TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO
PBL5 – DỰ ÁN KỸ THUẬT MÁY TÍNH

TÊN ĐỀ TÀI: DỰ ÁN NHÀ XE THÔNG MINH

Giảng viên hướng dẫn: TS. Trịnh Công Duy

NHÓM SINH VIÊN THỰC HIỆN	LỚP HỌC PHẦN
Nguyễn Quốc Anh	22T_KHDL
Nguyễn Quốc Tiến	22T_KHDL
Bùi Chí Cường	22T_KHDL

Đà Nẵng, 06/2025

TÓM TẮT ĐỒ ÁN

Trong thực tế, việc quản lý khu vực để xe tại các hộ gia đình hoặc khu nhà trọ còn mang tính thủ công, thiếu tính an toàn và không có khả năng giám sát thông minh, dẫn đến nguy cơ mất tài sản hoặc khó kiểm soát người ra vào.

Nhóm đã thiết kế và xây dựng một hệ thống nhà xe thông minh sử dụng Raspberry Pi 4 làm bộ điều khiển trung tâm, kết hợp với các mô hình trí tuệ nhân tạo gồm nhận diện khuôn mặt để xác thực người dùng, nhận diện người để phát hiện xâm nhập trái phép và nhận diện giọng nói để điều khiển thiết bị. Hệ thống được tích hợp với website giúp theo dõi trạng thái, quản lý thiết bị và điều khiển từ xa một cách thuận tiện.

Hệ thống hoạt động ổn định, các mô hình nhận diện cho độ chính xác cao, thời gian phản hồi nhanh và tiêu tốn ít tài nguyên. Website giao diện thân thiện, dễ sử dụng, giúp người dùng chủ động trong việc quản lý gara. Mô hình có khả năng mở rộng và ứng dụng thực tế cao trong các không gian sống hiện đại.

PBL5: DỰ ÁN KỸ THUẬT MÁY TÍNH

BẢNG PHÂN CÔNG NHIỆM VỤ

Sinh viên thực hiện	Các nhiệm vụ	Đánh giá
Nguyễn Quốc Anh	Thiết kế mô hình IOT	Hoàn thành
	Nhận diện gương mặt bằng Face Recognition	Hoàn thành
	Xây dựng Website quản lý gara	Hoàn thành
	Xây dựng thuật toán điều khiển phần cứng	Hoàn thành
	Triển khai mô hình AI	Hoàn thành
	Lắp rắp phần cứng	Hoàn thành
Nguyễn Quốc Tiến	Thiết kế mô hình IOT	Hoàn thành
	Nhận diện người bằng YOLO	Hoàn thành
	Xây dựng Website quản lý gara	Hoàn thành
	Xây dựng thuật toán điều khiển phần cứng	Hoàn thành
	Triển khai mô hình AI	Hoàn thành
	Lắp ráp phần cứng	Hoàn thành
Bùi Chí Cường	Thiết kế mô hình IOT	Hoàn thành
	Nhận diện giọng nói bằng Vosk	Hoàn thành
	Xây dựng Website quản lý gara	Hoàn thành
	Xây dựng thuật toán điều khiển phần cứng	Hoàn thành
	Triển khai kết nối không dây	Hoàn thành
	Lắp rắp phần cứng	Hoàn thành

MỤC LỤC

DANH SÁCH HÌNH ẢNH

DANH SÁCH BẢNG BIỂU

1. Giới thiệu

Trong thời đại công nghệ số phát triển mạnh mẽ, nhu cầu tự động hóa và nâng cao an toàn trong sinh hoạt, đặc biệt là khu vực để xe như gara, ngày càng trở nên cấp thiết. Với sự phát triển của trí tuệ nhân tạo (AI), thiết bị IoT và hệ thống nhúng, các giải pháp thông minh – hiệu quả – tiết kiệm đang trở nên khả thi hơn bao giờ hết.

Đồ án này hướng đến việc xây dựng một hệ thống gara xe thông minh, sử dụng Raspberry Pi 4 làm trung tâm điều khiển, tích hợp các công nghệ AI hiện đại nhằm mang lại sự tiện nghi, an toàn và dễ sử dụng cho người dùng tại hộ gia đình, nhà trọ hoặc các khu chung cư nhỏ.

