lập hiện có trong mạng thông qua các giao thức như UpnP (Universal Plug and Play), ZeroConf/mDNS.

- **DOMAIN**: miền, là tập hợp tất cả các ENTITY có chung một dạng STATE, được kích hoạt bởi các sự kiện giống nhau và cung cấp các SERVICE tương tự nhau. Để có thể truy cập vào một ENTITY cụ thể, Home Assistant có quy định bắt buộc phải đi kèm tên miền trước nó, ví dụ 'switch.den_cua_ra_vao'.
- **ENTITY**: thực thể, là đại điện của một chức năng, của một thiết bị hoặc một dịch vụ trong Home Assistant. ENTITY cũng là đơn vị cơ bản được dùng để quản lý và hiển thị chức năng hoặc thực hiện các ACTION. Một thiết bị hay dịch vụ có thể có một hay có nhiều các thực thể khác nhau.
- **EVENT**: sự kiện, một khái niệm rất quan trọng trong Home Assistant, đặc biệt nó đi kèm với các chức năng xử lý cơ bản bên trong. EVENT được lắng nghe và kích hoạt bởi các tích hợp hoặc bởi chính Home Assistant. Các AUTOMATION đều được kích hoạt dựa trên những EVENT này.
- **FRONTEND**: giao diện điều khiển của Home Assistant
- **GROUP**: nhóm, để có thể quản lý các thực thể (ENTITY) trong Home Assistant, các ENTITY có thể được sắp xếp vào cùng một nhóm. Thay vì phải tương tác với từng ENTITY mang cùng một thuộc tính hoặc dịch vụ nào đó, Home Assistant cho phép người dùng có thể tương tác với nhóm.
- **INTEGRATION**: tích hợp, là thành phần chịu trách nhiệm kết nối và hoạt động đa dạng với các thiết bị khác nhau của Home Assistant. INTEGRATION đóng vai trò duy trì trạng thái STATE, lắng nghe và tạo ra các EVENT, cung cấp các SERVICE.
- **LOVELACE**: giao diện điều khiển Frontend mặc định của Home Assistant. LOOVELACE DASHBOARD được sử dụng trong LOVELACE để tập hợp nhóm với các thẻ VIEW nhằm mục đích quản lý. VIEW đại diện bởi một tab bên trong bảng điều khiển.
- **NOTIFICATION**: thông báo, là một nhóm PLATFORM nằm dưới INTEGRATION notify có nhiệm vụ hỗ trợ người dùng.
- **PLATFORM**: nền tảng, giúp Home Assistant kết nối và tương tác với DEVICE hoặc SERVICE cụ thể. Tất cả các công việc để điều khiển đều cần phải thông qua PLATFORM.
- **SCRIPT**: kịch bản, là một INTEGRATION cho phép người dùng kích hoạt các ACTION theo các chế độ chạy khác nhau như single, restart, queued, parallel.
- **SERVICE**: dịch vụ, được sử dụng để tương tác với Home Assistant và các INTEGRATION với mục đích để kích hoạt các ACTION. Các SERVICE này thường được Home Assistant quy định sẵn.
- **STATE**: trạng thái, cho phép người dùng nắm được thông tin cơ bản nhất của một ENTITY. Các ENTITY này luôn hiện thị cho người dùng và các INTEGRATION truy cập STATE của chính nó.
- **TEMPLATE**: bản mẫu, đây cũng là một trong điều đặc biệt mạnh mẽ cho phép người dùng thay đổi theo ý muốn các thông tin trong Home Assistant. Chính điều này sẽ giúp người dùng linh hoạt hơn trong việc sử dụng Home Assistant để thực hiện ý tưởng.

- **TRIGGER**: kích hoạt, là một hoặc nhiều các điều kiện mà khi được thoả mãn sẽ kích hoạt các ACTION. TRIGGER có thể là điều kiện về thời gian, STATE, hoặc một điều kiện tuỳ chỉnh theo EVENT hoặc TEMPLATE.
- **ZONE**: vùng, là một phạm vi khu vực địa lý được giới hạn bởi toạ độ trung tâm về kinh tuyến/vĩ tuyến và bán kính. ZONE dùng để kiểm tra sự có mặt của một thiết bị dựa trên thông tin từ vị trí của thiết bị đó.

2.1.3.8. Những điểm nổi bật của Home Assistant

- Home Assistant không có các thành phần điện toán đám mây, điều này làm giảm thiểu khả năng an ninh, bảo mật, riêng tư và có tính ổn định cao hơn.

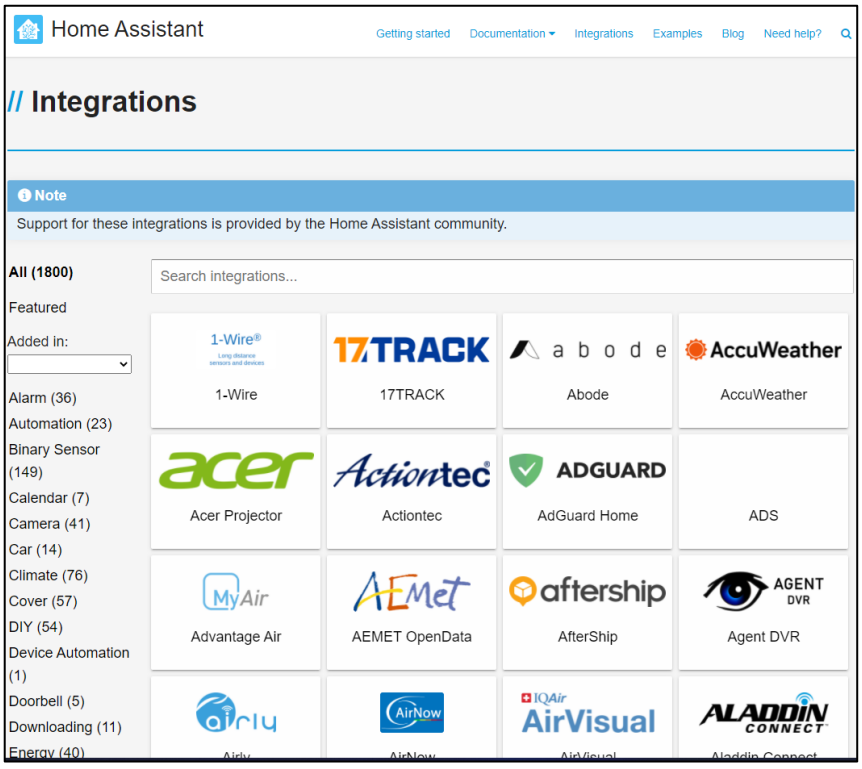


Hình 2.18 Home Assistant với phiên bản dùng trên điện thoại

- Home Assistant cung cấp bản client trên điện thoại và máy tính để người dùng có thể thao tác điều khiển từ xa thuận lợi hơn. Một điểm khác với các sản phẩm thương mại là nó không có các thiết bị trung tâm nên không có radio tích hợp sẵn.
- Home Assistant dễ dàng kết nối, đồng bộ với nhiều nền tảng khác nhau từ Nest đến Arduino, Kodi do không hoàn toàn khác biệt với các framework IOT.
- Được phát triển trên Python 3.x, Home Assistant có một điểm mạnh đó chính là sự linh hoạt bởi Python là một ngôn ngữ năng động. Điều này khiến cho Home Assistant có lợi thế về việc mở rộng hệ thống. Với python, việc kiểm tra và tạo các mẫu test thử cho từng thành phần mới trên phiên bản hiện có mà không bị ảnh hưởng vĩnh viễn đến các thành phần khác.
- Các hoạt động của Home Assistant là dựa trên sự kiện, kết hợp với theo dõi trạng thái thực thể. Mỗi thực thể đều có định danh, điều kiện và thuộc tính của riêng nó. Ví dụ đèn thông minh Xiaomi, cho phép bạn có thể điều chỉnh độ sáng, màu sắc, trạng thái bật tắt, … Ngoài ra, Home Assistant

