ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN - ĐIỆN TỬ
BỘ MÔN VIỄN THÔNG

----------o0o----------

ĐỒ ÁN TỐT NGHIỆP

XÂY DỰNG CỔNG IOT VỚI HOME ASSISTANT OS TÍCH HỢP AI TRONG XỬ LÝ ẢNH

SVTH: TỪ VĨNH MINH
MSSV: 2114085
GVHD: TS.VÕ QUẾ SƠN

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2025

KHÓA LUẬN TỐT NGHIỆP ĐƯỢC HOÀN THÀNH TẠI

TRƯỜNG ĐẠI HỌC BÁCH KHOA - ĐHQG - HCM

Cán bộ hướng dẫn Khóa luận tốt nghiệp : TS. Võ Quế Sơn

Cán bộ chấm phản biện : TS. Trịnh Xuân Dũng

Khóa luận tốt nghiệp được bảo vệ tại Trường Đại học Bách Khoa, ĐHQG TP.HCM ngày 30 tháng 12 năm 2025.

Thành phần Hội đồng đánh giá khoá luận tốt nghiệp gồm:

1. TS. Huỳnh Phú Minh Cường - Chủ tịch

2. TS. Võ Quế Sơn - Thư ký

3. TS. Trịnh Xuân Dũng - Ủy viên

Xác nhận của Chủ tịch Hội đồng đánh giá khóa luận tốt nghiệp và Chủ nhiệm Bộ môn sau khi đồ án đã được sửa chữa (nếu có).

CHỦ TỊCH HỘI ĐỒNG **CHỦ NHIỆM BỘ MÔN VIỄN THÔNG**

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA
-----☆-----

Số: _____ /BK.ĐĐT
Khoa: **Điện – Điện tử**
Bộ Môn: **Viễn Thông**

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc.
-----☆-----

NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP

1. HỌ VÀ TÊN: Từ Vĩnh Minh MSSV: 2114085
2. NGÀNH: Điện Tử - Viễn Thông LỚP : DD21DV3
3. Đề tài: Xây dựng cổng IoT với Home Assistant OS tích hợp AI trong xử lý ảnh.
4. Nhiệm vụ (Yêu cầu về nội dung và số liệu ban đầu):
   - Tìm hiểu về lý thuyết về Home Assistant OS và Raspberry pi 4
   - Tìm hiểu về các vi điều khiển, các cảm biến có trong hệ thống
   - Sơ đồ mạng và giao thức liên kết giữa các thiết bị
   - Tự động hóa trong Home Assistant: Thực hiện các kịch bản tự động hóa đã thiết lập sẵn
   - Ứng dụng xử lý ảnh chụp từ Camera với nhận diện đối tượng qua mô hình YOLOv11 và nhận diện khuôn mặt qua mô hình MTCNN và FaceNet.
5. Ngày giao nhiệm vụ luận văn: 15/09/2025
6. Ngày hoàn thành nhiệm vụ: 30/12/2025
7. Họ và tên người hướng dẫn: TS. Võ Quế Sơn Phần hướng dẫn: Toàn bộ

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

Tp.HCM, ngày..... tháng..... năm 20...

CHỦ NHIỆM BỘ MÔN **NGƯỜI HƯỚNG DẪN CHÍNH**

PHẦN DÀNH CHO KHOA, BỘ MÔN:
Người duyệt (chấm sơ bộ):.....
Đơn vị:.....
Ngày bảo vệ : .....
Điểm tổng kết: .....

LỜI CẢM ƠN

Trong quá trình thực hiện đồ án tốt nghiệp, em xin chân thành cảm ơn Thầy Võ Quế Sơn, người đã tận tình hướng dẫn, giúp đỡ và định hướng cho em hoàn thành đề tài cho đồ án tốt nghiệp trong HK251. Sự hỗ trợ và những góp ý chuyên môn của Thầy đã giúp em hiểu rõ hơn vấn đề cần nghiên cứu của đề tài, các hướng giải quyết khó khăn khi thực hiện và hoàn thành tốt đồ án.

Em cũng xin gửi lời cảm ơn chân thành đến Quý Thầy Cô Khoa Điện - Điện Tử của Trường đại học Bách Khoa - Đại học Quốc Gia TP.HCM, những người đã trang bị cho em những kiến thức nền tảng và chuyên môn về chuyên ngành Điện Tử - Viễn Thông trong suốt thời gian em học tập tại trường. Những bài giảng, kinh nghiệm và định hướng từ Quý Thầy Cô là hành trang vô cùng quan trọng giúp em có đủ tự tin và năng lực để thực hiện đề tài này.

Mặc dù đã rất cố gắng để thực hiện đồ án tốt nghiệp, nhưng bản thân em xin thừa nhận rằng đồ án tốt nghiệp của em vẫn còn nhiều thiếu sót, em rất mong nhận được sự góp ý của Thầy Cô để hoàn thiện hơn đề tài này cũng như nâng cao kiến thức chuyên môn cho công việc sau này.

TP. Hồ Chí Minh, ngày 30 tháng 12 năm 2025

Sinh viên thực hiện

TÓM TẮT ĐỒ ÁN TỐT NGHIỆP

Đồ án này trình bày về đề tài "Xây dựng cổng IoT với Home Assistant OS tích hợp AI trong xử lý ảnh" tập trung vào việc nghiên cứu và triển khai hệ thống nhà thông minh dựa trên nền tảng IoT và trí tuệ nhân tạo. Mục tiêu của đồ án là xây dựng một cổng IoT (IoT Gateway) có khả năng thu thập, xử lý và phân phối dữ liệu từ các thiết bị trong hệ thống Smart Home, đồng thời hỗ trợ các chức năng giám sát và tự động hóa nâng cao nhờ tích hợp khả năng xử lý ảnh.

Trong hệ thống này, Raspberry Pi được sử dụng làm trung tâm điều phối chạy Home Assistant OS, đảm nhiệm vai trò quản lý thiết bị, lưu trữ trạng thái và xử lý dữ liệu trong mạng nội bộ. Các thiết bị ngoại vi như ESP32 giao tiếp với Gateway thông qua MQTT và ESPHome API để truyền nhận dữ liệu cảm biến và thực hiện điều khiển theo thời gian thực. Kiến trúc này giúp hệ thống linh hoạt, dễ mở rộng và phù hợp cho mô hình Smart Home hiện đại.

Đề tài đồng thời tích hợp các mô hình AI xử lý ảnh để phục vụ mục tiêu giám sát và cảnh báo. Một ứng dụng Flask chạy trên Laptop thu nhận hình ảnh từ camera, xử lý bằng các mô hình như YOLOv11 (nhận diện đối tượng) và MTCNN – FaceNet (nhận diện khuôn mặt), sau đó gửi kết quả về Home Assistant OS dưới dạng JSON. Nhờ đó, hệ thống có thể hiển thị thông tin nhận diện trực quan và đưa ra cảnh báo một cách nhanh chóng và hiệu quả.

Kết quả thực nghiệm cho thấy hệ thống hoạt động ổn định, việc nhận diện đối tượng và khuôn mặt cho độ chính xác cao, tốc độ xử lý ảnh nhanh và hoạt động tốt trong mạng local. Giao diện Home Assistant OS hiển thị dữ liệu mượt mà, các cảm biến ESP32 phản hồi ổn định và các hành động điều khiển được thực hiện chính xác. Hệ thống thể hiện khả năng mở rộng tốt, dễ dàng tích hợp thêm thiết bị và kịch bản tự động hóa theo nhu cầu.

Trong tương lai, hệ thống có thể được nâng cấp theo hướng tăng cường bảo mật (VPN,HTTPS), mở rộng giao thức IoT (Zigbee, Lora,…), tối ưu mô hình AI để chạy trực tiếp trên thiết bị biên (Edge Computing), hoặc triển khai cơ chế truy cập từ xa qua Internet. Hiện tại đồ án chỉ được xây dựng trong mạng local nên có nhiều tính năng vẫn còn hạn chế.

MỤC LỤC

DANH SÁCH HÌNH MINH HỌA

DANH SÁCH BẢNG SỐ LIỆU

CHƯƠNG 1: GIỚI THIỆU

1.1. Tổng quan về đề tài:

Trong bối cảnh mạng lưới internet ngày càng phát triển, các hệ thống điều khiển từ xa phát triển mạnh mẽ, nhu cầu xây dựng một nền tảng điều khiển tập trung, linh hoạt và dễ mở rộng trở nên ngày càng quan trọng. Bên cạnh đó, lĩnh vực IoT và công nghệ trí tuệ nhân tạo (AI) đang ngày càng phát triển và có xu hướng hỗ trợ cho nhau trong các hệ thống như Smart Home, Smart Farm, điều khiển tự động trong các nhà máy đang được ứng dụng rộng rãi trong các hệ thống giám sát an ninh và tự động hóa nâng cao.

Đề tài "Xây dựng cổng IoT với Home Assistant OS tích hợp AI trong xử lý ảnh" được thực hiện nhằm kết hợp hai xu hướng công nghệ này, tạo ra một hệ thống IoT gateway thông minh thu thập dữ liệu, quản lý thiết bị, giám sát từ xa và kích hoạt các hành vi tự động trong môi trường nhà ở.

Hệ thống được xây dựng dựa trên Home Assistant OS, một nền tảng mã nguồn mở mạnh mẽ cho phép tích hợp đa thiết bị qua các giao thức như MQTT, ESPHome và API nội bộ. Gateway sử dụng Raspberry pi làm trung tâm điều phối, kết hợp với các node cảm biến và thiết bị IoT dựa trên ESP32. Đồng thời, mô hình AI như nhận diện đối tượng với Yolo, nhận diện khuôn mặt với MTCNN và FaceNet được tích hợp vào hệ thống giám sát ngôi nhà qua Camera.

Giải pháp không chỉ giúp hệ thống vận hành tự động và thông minh hơn, mà còn tăng tính an toàn, khả năng giám sát và quản lý toàn diện trong Smart Home. Với khả năng mở rộng linh hoạt, tính ổn định cao và giao diện người dùng trực quan, mô hình này là hướng tiếp cận hiệu quả và phù hợp cho các ứng dụng IoT hiện đại.

1.2. Lý do chọn đề tài:

Trong những năm gần đây, công nghệ IoT và hệ sinh thái nhà thông minh phát triển mạnh mẽ, trở thành một phần tất yếu trong các hệ thống tự động hóa hiện đại. Việc xây dựng một cổng IoT giúp kết nối và quản lý các thiết bị thông minh không chỉ mang tính nghiên cứu mà còn có giá trị ứng dụng thực tế rất lớn. Đặc biệt, Home Assistant - một nền tảng mã nguồn mở - đang được cộng

đồng quốc tế sử dụng rộng rãi nhờ khả năng mở rộng, linh hoạt và hoàn toàn miễn phí.

Bên cạnh IoT, trí tuệ nhân tạo (AI), đặc biệt là xử lý ảnh, đóng vai trò quan trọng trong việc nâng cao hiệu quả và tính thông minh của hệ thống. Nhà thông minh không chỉ dừng ở việc điều khiển tự động mà còn cần khả năng nhận diện, phân tích và đưa ra quyết định dựa trên hình ảnh thực tế. Ví dụ, nhận diện khuôn mặt, phát hiện đối tượng trong ảnh. Do đó, đề tài kết hợp IoT và AI mang lại tính thực tiễn cao, đáp ứng nhu cầu của nhiều ứng dụng đời sống.

Một lý do quan trọng nữa là khả năng triển khai thực tế của Home Assistant OS hỗ trợ rất nhiều thiết bị và giao thức như MQTT, Zigbee, Z-wave,… cho phép mở rộng mô hình mà không bị giới hạn về phần cứng. Đề tài hiện tại được triển khai trên Máy tính nhúng Raspberry pi 4 sử dụng hệ điều hành Home Assistant OS. Mặc dù Raspberry Pi 4 là một thiết bị phổ biến trong các hệ thống IoT nhờ chi phí thấp, khả năng linh hoạt và cộng đồng hỗ trợ rộng lớn, nhưng phần cứng của thiết bị vẫn còn những hạn chế nhất định khi xử lý các tác vụ yêu cầu hiệu năng cao. Vì vậy, Raspberry Pi 4 tập trung vào thu thập dữ liệu, quản lý thiết bị và giao tiếp với hệ thống, trong khi các tác vụ xử lý ảnh nâng cao được thực hiện trên dịch vụ ngoài hoặc các thiết bị như Laptop,... đảm bảo hiệu năng phù hợp với phần cứng.

1.3. Nhiệm vụ luận văn:

Nhiệm vụ 1: Tìm hiểu lý thuyết về Home Assistant OS và Raspberry pi 4

Nhiệm vụ 2: Tìm hiểu về các vi điều khiển , các cảm biến có trong hệ thống

Nhiệm vụ 3: Sơ đồ mạng và giao thức liên kết giữa các thiết bị

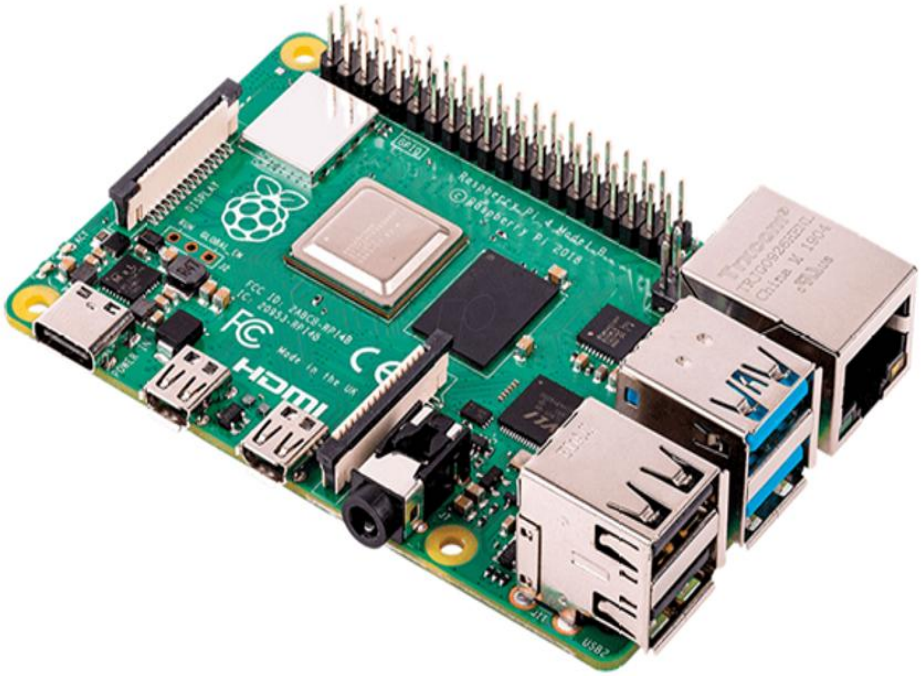
Nhiệm vụ 4: Tự động hóa trong Home Assistant: Thực hiện các kịch bản tự động hóa đã thiết lập sẵn

Nhiệm vụ 5: Ứng dụng xử lý ảnh chụp từ Camera với nhận diện đối tượng qua mô hình YOLOv11 và nhận diện khuôn mặt qua mô hình MTCNN và FaceNet.

CHƯƠNG 2: TỔNG QUAN VỀ HỆ THỐNG

2.1. Đặc tả phần cứng:

Máy tính nhúng Raspberry pi 4 module B 4 GB:



Hình 1: Máy tính Raspberry pi 4 model B

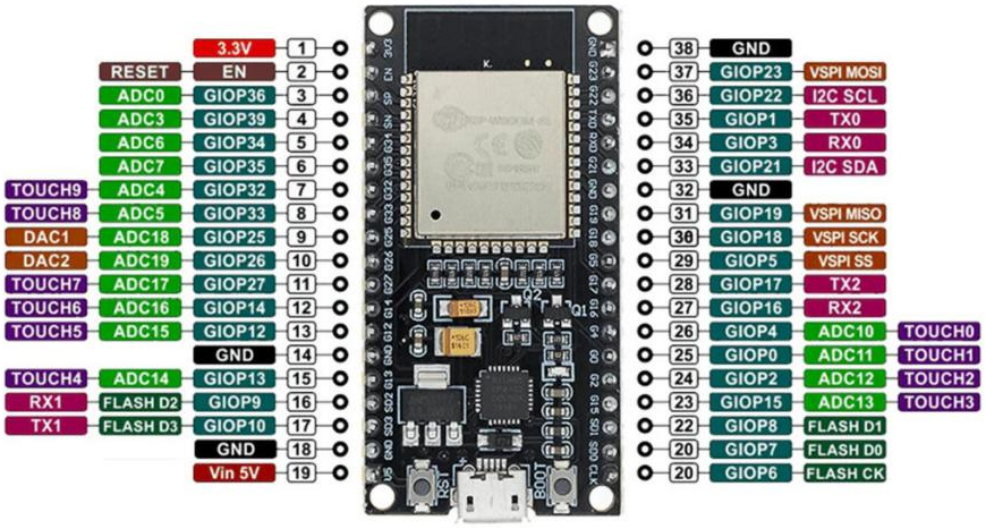
Raspberry Pi 4 Model B là một máy tính nhúng nhỏ gọn nhưng mạnh mẽ, được trang bị vi xử lý ARM Cortex-A72 4 lõi 64-bit với tốc độ lên đến 1.5 GHz, RAM tùy chọn từ 2GB đến 8GB, hỗ trợ xuất hình ảnh 4K qua cổng HDMI kép và tích hợp cổng USB 3.0, Gigabit Ethernet, Wi-Fi và Bluetooth 5.0. Với khả năng chạy nhiều hệ điều hành như Raspberry Pi OS, Ubuntu hoặc Home Assistant OS, thiết bị này trở thành nền tảng lý tưởng cho các dự án IoT, tự động hóa nhà thông minh, trí tuệ nhân tạo (AI) và xử lý dữ liệu thời gian thực. Ngoài ra, Raspberry Pi 4 hỗ trợ nhiều giao tiếp ngoại vi như GPIO, I2C, SPI, UART, giúp dễ dàng tích hợp các cảm biến, camera, module điều khiển và robot. Nhờ tính linh hoạt, chi phí thấp và cộng đồng hỗ trợ lớn, Raspberry Pi 4 Model B không chỉ là công cụ học tập lý tưởng cho sinh viên, mà còn là nền tảng thực nghiệm mạnh mẽ cho các ứng dụng nghiên cứu, phát triển prototyping và triển khai các hệ thống nhúng quy mô nhỏ đến vừa.

Bảng 1: Thông số kỹ thuật Raspberry pi 4

THÔNG SỐ KỸ THUẬT	
Vi xử lý	Broadcom BMC2711. Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5Ghz
RAM	4GB
Kết nối không dây	2,4GHz và 5GHz IEEE 802.11ac wireless LAN, Bluetooth 5.0, BLE
Kết nối có dây	Gigabit Ethernet
Cổng USB	2x USB 3.0, 2x USB 2.0
GPIO	40 pin GPIO
Video	2x Micro HDMI 4k
Camera	2x MIPI DSI/CSI
Âm thanh	4-pole stereo audio và composite video port
Lưu trữ	Micro SD
Nguồn	5,1V,3A /USB Type-C hoặc GPIO

Kit RF thu phát Wifi BLE ESP32 NodeMCU 38 chân

Hình 2: Module ESP32



Hình 3: Sơ đồ chân Module ESP32

ESP32 là vi điều khiển được ứng dụng rộng rãi trong các dự án về IoT, cần một thiết bị có khả năng điều khiển và kết nối với internet, bluetooth,… để điều khiển từ xa, hoạt động ổn định, tiêu thụ năng lượng thấp.