Tổng quan

Hiện nay, tại nhiều gia đình, khu trọ, hay nhà ở dân sinh, việc quản lý khu vực để xe vẫn chủ yếu được thực hiện theo phương pháp thủ công, phụ thuộc vào sự giám sát của con người như khóa tay, ghi chú giấy hoặc camera thông thường mà không có tích hợp thông minh. Điều này dẫn đến nhiều bất cập như thiếu tính bảo mật, khó kiểm soát ra vào, và không có khả năng giám sát từ xa hoặc cảnh báo kịp thời khi có sự cố.

Trong khi đó, trên thế giới, các mô hình nhà thông minh (Smart Home) đang ngày càng phát triển và được ứng dụng rộng rãi, từ điều khiển thiết bị điện, an ninh, cho đến tự động hóa sinh hoạt hàng ngày. Tuy nhiên, việc triển khai hệ thống thông minh trong gara để xe một phần thiết yếu của nhà ở vẫn chưa phổ biến, đặc biệt là tại Việt Nam. Lý do chủ yếu đến từ chi phí đầu tư cao, hạ tầng kỹ thuật phức tạp, và thiếu sự đơn giản trong triển khai cho người dùng không chuyên.

Trong bối cảnh đó, Raspberry Pi nổi lên như một giải pháp lý tưởng để xây dựng các hệ thống thông minh quy mô nhỏ. Đây là một dòng máy tính nhúng mini, có chi phí thấp, tiêu thụ điện năng ít, dễ lập trình, và đặc biệt là có khả năng tích hợp mạnh mẽ với các mô hình trí tuệ nhân tạo (AI), bao gồm nhận diện khuôn mặt, xử lý hình ảnh, nhận dạng giọng nói, ...

Việc kết hợp Raspberry Pi với các công nghệ AI và hệ thống điều khiển như Arduino, cảm biến, động cơ,... cho phép xây dựng một hệ thống gara xe thông minh, mang lại nhiều lợi ích thiết thực: từ việc nhận diện người dùng tự động để mở cửa, đến cảnh báo người lạ, điều khiển các thiết bị trong gara bằng giọng nói, hoặc quản lý từ xa thông qua giao diện web hoặc điện thoại.

Do đó, việc phát triển một hệ thống gara xe thông minh sử dụng nền tảng Raspberry Pi và AI là hướng đi vừa hiện đại, vừa phù hợp với điều kiện thực tế tại Việt Nam, góp phần đưa công nghệ đến gần hơn với đời sống hàng ngày.

Các vấn đề cần giải quyết

- Làm sao để tự động kiểm soát người ra vào khu vực để xe một cách an toàn, hiệu quả
    - Việc kiểm soát ra vào cần đảm bảo xác thực đúng người, tránh mở cửa cho người lạ hoặc bị giả mạo khuôn mặt. Hệ thống cần có tốc độ xử lý nhanh, hoạt động ổn định cả khi ánh sáng yếu, và ghi lại lịch sử truy cập để có thể kiểm tra sau này nếu cần
- Làm thế nào để phát hiện hành vi xâm nhập trái phép hoặc người lạ xuất hiện khi chủ vắng mặt.
    - Cần một mô hình phát hiện người có độ chính xác cao, hoạt động tốt trong các điều kiện ánh sáng khác nhau (ban ngày/ban đêm). Đồng thời, hệ thống nên có khả năng gửi cảnh báo theo thời gian thực (qua Telegram, email hoặc app) và ghi lại video bằng camera khi phát hiện chuyển động đáng ngờ.
- Cách thức điều khiển các thiết bị (đèn, cửa,…) trong gara mà không cần tiếp xúc vật lý.
    - Hệ thống nên tích hợp điều khiển bằng giọng nói để thao tác nhanh chóng, đặc biệt khi người dùng đang bận hoặc mang vác vật nặng. Ngoài ra, có thể bổ sung thêm điều khiển qua điện thoại hoặc website, để nâng cao tính tiện dụng.
- Làm sao để hệ thống có giao diện dễ sử dụng, thân thiện và có thể điều khiển từ xa qua internet.
    - Cần thiết kế một giao diện web trực quan, responsive, có thể truy cập từ máy tính, điện thoại, tablet. Giao diện nên thể hiện rõ trạng thái thiết bị, cho phép mở/tắt các thiết bị, cập nhật người dùng mới, xem lịch sử ra vào, và đảm bảo bảo mật bằng đăng nhập tài khoản và mã hóa kết nối.
- Lựa chọn và tối ưu kết nối các thiệt bị phần cứng
    - Việc lựa chọn giữa các loại động cơ như Step Motor, DC Motor cần cân nhắc về độ chính xác, tải trọng và tốc độ. Raspberry Pi sẽ xử lý các tác vụ

AI, trong khi Arduino được giao nhiệm vụ điều khiển ngoại vi để giảm tải xử lý.