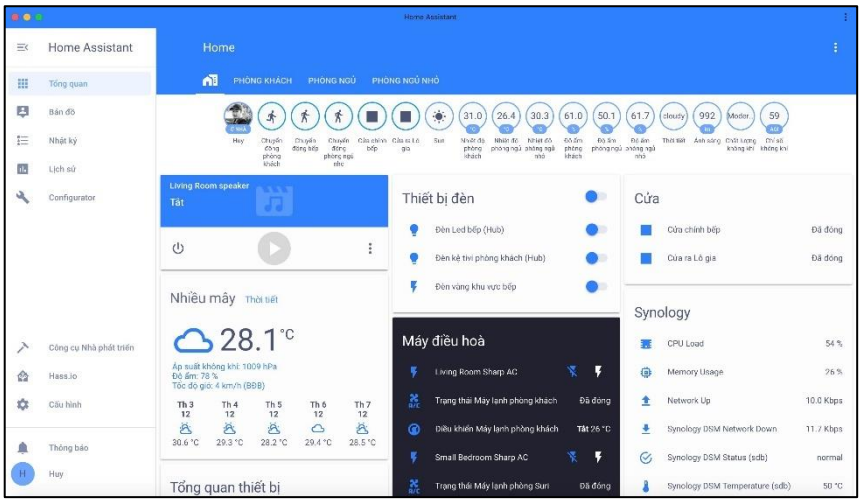
cho phép điều khiển theo nhóm. Quá trình thiết lập cũng dễ dàng nhờ khả năng phát hiện các thành phần và quét trong mạng. Việc thiết lập một thiết bị được hỗ trợ đã được đơn giản hoá đến mức trở thành một quá trình gần như tự động hoàn toàn.

2.1.3.9. Các tính năng Home Assistant mang lại



Hình 2.19 Danh sách các thiết bị đã được Home Assistant hỗ trợ

- Theo dõi và giám sát: tất cả các thiết bị thông minh trong nhà bạn đều có thể được Home Assistant theo dõi liên tục, miễn là các thiết bị đó đã được Home Assistant hỗ trợ. Danh sách các hãng thiết bị đã được Home Assistant hỗ trợ lên đến 1800 loại (tính đến thời điểm 12/6/2021). Trong đó, phải kể đến như Nest, IFTTT, Google, Hue, MQTT, Wemo, KODI, Plex, IKEA, vera, Arduino, Adobe, Amazon, Apple, Asus, Cisco, D-Link, Facebook, Huawei, LG, Microsoft, … điều này có thể thấy được sự lớn mạnh của cộng đồng Home Assistant. [6]



Hình 2.20 Giao diện Web điều khiển các thiết bị của ***Home Assistant***

- Điều khiển tất cả các thiết bị chỉ cần một giao diện duy nhất, thân thiện và dễ sử dụng với cách hiển thị dạng web.
- Home Assistant không lưu lại dữ liệu người dùng trên máy chủ để đảm bảo tính riêng tư.
- Tính năng hay nhất mà Home Assistant đem lại đó chính là sự tự động hoá của ngôi nhà. Ví dụ như vào buổi tối, sẽ tự động bật đèn nếu phát hiện bạn muốn đi vệ sinh, hay tự động phát nhạc, đọc báo hoặc báo thức vào buổi sáng, hay thông báo tắt đèn khi bạn đi ra khỏi nhà, … tất cả những điều này đều có thể thực hiện theo mong muốn và thói quen của người dùng.

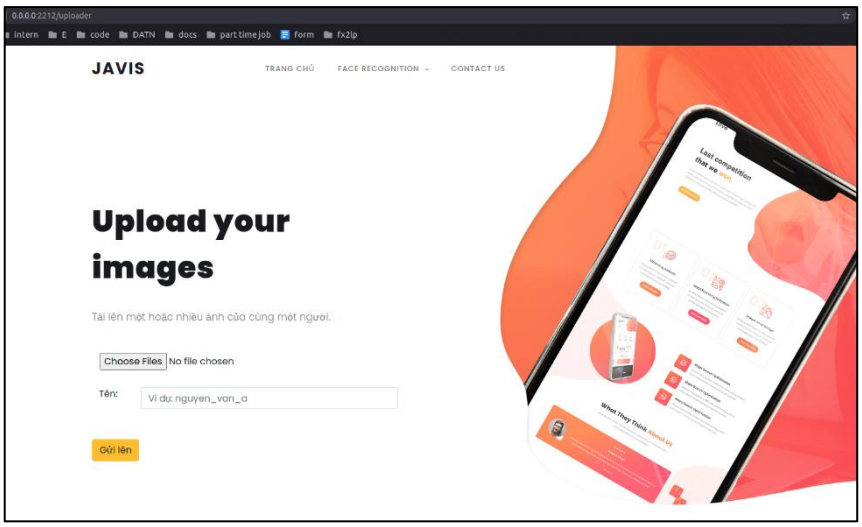
2.2 Dịch vụ Service để xử lý ảnh

Để có có thể phát hiện được khuôn mặt xuất hiện trong camera và nhận dạng được khuôn mặt đó là của ai, cần phải có một service chuyên để giải quyết bài toán này. Service dùng để xử lý ảnh sẽ được viết bằng Flask gồm có các chức năng như sau:

- Tải ảnh của các thành viên và lưu lại vào Database
- Chụp ảnh trong trường hợp không có ảnh có sẵn
- Đánh nhãn những ảnh trong Database
- Demo nhận diện khuôn mặt Realtime

Ngoài ra, còn một số tính năng phụ khác như là hỗ trợ báo cáo sự cố, góp ý phản hồi.

2.2.1.1. Chức năng tải ảnh lên

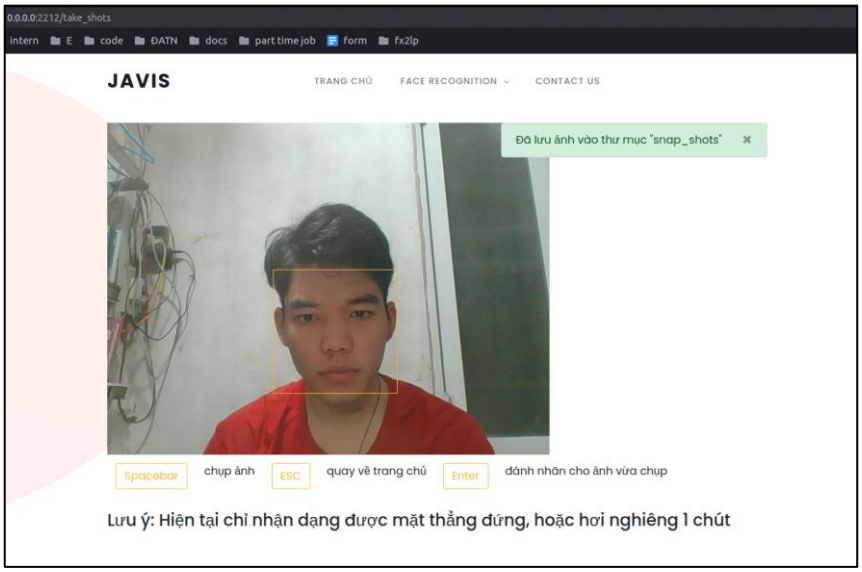


Hình 2.21 Giao diện trang tải ảnh lên Database

Đảm nhiệm vai trò tải ảnh lên và lưu lại chờ đánh nhãn, trang tải ảnh lên có thể tải lên từ một hay nhiều ảnh và lưu lại với tên được sinh ra tự động.