Bảng 2: Thông số kĩ thuật module ESP32

THÔNG SỐ KỸ THUẬT	
Vi xử lý	ESP32 (ESP32 - WROOM - 32)
Firmware	NodeMCU Lua
Chip USB-Serial	CP2102
Điện áp nguồn	5V DC
Tần số	Lên đến 240 MHz
Wifi	802.11 B/g/n/E/l (802.11N @ 2.4Ghz lên đến 150 Mbit/s)
Bluetooth	4.2 BR/EDR BLE 2 chế độ điều khiển
Bộ nhớ	448 Kbyte ROM, 520 Kbyte SRAM, 6 Kbyte SRAM trên RTC và QSPI Hỗ trợ đèn flash / SRAM chip
Bảo mật	IEEE 802.11, bao gồm cả WFA, WPA/WPA2 và WAPI

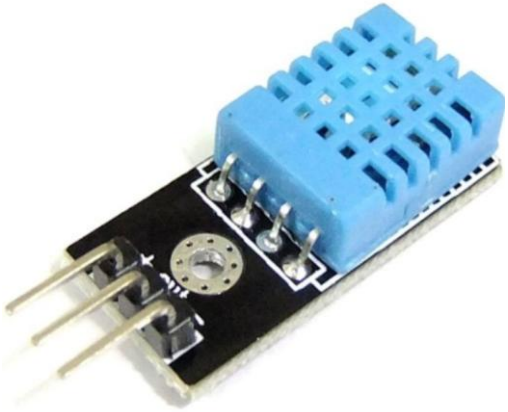
Cảm biến nhiệt độ, độ ẩm, áp suất BME280

Hình 4: Cảm biến nhiệt độ, độ ẩm, áp suất BME280

Bảng 3: Thông số kĩ thuật BME280

THÔNG SỐ KỸ THUẬT	
Giao tiếp	I2C (lên đến 3,4Mhz) và SPI(3 và 4 dây, lên đến 10Mhz).
Điện áp	1,71V đến 3,6V
Phạm vi đo độ ẩm	0 ~ 100 %
Phạm vi đo nhiệt độ	-40 ° C ~ + 85 ° C
Độ trễ	Độ ẩm tương đối 2%
Phạm vi đo áp suất	300 ~ 1100 hPa
Độ chính xác áp suất tuyệt đối	± 1hPa
Độ chính xác nhiệt độ tuyệt đối	± 0,5 ° C (ở 25 ° C)

Cảm biến nhiệt độ, độ ẩm DHT11

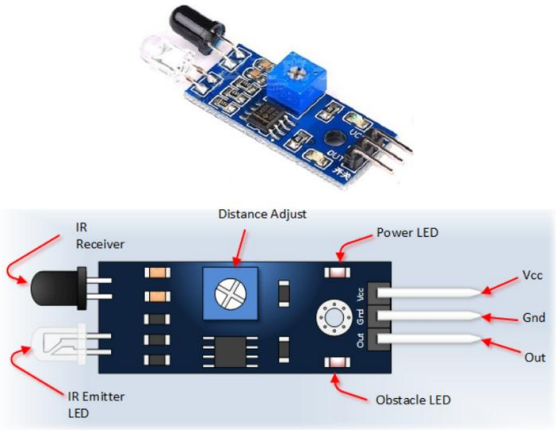


Hình 5: Cảm biến nhiệt độ, độ ẩm DHT11

Bảng 4: Thông số kĩ thuật DHT11

THÔNG SỐ KỸ THUẬT	
Điện áp hoạt động	3,3V - 5V
Dòng điện max	5mA
Dãi độ ẩm hoạt động	20 → 80%, độ chính xác ±5%
Dãi nhiệt độ hoạt động	0 → 50°C, độ chính xác ± 2°C
Khoảng cách truyền tối đa	20m
Kích thước	15,5 x 12 x 5,5 mm

Cảm biến vật cản hồng ngoại



Hình 6: Cảm biến vật cản hồng ngoại IR

Bảng 5: Thông số kĩ thuật cảm biến IR

THÔNG SỐ KỸ THUẬT	
IC so sánh	LM393
Điện áp	3,3V - 5V
Dòng hoạt động	> 20mA
Nhiệt độ làm việc	-10℃ ~ +50℃
Khoảng cách phát hiện	2 - 30 cm
Ngõ ra	Digital (0 và 1)
Kích thước	3,2 x 1,4 cm

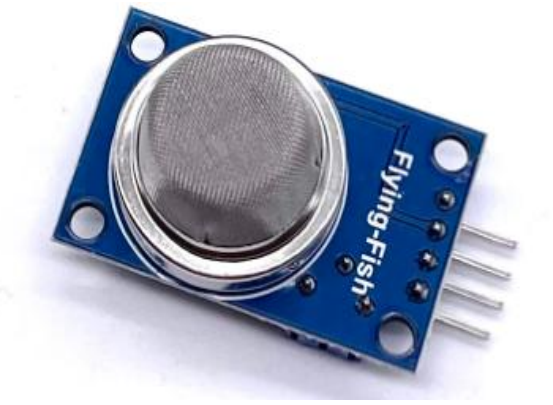
Cảm biến ánh sáng BH1750

Hình 7: Cảm biến ánh sáng BH1750

Bảng 6: Thông số kĩ thuật BH1750

THÔNG SỐ KỸ THUẬT	
Model	GY-302
Điện áp nguồn	3 - 5 VDC
Điện áp giao tiếp	TTL 3,3 - 5VDC
Chuẩn giao tiếp	I2C
Khoảng đo	1 - 65535 Lux
Ngõ ra	Analog, Digital
Kích thước	13,9 x 18,5 mm

Cảm biến khí gas

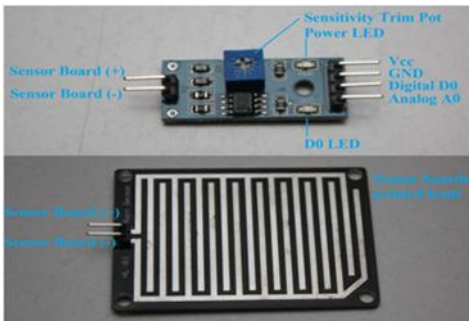


Hình 8: Cảm biến khí gas

Bảng 7: Thông số kĩ thuật cảm biến khí gas

THÔNG SỐ KỸ THUẬT	
Nguồn hoạt động	5V
Loại khí dò	LPG, iso-butan, propan và LNG
Chip	IC LM393 và MQ6
Loại dữ liệu	Analog
Phạm vi hoạt động	Khá rộng
Tốc độ phản hồi	Nhanh và độ nhạy cao
Kích thước	32 x 22 x 27mm

Cảm biến mưa



Hình 9: Cảm biến mưa

Bảng 8: Thông số kĩ thuật cảm biến mưa

THÔNG SỐ KỸ THUẬT	
Điện áp hoạt động	3V - 5V
Chip	IC LM393
Dòng hoạt động	15mA
Ngõ ra	Digital /Analog
Lá chắn	Sử dụng vật liệu chất lượng cao FR-04 hai mặt, bề mặt mạ niken, chống oxy hóa
Độ nhạy	Có biến áp để điều chỉnh

Camera EZVIZ C6N:

Hình 10: Camera EZVIZ C6N

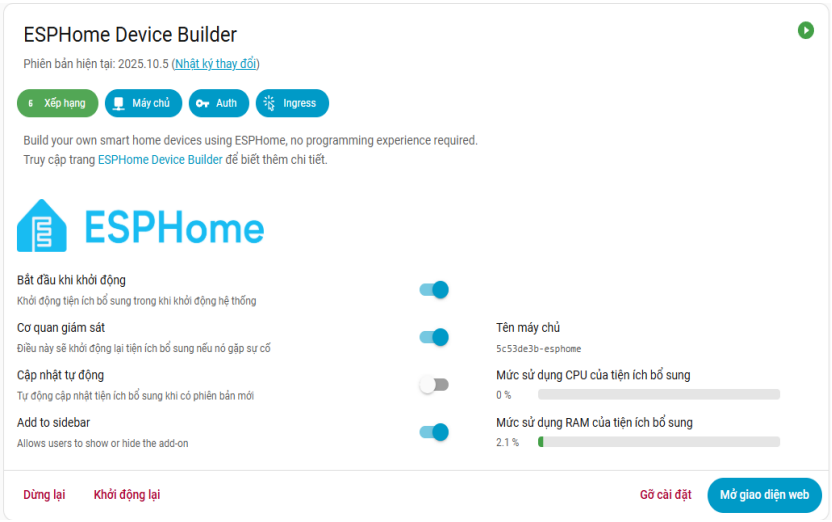
Ezviz C6N Pro 2k là một chiếc camera wifi trong nhà thông minh - lý tưởng để giám sát và bảo vệ ngôi nhà bạn. Với độ phân giải 2k (3MP), camera cho hình ảnh khá sắc nét và chi tiết, dễ dàng nhìn rõ toàn bộ không gian trong phòng. Thiết bị hỗ trợ xoay/trượt 360° nên bạn có thể quan sát toàn bộ khu vực mà không cần đặt nhiều camera; phù hợp để theo dõi trẻ nhỏ, thú cưng hay tài sản. Thêm vào đó, C6N Pro 2K còn có các tính năng "thông minh" như phát hiện chuyển động, phát hiện hình dáng người tự động (AI), chế độ đêm với màu/ hồng ngoại — giúp bạn kiểm soát tốt ngay cả khi trời tối. Nó cũng hỗ trợ đàm thoại 2 chiều, cho phép bạn nghe — nói với người ở nhà qua ứng dụng; và ghi hình + ghi âm, kèm khe thẻ nhớ tới 512 GB để lưu lại dữ liệu.

Bảng 9: Thông số kỹ thuật của Camera EZVIZ C6N

THÔNG SỐ KỸ THUẬT	
Độ phân giải hình ảnh	3MP (2K)
Ống kính	4mm, khẩu độ F1.6
Góc độ	Xoay 360 độ xem toàn cảnh
Chức năng	Tuần tra, ghi hình vào ban đêm, phát hiện chuyển động.
Hỗ trợ thẻ MicroSD	512GB , EZVIZ Cloud

2.2. Các Add on được sử dụng trong Home Assistant OS:

2.2.1. ESPHome Device Compiler:



Hình 11: ESPHome Device Compiler

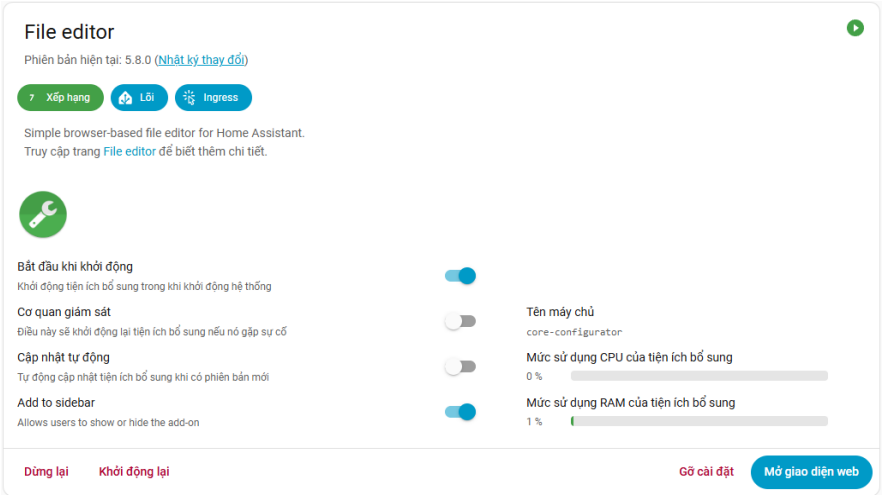
ESPHome Device Compiler là một add-on dành cho Home Assistant, cho phép người dùng biên dịch (compile) các cấu hình ESPHome trực tiếp bên trong hệ thống mà không cần cài đặt môi trường phát triển ESPHome riêng trên máy tính. Add-on này đặc biệt hữu ích khi Home Assistant được triển khai trên các thiết bị có tài nguyên hạn chế như Raspberry Pi hoặc chạy trong môi trường Docker, nơi việc cài thêm công cụ bên ngoài không thuận tiện.

Các điểm nổi bật:

- Biên dịch firmware ngay trong trong Home Assistant: Tự động chuyển đổi các file cấu hình YAML của ESPHome thành file firmware (.bin) sẵn sàng để nạp vào thiết bị, tích hợp liền mạch với giao diện của Home Assistant.
- Hỗ trợ nhiều loại thiết bị ESP: Cho phép tạo firmware cho ESP8266, ESP32 và các dòng vi điều khiển tương thích khác.
- Quản lý đơn giản: Không yêu cầu cài ESPHome CLI hay các thư viện phát triển trên máy tính cá nhân. Tất cả quá trình biên dịch được xử lý trong Add-on trên Home Assistant OS.

- Tối ưu hiệu năng và dễ triển khai: Môi trường biên dịch được cấu hình sẵn giúp rút ngắn thời gian thiết lập và giảm gánh nặng cho hệ thống chính.

2.2.2. File Editor:



Hình 12: File Editor

File Editor là một add-on dành cho Home Assistant, cho phép người dùng chỉnh sửa trực tiếp các file cấu hình (YAML) hoặc bất kỳ file văn bản nào ngay trong giao diện web của Home Assistant. Đây là công cụ rất hữu ích khi bạn muốn tùy chỉnh hệ thống mà không cần truy cập SSH, SFTP hay các phương thức kết nối phức tạp khác.

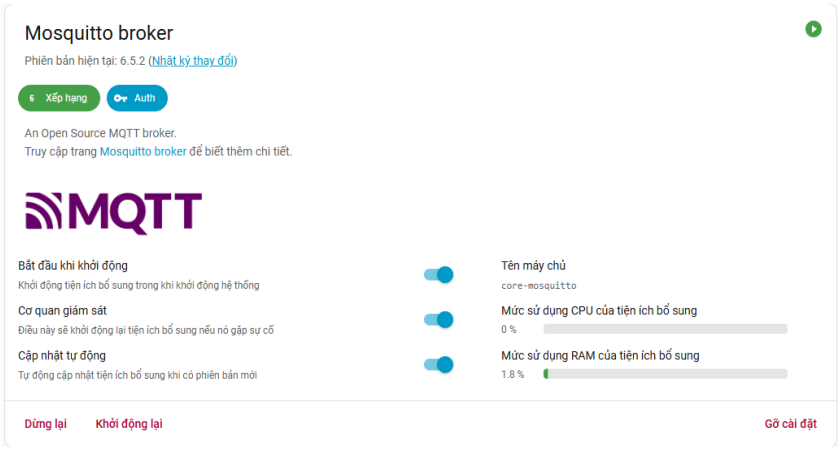
Chức năng chính của File Editor:

- Chỉnh sửa file cấu hình Home Assistant: Người dùng có thể mở và chỉnh sửa các file như configuration.yaml, automations.yaml, scripts.yaml… trực tiếp trong giao diện, giúp việc cấu hình trở nên nhanh chóng và thuận tiện.
- Giao diện trực quan, dễ sử dụng: File Editor cung cấp trình soạn thảo văn bản đơn giản với khả năng tô màu cú pháp (syntax highlighting) cho YAML, JSON và nhiều loại file khác, hỗ trợ người dùng dễ dàng đọc và chỉnh sửa nội dung.
- Tích hợp chặt chẽ với Home Assistant: Là một add-on chính thức, File Editor hoạt động liền mạch bên trong Home Assistant. Bạn có thể mở

trình chỉnh sửa ngay từ sidebar mà không cần rời khỏi môi trường quản lý hệ thống.

- Tìm kiếm và thay thế: Công cụ hỗ trợ chức năng tìm kiếm và thay thế nội dung trong file, giúp tối ưu thời gian chỉnh sửa, đặc biệt với những file cấu hình lớn.
- Quản lý file và thư mục: Bạn có thể duyệt cây thư mục trong khu vực cấu hình của Home Assistant, tạo hoặc xoá file, chỉnh sửa nội dung, và quản lý cấu trúc thư mục một cách trực quan.
- Bảo mật: File Editor hỗ trợ các cơ chế xác thực của Home Assistant, đảm bảo chỉ người dùng có quyền mới được phép truy cập và chỉnh sửa các file quan trọng của hệ thống.

2.2.3. Giao thức MQTT - MQTT Mosquitto:



Hình 13: Giao thức MQTT - MQTT Mosquitto

Mosquitto là một phần mềm mã nguồn mở dùng để triển khai giao thức MQTT (Message Queuing Telemetry Transport) – giao thức truyền thông nhẹ, tối ưu cho các thiết bị IoT. Trong hệ sinh thái Home Assistant, Mosquitto thường được sử dụng như MQTT broker, đóng vai trò trung gian nhận và phân phối thông điệp giữa các thiết bị hỗ trợ MQTT và hệ thống Home Assistant.

Chức năng chính của Mosquitto:

- Hỗ trợ các phiên bản MQTT tiêu chuẩn: Mosquitto tuân thủ đầy đủ các phiên bản MQTT phổ biến như 3.1, 3.1.1 và MQTT 5.0, đảm bảo khả năng tương thích với hầu hết thiết bị và dịch vụ IoT trên thị trường.

- Nhẹ và tối ưu hiệu suất: Phần mềm được thiết kế để hoạt động hiệu quả trên các hệ thống tài nguyên thấp như Raspberry Pi, thiết bị nhúng hoặc các môi trường chạy Home Assistant trong Docker.
- Tính bảo mật cao: Mosquitto hỗ trợ xác thực bằng username/password và mã hóa kết nối bằng SSL/TLS. Giúp đảm bảo an toàn dữ liệu khi truyền tải trong mạng IoT.
- Khả năng mở rộng linh hoạt: Có thể triển khai trên nhiều nền tảng: máy tính cá nhân, máy chủ, Home Assistant OS hoặc các thiết bị nhúng. Broker có thể kết nối cùng lúc nhiều client MQTT như cảm biến, công tắc, thiết bị ESPHome, Tasmota…
- Hỗ trợ QoS (Quality of Service)

Cung cấp các mức QoS 0, 1 và 2 để đảm bảo độ tin cậy của việc truyền nhận thông điệp tùy theo nhu cầu hệ thống:

- QoS 0: Gửi một lần, không đảm bảo nhận

- QoS 1: Đảm bảo tin nhắn được nhận ít nhất một lần

- QoS 2: Đảm bảo nhận đúng một lần, mức tin cậy cao nhất

2.2.4. Samba Share:

Hình 14: Samba Share

Samba Share là một add-on trong Home Assistant OS, cho phép người dùng truy cập và chỉnh sửa trực tiếp các file cấu hình của Home Assistant từ máy tính Windows, macOS hoặc Linux qua mạng nội bộ. Công cụ này giúp việc

quản lý cấu hình trở nên thuận tiện hơn so với việc chỉ sử dụng File Editor trong giao diện web.

Chức năng chính của Add on:

- Cho phép truy cập thư mục cấu hình config/ của Home Assistant từ máy tính.
- Hỗ trợ chỉnh sửa, xóa, tạo mới file và thư mục.
- Dễ dàng upload các file media, file custom component, hoặc các script YAML phức tạp.
- Hỗ trợ sao lưu và phục hồi cấu hình nhanh chóng, đặc biệt hữu ích khi dự án gồm nhiều file và thư mục.

Ưu điểm của Add on:

- Truy cập trực tiếp từ máy tính bằng phần mềm quen thuộc (VS Code, Notepad++, Sublime…).
- Hỗ trợ nhiều hệ điều hành: Windows, macOS, Linux.
- Thao tác nhanh, không cần đăng nhập SSH hay SFTP.
- Tối ưu cho việc chỉnh sửa nhiều file hoặc upload dữ liệu lớn mà File Editor khó xử lý.

2.2.5. Home Assistant Community Store

Hình 15: Home Assistant Community Store

HACS (Home Assistant Community Store) là một cửa hàng cộng đồng dành cho Home Assistant, cho phép người dùng dễ dàng tìm, cài đặt và quản lý các tích hợp (integration), giao diện Lovelace, theme, plugin, và custom component do cộng đồng phát triển.

HACS giúp mở rộng chức năng của Home Assistant vượt ra ngoài các tích hợp chính thức mà Home Assistant cung cấp.

Chức năng chính:

- Cài đặt tích hợp và plugin của cộng đồng: Bao gồm các custom component, Lovelace cards, automation, theme, icon pack…
- Cập nhật tự động: HACS có thể kiểm tra và thông báo khi có phiên bản mới của các gói đã cài.
- Quản lý dễ dàng: Giao diện trực quan, tích hợp liền mạch vào Home Assistant, giúp cài đặt hoặc xóa các plugin chỉ với vài cú click.
- Tìm kiếm nhanh: Cho phép tìm kiếm các dự án cộng đồng theo tên, tác giả, hoặc loại gói.

Ưu điểm:

- Mở rộng khả năng của Home Assistant mà không cần can thiệp trực tiếp vào file cấu hình.
- Cho phép cập nhật các tích hợp cộng đồng nhanh chóng, giảm rủi ro lỗi cấu hình thủ công.
- Hỗ trợ nhiều loại nội dung: từ custom component, Lovelace card, theme cho đến blueprint automation.
- Hỗ trợ cài đặt trực tiếp từ GitHub repository của cộng đồng.

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

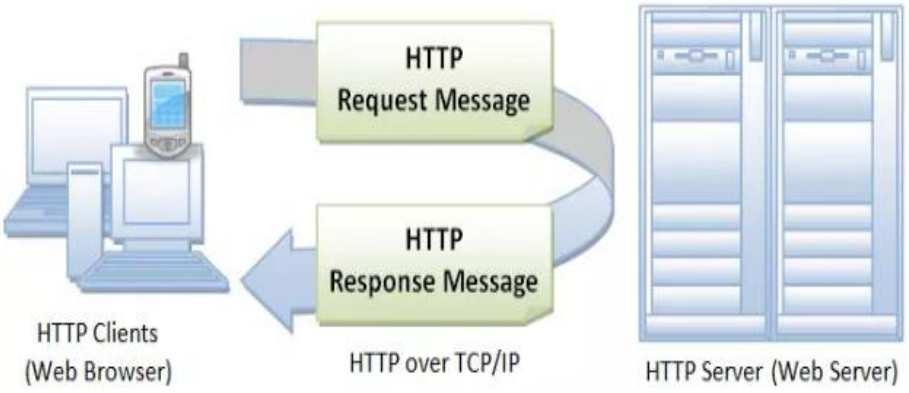
3.1. Sơ đồ giao thức mạng:

3.1.1. Các giao thức kết nối được sử dụng:

3.1.1.1. Giao thức HTTP:

Giao thức HTTP (HyperText Transfer Protocol) là một giao thức ứng dụng, được phát triển nhằm mục đích trao đổi siêu văn bản giữa các hệ thống trên các trang WEB. HTTP đóng vai trò quan trọng trong việc truyền tải tài liệu như văn bản, hình ảnh, video, âm thanh và nhiều dữ liệu khác trên internet.

HTTP là giao thức hoạt động dựa trên mô hình yêu cầu - phản hồi giữa máy khách (client) và máy chủ (server). Khi người dùng nhập một URL, trình duyệt (client) gửi yêu cầu tới máy chủ chứa nội dung đó và máy chủ sẽ phản hồi với dữ liệu được yêu cầu. Dưới đây là cấu trúc cơ bản của một yêu cầu HTTP:



Hình 16: Hoạt động của giao thức HTTP

HTTP hỗ trợ nhiều phương thức yêu cầu khác nhau, mỗi phương thức có mục đích riêng trong việc xử lý tài nguyên trên máy chủ:

- **GET**: Yêu cầu truy xuất tài nguyên từ máy chủ (ví dụ: tải một trang web).
- **POST**: Gửi dữ liệu đến máy chủ để tạo mới hoặc cập nhật tài nguyên.
- **PUT**: Tương tự POST, nhưng thường dùng để cập nhật toàn bộ tài nguyên.
- **DELETE**: Xóa tài nguyên trên máy chủ.

- **HEAD**: Tương tự GET nhưng không trả về phần nội dung chính của tài nguyên, chỉ trả về các tiêu đề.
- **PATCH**: Cập nhật một phần tài nguyên (khác với PUT cập nhật toàn bộ).
- **OPTIONS**: Lấy thông tin về các phương thức HTTP mà máy chủ hỗ trợ cho tài nguyên cụ thể.

Địa chỉ tài nguyên (URL): là địa chỉ của tài nguyên trên mạng mà client muốn truy cập. Một URL tiêu chuẩn bao gồm:

- **Scheme** (HTTP/HTTPS): Định nghĩa giao thức truyền tải.
- **Host** (Tên miền): Địa chỉ của máy chủ lưu trữ tài nguyên.
- **Path** (Đường dẫn): Vị trí cụ thể của tài nguyên trên máy chủ.

Các tiêu đề yêu cầu chứa thông tin về trình duyệt, hệ điều hành, ngôn ngữ ưa thích, định dạng dữ liệu mong muốn và các thông số liên quan khác mà máy khách gửi kèm theo yêu cầu.