Đề xuất giải pháp tổng quan

Hệ thống được đề xuất sử dụng Raspberry Pi 4 làm bộ điều khiển trung tâm, tích hợp 3 mô hình AI chính:

- Nhận diện khuôn mặt: Xác thực người dùng hợp lệ để tự động mở cửa.
- Phát hiện người lạ: Giám sát và cảnh báo xâm nhập trái phép qua camera.
- Nhận diện giọng nói: Điều khiển thiết bị thông qua lệnh.

Hệ thống kết nối với Arduino để điều khiển các thiết bị vật lý như cửa, đèn, còi… thông qua động cơ Step Motor, DC, LED, buzzer... giúp tối ưu hóa hiệu suất và độ trễ. Tất cả được quản lý qua giao diện web trực quan, hỗ trợ người dùng:

- Theo dõi trạng thái hệ thống theo thời gian thực
- Điều khiển thiết bị từ xa
- Cập nhật người dùng, dữ liệu khuôn mặt
- Tùy chỉnh lệnh giọng nói theo nhu cầu

Giải pháp hướng đến sự tiết kiệm, tiện lợi và dễ mở rộng, phù hợp với điều kiện và trình độ kỹ thuật tại Việt Nam.

Vấn đề	Giải pháp đề xuất
Phần cứng	Raspberry Pi 4 Arduino Uno R3 Camera Raspberry V1.3 Driver ULN2003 L298N
Nhận diện con người	Thử nghiện với các mô hình YOLO, Fast – CNN, ….
Nhận diện mặt người	Thử nghiệm với các mô hình MobileNetv1, MobileNetv2, DeepFace MobileFaceNet, , Face_recognition, …
Chuyển lời nói thành văn bản	Thử nghiệm với các mô hình: Open AI Whisper, VinAI PhoWhisper,..

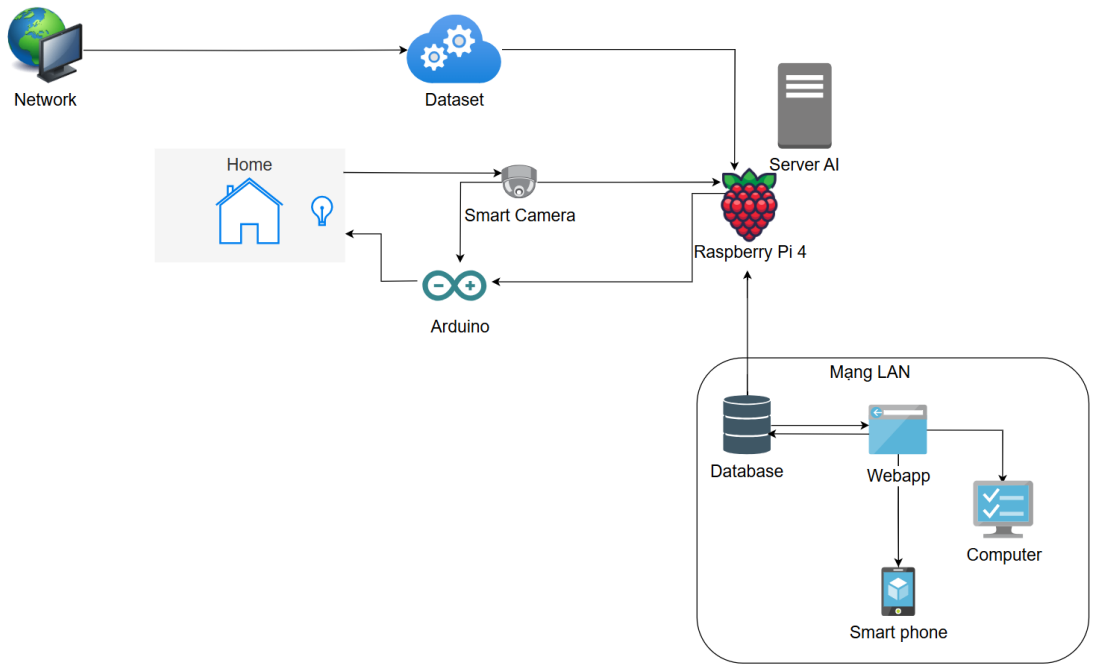
	Thử nghiệm với Google Cloud Speech API
Ứng dụng	Xây dựng Website quản lý người dùng Kết nối với Ardunio để gởi kết quả cũng như trạng thái xử lý
Server	Viết bằng FastAPI