Ảnh tải lên phải có mặt người cần nhận diện trong đó, khi tải ảnh lên, hệ thống sẽ phát hiện và thực hiện cắt mặt và lưu lại ảnh dưới dạng file JPG trong thư mục static/snap_shots.

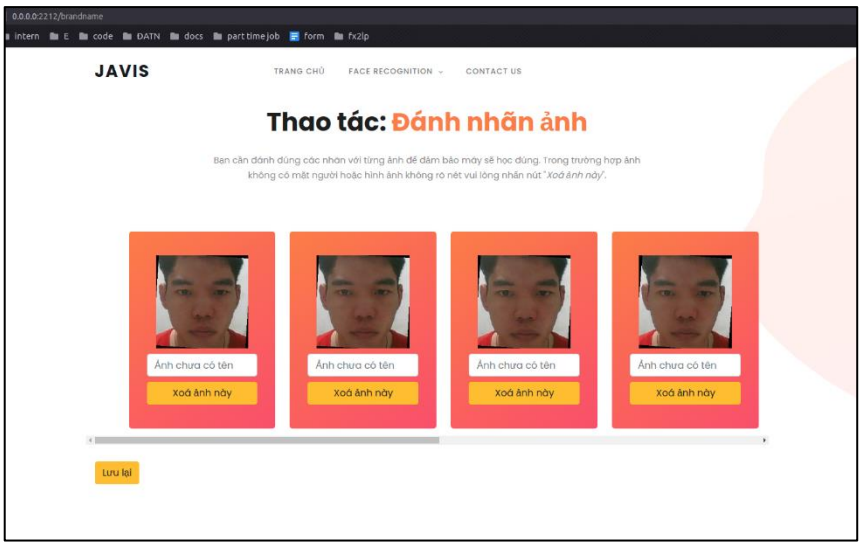
2.2.1.2. Chức năng chụp ảnh



Hình 2.22 Giao diện trang chụp ảnh

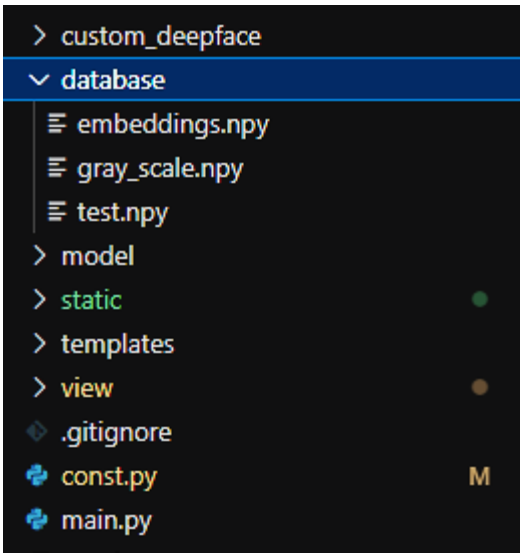
Ảnh sẽ được service chụp trực tiếp từ camera đã được kết nối đến service, ảnh sẽ được lưu lại dưới dạng JPG sau khi đã cắt ra khuôn mặt có trong ảnh và căn chỉnh lại ảnh khuôn mặt đứng thẳng.

2.2.1.3. Chức năng đánh nhãn ảnh



Hình 2.23 Giao diện trang đánh nhãn ảnh

Ảnh sau khi đã được chụp cần thực hiện thao tác tiếp theo đó chính là đánh nhãn cho mỗi ảnh. Thao tác này nhằm loại bỏ đi những ảnh không đạt tiêu chuẩn như ảnh khuôn mặt bị căn chỉnh đã xoay ảnh quá nghiêng, ảnh chụp được không rõ ràng. Sau khi đã đánh nhãn những ảnh có chất lượng tốt, phù hợp để lưu lại và loại bỏ những ảnh không phù hợp, hệ thống sẽ tự động lưu lại nó vào trong file embeddings.npy và xoá ảnh đã đánh nhãn để tránh tiết kiệm không gian lưu trữ.



Hình 2.24 Vị trí file embeddings.npy trong thư mục database

File embeddings này sẽ được lưu lại trong thư mục database của service nhằm mục đích lưu lại thông tin để nhận dạng.

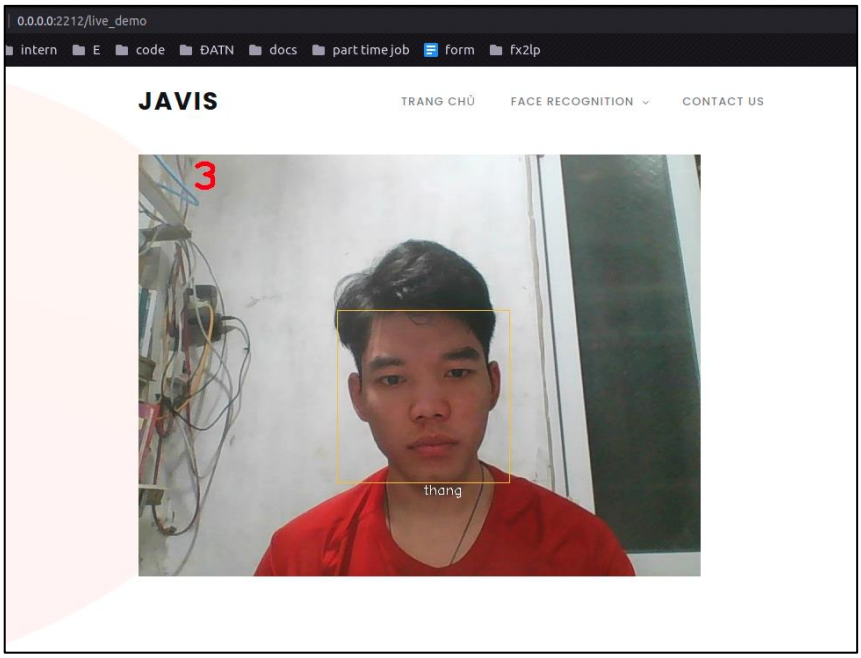
```
import numpy as np
import pandas as pd
from const import embedding_path
embeddings = np.load(embedding_path, allow_pickle=True)
df = pd.DataFrame(embeddings, columns=['employee', 'embedding'])
print(df.shape)
print(df)
```

```
(20, 2)
    employee                                          embedding
0        huy  [0.22873210906982422, -0.9564796686172485, -2....
1        huy  [-0.1191978007555008, -1.648725986480713, -2.4...
2        huy  [0.5203070640563965, -0.9194456338882446, -2.1...
3        huy  [0.20127424597740173, -0.8754792213439941, -2....
4        huy  [-0.11226420104503632, -0.8447368144989014, -2...
5    dung ai  [0.9970278739929199, -1.1126073598861694, -3.1...
6    dung ai  [0.8800203800201416, -1.1757560968399048, -3.0...
7    dung ai  [1.2486956119537354, -1.0430350303649902, -3.2...
8    dung ai  [1.0159077644348145, -1.186078429222107, -2.83...
9    dung ai  [0.7063747644424438, -1.053912878036499, -2.80...
10  dung iot  [-0.014605514705181122, -0.7461525201797485, -...
11  dung iot  [0.022123202681541443, -0.9281516075134277, -0...
12  dung iot  [-0.35685089230537415, -0.5286805629730225, -1...
13  dung iot  [-0.33963507413864136, -0.3695027530193329, -0...
14  dung iot  [-0.5359537601470947, -0.7440876960754395, -0....
15     thang  [0.5457626581192017, -1.7796396017074585, -1.8...
16     thang  [0.6490243673324585, -1.6345521211624146, -1.6...
```