Sau khi xử lý yêu cầu, máy chủ trả về phản hồi bao gồm:

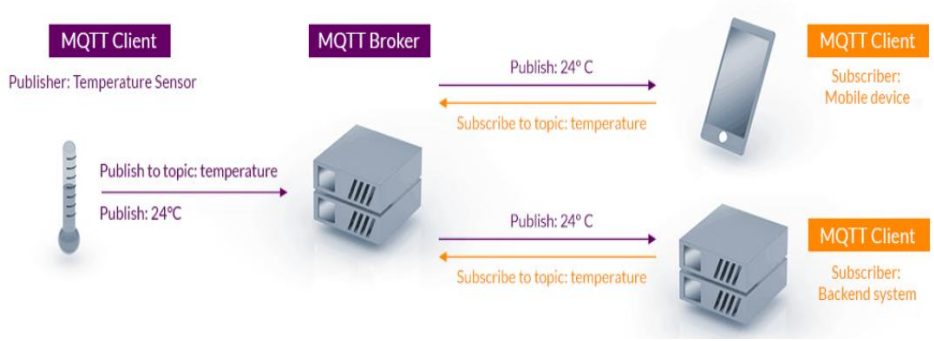
- **Mã trạng thái HTTP** (HTTP Status Code): Cho biết kết quả của yêu cầu. Ví dụ: 200 (OK), 404 (Not Found), 500 (Internal Server Error).
- **Nội dung phản hồi** (Response Body): Phần dữ liệu thực tế mà máy chủ trả về, có thể là HTML, JSON, XML hoặc tệp tin khác.
- **Tiêu đề phản hồi** (Response Headers): Chứa các thông tin bổ sung về phản hồi như định dạng nội dung, mã hóa, hoặc các thông tin điều khiển bộ nhớ đệm (cache control).

3.1.1.2. Giao thức MQTT:

Giao thức MQTT là một giao thức nhắn tin dựa trên các tiêu chuẩn hoặc một bộ các quy tắc được sử dụng cho việc giao tiếp máy với máy. Cảm biến thông minh, thiết bị đeo trên người và các thiết bị Internet vạn vật (IoT) khác thường phải truyền và nhận dữ liệu qua mạng có tài nguyên và băng thông hạn chế. Các thiết bị IoT này sử dụng MQTT để truyền dữ liệu vì giao thức này dễ

triển khai và có thể giao tiếp dữ liệu IoT một cách hiệu quả. MQTT hỗ trợ nhắn tin giữa các thiết bị với Cloud và từ Cloud đến thiết bị.

Cách thức MQTT hoạt động:



Hình 17: Cách thức MQTT hoạt động

Chức năng chính:

Giao thức nhắn tin nhẹ (Lightweight Messaging Protocol): MQTT là một giao thức nhắn tin rất nhẹ, tối ưu hóa cho việc truyền tải dữ liệu qua các mạng có băng thông hạn chế và thiết bị có tài nguyên hạn chế (ví dụ như cảm biến, thiết bị IoT).

Dựa trên mô hình Publisher-Subscriber: MQTT sử dụng mô hình Publisher Subscriber (Phát hành - Đăng ký). Máy chủ MQTT (broker) nhận và phân phối các thông điệp giữa các Publisher (người gửi) và Subscriber (người nhận). Các Publisher gửi thông điệp tới một Topic (chủ đề), và các Subscriber đăng ký nhận thông điệp từ một hoặc nhiều chủ đề đó.

Nhắn tin bất đồng bộ (Asynchronous Messaging): Các thiết bị gửi và nhận thông điệp mà không cần đồng bộ hóa trực tiếp giữa các thiết bị. Điều này giúp giảm độ trễ và cải thiện hiệu quả trong các hệ thống phân tán.

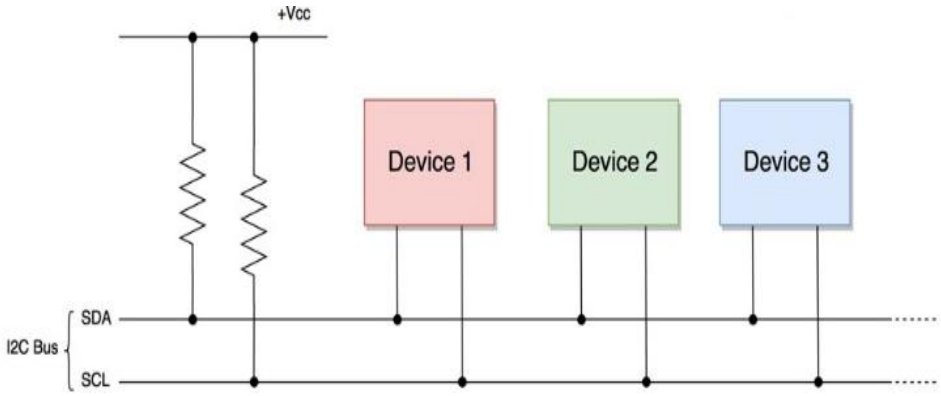
Mạng nhúng và tài nguyên hạn chế: MQTT được thiết kế để hoạt động tốt trong các môi trường mạng với tài nguyên hạn chế (như băng thông thấp hoặc các thiết bị IoT với bộ xử lý yếu). Nó sử dụng một cơ chế tối giản để giảm thiểu lượng băng thông cần thiết cho việc truyền tải dữ liệu.

Đảm bảo tin cậy (QoS - Quality of Service): MQTT hỗ trợ 3 mức độ QoS để đảm bảo chất lượng dịch vụ của các thông điệp, từ đảm bảo không bị mất tin nhắn cho đến việc đảm bảo mỗi tin nhắn chỉ gửi một lần.

Kết nối duy trì (Keep-alive) và thông báo thất bại: MQTT có cơ chế để duy trì kết nối giữa thiết bị và máy chủ broker trong thời gian dài, và nếu kết nối bị gián đoạn, nó có thể tự động thử kết nối lại.

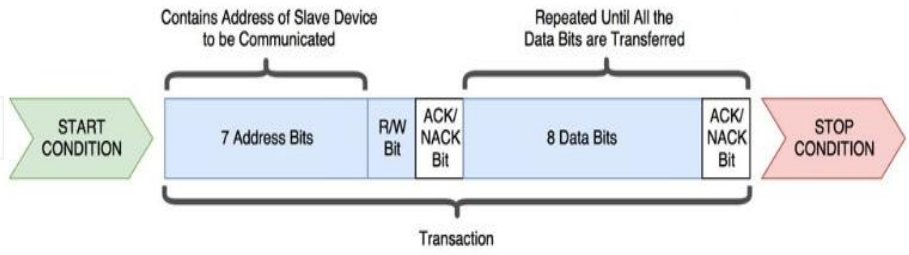
3.1.1.3. Giao thức I2C:

Giao thức I2C (Inter - Intergrated Circuit) là một giao thức truyền thông nối tiếp được thiết kế để cho phép nhiều thiết bị giao tiếp trên cùng một bus chỉ với hai dây là SDA (Serial Data Line) để truyền dữ liệu và SCL (Serial Clock Line) để đồng bộ xung clock do thiết bị master tạo ra. I2C là giao tiếp half-duplex, dữ liệu truyền theo một chiều tại một thời điểm, và các thiết bị trên giao tiếp I2C được kết nối cùng đường SCL và SDA như sau:



Hình 18: Thiết bị kết nối qua giao thức I2C

Cách hoạt động của giao thức I2C:



Hình 19: Cách thức hoạt động của I2C protocol

Cụ thể:

- Điều kiện bắt đầu: Đường dây SDA chuyển mức điện áp từ cao thành thấp, trước khi đường dây SCL chuyển từ mức cao xuống thấp.
- Điều kiện kết thúc: Đường dây SDA chuyển mức điện áp từ thấp thành cao, sau khi đường dây SCL chuyển từ mức thấp lên cao.
- Khung địa chỉ: Gồm một chuỗi 7 hoặc 10 bit duy nhất cho mỗi thiết bị Slave, các máy chủ Master sẽ dựa vào đây để xác định thiết bị mà mình cần giao tiếp.
- Bit đọc / ghi: Một bit đơn để hệ thống biết rằng Master đang gửi dữ liệu đến Slave (mức điện áp thấp) hay là Master cần nhận dữ liệu từ Slave (mức điện áp cao).
- Bit ACK / NACK: Một một khung dữ liệu bên trong tin nhắn đều được theo dõi bởi các bit Acknowledge / No-Acknowledge này. Nếu địa chỉ khung hoặc khung dữ liệu được gửi thành công, bit ACK sẽ được gửi trả về cho người gửi từ thiết bị nhận.

Địa chỉ của giao thức I2C: các thiết bị Slave muốn biết rằng thiết bị Master đang gửi dữ liệu đến nó chứ không phải một thiết bị Slave khác thì thiết bị Master sẽ cần phải biết địa chỉ I2C của thiết bị Slave đó. Đó là lý do phải có địa chỉ I2C và khung địa chỉ luôn là khung đầu tiên sau bit bắt đầu của một tin nhắn mới.

Bit đọc/ghi:

- Nếu Master muốn gửi dữ liệu tới Slave: Bit đọc / ghi có mức điện áp thấp.
- Nếu Master muốn nhận dữ liệu từ Slave: Bit đọc / ghi có mức điện áp cao.

Khung dữ liệu:

Khi Master nhận được bit ACK từ Slave, hệ thống sẽ gửi khung dữ liệu đầu tiên đi.

Khung dữ liệu này luôn dài 8 bit, trong đó các bit quan trọng được đặt ở phần đầu. Mỗi một khung dữ liệu đều được đặt đằng sau bit ACK/NACK để xác

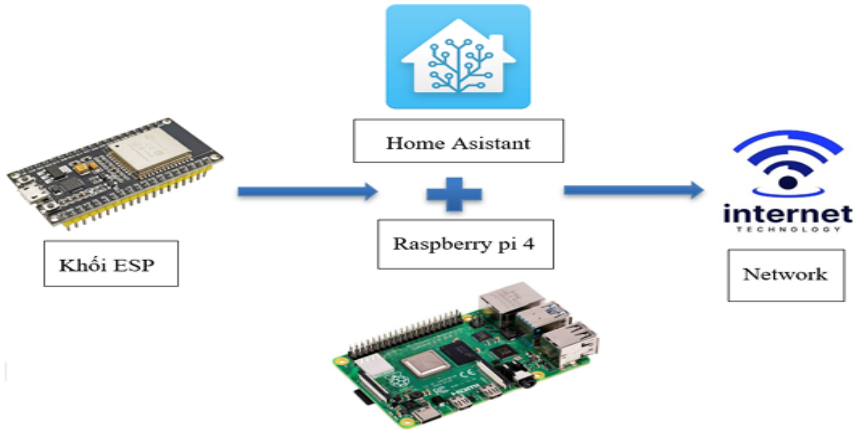
nhận là khung đã được gửi và nhận thành công chưa. Bit ACK này phải được nhận bởi một Master hoặc Slave tùy theo từng trường hợp, thì khung dữ liệu khác mới được gửi đi.

Sau khi đã gửi đi tất cả các khung dữ liệu, Master có thể gửi điều kiện kết thúc tới Slave để ngừng quá trình truyền dữ liệu, thông qua việc chuyển điện áp từ mức thấp lên mức cao trên SCL và sau đó là trên đường dây SDA.

Các bước truyền dữ liệu của giao thức I2C:

- Master gửi điều kiện bắt đầu đến tất cả các thiết bị Slave bằng cách đổi điện áp của đường dây SDA từ mức cao sang mức thấp, trước khi chuyển dây SCL từ cao thành thấp.
- Master gửi địa chỉ của Slave (ở dạng 7bit hoặc 10bit) mà nó muốn liên lạc cùng với bit đọc / ghi cho tất cả các thiết bị Slave
- Mỗi Slave so sánh địa chỉ của mình với địa chỉ được Master gửi đến. Nếu trùng khớp địa chỉ thì Slave sẽ gửi trả bit ACK về, thôn qua việc kéo dây SDA từ mức cao xuống mức thấp 1 bit. Nếu khác địa chỉ thì Slave không làm gì cả.
- Master gửi hoặc nhận khung dữ liệu.
- Khi đã gửi khung dữ liệu thành công, các thiết bị nhận dữ liệu sẽ gửi trả một bit ACK cho thiết bị gửi để xác nhận rằng quá trình gửi nhận khung dữ liệu đã thành công.

3.1.2. Sơ đồ mạng tổng quan:



Hình 20: Sơ đồ tổng quan của hệ thống

Khối ESP trong hệ thống bao gồm các module ESP32, hoạt động như các node ngoại vi kết nối không dây qua WiFi, có khả năng điều khiển các thiết bị điện như công tắc relay, đèn, quạt và thu thập dữ liệu từ nhiều loại cảm biến khác nhau như nhiệt độ, độ ẩm, ánh sáng hay cảm biến chuyển động. Các module này gửi dữ liệu về Raspberry Pi 4 đang chạy Home Assistant OS (HAOS) để xử lý tập trung, giúp việc điều khiển thiết bị diễn ra nhanh chóng, chính xác và đồng bộ. Nhờ đó, các node ESP32 vừa thực hiện thu thập thông tin môi trường, vừa tham gia vào các kịch bản tự động hóa của hệ thống.

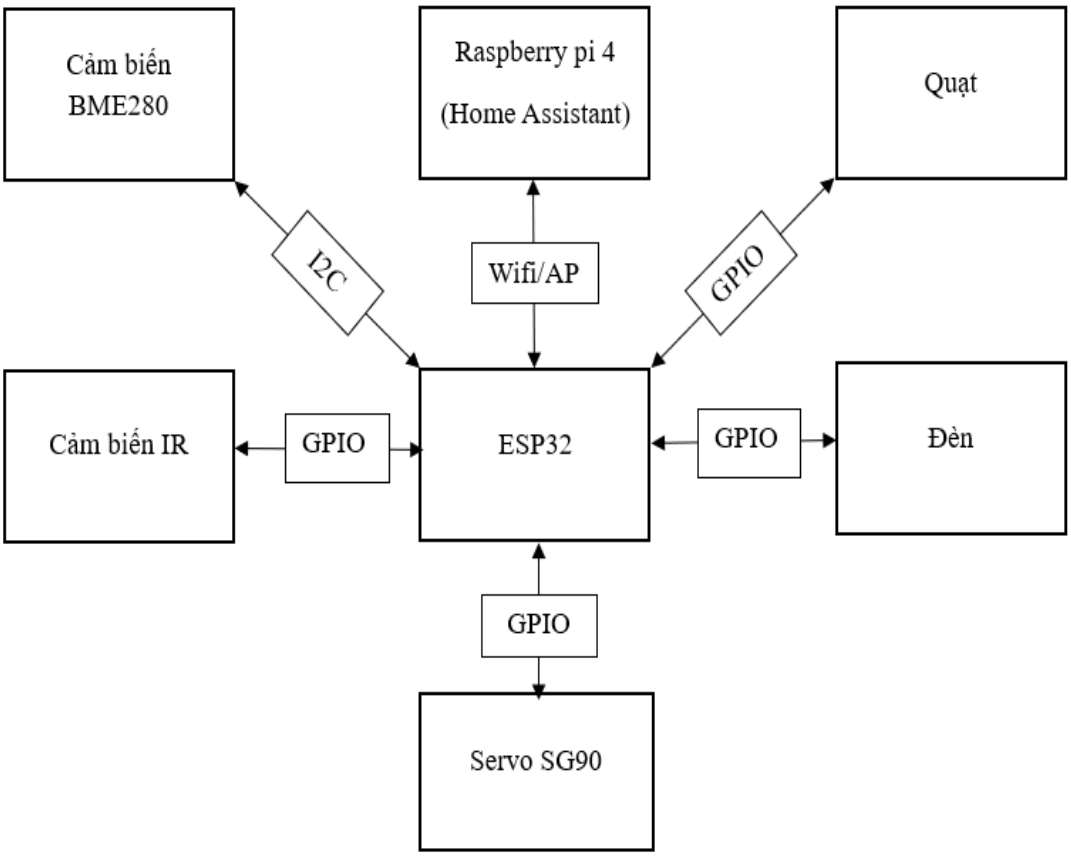
Raspberry Pi 4 đảm nhiệm vai trò trung tâm điều phối, tiếp nhận và xử lý dữ liệu từ các module ESP32, đồng thời thực hiện các lệnh điều khiển thiết bị điện dựa trên các kịch bản do người dùng thiết lập. Khi chạy trên Home Assistant OS, Raspberry Pi 4 trở thành cổng IoT mạnh mẽ, có khả năng quản lý, đồng bộ hóa và chuyển đổi dữ liệu giữa các thiết bị ngoại vi và mạng trung tâm. HAOS cung cấp các giao diện API, MQTT, HTTP để tương tác với các node ESP32 và tận dụng các tiện ích Add-on, plugin cũng như các tích hợp sẵn có, giúp việc quản lý hệ thống trở nên trực quan, linh hoạt và dễ dàng mở rộng khi thêm các thiết bị mới.

Home Assistant là nền tảng mã nguồn mở mạnh mẽ, miễn phí, được cộng đồng rộng rãi hỗ trợ, giúp kết nối, đồng bộ và quản lý nhiều thiết bị IoT từ các nhà sản xuất khác nhau. Nền tảng này cung cấp các tính năng tự động hóa nâng cao, cho phép lập lịch, tạo kịch bản cảnh báo, điều khiển từ xa và phân quyền truy cập. Đồng thời, Home Assistant hỗ trợ giao diện điều khiển trên cả máy tính và thiết bị di động, tích hợp các nền tảng đa dạng như MQTT, ESPHome, Zigbee, Google Assistant, Alexa, giúp người dùng giám sát và quản lý toàn bộ hệ thống Smart Home một cách tiện lợi và hiệu quả.

Với kiến trúc này, hệ thống không chỉ đáp ứng nhu cầu điều khiển và giám sát nội bộ mà còn mở rộng khả năng tích hợp AI và các mô hình xử lý dữ liệu thông minh, cho phép nhận diện đối tượng, phát hiện sự kiện, cảnh báo an ninh và tối ưu hóa tự động hóa. Hệ thống có thể mở rộng linh hoạt bằng cách bổ sung thêm các module ESP32 hoặc cảm biến mới mà không làm thay đổi cấu trúc.

3.2. Sơ đồ kết nối của hệ thống giữa ESP32 và Home Assistant OS:

3.2.1. Sơ đồ kết nối của phòng khách:



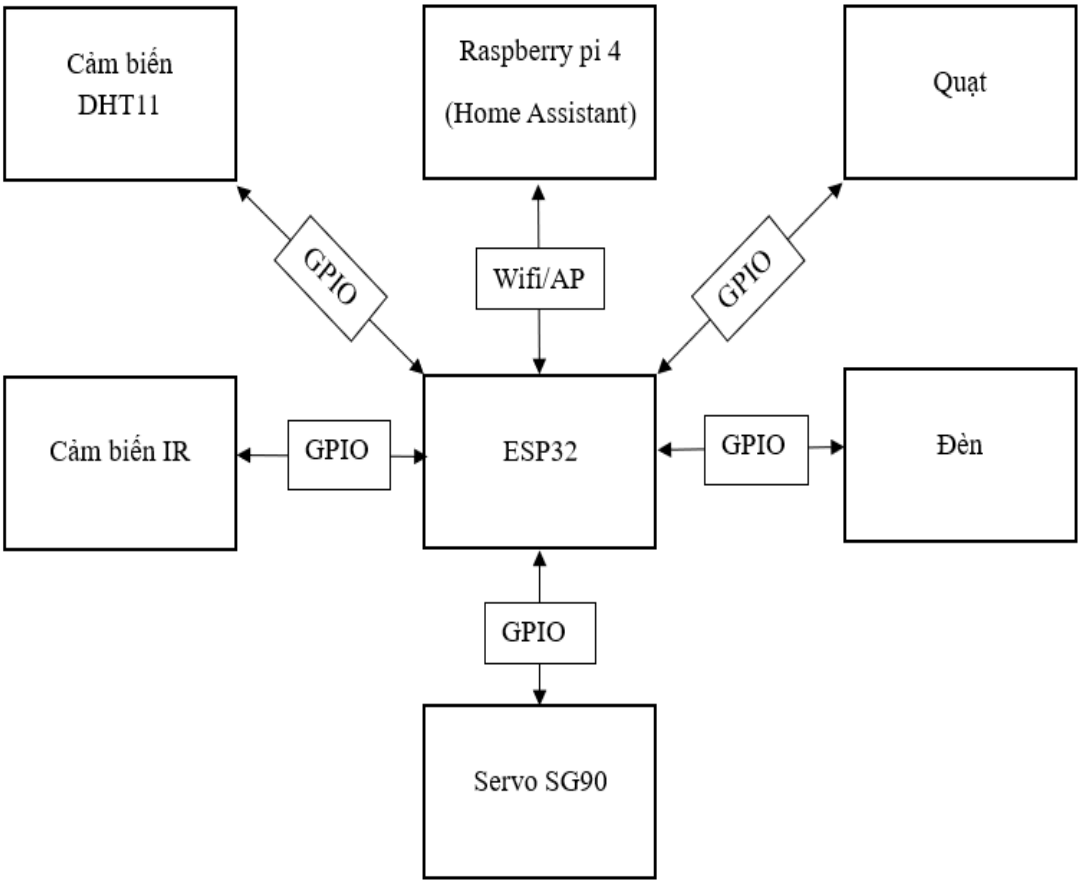
Hình 21: Sơ đồ kết nối của phòng khách

Kết nối I2C: Giữa cảm biến BME280 và ESP32, đây là giao tiếp số hai dây SDA và SCL, cho phép truyền dữ liệu giữa hai thiết bị. Cảm biến BME280 sẽ gửi dữ liệu đo được (nhiệt độ, độ ẩm, áp suất) tới ESP32 thông qua giao tiếp này.

Kết nối GPIO: ESP32 và các thiết bị cảm biến IR, hệ thống đèn và quạt cùng với servo SG90 được điều khiển thông qua các chân GPIO trên ESP32 để bật tắt các thiết bị và thu thập dữ liệu từ cảm biến.

Kết nối Wifi/API: ESP32 kết nối với Raspberry pi 4 chạy hệ điều hành Home Assistant thông qua kết nối không dây Wifi, ESP32 sẽ thu thập các dữ liệu từ các thiết bị trong khu vực qua các chân kết nối để gửi dữ liệu lên Home Assistant và điều khiển thiết bị từ xa.