Bảng 1. Đề xuất các giải pháp tổng quan

2. Giải pháp

2.1. Giải pháp phần cứng và truyền thông

2.1.1. Sơ đồ tổng quan hệ thống



Hình 1. Sơ đồ tổng quan hệ thống

Hệ thống được thiết kế dựa trên kiến trúc điều khiển phân tán, trong đó Raspberry Pi 4 đóng vai trò là bộ điều khiển trung tâm, chịu trách nhiệm xử lý các thuật toán nhận diện thông minh như nhận diện khuôn mặt, phát hiện người và xử lý giọng nói thông qua các mô hình AI đã được tích hợp. Sau khi xử lý dữ liệu đầu vào, Raspberry Pi 4 sẽ truyền các lệnh điều khiển cụ thể xuống vi điều khiển Arduino thông qua giao tiếp UART (serial) hoặc I2C.

Arduino, với nhiệm vụ là bộ điều khiển cấp thấp, sẽ nhận các tín hiệu điều khiển từ Raspberry Pi và thực hiện các thao tác vật lý tương ứng như:

- Bật/tắt đèn LED (để báo trạng thái)
- Điều khiển động cơ DC thông qua module L298N (cho các chuyển động đơn giản như đóng/mở cửa hoặc di chuyển)
- Bật/tắt còi tương ứng với yêu cầu được lập trình

- Điều khiển động cơ bước Stepper (28BYJ-48 qua Driver ULN2003) để thực hiện chuyển động chính xác và có kiểm soát hơn như quay camera giám sát.

Việc tách riêng xử lý giữa Raspberry Pi (xử lý cao cấp, trí tuệ nhân tạo) và Arduino (điều khiển thiết bị ngoại vi thời gian thực) giúp hệ thống hoạt động ổn định, hiệu quả và có thể mở rộng dễ dàng. Khi có sự kiện xảy ra (ví dụ có người được nhận diện đúng), Raspberry Pi sẽ gửi lệnh xuống Arduino để thực thi hành động tương ứng một cách nhanh chóng và chính xác.