Hình 2.25 Cấu trúc của file embeddings

File embeddings được lưu lại có đuôi ‘.npy’ và chứa các thông tin: tên người, vector ảnh của người đó.

2.2.1.4. Demo nhận diện realtime để trải nghiệm



Hình 2.26 Giao diện trang demo nhận diện khuôn mặt realtime

Service nhận diện khuôn mặt sẽ tự động phát hiện khuôn mặt và nhận diện khuôn mặt. Thông qua so sánh, đối chiếu với thông tin đọc ra từ file embeddings.npy, kết quả trả về sẽ có dạng:


```
{
  "duration": 0.5837879180908203,
  "identity": "thang"
}
```


Hình 2.27 Kết quả trả về dưới dạng JSON

- **Duration**: thời gian xử lý nhận dạng và trả về kết quả tính theo giây (trong ví dụ thì thời gian dùng để nhận dạng là 0.58 giây)
- **Identity**: tên người đã nhận diện được (trong ví dụ này thì người được nhận ra có tên là 'thang')

2.2.1.5. API điều khiển dùng để giao tiếp giữa Service xử lý ảnh và Home Assistant

Ngoài service cung cấp RESTful API như là giao diện Web frontend (mặc định là cổng 2212), service cũng cung cấp các API để gọi đến và trả về kết quả dưới dạng mã hoá JSON.

Để Home Assistant có thể gọi đến Service xử lý ảnh, cần phải thêm API component vào trong file 'rest_command.yaml' của Home Assistant. [7]

```
predict_snapshot:
  content_type: application/json; charset=utf-8
  method: POST
  payload: '{"title": "{{ title }}", "message": "{{ message }}"}'
  url: http://127.0.0.1:8123/predict_snapshots
snapshot:
  content_type: application/json; charset=utf-8
  method: POST
  payload: '{}'
  url: http://127.0.0.1:2212/snapshot
```

Hình 2.28 API được gọi từ Home Assistant đến Service xử lý ảnh được thêm vào trong file 'rest_command.yaml'

Hình 2.25 là minh hoạ cho 2 thực thể trong Home Assistant, được gắn với 2 API để gọi đến Service xử lý ảnh. Địa chỉ IP 127.0.0.1 với cổng 2212 là định tuyến của Service xử lý ảnh do Service xử lý ảnh và Home Assistant đều được tích hợp trong bộ điều khiển trung tâm.

```python
@app.route('/predict_snapshots', methods=['POST'])
def predict_snapshots():
    payload = request.get_json()
    if (payload['message'] == 'service_snapshot'):
        return predict_service_snapshots()
    else:
        img = os.path.join(img_path, payload['title'])
        if check_file_exist(img):
            return predict_snapshot(img)
        else:
            logging.warning("File is not found")
            return "File is not found"
```

Hình 2.29 URL predict_snapshots nhận sự kiện khi được Home Assistant gọi đến Service xử lý ảnh

Hình 2.26 là minh hoạ cho hàm xử lý nhận sự kiện gọi từ Home Assistant khi thực thể predict_snapshot được kích hoạt trên Home Assistant.

Các thực thể predict_snapshot, snapshot khi được kích hoạt trên Home Assistant sẽ thực hiện gọi đến Service xử lý ảnh và thực hiện xử lý và trả về kết quả như hình 2.24.

2.3 Điều khiển các thiết bị được tích hợp với Home Assistant

2.3.1 Những yêu cầu về phần cứng và cài đặt phần mềm

2.3.1.1. Những yêu cầu về phần cứng

Hình 2.30 TV box TX3 được lựa chọn làm bộ điều khiển trung tâm

- **Bộ điều khiển trung tâm**: là một máy tính bình thường, cấu hình không cần quá mạnh, hoặc Raspberry Pi hoặc Tv box dùng để chứa Service xử lý

ảnh và Home Assistant. Ở đây, TX3 là bộ có chip Arm với CPU 2.1GHz, ram 4GB, bộ nhớ trong 16-32GB, được cài hệ điều hành Armbian 20.09 Bionic.

- **Camera**: là những loại camera bình thường có trên thị trường, thậm chí cả cam của laptop.

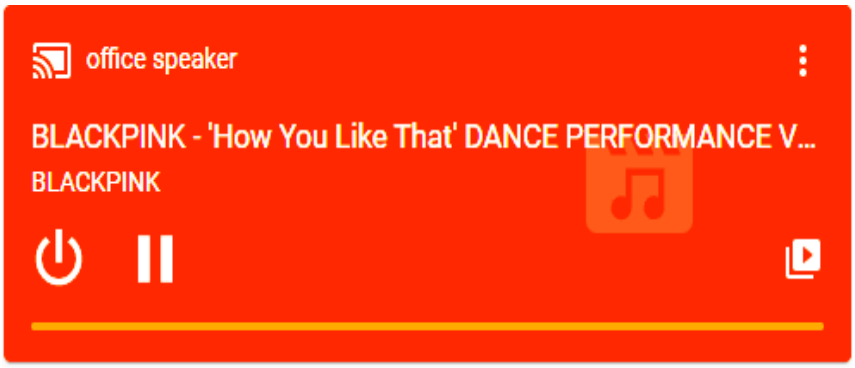
2.3.1.2. Những yêu cầu về cài đặt phần mềm

- Bộ xử lý trung tâm được cài đặt hệ điều hành Armbian 20.09 Bionic, là một phiên bản hệ điều hành Linux dành cho dòng Arm
- Phiên bản Home Assistant version core-2021.5.5 ra mắt ngày 5/5/2021

2.3.2 Các thiết bị được tích hợp trong bài toán

Một trong những điểm mạnh của Home Assistant đó chính là việc tích hợp các thiết bị trong danh sách hỗ trợ của Home Assistant gần như trở thành một quá trình tự động hoá.

2.3.2.3. Google Home Mini



Hình 2.31 Thực thể Google Home Mini đã được tích hợp trên Home Assistant

Google Home Mini được sử dụng để đọc thông báo khi Home Assistant nhận được sự kiện trả về kết quả nhận diện.

2.3.2.4. Khoá cửa thông minh tích hợp vân tay

Hình 2.32 Khoá của thông minh có tích hợp mật mã vân tay

Khoá cửa thông minh được tích hợp vào Home Assistant sẽ tự động được mở ra trong trường hợp Home Assistant nhận được sự kiện trả về kết quả đã nhận ra người trong camera.