3.2.2. Sơ đồ kết nối của phòng ngủ:

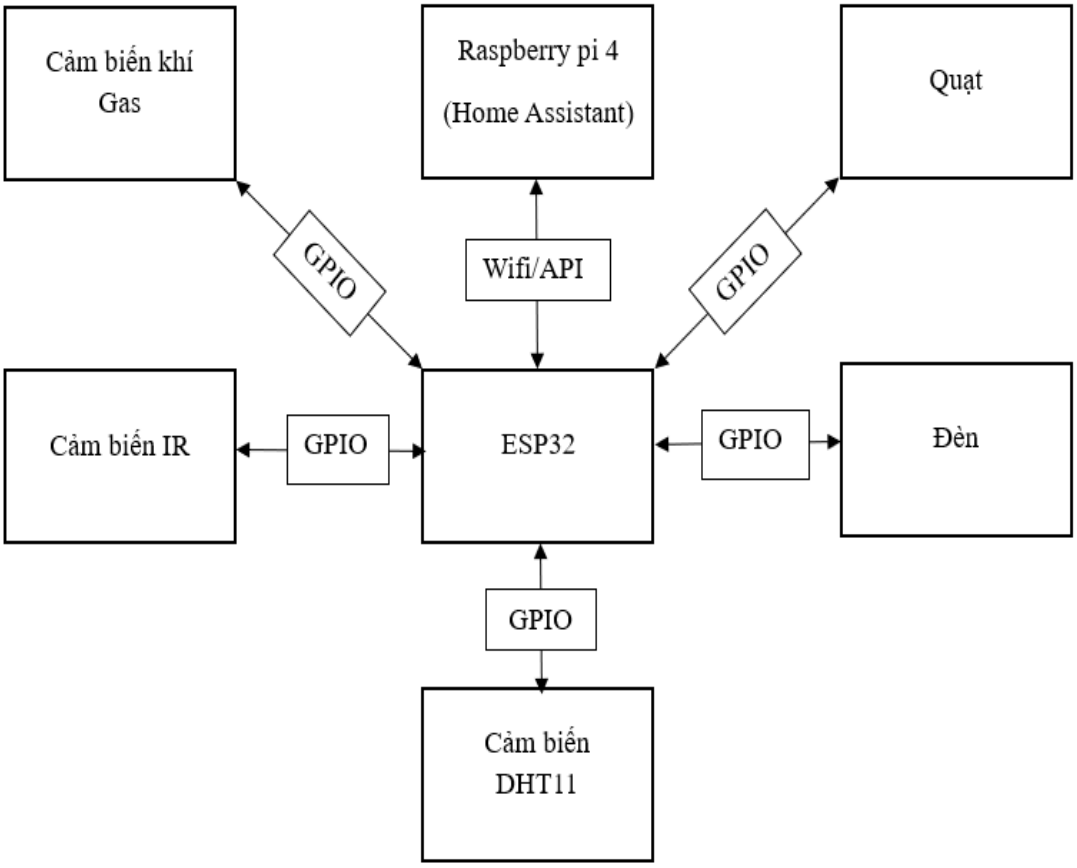


Hình 22: Sơ đồ kết nối của phòng ngủ

Kết nối GPIO: ESP32 kết nối với các cảm biến, đèn, quạt và servo SG90 thông qua các chân GPIO, cho phép thu thập dữ liệu từ các thiết bị và điều khiển chúng trực tiếp trên hệ thống. Dữ liệu thu thập được sẽ được ESP32 gửi lên Raspberry pi 4 chạy Home Assistant OS, đồng thời Home Assistant OS cũng có thể điều khiển các thiết bị này từ xa.

Kết nối Wifi/API: ESP32 kết nối với Raspberry pi 4 chạy hệ điều hành Home Assistant thông qua kết nối không dây Wifi, ESP32 sẽ thu thập các dữ liệu từ các thiết bị trong khu vực qua các chân kết nối để gửi dữ liệu lên Home Assistant và điều khiển thiết bị từ xa.

3.2.3. Sơ đồ kết nối của phòng bếp:

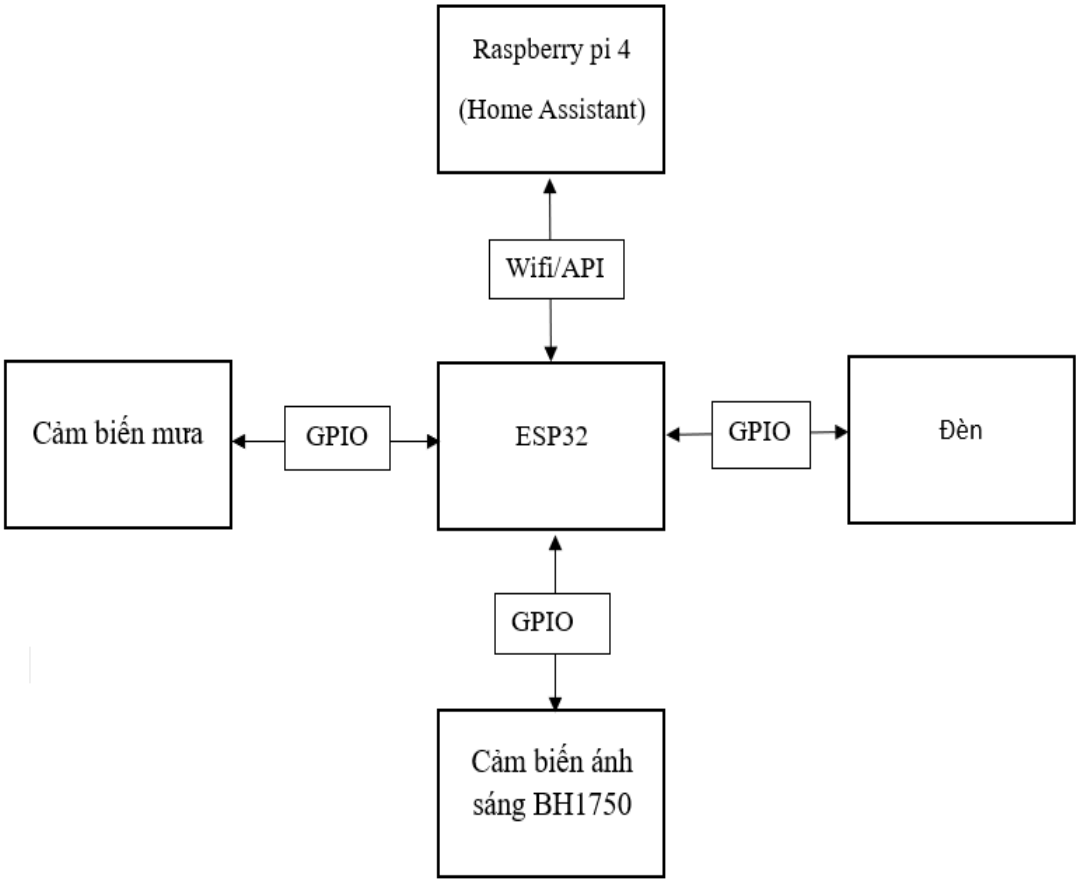


Hình 23: Sơ đồ kết nối của phòng bếp

Kết nối GPIO: ESP32 kết nối với các thiết bị cảm biến, đèn, quạt, servo SG90 thông qua các chân GPIO để thu thập dữ liệu và điều khiển các thiết bị trong hệ thống, ESP32 gửi các dữ liệu này lên Raspberry pi 4 chạy Home Assistant OS và điều khiển thiết bị trực tiếp từ Home Assistant OS.

Kết nối Wifi/API: ESP32 giao tiếp với Raspberry Pi 4 chạy Home Assistant thông qua kết nối Wi-Fi, thu thập dữ liệu từ các thiết bị qua GPIO và gửi trực tiếp lên Home Assistant để giám sát. Thay vì sử dụng giao thức MQTT, ESP32 kết nối với Home Assistant OS thông qua Add-on ESPHome API, cho phép giao tiếp nội bộ, ổn định và nhanh chóng. Nhờ vậy, dữ liệu từ các cảm biến được cập nhật gần như thời gian thực. Hệ thống này không chỉ giảm thiểu độ trễ so với các giải pháp dựa trên MQTT mà còn đơn giản hóa việc tích hợp nhiều loại thiết bị khác nhau vào cùng một mạng nội bộ.

3.2.4. Sơ đồ kết nối của khu vực ngoài sân:



Hình 24: Sơ đồ kết nối của khu vực ngoài sân

Kết nối GPIO: ESP32 kết nối với các thiết bị cảm biến, đèn thông qua các chân GPIO để thu thập dữ liệu và điều khiển các thiết bị trong hệ thống, ESP32 gửi các dữ liệu này lên Raspberry pi 4 chạy Home Assistant OS và điều khiển thiết bị trực tiếp từ Home Assistant OS.

Kết nối Wi-Fi/API: ESP32 giao tiếp với Raspberry Pi 4 chạy Home Assistant thông qua kết nối Wi-Fi, thu thập dữ liệu từ các thiết bị qua GPIO và gửi trực tiếp lên Home Assistant để giám sát. Thay vì sử dụng giao thức MQTT, ESP32 kết nối với Home Assistant OS thông qua Add-on ESPHome API, cho phép giao tiếp nội bộ, ổn định và nhanh chóng. Nhờ vậy, dữ liệu từ các cảm biến như ánh sáng hay trạng thái thiết bị như đèn được cập nhật gần như thời gian thực.

3.3. Kết nối Camera EZVIZ với Home Assistant OS:

Trong đề tài này, việc tích hợp camera EZVIZ vào hệ thống Home Assistant OS được thực hiện thông qua EZVIZ Cloud. Camera EZVIZ là thiết bị giám sát thông minh phổ biến, cung cấp hình ảnh chất lượng cao và các tính năng nhận diện chuyển động, nhận diện con người. EZVIZ Cloud là nền tảng đám mây chính thức của hãng EZVIZ, cho phép camera truyền tải dữ liệu hình ảnh, trạng thái thiết bị và thông báo sự kiện từ xa.

Khi sử dụng Home Assistant OS, hệ thống không kết nối trực tiếp với camera qua mạng nội bộ (LAN) mà thông qua giao thức API của EZVIZ Cloud. Mô hình kết nối được mô tả như sau: Camera EZVIZ → EZVIZ Cloud → Home Assistant OS

Camera kết nối Wi-Fi và đăng ký với EZVIZ Cloud. Home Assistant OS sử dụng EZVIZ Integration để đăng nhập tài khoản EZVIZ, từ đó truy xuất video trực tiếp, snapshot và các sự kiện phát hiện chuyển động. Cách tiếp cận này cho phép người dùng quản lý camera mà không cần cấu hình port, NAT hay RTSP, giúp quá trình triển khai nhanh chóng và thuận tiện.

Quy trình thiết lập trong Home Assistant bao gồm các bước:

1. Đăng ký tài khoản EZVIZ và đảm bảo camera đã được thêm vào ứng dụng EZVIZ.
2. Lấy mã xác thực (verification code) từ tem trên camera nếu cần.
3. Thêm Integration EZVIZ trong Home Assistant, nhập thông tin tài khoản và mã xác thực.

Sau khi hoàn tất, camera sẽ xuất hiện trong Home Assistant như một thiết bị thông minh. Việc kết nối qua EZVIZ Cloud có ưu điểm là cấu hình đơn giản, ổn định, không phụ thuộc vào mạng LAN hoặc port forwarding. Tuy nhiên, phương thức này có hạn chế về độ trễ hình ảnh so với các kết nối trực tiếp như RTSP, và phụ thuộc vào hoạt động của máy chủ EZVIZ. Trong bối cảnh triển khai đồ án, phương thức này vẫn phù hợp với yêu cầu thu thập dữ liệu, giám sát và tích hợp automation cơ bản của hệ thống IoT.

3.4. Xử lý ảnh với thuật toán AI:

3.4.1. Xử lý ảnh với nhận diện đối tượng bằng Yolo:

Hình 25: YOLOv11

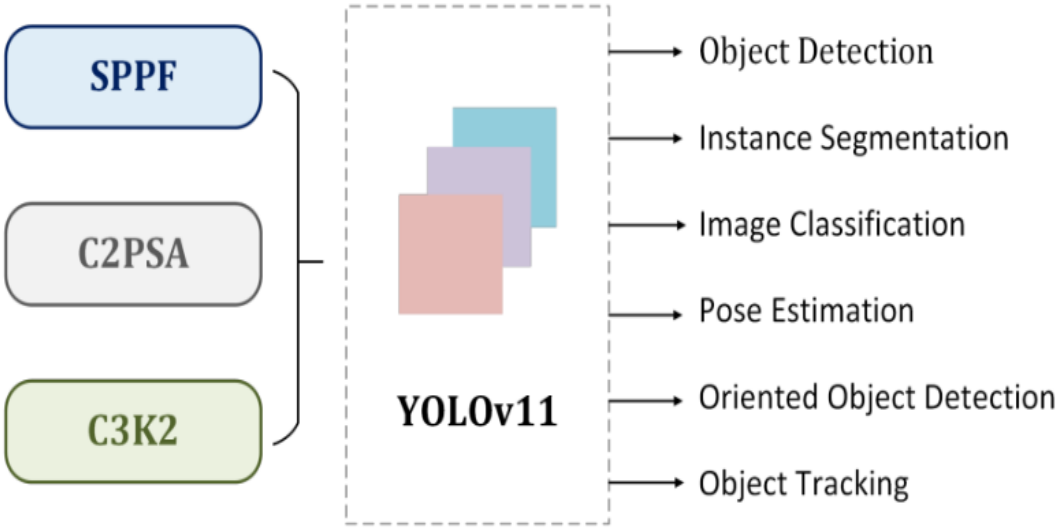
3.4.1.1. Giới thiệu tổng quan về mô hình YOLOv11:

Trong hệ thống IoT của đề tài, việc xử lý ảnh và nhận diện đối tượng đóng vai trò quan trọng để nâng cao khả năng thông minh và tự động hóa. YOLO (You Only Look Once) là một trong những mô hình học sâu (deep learning) phổ biến, được thiết kế để phát hiện và phân loại đối tượng trong hình ảnh theo thời gian thực. Điểm mạnh của YOLO là tốc độ xử lý nhanh và khả năng nhận diện nhiều loại đối tượng trên cùng một khung hình.

Mô hình YOLO hoạt động theo nguyên tắc xem toàn bộ hình ảnh một lần duy nhất (single-pass), khác với các phương pháp truyền thống dựa trên sliding window hoặc region proposal. Cấu trúc mạng YOLO chia hình ảnh thành các lưới (grid) và dự đoán đồng thời các hộp giới hạn (bounding box) cùng với xác suất của các lớp đối tượng trong từng ô lưới. Nhờ đó, mô hình vừa đạt độ chính xác cao, vừa giảm độ trễ, phù hợp với các ứng dụng giám sát thời gian thực như camera an ninh trong hệ thống IoT.

Kiến trúc YOLOv11 được ra mắt năm 2024, giới thiệu kiến trúc hiệu quả hơn với các khối C3K2, SPFF (Spatial Pyramid Pooling Fast) và các cơ chế chú ý nâng cao như C2PSA. YOLOv11 được thiết kế để tăng cường khả năng phát hiện đối tượng nhỏ và cải thiện độ chính xác trong khi vẫn duy trì tốc độ suy luận thời gian thực mà YOLO nổi tiếng.

3.4.1.2. Kiến trúc YOLOv11:



Hình 26: Kiến trúc của YOLOv11

YOLOv11 giới thiệu các thành phần khối C3k2 (Cross Stage Partial with kernel size 2), SPFF (Spatial Pyramid Pooling - Fast) và C2PSA (Convolution block with Parallel Spatial Attention). Những kỹ thuật mới này nâng cao khả năng trích xuất tính năng và cải thiện độ chính xác của mô hình giúp mô hình tốt hơn cho các trường hợp sử dụng phát hiện đối tượng theo thời gian thực.

Backbone: là một thành phần quan trọng của kiến trúc YOLO, chịu trách nhiệm trích xuất các đặc trưng từ hình ảnh đầu vào ở nhiều mức độ khác nhau. Quá trình này bao gồm việc xếp chồng các lớp convolution và các block chuyên biệt để tạo ra các bản đồ đặc trưng (feature map) ở nhiều độ phân giải khác nhau.

Convolution layers: sử dụng cấu trúc tương tự với các YOLO phiên bản trước để giảm kích thước hình ảnh. Một cải tiến quan trọng trong YOLOv11 là sự ra đời của block C3K2, thay thế block C2F được sử dụng ở phiên bản trước. Block C3k2 là một triển khai hiệu quả hơn về mặt tính toán của Cross Stage Partial (CSP) Bottleneck, sử dụng hai khối convolution nhỏ thay vì một convolution lớn như trong YOLOv8.

SPPF: giúp model tích hợp thông tin từ nhiều ảnh scale khác nhau trong một khu vực đặc trưng, đặc biệt hữu ích khi đối tượng có các kích thước khác nhau (nhỏ, vừa, lớn).

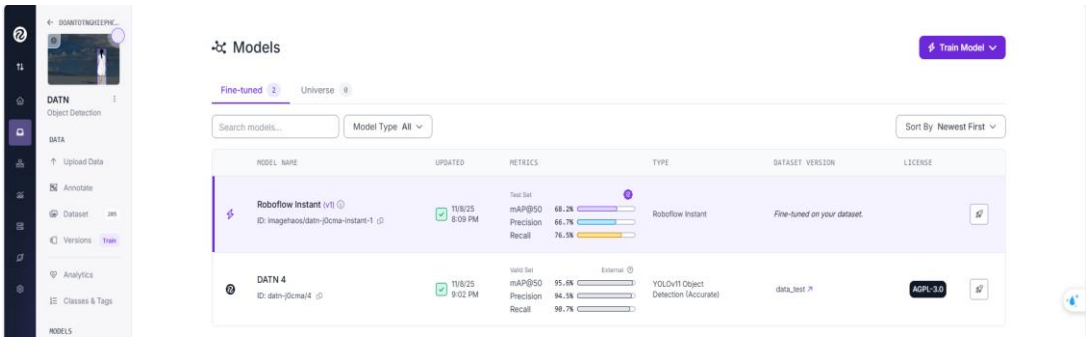
C2PSA: giúp model tập trung vào các vùng quan trọng của ảnh (ví dụ các đối tượng chính, tránh các vật không liên quan).

C3K2: YOLOv11 sử dụng khối C3K2 để xử lý việc trích xuất đặc trưng ở các giai đoạn khác nhau của Backbone. Các kernel nhỏ 3x3 giúp tính toán hiệu quả hơn, đồng thời vẫn giữ khả năng nắm bắt các đặc trưng quan trọng trong hình ảnh.

3.4.1.3. Cách sử dụng YOLOv11 cho nhận diện đối tượng:

Bước 1: YOLOv11 sử dụng Pytorch với câu lệnh: “pip install torch torchvision ultralytics”.

Bước 2: Sử dụng Roboflow để gán nhãn cho các tệp hình ảnh làm dữ liệu huấn luyện mô hình theo mô hình YOLOv11.

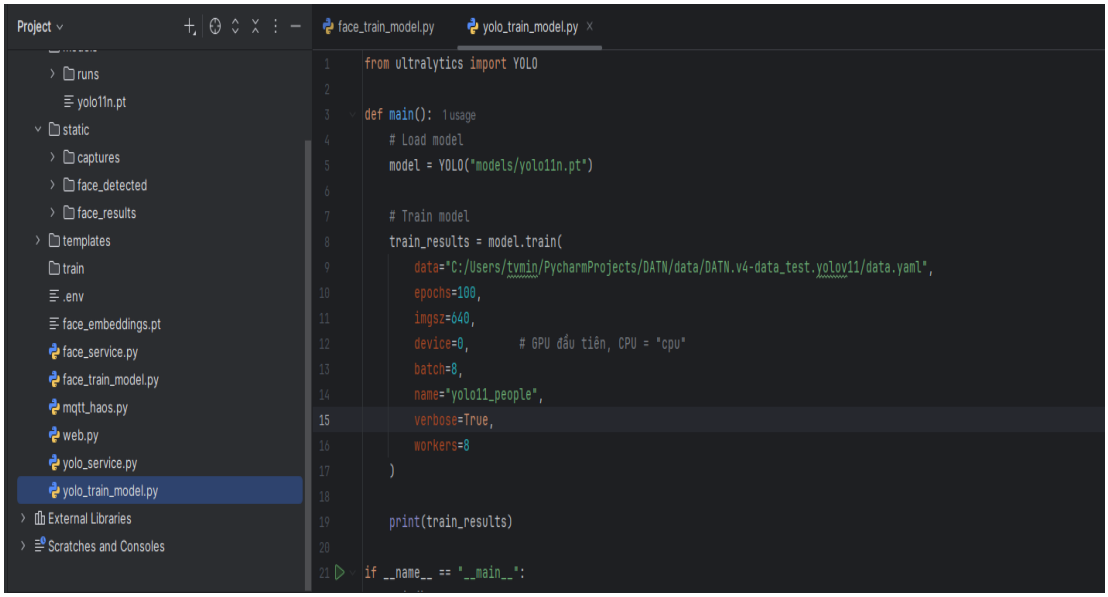


Hình 27: Train mô hình với Roboflow

Roboflow là một nền tảng mạnh mẽ hỗ trợ toàn bộ quy trình xây dựng và triển khai các mô hình computer vision (thị giác máy tính), đặc biệt cho các tác vụ như phát hiện đối tượng (object detection), phân loại ảnh (image classification) và phân đoạn ảnh (image segmentation). Hỗ trợ tải ảnh, gán nhãn và tổ chức hình ảnh theo từng mục đích của dự án. Hỗ trợ tăng cường dữ liệu với các chức năng: xoay, lật, thay đổi độ sáng,.. giúp tăng tính đa dạng của hình ảnh. Hỗ trợ chuẩn hóa dữ liệu, xuất dataset với mô hình theo nhiều framework.