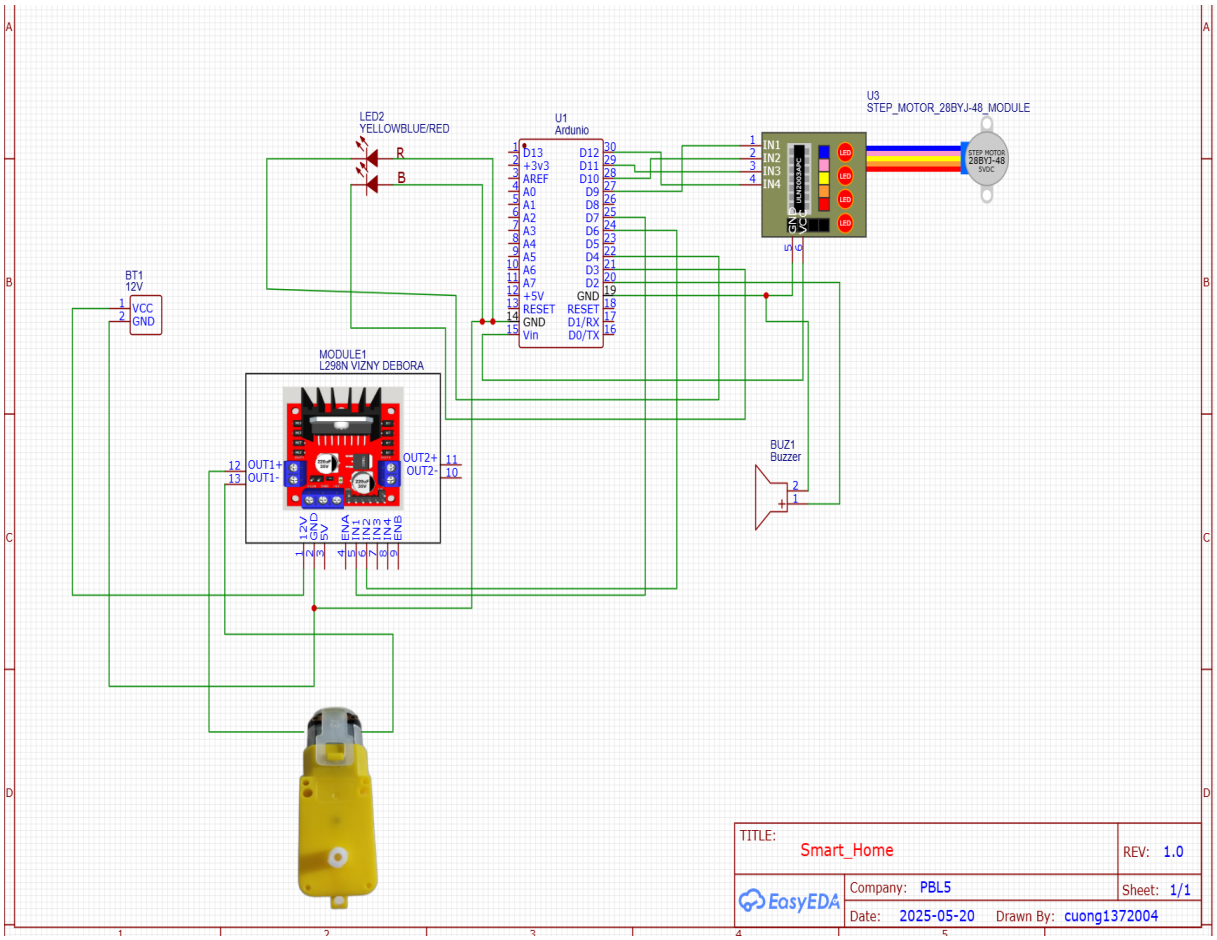
2.1.2. Sơ đồ kết nối các linh kiện phần cứng

1. Các thiết bị phần cứng lắp đặt:
    1. Máy tính nhúng Raspberry Pi 4
        - Vai trò: Xử lý chính, chạy hệ điều hành Linux, mô hình AI (như YOLO, nhận diện giọng nói...), giao tiếp với Arduino và điều khiển tín hiệu cao cấp.
        - Nối với:
            - Arduino qua USB.
            - Mạng Wi-Fi hoặc Ethernet để điều khiển từ xa qua web/app.
    2. Arduino
        - Vai trò: Điều khiển các thiết bị phần cứng (motor, LED, buzzer) theo lệnh từ Raspberry Pi.
        - Nối với Raspberry Pi:
            - Qua cổng USB (serial).
            - Nhận lệnh điều khiển từ Raspberry Pi để điều động động cơ, cảnh báo...
    3. Module L298N
        - Vai trò: Điều khiển động cơ DC về hướng quay và tốc độ.
        - Nối với Arduino:
            - IN1 → chân digital
            - IN2 → chân digital
            - ENA → PWM
            - OUT1, OUT2 → 2 chân motor DC
        - Nguồn cấp:

- 12V vào chân +12V
- GND nối GND chung với Arduino
- Jumper ENA để dùng PWM từ Arduino

4. Động cơ DC Motor
   - Vai trò: Quay liên tục, có thể dùng để đóng/mở cửa, quạt, băng chuyền.
   - Nối: 2 dây vào OUT1, OUT2 của L298N.
5. Động cơ bước 28BYJ48-5V
   - Vai trò: Điều khiển góc quay chính xác theo bước (mở cửa từng giai đoạn, quay camera...).
   - Nối với ULN2003 Driver:
     - Dây 5 chân từ động cơ cắm vào driver (theo chuẩn màu).
   - Nối ULN2003 với Arduino:
     - IN1 → chân digital
     - IN2 → chân digital
     - IN3 → chân digital
     - IN4 → chân digital
   - Nguồn cấp:
     - +5V → VCC driver
     - GND nối chung Arduino
6. Driver Điều Khiển Động Cơ Bước ULN2003
   - Vai trò: ULN2003 là mạch tích hợp 7 transistor Darlington, thường dùng để điều khiển động cơ bước loại 28BYJ-48 5V.
   - Hỗ trợ điều khiển chính xác các bước xoay (mỗi bước ~5.625°), thích hợp cho các ứng dụng như đóng/mở cửa tự động, quay camera,...
7. Đèn Led
   - Vai trò: Báo hiệu trạng thái (mở khóa, nhận diện thành công, cảnh báo...).
   - Nối với Arduino:
     - Anode (+) → điện trở 220Ω → chân digital
     - Cathode (-) → GND
8. Buzzer
   - Vai trò: Cảnh báo bằng âm thanh khi có sự kiện (nhận diện sai, mở cửa...).
   - Nối với Arduino:

- Chân + → Điện trở 100–220Ω → chân digital
- Chân - → GND



Hình 2. Sơ đồ lắp mạch

2.1.3. Các thiết bị phần cứng sử dụng

Raspberry Pi 4 Model B

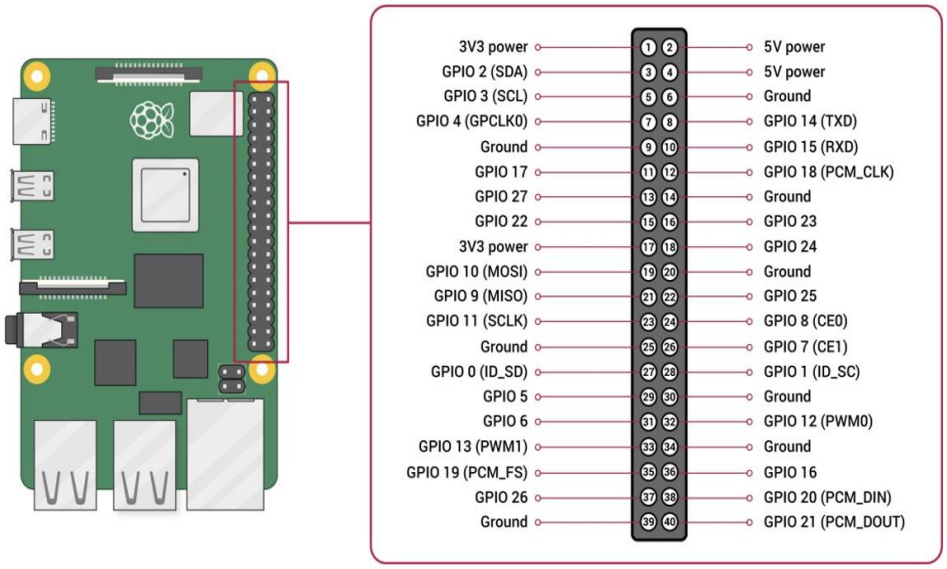
Hình 3. Rasberry Pi 4 Model B

Raspberry Pi 4 Model B là phiên bản mới nhất trong dòng máy tính nhỏ gọn nổi tiếng Raspberry Pi, được thiết kế để mang lại hiệu năng cao hơn và hỗ trợ nhiều tính năng tiên tiến cho các ứng dụng giáo dục, nghiên cứu, và phát triển nhúng. Với khả năng xử lý mạnh mẽ hơn, bộ nhớ RAM đa dạng, và khả năng kết nối phong phú, Raspberry Pi 4 Model B mở rộng đáng kể tiềm năng sáng tạo cho người dùng.

Các thông số kỹ thuật chính của máy tính nhúng:

- Bộ vi xử lý: Broadcom BCM2711, lõi tứ ARM Cortex-A72 (ARM v8) 64-bit, tốc độ 1.5 GHz.
- Bộ nhớ RAM: Tùy chọn 2 GB, 4 GB, hoặc 8 GB LPDDR4-3200 SDRAM.
- Đồ họa: Broadcom VideoCore VI, hỗ trợ OpenGL ES 3.1, 4Kp60 HEVC video decode
- Lưu trữ: Khe cắm thẻ microSD (hỗ trợ UHS-I), USB 3.0, USB 2.0.
- Cổng USB: 2 x USB 3.0, 2 x USB 2.0.
- Kết nối mạng: Ethernet Gigabit, hỗ trợ PoE (yêu cầu bổ sung HAT PoE).
- Không dây: Wi-Fi 802.11 b/g/n/ac (2.4 GHz và 5.0 GHz), Bluetooth 5.0, BLE.
- Cổng màn hình: 2 x micro-HDMI, hỗ trợ độ phân giải lên đến 4Kp60.