2.3.2.5. Camera

Để có thể tích hợp camera vào Home Assistant, camera sẽ sử dụng giao thức RTSP như hình sau:

```
name: Camera_ATV
platform: ffmpeg
input: rtsp://admin:ECSIAQ@192.168.1.47:554
```

Hình 2.33 Tích hợp Camera vào Home Assistant sử dụng giao thức RTSP

Giao thức **RTSP** (**R**eal **T**ime **S**treaming **P**rotocol) – giao thức truyền tin thời gian thực là một giao thức điều khiển truyền thông mạng ở tầng ứng dụng để điều khiển máy chủ chứa các dữ liệu truyền tin đa phương tiện. Giao thức này được sử dụng để thiết lập và điều khiển các phiên truyền thông giữa các trạm cuối. RTSP được phát triển bởi RealNetworks, Netscape và Đại học Columbia, với dự thảo đầu tiên đệ trình lên IETF năm 1996.

2.3.2.6. Sensor dùng để kiểm tra kết quả

```
- platform: rest
  resource: "http://192.168.0.106:2212/get_results"
  method: POST
  name: check_person
  value_template: "{{ value_json.identity }}"
  force_update: true
  scan_interval: 1
- platform: rest
  resource: "http://192.168.0.106:2212/get_results"
  method: POST
  name: get_duration
  value_template: "{{ value_json.duration }}"
  force_update: true
  scan_interval: 1
```

Hình 2.34 Thiết lập Sensor kiểm tra kết quả trả về trong Home Assistant

Để có thể lấy được kết quả ngay khi nhận được sự kiện trả về kết quả nhận dạng, Home Assistant cần dùng sensor ảo được tạo ra để liên tục gửi đến Service xử lý ảnh và trả về thông tin. [7]

2.3.3 Kịch bản script để điều khiển thiết bị chạy tự động

```
chup_anh_tu_services:
  alias: chụp 3 ảnh
  sequence:
  - data:
      message: service_snapshot
    service: rest_command.predict_snapshot
  - data:
      entity_id: media_player.office
      message: "{% if states('sensor.check_person') != 'unknown' %}
                Chào mừng
                {{ states('sensor.check_person') }}
                đã trở về nhà.
                {% else %}
                Không nhận dạng được khuôn mặt. Xin hãy thử lại.
                {% endif %}"
    service: tts.google_translate_say
  mode: single
```

Hình 2.35 Kịch bản tự động hoá chụp ảnh và đọc thông báo ra loa Google Home Mini

Một trong những điểm mạnh khác của Home Assistant được nhắc đến nhiều nhất đó chính là sự mạnh mẽ của những kịch bản tự động hoá bởi sự ổn định, nhanh và thân thiện với người dùng.

Người dùng có thể thiết lập các kịch bản tự động hoá bằng cách viết trực tiếp vào file yaml hoặc sử dụng giao diện được hỗ trợ sẵn của Home Assistant.

Hình 2.32 minh hoạ một kịch bản tự động hoá có trình tự thực hiện như sau:

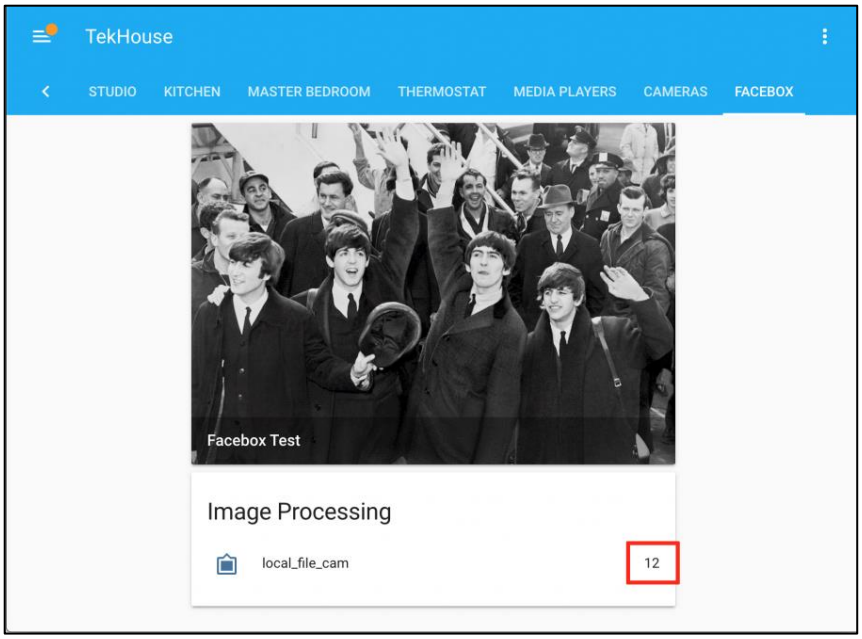
- Thực hiện gọi đến Service xử lý ảnh để chụp ảnh người xuất hiện trong camera thông qua API: ‘predict_snapshot’
- Đọc thông báo ra loa Google Home Mini khi phát hiện được người trong camera. Thông báo này được viết theo cú pháp và biến của công cụ Jinja2 templating. [8]

Kịch bản mở khoá cửa thông minh với Home Assistant cũng tương tự như kịch bản trên.

CHƯƠNG 3. **CÁC MÔ HÌNH THUẬT TOÁN ĐÃ TÌM HIỂU**

3.1 Mô hình FACEBOX

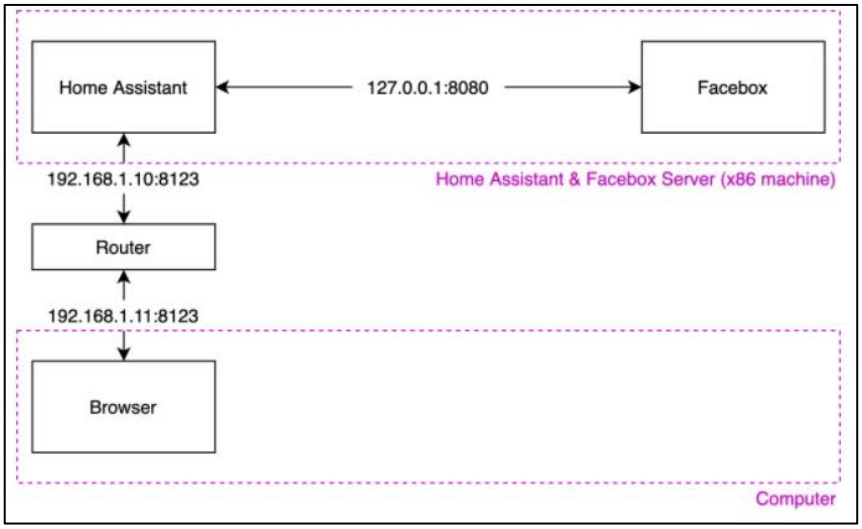
Nền tảng xử lý ảnh Facebox cho phép thực hiện và phát hiện khuôn mặt xuất hiện trong ảnh camera.