Bước 3: Sau khi có tập dữ liệu với Roboflow, ta sẽ tải về và huấn luyện trên máy tính:

File yolo_train_model.py: Huấn luyện mô hình YOLO từ tệp dataset lấy từ Roboflow



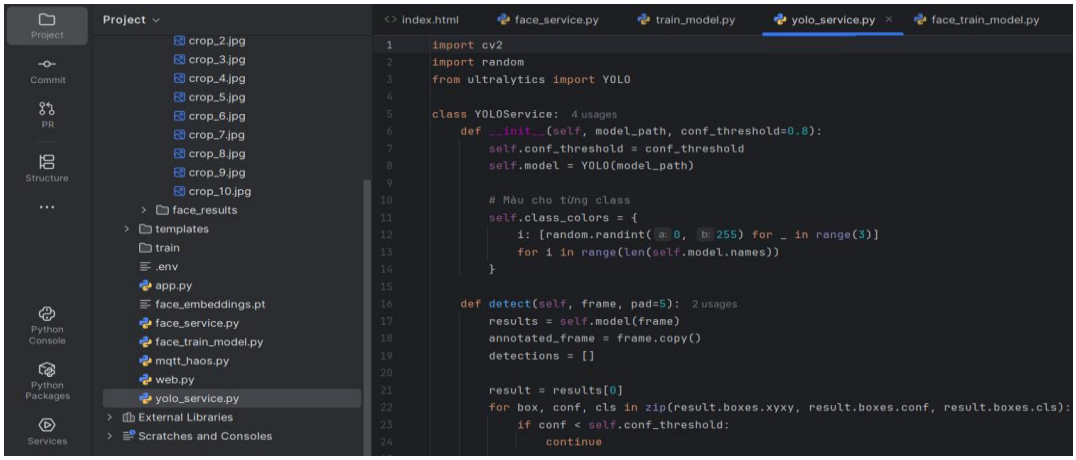
Hình 28: YOLO_train

Mô tả chương trình:

File yolo_train_model.py là một script Python được thiết kế để huấn luyện mô hình YOLOv11 cho bài toán nhận diện đối tượng, chẳng hạn như phát hiện người hoặc các đối tượng cụ thể trong dataset của dự án. Script sử dụng thư viện Ultralytics YOLO để tải mô hình khởi tạo (yolo11n.pt) và thực hiện fine-tune trên dữ liệu riêng. Trong quá trình huấn luyện, các tham số quan trọng được cấu hình bao gồm đường dẫn dataset (data.yaml) chứa thông tin về các lớp và file train/test, số epoch huấn luyện (epochs=100), kích thước ảnh đầu vào (imgsz=640), batch size (batch=8), thiết bị tính toán (device=0 cho GPU hoặc "cpu"), tên project huấn luyện (name="yolo11_people") và số luồng xử lý dữ liệu (workers=8). Sau khi huấn luyện hoàn tất, kết quả về loss, mAP và các metric khác được in ra để người dùng đánh giá hiệu năng mô hình.

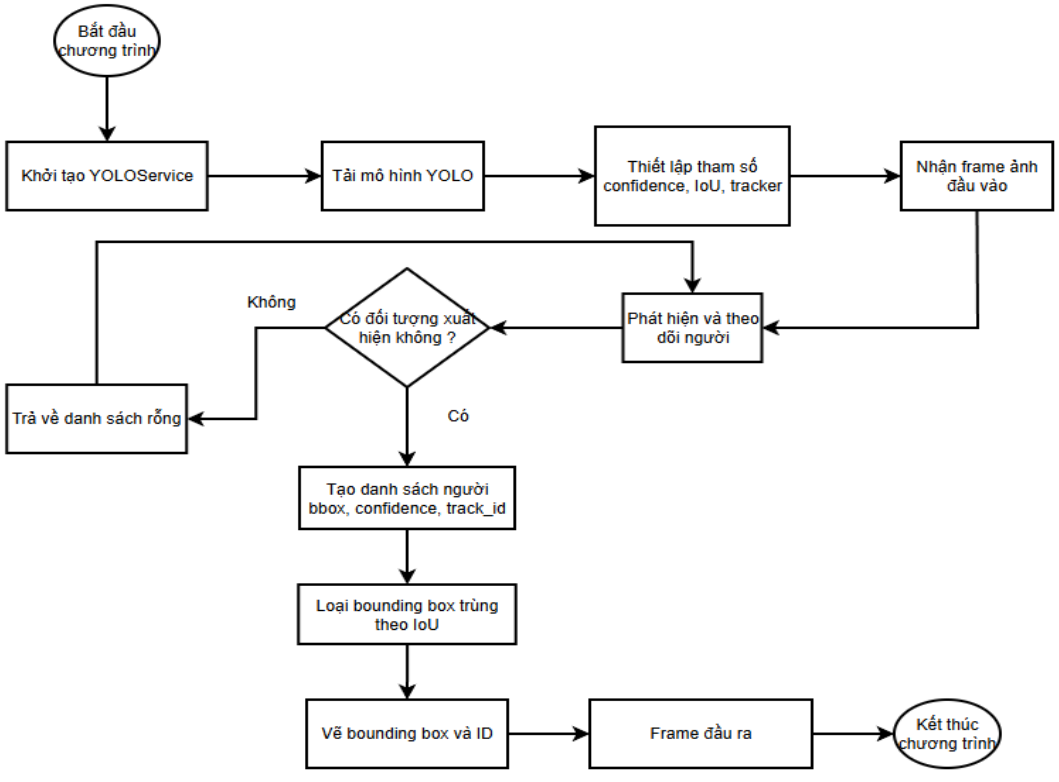
File yolo_service.py: là module Python được thiết kế để xử lý và nhận diện đối tượng từ ảnh hoặc video stream từ Camera sử dụng mô hình YOLO của

Ultralytics. Module này định nghĩa lớp YOLOService, cho phép tải mô hình YOLO đã huấn luyện, thiết lập ngưỡng confidence để lọc các dự đoán kém tin cậy, và gán màu sắc riêng cho từng lớp đối tượng để vẽ trực quan bounding box trên ảnh.



Hình 29: YOLO_service

Lưu đồ giải thuật:



Hình 30: Sơ đồ giải thuật của file yolo_service.py

Phân tích chương trình:

Chương trình khởi tạo lớp dịch vụ YOLOService, trong đó mô hình YOLO được tải lên và các tham số quan trọng như ngưỡng confidence, ngưỡng IOU và chế độ tracker được thiết lập. Sau khi hoàn tất bước khởi tạo, hệ thống liên tục nhận frame ảnh đầu vào từ nguồn video.

Mỗi frame được đưa vào mô hình YOLO để phát hiện và theo dõi người. Tại đây, hệ thống kiểm tra xem frame ảnh có tồn tại đối tượng hợp lệ hay không. Nếu không phát hiện được đối tượng, chương trình trả về danh sách rỗng và chờ frame tiếp theo.

Trong trường hợp phát hiện được người, hệ thống tiến hành tạo danh sách đối tượng, bao gồm thông tin của ảnh vẽ bounding box, độ tin cậy (confidence) và mã định danh theo dõi (track_id). Danh sách này sau đó được xử lý để loại bỏ các bounding box trùng lặp trong quá trình phát hiện dựa trên chỉ số IOU nhằm đảm bảo mỗi người chỉ được biểu diễn bởi một bounding box duy nhất.

Cuối cùng, hệ thống vẽ bounding box và ID tương ứng lên frame ảnh và xuất ra frame đầu ra đã được xử lý, phục vụ cho việc hiển thị, lưu trữ hoặc truyền sang các module khác trong hệ thống, sau đó kết thúc chu trình xử lý frame của hiện tại.

3.4.2. Xử lý ảnh với nhận diện khuôn mặt bằng MTCNN và FaceNet:

3.4.2.1. Giới thiệu tổng quan về MTCNN:

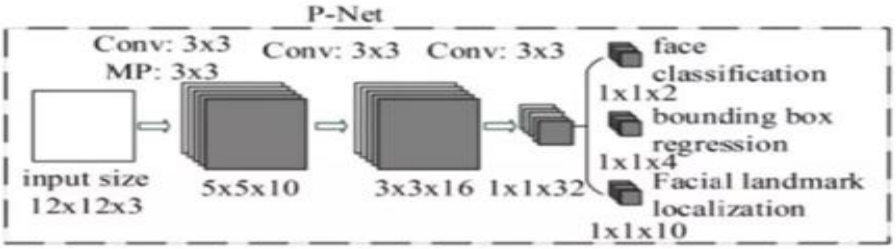
MTCNN (Multi-task Cascaded Convolutional Networks) là một mô hình mạng nơ-ron sâu được thiết kế đặc biệt để phát hiện khuôn mặt và xác định các đặc điểm trên khuôn mặt (facial landmarks). Nó thực hiện nhiệm vụ này thông qua ba bước chính được tổ chức trong ba mạng con riêng biệt:

P-Net (Proposal Network):

Sử dụng phép Resize ảnh, để tạo một loạt các bản copy từ ảnh gốc với kích cỡ khác nhau, từ to đến nhỏ, tạo thành 1 Image Pyramid.

Hình 31: Image Pyramid

Với mỗi phiên bản copy-resize của ảnh gốc, ta sử dụng Sliding Kernel 12x12x3 đi qua toàn bộ ảnh để tìm khuôn mặt, sau khi nhận thấy được ảnh khuôn mặt thì vẽ bouding box và tính confidence score (xác suất patch chứa khuôn mặt). Thực hiện loại bỏ các box có confidence threshold quá thấp (< 0.5) và loại bỏ tiếp các vùng không phải khuôn mặt thông qua kỹ thuật Non-Maximum Suppression (NMS). Những box còn lại đươc quy đổi lại về tọa độ ảnh gốc , để tiếp tục sang bước kế tiếp (R-Net).

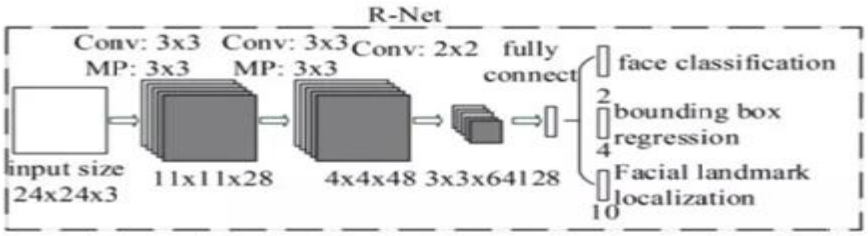


Hình 32: Lớp P-Net của MTCNN

R-Net (Refinement Network):

Nhiệm vụ là tinh chỉnh bounding box từ P-Net để xác định vị trí khuôn mặt chính xác hơn và lọc bỏ các bounding box sai hoặc không liên quan. Là mạng CNN sâu hơn P-Net, hoạt động trên các patch ảnh được crop từ bounding boxes P-Net. Tuy nhiên, mạng còn sử dụng thêm một phương pháp tên là padding, nhằm thực hiện việc chèn thêm các zero-pixels vào các phần thiếu của bounding box nếu bounding box vượt quá bên của ảnh. Với đầu vào là các ảnh đã được resize về 24x24 pixel, các hoạt động tiếp theo cũng như mạng P-Net,

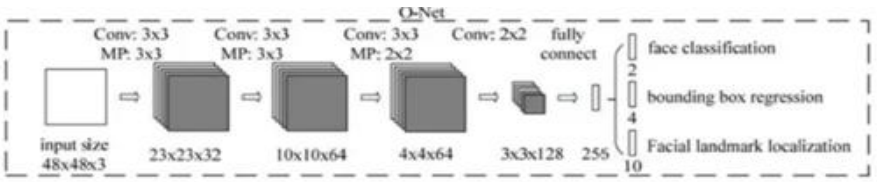
kết quả đầu ra là những tọa độ mới của các box còn lại và đưa vào mạng tiếp theo O-Net.



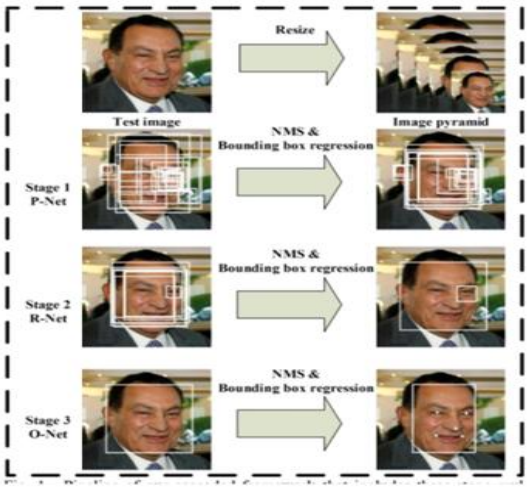
Hình 33: Lớp R-Net của MTCNN

O-Net (Output Network):

Nhiệm vụ của mạng O-Net cũng tương tự như R-Net nhưng thay đổi kích thước thành 48x48. Tuy nhiên, kết quả đầu ra của mạng lúc này không còn là tọa độ của các box nữa, mà trả về các giá trị gồm tọa độ của bounding box và các đặc điểm khuôn mặt như mắt, mũi, miệng.



Hình 34: Lớp O-Net của MTCNN



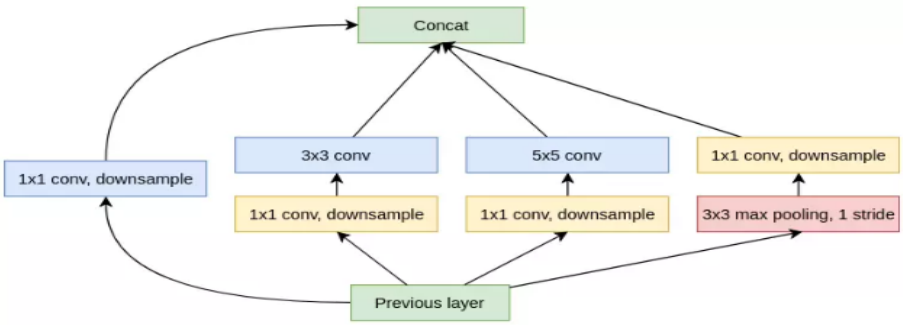
Hình 35: Toàn bộ quá trình của MTCNN

3.4.2.2. Giới thiệu tổng quan về FaceNet:

FaceNet là một mô hình học sâu được phát triển bởi Google, sử dụng để nhận diện khuôn mặt thông qua việc mã hóa khuôn mặt thành một vector đặc trưng duy nhất (embedding). Vector này có thể được sử dụng để:

Nhận diện khuôn mặt: So sánh embedding của khuôn mặt mới với các embedding đã lưu trước đó. Xác minh khuôn mặt: Kiểm tra xem hai khuôn mặt có thuộc cùng một người hay không.

Inception V1 là một kiến trúc mạng CNN được Google giới thiệu vào năm 2014, nổi bật với các khối Inception. Điểm đặc trưng của khối này là khả năng xử lý song song: cùng một đầu vào có thể được truyền qua nhiều lớp Convolution khác nhau để trích xuất các loại đặc trưng khác nhau, sau đó các kết quả này được concatenate thành một output duy nhất. Cách học song song này giúp mạng nắm bắt được nhiều chi tiết hơn và trích xuất được nhiều feature phong phú hơn so với mạng CNN truyền thống. Ngoài ra, Inception V1 còn sử dụng các lớp Convolution 1×1 để giảm số lượng tham số, từ đó giảm kích thước mạng và giúp quá trình huấn luyện trở nên nhanh hơn.



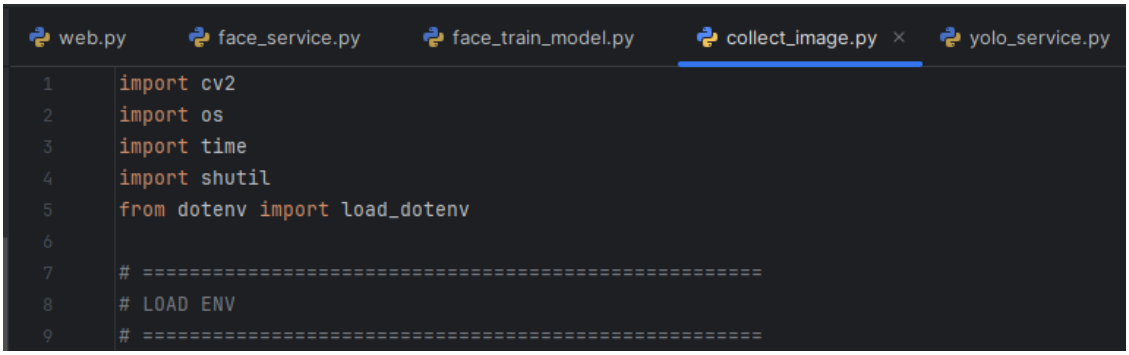
Hình 36: Cấu trúc của mạng CNN Inception V1

Cụm khuôn mặt: Nhóm các khuôn mặt tương tự nhau trong một tập dữ liệu. FaceNet sử dụng kiến trúc mạng nơ-ron sâu Inception và được huấn luyện dựa trên triplet loss để tối ưu hóa embedding:

Triplet Loss: Tối thiểu hóa khoảng cách giữa một khuôn mặt anchor và một khuôn mặt positive (cùng người) và tối đa hóa khoảng cách giữa anchor và negative (người khác).

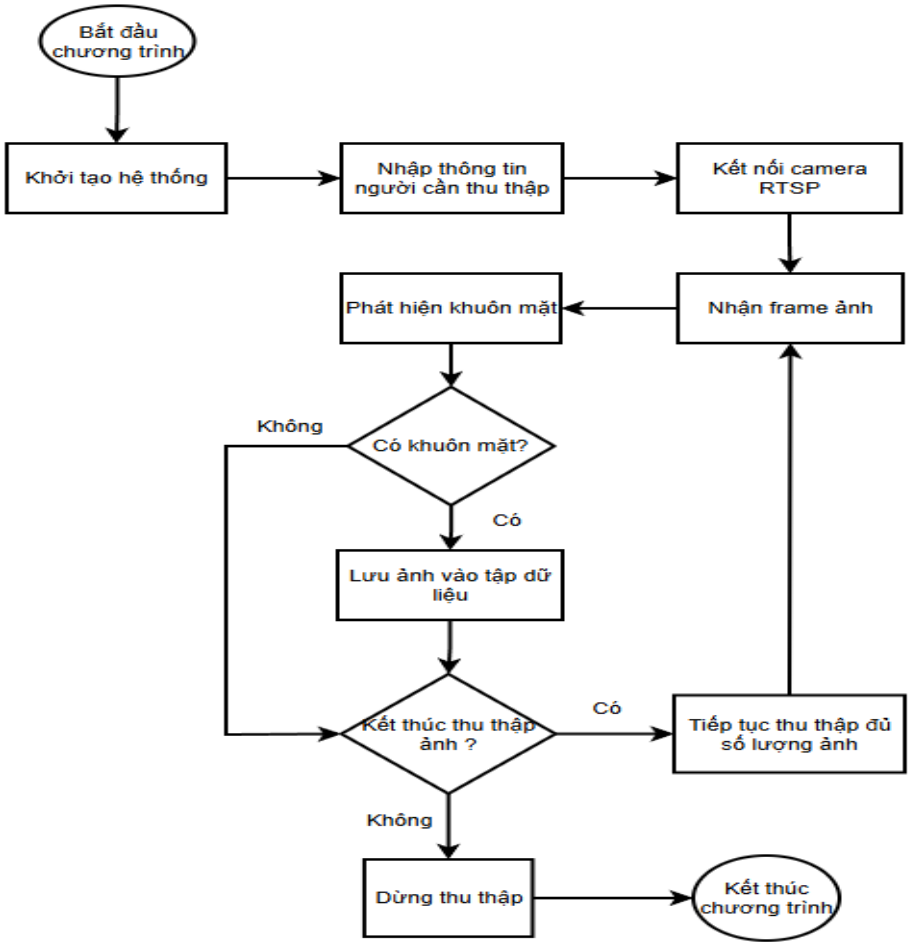
3.4.2.3. Xử lý ảnh với nhận diện khuôn mặt:

1. File collect_image.py: giúp thu thập dữ liệu khuôn mặt từ camera RTSP để phục vụ mục đích huấn luyện hoặc đăng ký khuôn mặt.



Hình 37: File collect_image.py

Lưu đồ giải thuật:



Hình 38: Lưu đồ giải thuật file collect_image.py

Phân tích chương trình:

1. cv2: Xử lý ảnh và video thời gian thực, kết nối và đọc luồng stream.

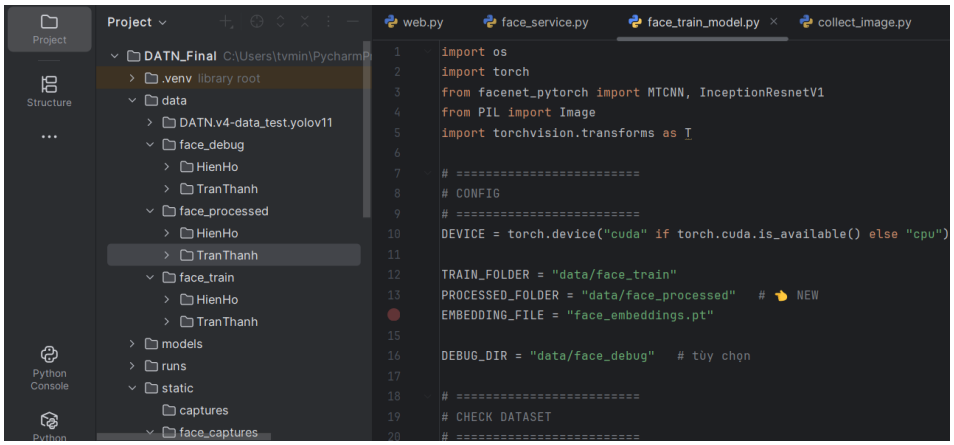
2. os: Làm việc với hệ thống tệp và biến môi trường, tạo thư mục và lưu ảnh.

3. time: Xử lý thời gian và delay.

4. shutil: Quản lý thư mục và dữ liệu của hệ thống.

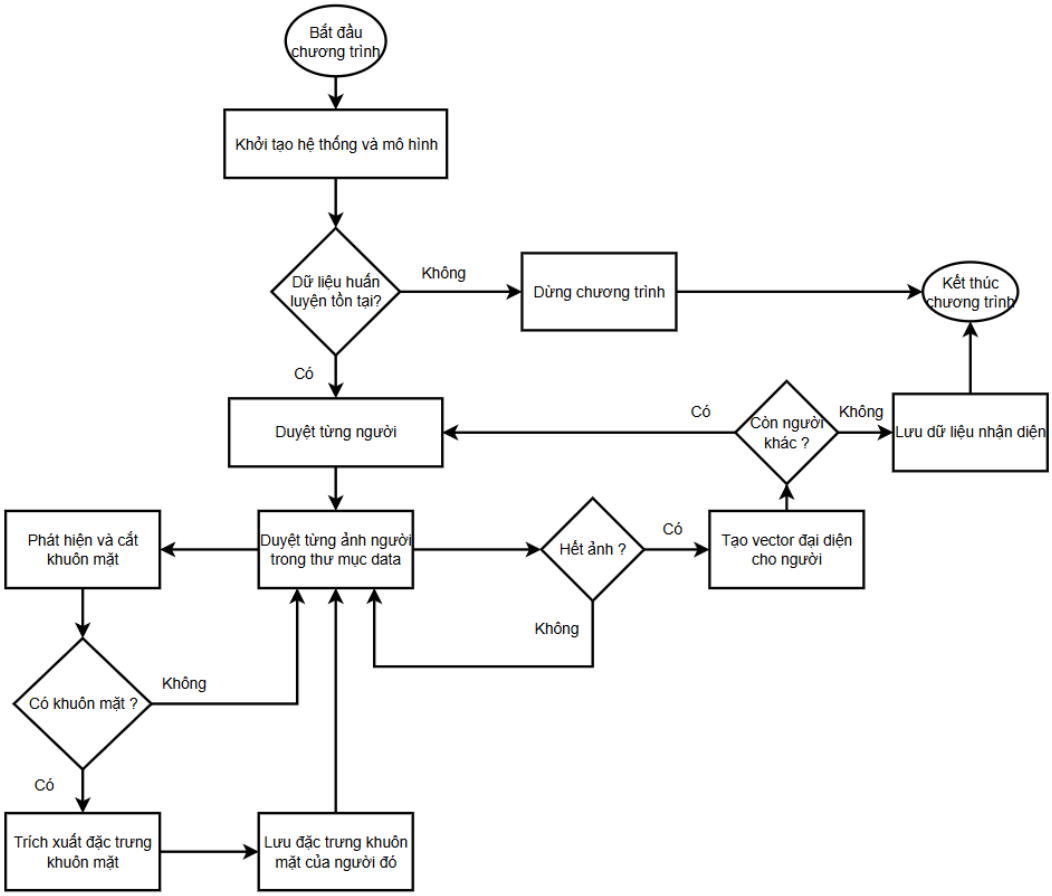
Chương trình bắt đầu bằng việc tải các cấu hình cần thiết và nạp mô hình Haar Cascade phục vụ cho phát hiện khuôn mặt. Sau đó, hệ thống yêu cầu người dùng nhập tên đối tượng cần lưu dữ liệu và chuẩn bị thư mục lưu trữ tương ứng. Tiếp theo, chương trình mở kết nối đến luồng camera RTSP và tiến hành đọc từng khung hình video. Nếu việc đọc khung hình không thành công, hệ thống sẽ thực hiện kết nối lại luồng RTSP và tiếp tục quá trình. Khi đọc khung hình thành công, ảnh được xử lý để phát hiện khuôn mặt bằng Haar Cascade. Trường hợp phát hiện có khuôn mặt, chương trình sẽ lưu ảnh chụp khuôn mặt theo điều kiện đã thiết lập. Quá trình lưu ảnh được điều khiển bởi người dùng, cho phép tiếp tục hoặc thoát khỏi quá trình thu thập dữ liệu. Khi người dùng chọn kết thúc hoặc đạt đủ số lượng ảnh yêu cầu, chương trình dừng việc lưu ảnh và kết thúc quá trình thực thi.

2. File face_train_model.py: xây dựng cơ sở dữ liệu đặc trưng khuôn mặt (face_embeddings) từ tập ảnh huấn luyện. Mỗi người được biểu diễn bởi một vector đặc trưng.



Hình 39: Huấn luyện mô hình khuôn mặt

Lưu đồ giải thuật:



Hình 40: Lưu đồ giải thuật của file face_train_model.py

Phân tích chương trình:

1. os: Làm việc với hệ thống tệp và thư mục.

2. torch: Thư viện học sâu dùng cho xử lý tensor và suy luận mô hình.

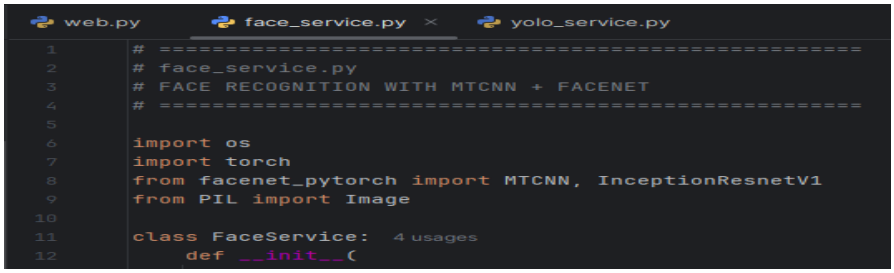
3. face_pytorch: Thư viện triển khai sẵn các mô hình nhận diện khuôn mặ.

4. PIL: Xử lý và thao tác với ảnh, đọc và lưu ảnh.

Chương trình có chức năng tạo dữ liệu nhận diện khuôn mặt bằng cách trích xuất đặc trưng từ nhiều ảnh và xây dựng vector đại diện cho mỗi người. Sử dụng mô hình phát hiện khuôn mặt MTCNN và trích xuất đặc trưng FaceNet để thực hiện. Đầu tiên, kiểm tra sự tồn tại của dữ liệu khuôn mặt đã được lấy từ file collect_image.py, nếu không có thì sẽ dừng chương trình.

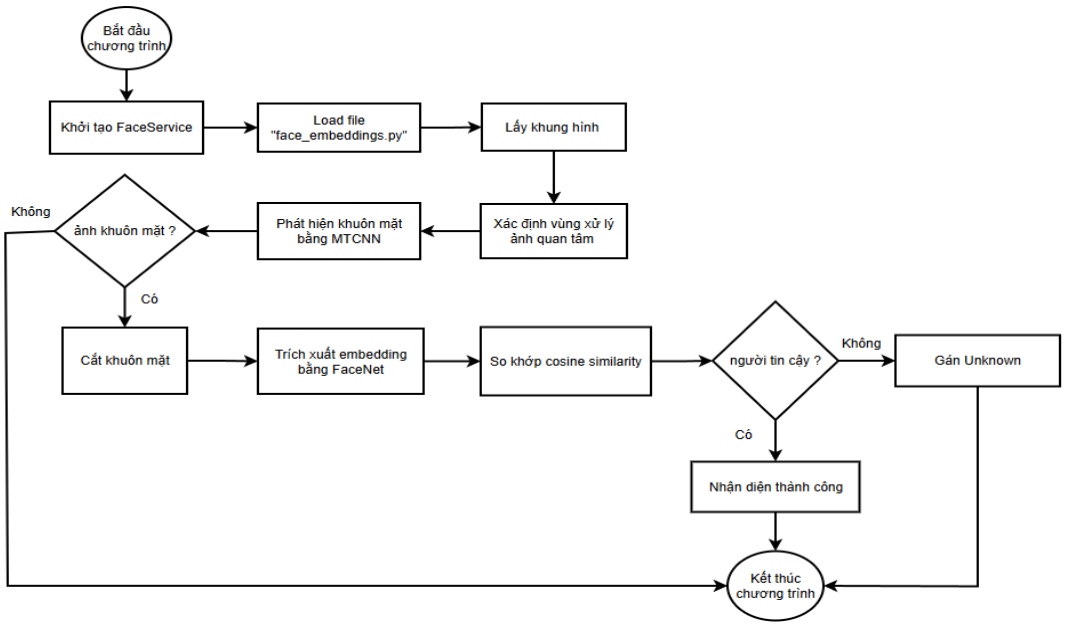
Tiếp theo, chương trình sẽ duyệt qua từng người trong tập dữ liệu huấn luyện. Với mỗi người, sẽ thực hiện phát hiện và cắt khuôn mặt. Những ảnh mờ hoặc bị lỗi thì sẽ bỏ qua. Ảnh hợp lệ thì sẽ được trích xuất thành các vector đặc trưng khuôn mặt và lưu lại. Sau khi xử lý toàn bộ ảnh của mỗi người, chương trình sẽ tính toán vector đặc trưng đại diện bằng cách lấy trung bình các vector đã trích xuất. Vector đại diện này được lưu lại tương ứng với từng người. Quá trình này được lặp lại cho đến khi tất cả thư mục chứa ảnh người đại diện đều được duyệt qua. Cuối cùng, toàn bộ dữ liệu đặc trưng khuôn mặt được lưu ra file "face_embeddings.pt" để phục vụ cho mục đích nhận diện khuôn mặt.

3. File face_service.py: là một module Python chuyên biệt dùng để nhận diện và quản lý khuôn mặt trong các ứng dụng giám sát hoặc hệ thống Smart Home.



Hình 41: Module face_service.py

Lưu đồ giải thuật:



Hình 42: Lưu đồ giải thuật của file face_service.py

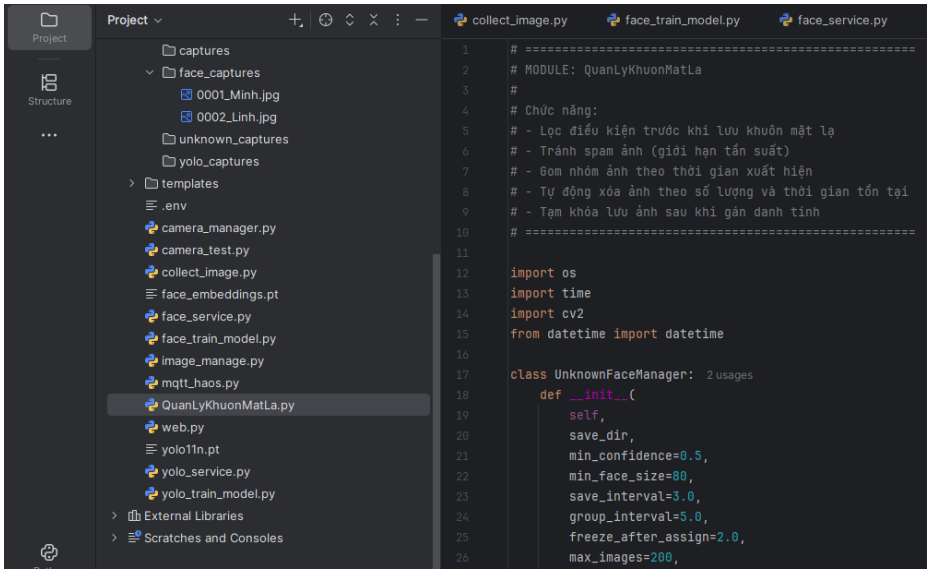
Phân tích chương trình:

Chương trình khởi tạo đối tượng FaceService, trong đó hệ thống thiết lập thiết bị xử lý và nạp dữ liệu vector đặc trưng khuôn mặt từ file face_embeddings.pt. Các vector này đại diện cho những người đã được đăng ký trước đó. Sau khi khởi tạo, hệ thống nhận khung hình ảnh đầu vào và xác định vùng ảnh quan tâm để xử lý. Trong vùng ảnh này, mô hình MTCNN được sử dụng để phát hiện khuôn mặt và loại bỏ các trường hợp có độ tin cậy thấp.

Đối với các khuôn mặt hợp lệ, chương trình cắt ảnh khuôn mặt và trích xuất vector đặc trưng bằng mô hình FaceNet. Vector thu được sau đó được so khớp với cơ sở dữ liệu bằng phương pháp cosine similarity. Nếu độ tương đồng vượt ngưỡng cho phép, khuôn mặt được nhận diện là người tương ứng; ngược lại, khuôn mặt được gán nhãn "Unknown". Kết quả nhận diện được trả về để phục vụ các bước xử lý tiếp theo. Toàn bộ quy trình trên được lặp lại liên tục cho các khung hình kế tiếp cho đến khi chương trình kết thúc quá trình xử lý ảnh.

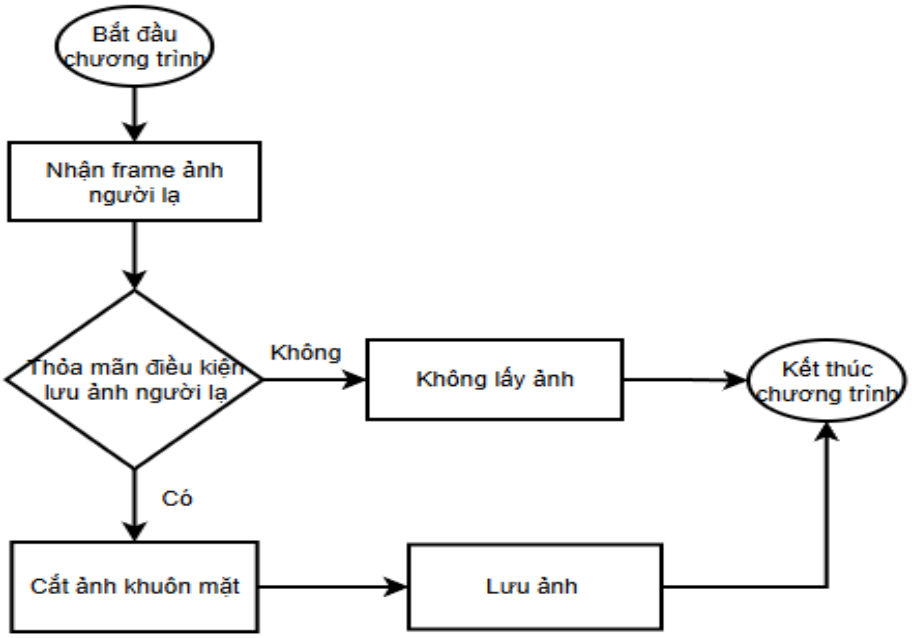
3.4.3. Xây dựng Web để hỗ trợ tích hợp AI trong xử lý ảnh chụp từ Camera và gửi lên Home Assistant:

1. File QuanLyKhuonMatLa.py: quản lý các khuôn mặt chưa được nhận diện trong hệ thống AI Camera, tránh việc lưu trữ dư thừa, kiểm soát và hỗ trợ gán danh tính.



Hình 43: Module QuanLyKhuonMatLa.py

Lưu đồ giải thuật:



Hình 44: Lưu đồ giải thuật file QuanLyKhuonMatLa.py

Phân tích chương trình:

1. os: làm việc với hệ thống file và thư mục.

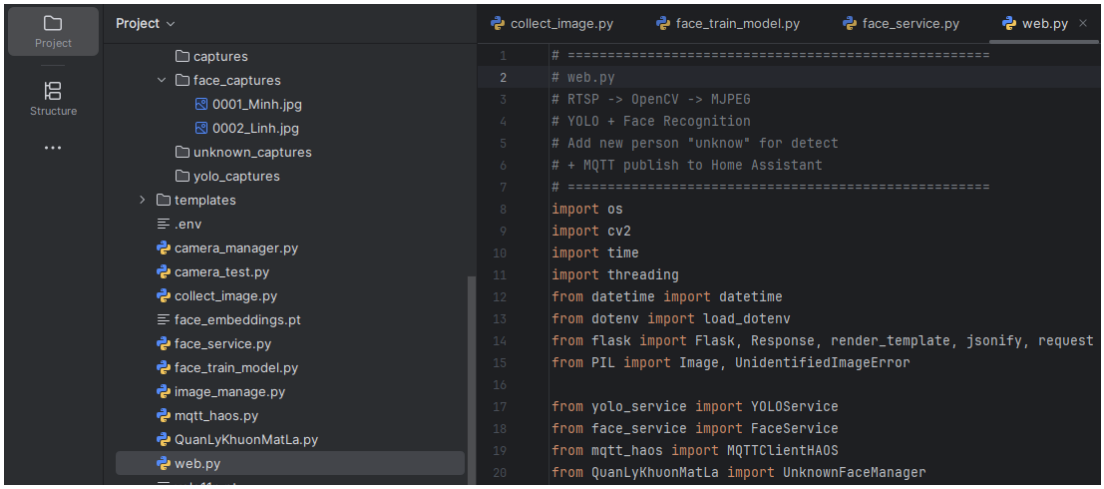
2. time: cập nhật thời gian hiện tại của hệ thống.

3. cv2: thư viện OpenCV, cắt ảnh, lưu ảnh.

4. datetime: dùng để tạo thời gian theo định dạng ngày giờ.

Chương trình được xây dựng nhằm quản lý việc lưu trữ khuôn mặt lạ trong hệ thống camera thông minh. Khi phát hiện một khuôn mặt chưa được xác định danh tính, hệ thống tiến hành kiểm tra những điều kiện như độ tin cậy nhận diện, kích thước khuôn mặt, tính hợp lệ của vùng khuôn mặt và khoảng thời gian tối thiểu giữa các lần nhận diện để tránh lưu nhầm ảnh trùng. Nếu các điều kiện được thỏa mãn, ảnh khuôn mặt lạ mới được cắt từ khung hình gốc và lưu vào thư mục chỉ định. Sau mỗi lần lưu, chương trình tự động cập nhật thời gian lưu, ghi log và thực hiện dọn dẹp lại dữ liệu bằng cách xóa các ảnh cũ quá giới hạn số lượng và thời gian tồn tại. Ngoài ra, chương trình sẽ kết hợp với giao diện điều khiển của trang Web để gán danh tính phục vụ cho hệ thống nhận diện dễ dàng hơn.

2. File web.py: chịu trách nhiệm việc giám sát hình ảnh từ Camera, xử lý ảnh với các mô hình AI, cung cấp giao diện điều khiển cho người dùng. Ngoài ra, chương trình còn đáp ứng khả năng gửi thông tin phát hiện nhận diện lên Home Assistant OS thông qua giao thức MQTT.



Hình 45: File web.py

Phân chia luồng hệ thống:

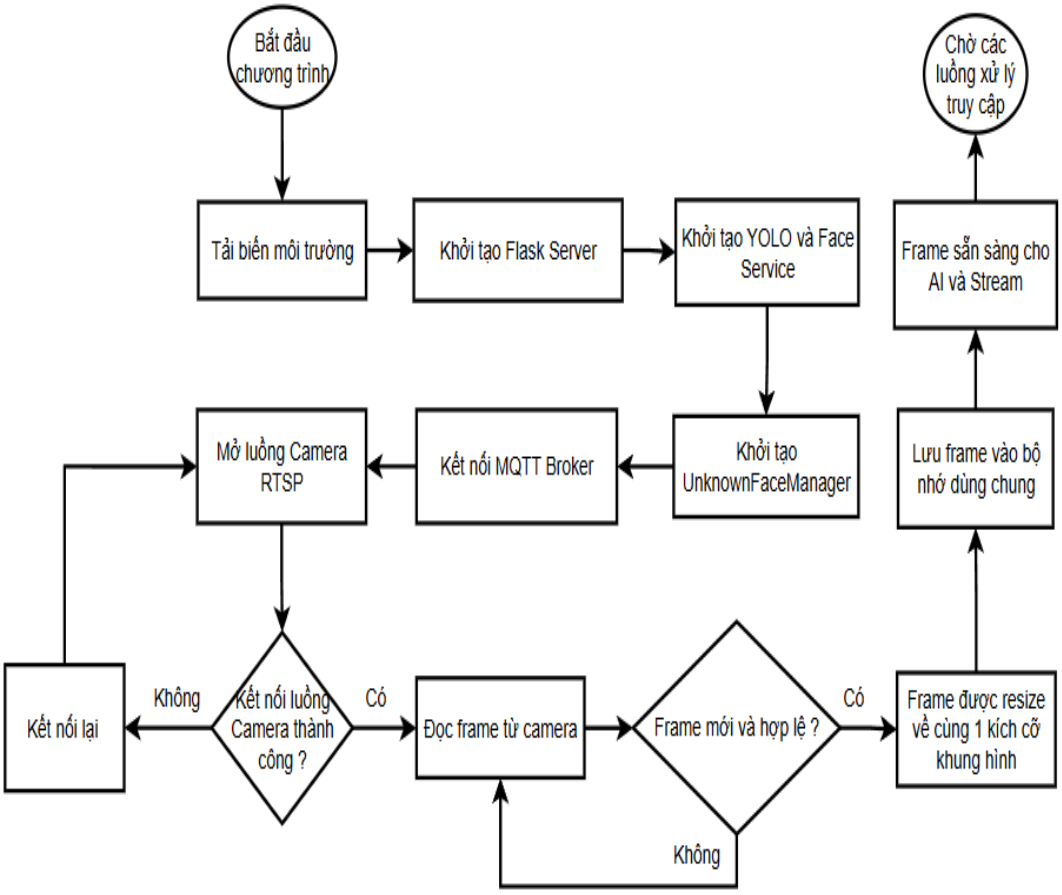
Hệ thống được thiết kế theo mô hình đa luồng nhằm đảm bảo khả năng xử lý thời gian thực và tách biệt rõ ràng giữa các chức năng. Cụ thể, chương trình web.py được chia thành bốn luồng chính hoạt động song song và độc lập với nhau.

Luồng camera chịu trách nhiệm kết nối với camera RTSP, thu nhận và cập nhật liên tục khung hình mới nhất. Luồng xử lý AI tự động là luồng quan trọng nhất, thực hiện phát hiện người bằng YOLO, nhận diện khuôn mặt, xử lý khuôn mặt lạ, hiển thị kết quả và gửi dữ liệu về hệ thống Home Assistant thông qua MQTT. Bên cạnh đó, luồng khởi tạo và hiển thị Camera đảm nhiệm việc mã hóa và truyền hình ảnh dưới dạng MJPEG để hiển thị trên giao diện web, không ảnh hưởng đến logic xử lý AI. Cuối cùng, luồng Web API phục vụ các thao tác điều khiển từ người dùng như bật hoặc tắt chế độ AI, chụp ảnh thủ công và gán danh tính cho khuôn mặt lạ.

Việc phân chia hệ thống theo các luồng chức năng như trên giúp hệ thống hoạt động ổn định, dễ mở rộng và thuận tiện cho việc phân tích, thiết kế cũng như triển khai thực tế.