Hình 3.1 Facebox khi được tích hợp trên trang điều khiển của Home Assistant

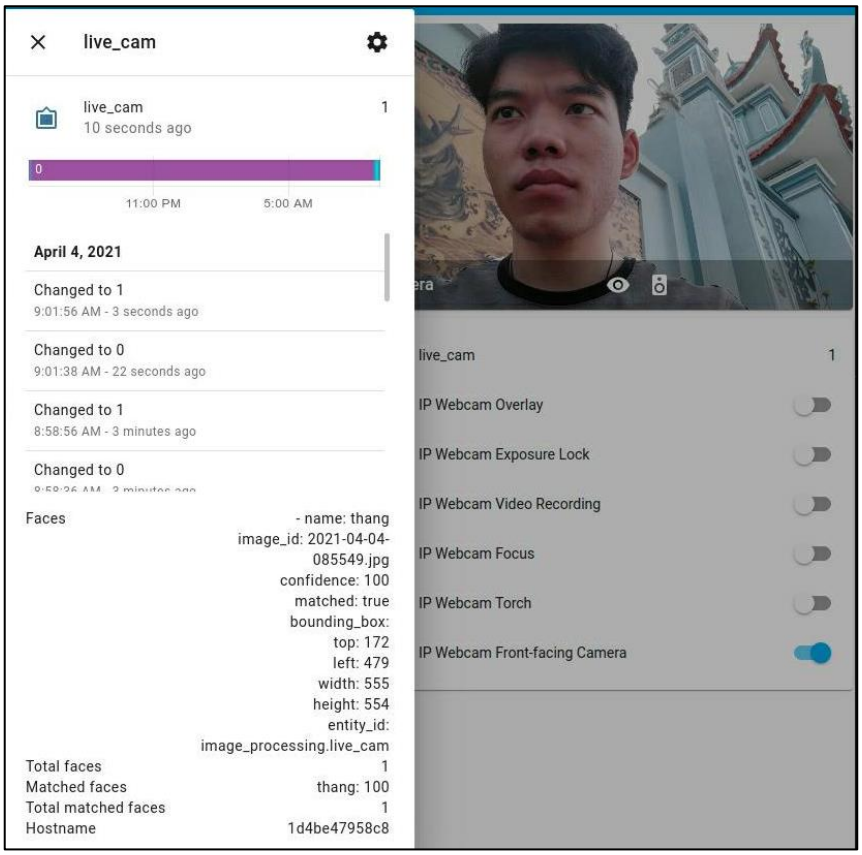
3.1.1 Ưu điểm

- Serivce Facebox chạy bằng Docker giống như Home Assistant. Điều này khiến việc cài đặt và chạy trở nên đơn giản hơn rất nhiều. [9]



Hình 3.2 Mô hình Facebox

- Đã được Home Assistant hỗ trợ nên việc tích hợp với Home Assistant vô cùng đơn giản



Hình 3.3 Nhận diện khuôn mặt sử dụng Facebox

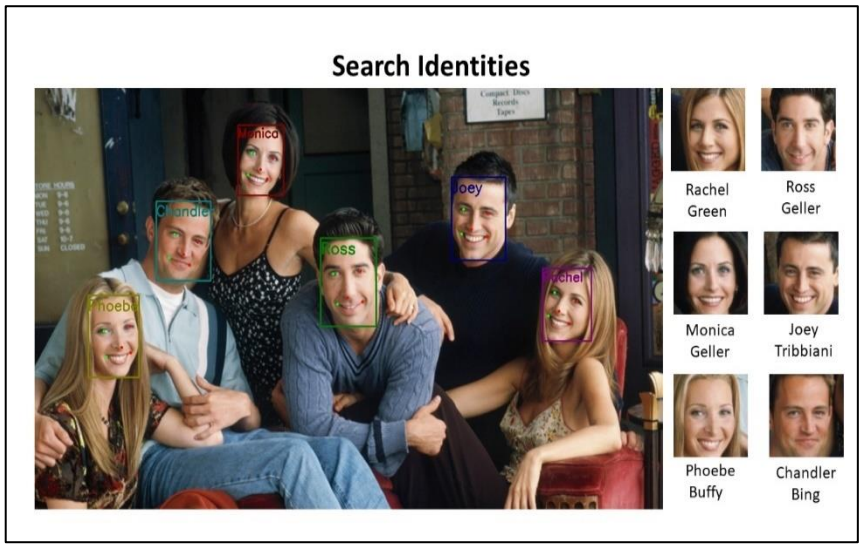
- Yêu cầu phần cứng không mạnh, Facebox hoạt động bằng cách chạy Docker container trên máy x86 với bộ nhớ ram tối thiểu 2GB

3.1.2 Khuyết điểm

- Chỉ những máy tính tương thích với x86 mới có thể chạy, không hỗ trợ phiên bản ARM
- Độ chính xác không cao
- Độ trễ khi thực hiện lằm mới ảnh truyền từ camera khá cao nên không phù hợp với những trường hợp cần sử dụng với thời gian thực

3.2 Mô hình INSIGHTFACE

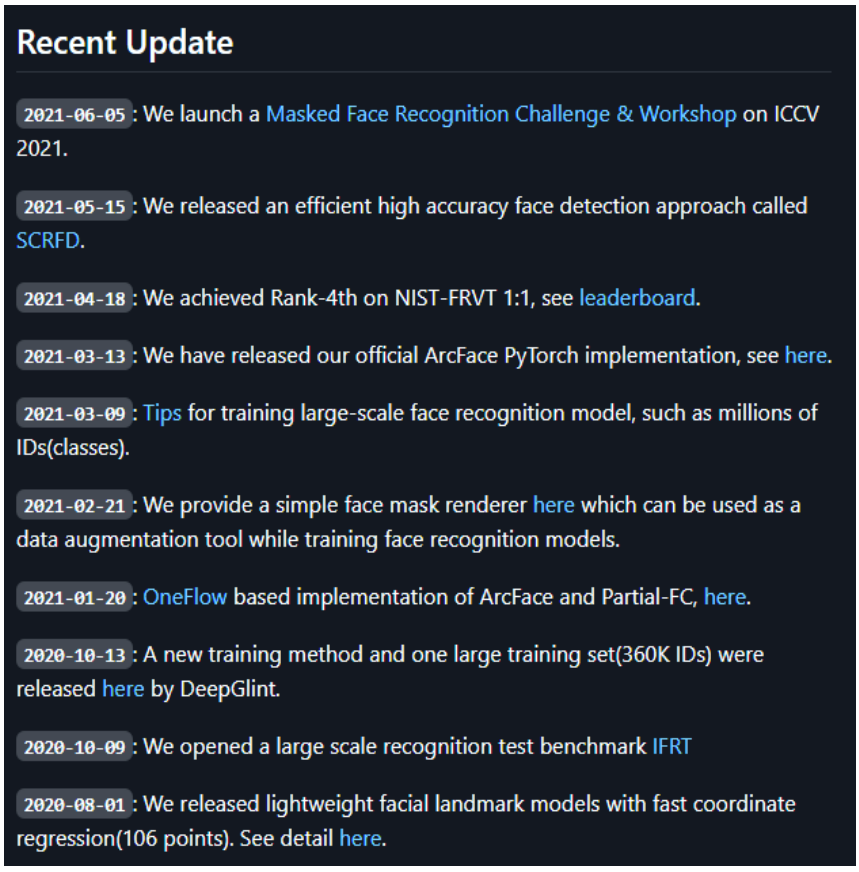
Insightface là một công cụ phân tích khuôn mặt với mã nguồn mở, dựa trên MXNet và PyTorch. [10]



Hình 3.4 ArcFace video demo

3.2.1 Ưu điểm

- Cộng đồng lớn mạnh nên các phiên bản mới được cập nhật rất thường xuyên và liên tục.