Lưu đồ giải thuật:

1. Luồng khởi tạo và hiển thị Camera:

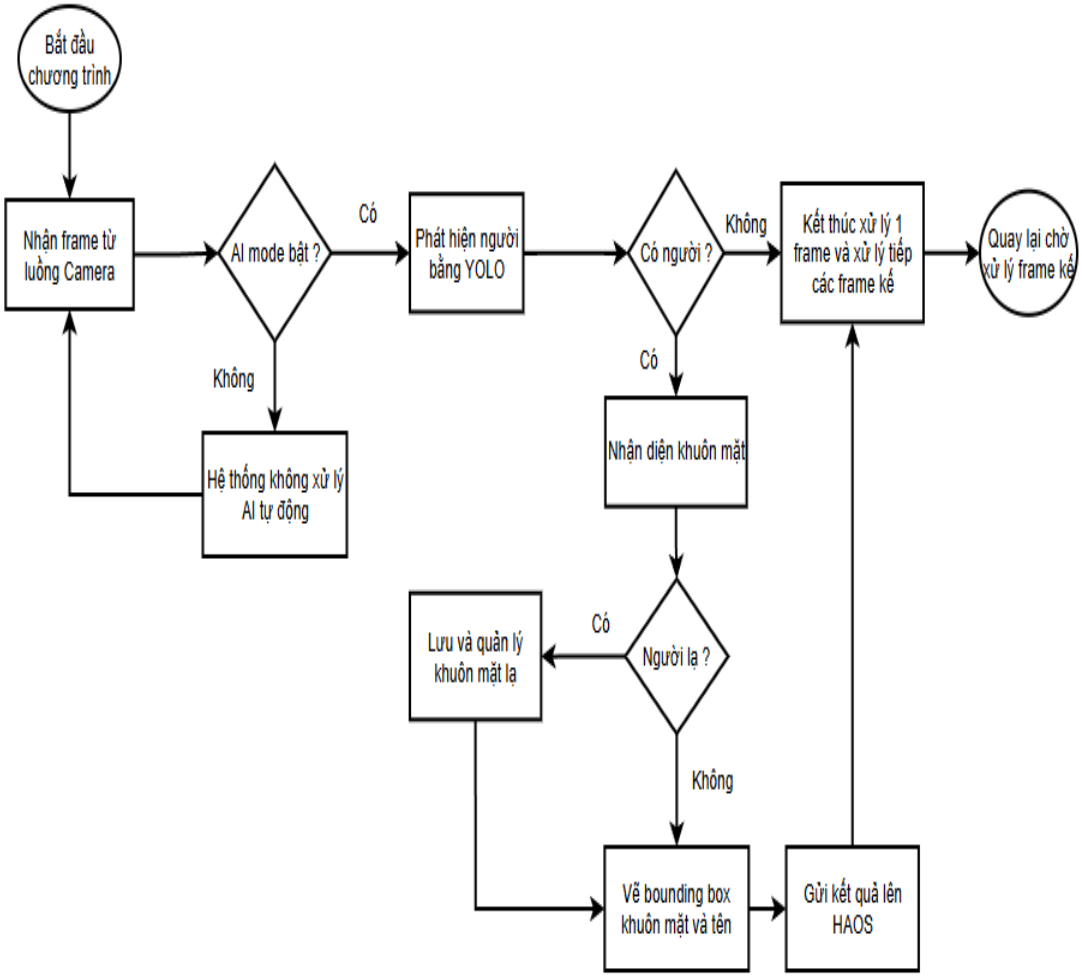


Hình 46: Lưu đồ giải thuật luồng khởi tạo và hiển thị Camera

Phân tích luồng khởi tạo và hiển thị Camera:

Sau khi chương trình bắt đầu, hệ thống tiến hành tải các biến môi trường và khởi tạo các thành phần chính như Flask Server, dịch vụ YOLO, Face Recognition và bộ quản lý khuôn mặt lạ. Tiếp theo, hệ thống kết nối đến MQTT Broker để phục vụ việc gửi dữ liệu trạng thái. Sau quá trình khởi tạo, một luồng riêng được tạo ra để kết nối đến camera thông qua giao thức RTSP. Luồng này chịu trách nhiệm đọc các khung hình mới nhất từ camera, kiểm tra tính hợp lệ của khung hình và chuẩn hóa kích thước trước khi lưu vào bộ nhớ dùng chung. Các khung hình đã sẵn sàng sau đó được cung cấp cho luồng xử lý AI và luồng hiển thị video MJPEG, cho phép hệ thống vừa xử lý nhận diện vừa hiển thị hình ảnh theo thời gian thực.

2. Luồng xử lý AI tự động (luồng hoạt động chính):

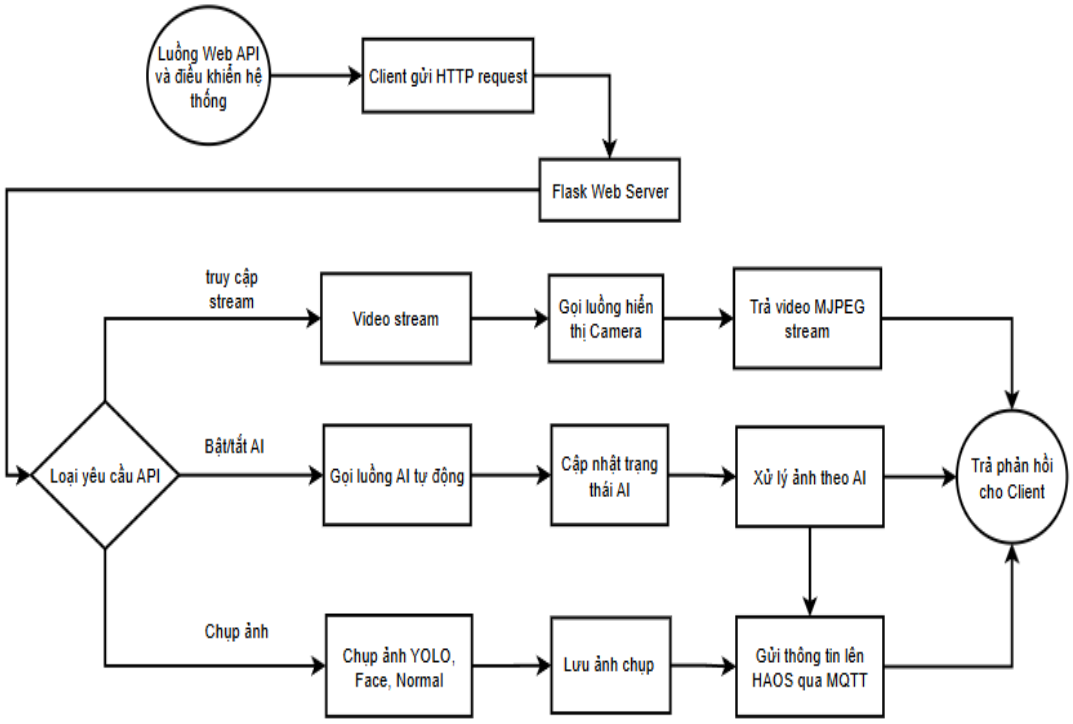


Hình 47: Lưu đồ giải thuật luồng xử lý AI tự động

Phân tích luồng AI tự động:

Hệ thống bắt đầu bằng việc lấy khung hình từ luồng khởi tạo và hiển thị Camera. Nếu chế độ AI tự động được bật, khung hình sẽ chạy mô hình nhận diện đối tượng YOLO để phát hiện đối tượng là người, khi không phát hiện người thì hệ thống kết thúc xử lý khung hình hiện tại và chuyển sang khung hình tiếp theo. Khi có người xuất hiện, hệ thống tiếp tục thực hiện nhận diện khuôn mặt trong vùng phát hiện người đó. Nếu khuôn mặt chưa có danh tính, ảnh khuôn mặt lạ sẽ được lưu và quản lý để gán danh tính sau này. Ảnh sau khi xử lý sẽ hiển thị trực tiếp trên khung Video với ảnh có bounding box và tên của người quen hoặc người lạ. Sau đó, kết quả thông tin của ảnh sẽ được gửi về Home Assistant qua giao thức MQTT.

3. Luồng Web API điều khiển hệ thống:



Hình 48: Lưu đồ giải thuật của luồng Web API điều khiển hệ thống

Phân tích luồng Web API điều khiển hệ thống:

Luồng Web API đóng vai trò là lớp điều phối trung tâm của hệ thống, tiếp nhận các yêu cầu HTTP từ phía client và phân loại theo từng chức năng cụ thể. Khi client gửi yêu cầu đến, Flask Web Server chịu trách nhiệm tiếp nhận và xác định loại API tương ứng. Đối với yêu cầu truy cập luồng video, Web API gọi luồng hiển thị camera để lấy khung hình hiện tại từ bộ nhớ dùng chung và trả về cho client dưới dạng luồng video MJPEG theo thời gian thực.

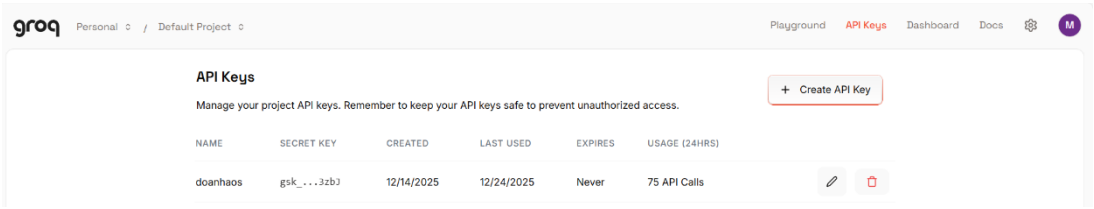
Trong trường hợp client gửi yêu cầu bật hoặc tắt chế độ xử lý AI, Web API chỉ thực hiện việc gọi luồng AI tự động và cập nhật trạng thái hoạt động của hệ thống, trong khi toàn bộ quá trình phát hiện và nhận diện đối tượng được thực hiện bởi luồng xử lý AI chạy song song. Đối với các yêu cầu chụp ảnh, Web API sử dụng khung hình hiện tại để thực hiện chụp ảnh theo các chế độ khác nhau như ảnh thường, ảnh có bounding box YOLO hoặc ảnh khuôn mặt, sau đó lưu ảnh vào hệ thống và gửi thông tin liên quan đến Home Assistant thông qua giao thức MQTT. Sau khi hoàn tất việc xử lý từng loại yêu cầu, Web API trả phản hồi HTTP tương ứng về cho client, đánh dấu kết thúc vòng xử lý của yêu cầu đó.

3.4.4. Phân tích hình ảnh bằng AI dùng LLM Vision:

Cài đặt LLM Vision bằng cách tải xuống từ HACS.

Trong phần Cài đặt, chọn Thiết bị và dịch vụ => Thêm bộ tích hợp => LLM Vision.

Tại LLM Vision , thêm cấu hình cho Groq AI, truy cập vào Groq API để tạo 1 API miễn phí.

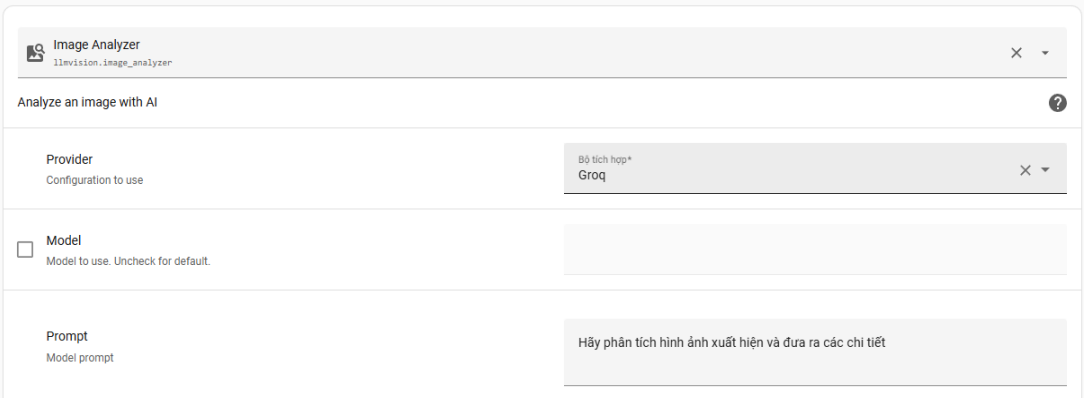


Hình 49: Lấy API từ Groq.

Kiểm thử Groq AI với hình ảnh:

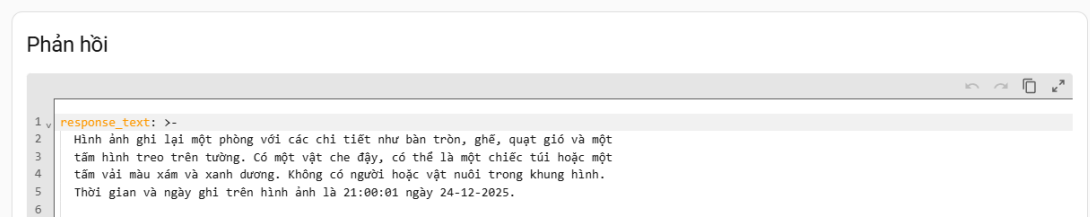
Provider: chọn Groq.

Prompt: ghi nội dung muốn Groq xử lý.



Hình 50: Groq phân tích hình ảnh

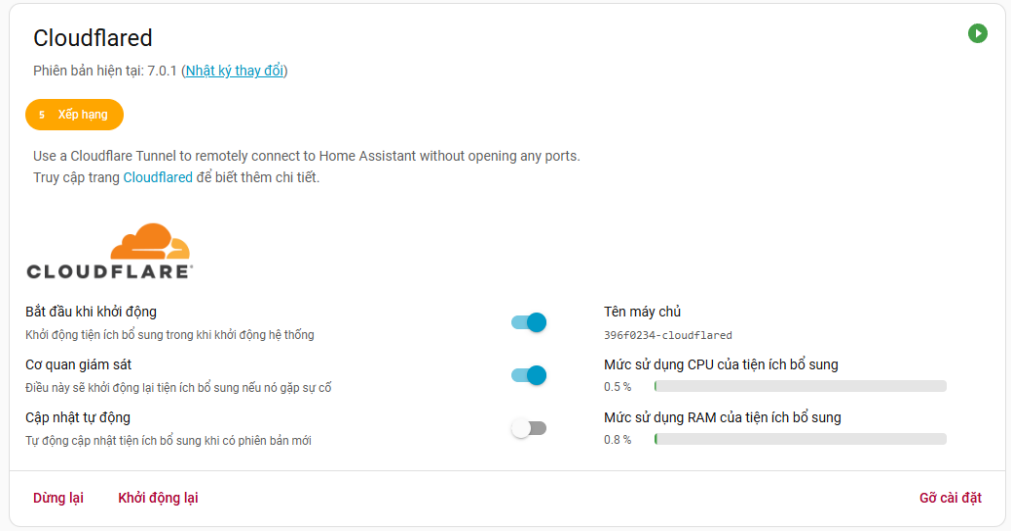
Image entity: lựa chọn thực thể chứa hình ảnh.



Hình 51: Kết quả phân tích ảnh với LLM Vision

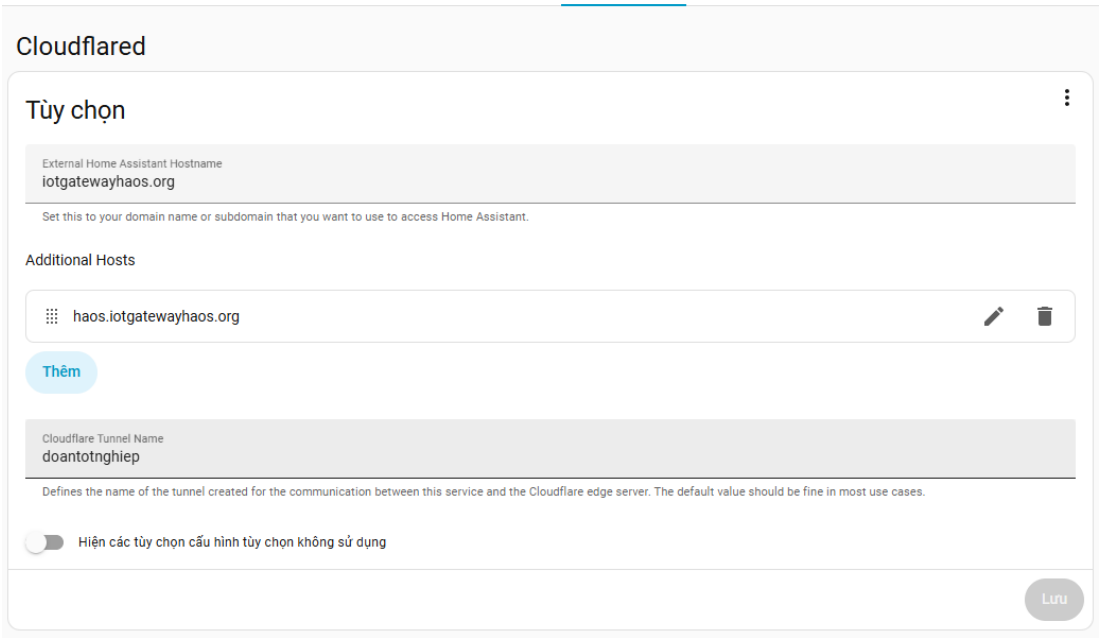
3.5. Thiết lập truy cập từ xa:

Cloudflare Tunnel: là một cơ chế cho phép truy cập dịch vụ nội bộ ra ngoài Internet mà không cần mở port. Thay vì Client truy cập trực tiếp vào IP public, server nội bộ sẽ chủ động tạo một kết nối outbound an toàn (https) tới Cloudflare.



Hình 52: Add ons Cloudflared

Cần phải mua 1 domain từ Cloudflare: https://www.cloudflare.com/



Hình 53: Cấu hình cho Cloudflare Tunnel

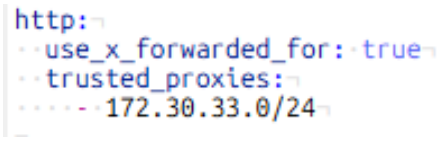
Tên miền domain đã mua: iotgatewayhaos.org

Cloudflare sẽ truy cập vào domain này để cho phép truy cập mạng nội bộ của HAOS. Ta tạo thêm các domain phụ cùng trỏ vào cùng 1 cloudflare tunnel, để sau này cần thêm dịch vụ khác hoặc sài nhiều tên miền thì vẫn có thể thực hiện.

Đường dẫn truy cập sẽ là https://haos.iotgatewayhaos.org

Cloudflare tunnel name là tên tunnel nội bộ trên Cloudflare chỉ để quản lý và định danh, không ảnh hưởng đến domain hay truy cập.

Trong file configuration.yaml phải cho phép nhận yêu cầu từ internet và chuyển tiếp đến máy chủ nội bộ (reverse proxy).



Hình 54: Enabled Reverse Proxy

CHƯƠNG 4: KẾT QUẢ THỰC HIỆN

4.1. Thiết kế phần cứng:

Hình 55: Mô hình tổng quan của hệ thống

Sơ đồ chân kết nối mạch của ESP32 với phòng khách:

Bảng 10: Sơ đồ chân kết nối mạch phòng khách

Tên linh kiện	Chân linh kiện	Chân ESP32
BME280	VCC	3v3
	GND	GND
	SDA	GPIO21
	SCL	GPIO22
Đèn 12V	Chân dương	GPIO19
Quạt 12V	Chân dương	GPIO18
IR sensor	OUT	GPIO27
	VCC	5v
	GND	GND
Servo SG90	OUT	GPIO23
	VCC	5v
	GND	GND

Sơ đồ chân kết nối mạch của ESP32 với phòng ngủ:

Bảng 11: Sơ đồ chân kết nối mạch phòng ngủ

Tên linh kiện	Chân linh kiện	Chân ESP32
DHT11	VCC	3v3
	GND	GND
	OUT	GPIO32
Đèn 12V	Chân dương	GPIO25
Quạt 12V	Chân dương	GPIO26
IR sensor	Out	GPIO33
	VCC	5v
	GND	GND
Servo SG90	OUT	GPIO23
	VCC	5v
	GND	GND

Sơ đồ chân kết nối mạch của ESP32 với nhà bếp:

Bảng 12: Sơ đồ chân kết nối mạch nhà bếp

Tên linh kiện	Chân linh kiện	Chân ESP32
DHT11	VCC	3v3
	GND	GND
	OUT	GPIO32
Đèn 12V	Chân dương	GPIO25
Quạt 12V	Chân dương	GPIO26
IR sensor	Out	GPIO27
	VCC	5v
	GND	GND
Servo SG90	OUT	GPIO23
	VCC	5v
	GND	GND
Gas sensor	OUT	GPIO34
	VCC	5v
	GND	GND

Sơ đồ chân kết nối mạch của ESP32 với ngoài sân:

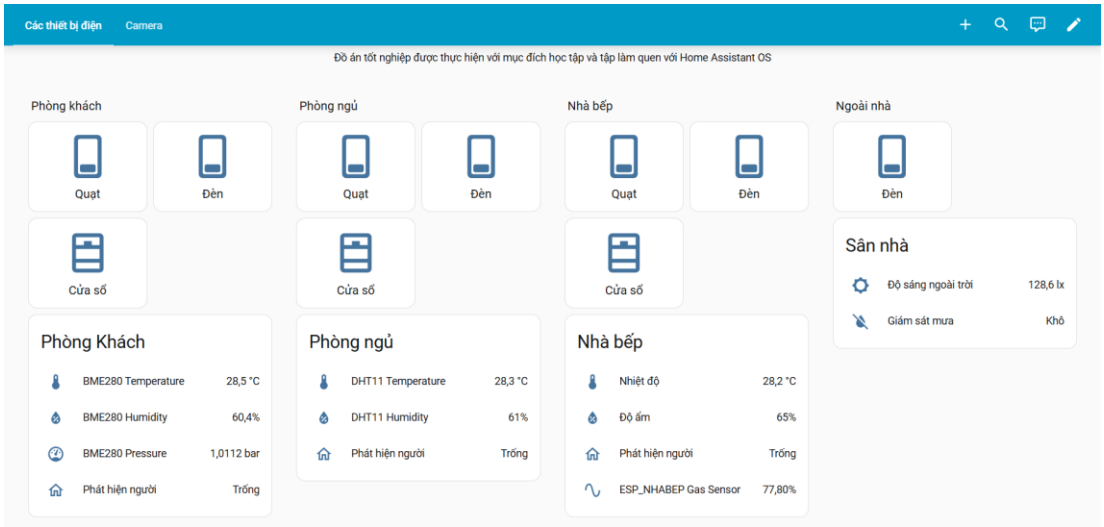
Bảng 13: Sơ đồ chân kết nối mạch ngoài sân

Tên linh kiện	Chân linh kiện	Chân ESP32
BH1750	VCC	5v
	GND	GND
	SDA	GPIO21
	SCL	GPIO22
Đèn 12V	Chân dương	GPIO25
Cảm biến mưa	OUT	GPIO27
	VCC	5v
	GND	GND

4.2. Thiết kế phần mềm:

Kết quả sau khi thiết kế hệ thống:

Hệ thống sẽ được chia làm bố cục 2 phần gồm: thẻ điều khiển các thiết bị điện, thẻ điều khiển camera.

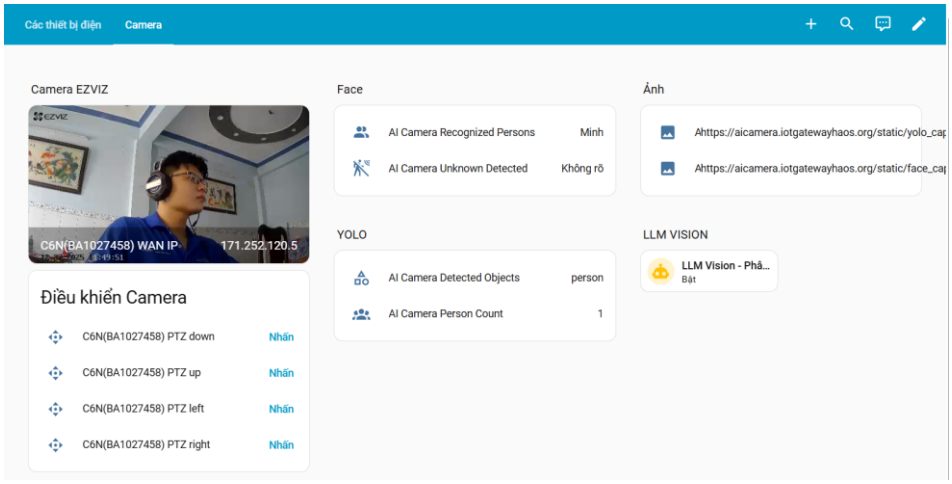


Hình 56: Thẻ điều khiển hệ thống trên Home Assistant OS

Kết quả sau khi thiết kế thẻ điều khiển thiết bị điện:

Thẻ điều khiển các thiết bị điện gồm: điều khiển công tắc đèn, quạt và đóng/mở cửa sổ của từng khu vực. Đọc các thông tin thu thập được từ cảm biến

và giám sát được môi trường xung quanh khu vực trong ngôi nhà. Nhiệt độ và độ ẩm của từng khu vực được hiển thị lên trang điều khiển, cảnh báo từ các cảm biến IR, cảm biến khí gas và cảm biến ánh sáng được thu thập và gửi trực tiếp lên Home Assistant, điều khiển đóng/mở cửa sổ từng khu vực. Mọi việc đều được thực hiện thông qua ESPHome API giao tiếp với Home Assistant OS, được tích hợp sẵn nên độ ổn định và khả năng phản hồi nhanh hơn.



Hình 57: Thẻ điều khiển chức năng từ Camera

Kết quả sau khi thiết kế thẻ Camera:

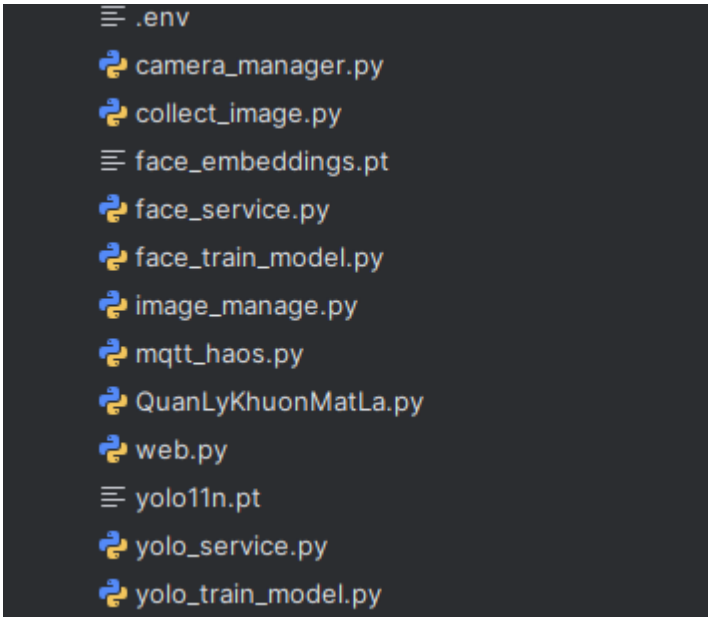
Hệ thống được chia ra làm 3 phần:

Camera EZVIZ: giao diện xem stream video RTSP của Camera và điều khiển Camera xoay theo chiều 360 độ để quan sát hết toàn cảnh khu vực.

Thông tin hệ thống giám sát: Phát hiện đối tượng và nhận diện khuôn mặt. Hệ thống còn gửi url ảnh lên HAOS để có thể xem trực tiếp.

Phân tích thực thể Camera với LLM Vision: Ứng dụng LLM Vision từ HACS để phân tích bức ảnh theo mô hình AI API của Groq để mô tả xem có người nào xuất hiện và người đó đang làm gì ?

4.3. Chạy hệ thống:



Hình 58: Cấu trúc file python của dự án

Giai đoạn 1: Thu thập hình ảnh làm dataset từ file collect_image.py

Nhập tên người lần lưu khuôn mặt và nếu dữ liệu có tồn tại thì ghi đè hoặc dùng tiếp.

```
(.venv) PS C:\Users\tvmin\PycharmProjects\DATN_Final> python collect_image.py
Nhập tên người cần lưu khuôn mặt: Minh

Đã tồn tại dữ liệu cho người 'Minh'
1 - Dùng tiếp (append ảnh mới)
2 - Ghi đè (xóa toàn bộ dữ liệu cũ)
3 - Hủy
Nhập lựa chọn (1/2/3): 1

[INFO] Thu thập ảnh FULL FRAME bằng Haar Cascade
      P : Pause / Resume
      Q : Quit
```

Hình 59: Thu thập ảnh làm dữ liệu huấn luyện khuôn mặt

Ở giai đoạn 1 này dataset sẽ được chụp liên tục từ Camera tương ứng với mỗi người là tối đa 100 bức ảnh.

```
[SAVE] Minh_100.jpg
[INFO] Đã đủ số lượng ảnh
[DONE] Hoàn tất thu thập dữ liệu
(.venv) PS C:\Users\tvmin\PycharmProjects\DATN_Final>
```

Hình 60: Kết quả sau khi thu thập

Nếu dữ liệu đã có và muốn dùng dữ liệu mới thì:

```
Đã tồn tại dữ liệu cho người 'Minh'
1 - Dùng tiếp (append ảnh mới)
2 - Ghi đè (xóa toàn bộ dữ liệu cũ)
3 - Hủy
Nhập lựa chọn (1/2/3): 2
[INFO] Đã xóa dữ liệu cũ của Minh
```

Khi đó ảnh sẽ được xóa hết từ thư mục cũ và thực hiện việc lưu ảnh mới tối đa là 100 ảnh trong thư mục đó.

Giai đoạn 2: Huấn luyện mô hình nhận diện khuôn mặt:

Mô hình sẽ được huấn luyện với file face_train_model.py để thực hiện việc cắt khuôn mặt với MTCNN và trích xuất đặc trưng khuôn mặt với FaceNet.

```
(.venv) PS C:\Users\tvmin\PycharmProjects\DATN_Final> python face_train_model.py
[INFO] Starting face embedding processing
[INFO] Using device: cuda
[INFO] HienHo: 15/15 images used
[INFO] Minh: 100/100 images used
[INFO] TranThanh: 15/15 images used
[DONE] Saved 3 identities to face_embeddings.pt
[INFO] Processed face images saved to: data/face_processed
[INFO] Debug images saved to: data/face_debug
(.venv) PS C:\Users\tvmin\PycharmProjects\DATN_Final>
```

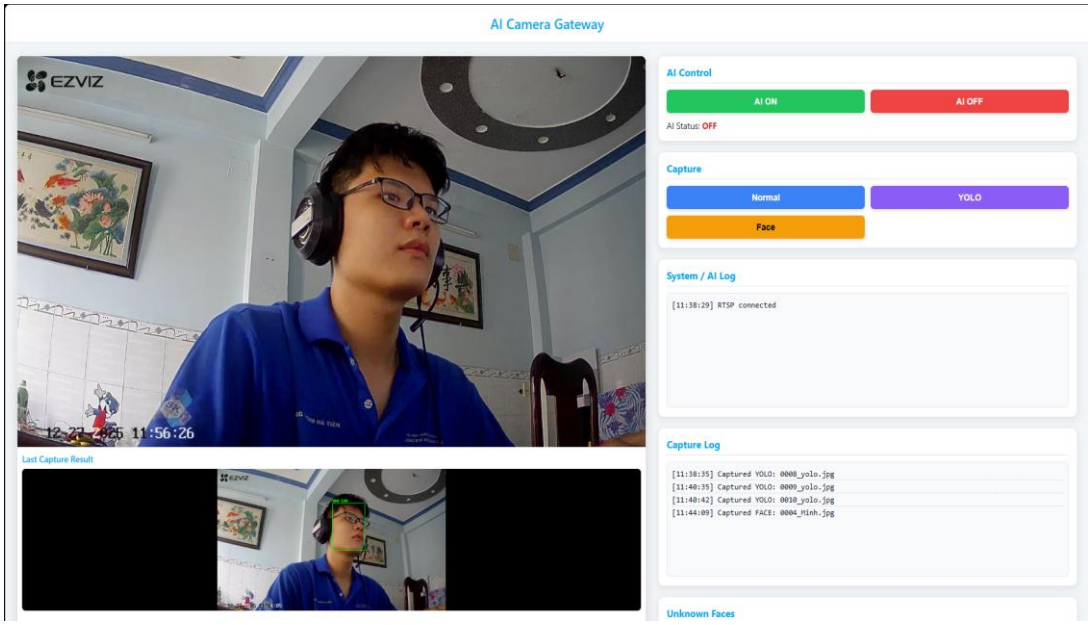
Giai đoạn 3: Thực hiện hệ thống web flask để phục vụ việc giám sát:

Khởi động web flask: chạy lệnh sau : cd DATN => python web.py

```
(.venv) PS C:\Users\tvmin\PycharmProjects\DATN_Final> python web.py
[FaceService] Device: cuda
[FaceService] Loaded 4 identities
 * Serving Flask app 'web'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.2.17:5000
```

Hình 61: Khởi động web flask

Truy cập vào đường dẫn FLASK_URL từ trình duyệt web.

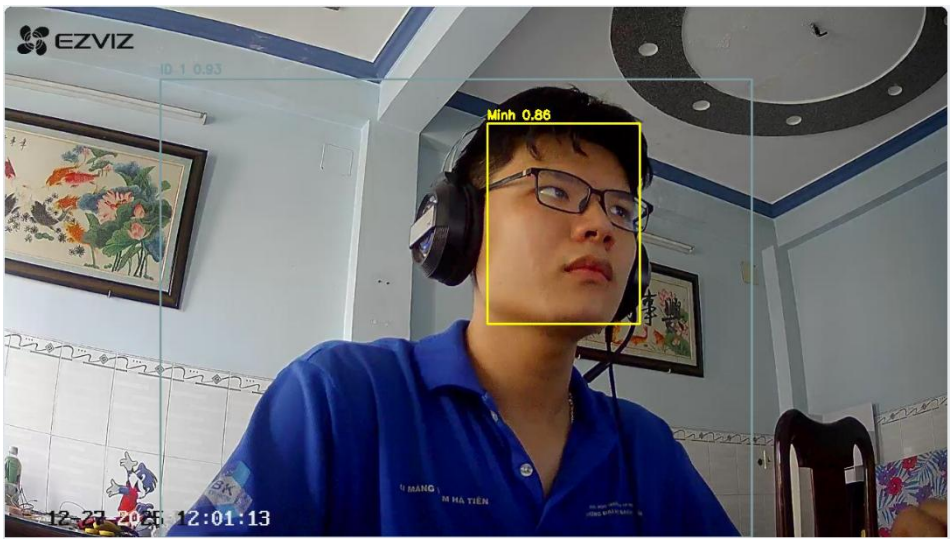


Hình 62: Giao diện trang Web điều khiển Camera

Các chức năng chính:

Hệ thống điều khiển Camera với 2 chế độ: AI tự động và chụp ảnh thủ công.

AI tự động: Hệ thống sẽ kích hoạt mô hình YOLO để quét qua các frame ảnh để phát hiện đối tượng con người. Nếu có người đang nhìn vào Camera sẽ giám sát để phân tích xem người đó là người quen hay người lạ dựa vào mô hình Nhận diện khuôn mặt với MTCNN và FaceNet.



Hình 63: Kết quả của mô hình AI tự động

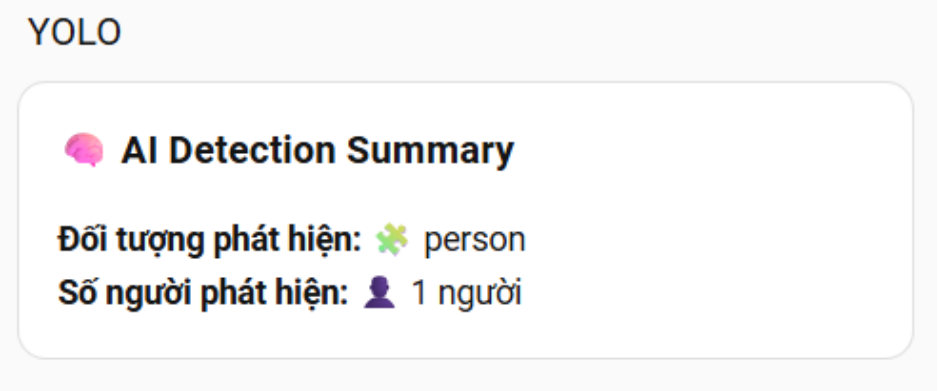
Chụp ảnh thủ công gồm có 3 chế độ: chụp ảnh bình thường, chụp ảnh với YOLO và chụp ảnh với Face_Detect. Kết quả sẽ gửi thông tin lên HAOS qua giao thức MQTT.

Kết quả ảnh chụp với YOLO:

Hình 64: Ảnh chụp với YOLO

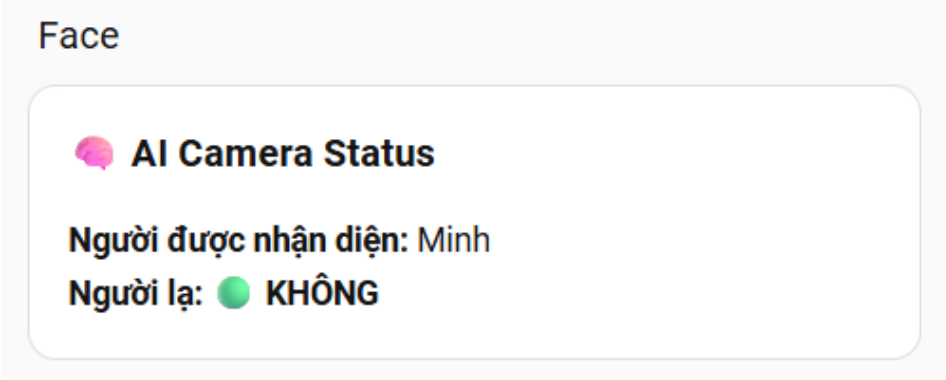
Kết quả ảnh chụp Face_Detect:

Hình 65: Ảnh chụp với Face_Detect



Hình 66: Thông tin đối tượng trong ảnh

Nhận diện khuôn mặt của người xuất hiện trong ảnh:



Hình 67: Thông tin người trong ảnh

4.4. Thực hiện các chức năng Automation:

Bảng 14: Thực hiện các chức năng Automation

Automation	Chức năng
1	Cảnh báo khí gas khi vượt ngưỡng thì gửi kết quả đến điện thoại
2	Cảnh báo phát hiện người trong khu vực khi camera chụp ảnh phát hiện có người
3	Cảnh báo phát hiện người tại phòng khách khi có người xuất hiện
4	Cảnh báo phát hiện người tại phòng ngủ khi có người xuất hiện
5	Cảnh báo phát hiện người tại phòng nhà bếp khi có người xuất hiện
6	Tự động bật / tắt đèn khi có người xuất hiện trong phòng khách
7	Tự động bật / tắt đèn khi có người xuất hiện trong phòng ngủ
8	Tự động bật / tắt đèn khi có người xuất hiện trong phòng nhà bếp
9	Tự động bật / tắt đèn ngoài sân theo độ sáng môi trường
10	Tự động đóng cửa sổ khi có trời mưa
11	Cảnh báo phát hiện người “ kiểm tra”: mỗi khi phát hiện thấy người
12	LLM Vision - Phân tích ảnh: gửi thông báo mỗi lần có sự kiện được phát hiện về người xuất hiện trong khu vực.

Kết quả của việc thực hiện các chức năng Automation với Home Assistant gửi qua lại giữa web và ứng dụng trên điện thoại.

Hình 68: Phân tích ảnh của LLM Vision

Hình 69: Cảnh báo phát hiện chuyển động

Hình 70: Cảnh báo từ Camera khi phát hiện người

CHƯƠNG 5: TỔNG KẾT VÀ ĐÁNH GIÁ ĐỀ TÀI

5.1. Những thành tựu đạt được của đề tài:

Đồ án tốt nghiệp đã đạt được một số thành tựu cũng như mục tiêu đề ra bao gồm:

- Xây dựng được hệ thống liên kết với ESP32 và Raspberry pi 4 với hệ điều hành Home Assistant.
- Xây dựng được hệ thống giám sát với các cảm biến môi trường, điều khiển đèn và quạt của ngôi nhà, giám sát với Camera EZVIZ.
- Xây dựng được các mô hình phát hiện đối tượng và nhận diện khuôn mặt qua ảnh chụp từng frame ảnh của video stream RTSP của Camera EZVIZ.
- Xây dựng hệ thống với khả năng truy cập từ xa qua Cloudflare Tunnel.
- Tạo ra nhiều kịch bản tối ưu hóa khả năng của hệ thống.

5.2. Những hạn chế của đề tài:

Với những thành tựu đã đạt được thì đề tài đồ án tốt nghiệp vẫn còn những hạn chế:

- Hệ thống hiện tại chạy ổn định nhưng về phần tích hợp AI trong xử lý ảnh với việc giám sát thời gian thực từ xa vẫn còn bị giới hạn do việc truyền video liên tục qua giao thức HTTPS với Cloudflare Tunnel gây độ trễ cao, không đảm bảo tính ổn định.
- YOLO và MTCNN/FaceNet xử lý tốt với các điều kiện ánh sáng và góc nhìn chuẩn, nhưng khi ánh sáng yếu, vật thể che khuất, hoặc số lượng đối tượng nhiều, độ chính xác giảm.
- Các mô hình AI được sử dụng chủ yếu trên mô hình huấn luyện sẵn và tệp dữ liệu giới hạn do người dùng cung cấp.
- Chất lượng và độ ổn định của video phụ thuộc vào đường truyền RTSP của camera. Nếu camera gặp vấn đề mạng hoặc stream không ổn định, việc nhận diện đối tượng và khuôn mặt sẽ bị gián đoạn.
- Giao diện người dùng còn đơn giản chủ yếu phục vụ mục đích giám sát và hiển thị kết quả, chưa được tối ưu về mặt trải nghiệm người dùng.

5.3. Hướng phát triển của đề tài:

Đề tài hiện tại xây dựng một hệ thống điều khiển và giám sát ở phạm vi nhỏ, còn ở mức sơ khai, chủ yếu phục vụ mục đích nghiên cứu và trình diễn (demo). Tuy nhiên, từ nền tảng đã xây dựng, hệ thống có tiềm năng phát triển theo nhiều hướng trong tương lai, cụ thể như sau:

Thứ nhất, Mở rộng phần cứng và tích hợp thêm cảm biến: Hệ thống có thể được nâng cấp để tích hợp thêm nhiều loại cảm biến và thiết bị ngoại vi nhằm mở rộng khả năng giám sát và điều khiển. Tuy nhiên, việc mở rộng này đòi hỏi nâng cấp phần cứng xử lý, do Raspberry Pi 4 vẫn còn những hạn chế về hiệu năng khi phải xử lý đồng thời nhiều camera hoặc nhiều tác vụ tính toán nặng liên quan đến trí tuệ nhân tạo.

Thứ hai, tích hợp tự động hóa đa nền tảng: Trong tương lai, hệ thống có thể được phát triển để hỗ trợ thêm các kịch bản tự động hóa thông minh và tương tác với nhiều nền tảng khác nhau. Việc tích hợp các giao thức và hệ sinh thái IoT phổ biến như Zigbee, trợ lý ảo (Alexa) hoặc các nền tảng IoT khác sẽ giúp hệ thống trở nên linh hoạt hơn và phù hợp với các ứng dụng nhà thông minh thực tế.

Thứ ba, hỗ trợ truy cập từ xa và nâng cao bảo mật: Hiện tại, hệ thống hoạt động dựa trên mô hình mạng nội bộ kết hợp với Cloudflare Tunnel nên vẫn cho phép truy cập từ xa thông qua Internet. Tuy nhiên, hình thức này phụ thuộc vào dịch vụ trung gian và chưa tối ưu cho các kịch bản truy cập linh hoạt hoặc yêu cầu độ trễ thấp. Trong tương lai, để hỗ trợ khả năng truy cập từ xa trực tiếp mà không cần thông qua nền tảng trung gian, hệ thống cần được mở rộng bằng các cơ chế kết nối an toàn như VPN (WireGuard, OpenVPN) hoặc triển khai các dịch vụ truy cập có xác thực và mã hóa mạnh (HTTPS, TLS).

Thứ tư, nâng cao cơ chế cảnh báo: Hệ thống nên được mở rộng để bổ sung các cơ chế gửi cảnh báo từ Home Assistant OS đến các ứng dụng nhắn tin phổ biến như Telegram, Zalo hoặc các nền tảng thông báo khác. Home Assistant có thể tự động gửi thông báo kèm theo thông tin chi tiết hoặc hình ảnh chụp được từ hệ thống AI Camera.

TÀI LIỆU THAM KHẢO

[1] Home Assistant - Trang web chính thức của Home Assistant.

[2] ESPHome - Smart Home Made Simple - Trang web hỗ trợ của ESPHome

[3] ESP32 Development Boards List with Features, Pinouts and more

[4] MQTT - Home Assistant

[5] HACS - Trang Home Assistant Community Store

[6] Tìm Hiểu Mô Hình YOLO Cho Bài Toán Object Detection - Understanding YOLO – Quoc Pham – Data Scientist at Overspace

[7] Nhận diện khuôn mặt với mạng MTCNN và FaceNet (Phần 1)

[8] Nhận diện khuôn mặt với mạng MTCNN và FaceNet (Phần 2)

[9] Roboflow: Computer vision tools for developers and enterprises

[10] Ultralytics YOLO11 - Tài liệu Ultralytics YOLO

[11] API là gì? Đặc điểm nổi bật, thông tin về Web API mới 2025

[12] Mì AI - Nhận diện khuôn mặt trong video bằng MTCNN và Facenet

[13] How to Train a YOLOv11 Object Detection Model on a Custom Dataset

[14] MQTT là gì? Vai trò của MQTT trong IoT

[15] Tìm hiểu về HTTP (HyperText Transfer Protocol)

[16] Welcome to Flask — Flask Documentation (3.1.x)

[17] Face Detection using Python and OpenCV with webcam - GeeksforGeeks

[18] GitHub - SARAH-HADDAD/Face-Recognition: Face recognition with mtcnn and facenet.

[19] Gắn tên miền truy cập cho Home Assistant không cần mở port

[20] How to Use ESPHome with ESP32: A Beginner's Guide | Microcontroller Tutorials