Hình 3.5 Những phiên bản cập nhất gần đây nhất của INSIGHTFACE

- Độ chính xác rất cao lên đến 99,86% trên tập dữ liệu LFW
- Thuật toán Face detection với mtcnn có độ chính xác tương đối và khá nhanh
- Nhận diện khá tốt với mặt thẳng

3.2.2 Khuyết điểm

- Mặt nghiêng do thông tin khuôn mặt bị mất mát khá nhiều nên không tốt
- Không dễ sử dụng, code khá phức tạp
- Tốn rất nhiều Ram, yêu cầu phần cứng máy tính phải đủ mạnh

3.3 Mô hình DEEPFACE

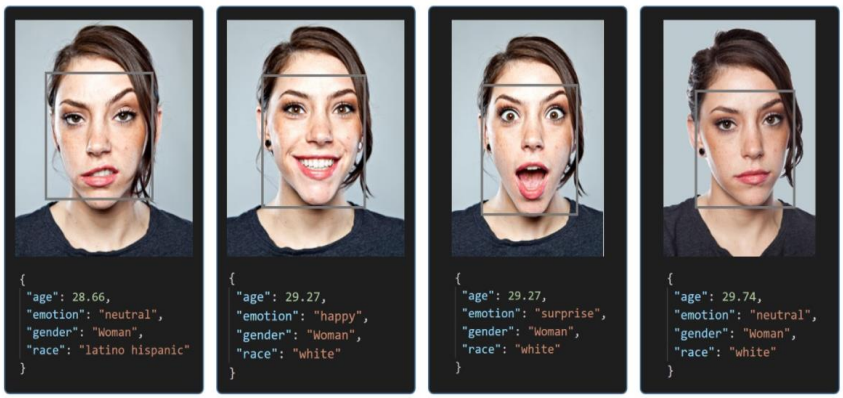
Hình 3.6 Logo Deepface

Deepface là một framework nhận dạng khuôn mặt đóng gói của nhiều mô hình hiện nay: VGG-Face, Google Facenet, Openface, Facebook, Deepface, DeepID, ArcFace và Dlib. Thư viện này chủ yếu dựa trên Keras và TensorFlow. [11]

Dự án Deepface được phát triển bởi Facebook, ra mắt tại hội thảo IEEE (tổ chức chuyên định ra các tiêu chuẩn cho nghành công nghệ) với chuyên đề về Thị giác máy vi tính và Nhận diện Biểu mẫu.

3.3.1 Ưu điểm

- Hướng dẫn sử dụng và cài đặt rất đơn giản, thân thiện, không mất quá nhiều thời gian đọc hiểu để có thể sử dụng



Hình 3.7 Xác định về độ tuổi, giới tính, chủng tốc, biểu cảm trong Deepface

- Thông tin kết quả trả về đa dạng: độ tuổi, giới tính, chủng tộc, biểu cảm, tên người. Trong đó Age model đạt ± 4.65 MAE, gender model đạt độ chính xác 97.44%, …
- Độ chính xác cao, cho phép chọn nhiều phương pháp nhận dạng khuôn mặt như độ chính xác Facenet đạt 99,65%, ArcFace đạt 99.4%, Dlib đạt 99.38%, VGG-Face đạt 98.78%, OpenCV đạt 93.80% trên tập dữ liệu LFW dataset trong khi mắt người đạt 97.53% trong các thử nghiệm chuẩn.
- Vẫn luôn được cập nhật và hoàn thiện qua từng phiên bản
- Do được phát triển bởi Facebook, Deepface có dữ liệu khuôn mặt của 4,4 triệu người trên mạng xã hội này khiến cho Deepface trở nên rất dược kỳ vọng (theo số liệu 2010) và xuất hiện trên Github vào tháng 7/2020.
- Là một trong ố ít các giải pháp nhận diện khuôn mặt REST API tự lưu trữ có thể được bắt đầu bằng một lệnh Docker-comp.
- Dễ dàng tích hợp trong hệ thống mà không cần có kiến thức về học máy, đồng thời có khả năng mở rộng cho phép nhận dạng đồng thời nhiều khuôn mặt trên nhiều luồng video.

3.3.2 Khuyết điểm

- Deepface cũng cung cấp API REST, nhưng nó chỉ hỗ trợ các phương pháp xác minh mà không thể tạo bộ sưu tập các khuôn mặt và tìm một trong số các khuôn mặt đó

3.4 Nhận định phương hướng tiếp cận

Tất cả mô hình thuật toán ở trên đều có những khuyết điểm và nhược điểm riêng.

Trong đó, Deepface là lựa chọn vô cùng phù hợp với mục tiêu ban đầu bởi những lý do sau:

- Độ chính xác cao > 99%, tương tự với INSIGHTFACE
- Phần cứng yêu cầu không cần quá mạnh, phù hợp với các máy tính yếu hoặc Raspberry Pi hoặc Tv box, không yêu cầu chip phù hợp với x86 như Facebox

- Khả năng tích hợp và mở rộng dễ dàng là tiêu chí rất quan trọng để có thể liên kết với các thành phần khác ví dụ như Home Assistant

Ngoài ra, để giảm tải yêu cầu về phần cứng, do deepface được viết dựa trên Keras và Tensorflow nên ở dự án này, Service xử lý ảnh viết lại Deepface sử dụng với Tensorflow Lite với ưu điểm của Tensorflow Lite là nhẹ hơn, tốc độ nhanh hơn, có thể cài trên hệ điều hành của Linux. Service xử lý ảnh được được viết bằng cách chuyển đổi mô hình có sẵn sang Tflite, viết lại các hàm dựa theo Tflite để có thể chuyển từ Tensorflow sang Tensorflow Lite.

CHƯƠNG 4. **ĐÁNH GIÁ VÀ THẢO LUẬN**

4.1 Kết quả

4.1.1 Kết quả của bài toán

4.1.1.1. Độ chính xác

Theo kết quả thử nghiệm, độ chính xác với ảnh ~80% trong điều kiện phòng làm việc có đủ ánh sáng, khuôn mặt thẳng và kết quả còn thấp hơn với mặt nghiêng.

Giải thích cho vì sao độ chính xác lại thấp hơn so với độ chính xác trong dự án Deepface (đã được nhắc đến ở mục 3.3.1), nguyên nhân quan trọng bắt nguồn từ các lý do như sau:

- Điều kiện môi trường bên ngoài như ánh sáng làm ảnh hưởng. Thậm chí trong một vài trường hợp ánh sáng môi trường cao hơn nhiều ánh sáng trong khuôn mặt, ví dụ như trường hợp camera chụp ngược sáng, service không phát hiện được ra có khuôn mặt trong ảnh.

Hình 4.1 Điều kiện về độ sáng giữa 2 ảnh khác nhau dẫn tới sai lệch

- Ảnh đầu vào để nhận diện là khuôn mặt dựa vào thuật toán cắt ảnh của OpenCV được cắt ra để test vẫn còn nhiều chỗ thừa, và bị nghiêng quá nhiều sau khi căn chỉnh dẫn tới độ chính xác thấp.

Hình 4.2 Ảnh được cắt ra do thuật toán cắt ảnh của OpenCV khi demo realtime

- Các trọng số của mô hình chưa được tối ưu sau khi chuyển đổi từ Tensorflow sang Tensorflow Lite.

4.1.1.2. Hiệu năng và tốc độ xử lý

Thời gian xử lý mất từ 0.4-1 giây cho để xử lý kết quả với tập dữ liệu test có 50 ảnh trên bộ điều khiển trung tâm TV box TX3 ram 4GB, cpu 2.1GHz.

4.1.2 Thế mạnh

- Tính đến thời điểm hiện tại, DeepFace là một trong những thuật toán nhận dạng khuôn mặt với độ chính xác thuộc top đầu.
- Tốc độ nhận diện khuôn mặt nhanh với số lượng người không quá lớn
- Không cần kết nối Internet để hoạt động, điều này góp phần giúp hoạt động của hệ thống ổn định hơn.
- Trong trường hợp chi chụp ảnh để xử lý chứ không dùng camera realtime liên tục sẽ giảm tải được công việc mà cpu phải xử lý, tăng độ bền của thiết bị phần cứng.
- Service xử lý ảnh không lưu lại ảnh sau khi đã được đánh nhãn mà mã hoá thành vector lưu vào trong file embeddings.npy góp phần tiết kiệm tài nguyên bộ nhớ với những hệ thống có không gian lưu trữ nhỏ.
- Kết hợp với Home Assistant khiến việc điều khiển các thiết bị theo một kịch bản tự động hoá đã thiết lập trước trở nên dễ dàng hơn.
- Cả Service xử lý ảnh cũng như Home Assistant đều sử dụng giao diện Web nên rất thân thiện với người dùng bình thường

4.1.3 Nhược điểm

- Độ chính xác không ổn định với khuôn mặt nghiêng hoặc trong môi trường ánh sáng yếu
- DeepFace được huấn luyện với bộ dữ liệu riêng biệt, gồm hàng triệu ảnh truyền thông, xã hội với kích thước ảnh lớn hơn những bộ dữ liệu khác trong nghiên cứu học thuật cũng như trong thử nghiệm test.

4.2 Tính ứng dụng của hệ thống

Bởi tốc độ xử lý nhanh, thiết bị phàn cứng nhỏ gọn không cồng kềnh, thích hợp với những văn phòng nhỏ, cơ quan hoặc trong các hộ gia đình, đặc biệt là các hộ gia đình ở các thành phố lớn thường có phòng cho sinh viên thuê.

Việc kiểm soát và theo dõi của hệ thống cũng như sự tiện lợi của một ngôi nhà thông minh giúp nâng cao chất lượng đời sống, đồng thời đảm bảo về sự an toàn và tính riêng tư.

CHƯƠNG 5. **KẾT LUẬN**

5.1 Kết luận chung

Công nghệ ngày càng phát triển mạnh, theo đó xu hướng nhà thông minh với sự tự động hoá của các thiết bị trong nhà từ đèn, quạt, đồ gia dụng, … cũng trở nên được ưa chuông hơn. Một giải pháp sử dụng khuôn mặt để mở khoá là thiết yếu trong một ngôi nhà như vậy để để bảo đảm sự an toàn cũng như góp phần cải thiện chất lượng cuộc sống.

Tính ứng dụng của giải pháp sử dụng khuôn mặt để mở khoá cũng rất rộng, không bị giới hạn bởi các hộ gia đình mà có thể ứng dụng với rất nhiều các công ty, cơ quan, … muốn theo dõi và kiểm soát chất lượng công việc để đảm bảo năng suất công việc.

Một hệ thống nhận diện khuôn mặt không đòi hỏi phần cứng quá mạnh, đồng thời không yêu cầu nhiều về thiết bị đi kèm có thể giúp tiết kiệm chi phí cũng như dễ dàng triển khai, lắp đặt hơn mà vẫn đảm bảo được hiệu quả khi hệ thống hoạt động.

Hệ thống hoạt động sử dụng giao diện Web thân thiện với người dùng bình thường dễ làm quen, ngoài ra còn dễ bảo trì và nâng cấp.

Tuy nhiên, giải pháp này vẫn còn gặp nhiều vấn đề ví dụ như độ chính xác không ổn định, phụ thuộc nhiều vào chất lượng của camera.

5.2 Hướng phát triển tiếp theo của đồ án trong tương lai

- Cải thiện độ chính xác bằng cách điều chỉnh trọng số sau khi chuyển đổi model từ Tensorflow sang Tensorflow Lite.
- Giảm bớt ảnh hưởng của camera với các yếu tố của môi trường như ánh sáng.
- Đóng gói service xử lý ảnh thành Docker để tích hợp lên Addons của Home Assistant.

TÀI LIỆU THAM KHẢO

[1] "Flask trong Python là gì? Lý do nên dùng Flask," [Online]. Available: https://vn.got-it.ai/blog/flask-trong-python-la-gi-ly-do-nen-dung-flask.

[2] "Flask là gì? Bạn biết gì về Framework này?," [Online]. Available: https://niithanoi.edu.vn/flask-la-gi.html.

[3] "RESTful API là gì? Cách thiết kế RESTful API," [Online]. Available: https://topdev.vn/blog/restful-api-la-gi/.

[4] "Apple HomeKit, Alexa và Google Home – Nền tảng Smart Home nào là tốt nhất?," [Online]. Available: https://smarthomekit.vn/chon-nen-tang-smart-home-phu-hop/.

[5] konnected, "Home Assistant: khái niệm và từ khoá cơ bản, thông dụng," [Online]. Available: https://konnected.vn/home-assistant/home-assistant-thuat-ngu-thong-dung-2020-03-19.

[6] H. Assistant, "Integrations," [Online]. Available: https://www.home-assistant.io/integrations/#camera.

[7] H. Assistant, "Command line Sensor," [Online]. Available: https://www.home-assistant.io/integrations/sensor.command_line.

[8] H. Assistant, "Templating," [Online]. Available: https://www.home-assistant.io/docs/configuration/templating.

[9] H. Assistant, "Facebox," [Online]. Available: https://www.home-assistant.io/integrations/facebox/.

[10] deepinsight/insightface, "Face Analysis Project on PyTorch and MXNet," [Online]. Available: https://github.com/deepinsight/insightface.

[11] serengil, "A Lightweight Deep Face Recognition and Facial Attribute Analysis (Age, Gender, Emotion and Race) Framework for Python," [Online]. Available: https://github.com/serengil/deepface.

[12] H. Assistant, "Home assistant face recognition (Facebox step-by-step guild)," [Online]. Available: https://siytek.com/home-assistant-face-recognition/.

[13] "Local presense detection using face recognition and tensorflow.js for Home Assistant," [Online]. Available: https://www.wouterbulten.nl/blog/tech/presence-detection-face-recognition-part-1/.

[14] "Python OpenCV: Capture video from Camera," [Online]. Available: https://www.geeksforgeeks.org/python-opencv-capture-video-from-camera/.

[15] "Tensorflow guild, Tensorflow Lite runtime package," [Online]. Available: https://www.tensorflow.org/lite/guide/python.

[16] "Streaming video with Flask on Web frontend," [Online]. Available:

https://github.com/miguelgrinberg/flask-video-streaming.

[17] H. Assistant, "RESTful API," [Online]. Available: https://www.home-assistant.io/integrations/rest/.

[18] "RESTful Sensor," [Online]. Available: https://www.home-assistant.io/integrations/sensor.rest.

[19] "Flask tutorial," [Online]. Available: https://www.tutorialspoint.com/flask/index.htm.