- Cổng âm thanh: Jack âm thanh 3.5 mm (âm thanh và video composite), hỗ trợ HDMI audio.
- Nguồn điện: Cổng USB-C, cung cấp nguồn 5V/3A. [1]



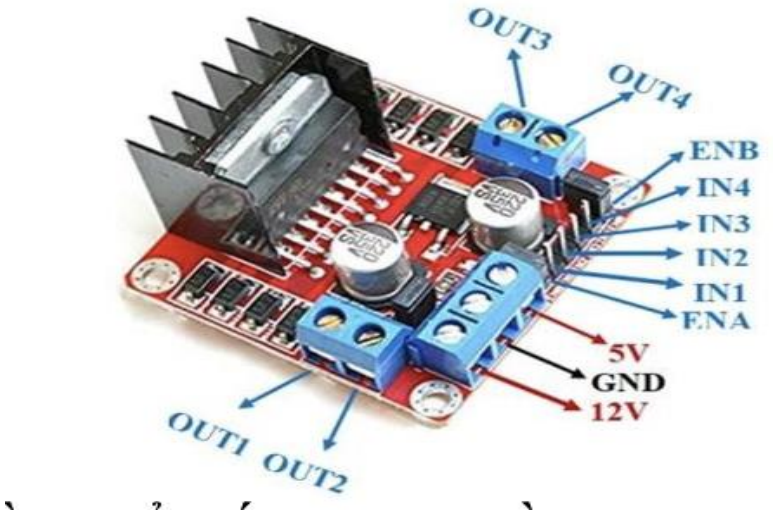
Hình 4: Sơ đồ chân của Rasperry Pi 4

Giao thức truyền thông và các chân giao tiếp:

- GPIO: 40 chân GPIO tiêu chuẩn, cung cấp giao diện để kết nối với các thiết bị phần cứng bên ngoài như cảm biến, đèn LED, hoặc module giao tiếp.
- UART: Universal Asynchronous Receiver/Transmitter, dùng để giao tiếp nối tiếp với các thiết bị khác như mô-đun GPS hoặc module không dây.
- SPI: Serial Peripheral Interface, một giao thức truyền thông nối tiếp tốc độ cao dùng để kết nối với các thiết bị như màn hình OLED hoặc cảm biến.
- I2C: Inter-Integrated Circuit, một giao thức truyền thông nối tiếp cho phép kết nối với các thiết bị như bộ giải mã ADC, cảm biến nhiệt độ, hoặc EEPROM.
- Ethernet: Hỗ trợ truyền thông mạng có dây với tốc độ Gigabit Ethernet, giúp tăng tốc độ truyền dữ liệu và độ tin cậy khi kết nối mạng.
- USB: Các cổng USB 3.0 và USB 2.0 cho phép kết nối với nhiều loại thiết bị ngoại vi như ổ cứng ngoài, bàn phím, chuột, và thiết bị lưu trữ USB.

Module điều khiển động cơ L298N

Module điều khiển động cơ L298N là một thiết bị thông dụng được sử dụng để điều khiển động cơ DC và động cơ bước trong các dự án điện tử và robot. Nó sử dụng chip L298N, một cầu H kép mạnh mẽ cho phép điều khiển độc lập hai động cơ DC hoặc một động cơ bước. Module này hỗ trợ nhiều loại động cơ và cung cấp khả năng điều khiển tốc độ và chiều quay một cách linh hoạt.



Hình 5:Module điều khiển động cơ L298N

Dưới đây là một vài thông số cơ bản về nó:

1. **Thông số điện:**
    - Điện áp đầu vào cho động cơ (Vms): 5V đến 35V
    - Điện áp logic: 5V
    - Dòng điện đầu ra: Lên đến 2A mỗi cầu H (tổng cộng 4A với cả hai cầu H)
    - Dòng điện cực đại: 3A mỗi cầu H (trong thời gian ngắn)
2. **Đặc điểm vật lý:**
    - Kích thước: Khoảng 43 mm x 43 mm x 27 mm
    - Trọng lượng: Khoảng 30 grams
3. **Giao tiếp và điều khiển:**
    - Điều khiển tốc độ: PWM (Pulse Width Modulation)
    - Điều khiển chiều quay: Sử dụng các chân Input (IN1, IN2, IN3, IN4) để xác định chiều quay của động cơ
4. **Chân điều khiển:**
    - ENA: Điều khiển tốc độ động cơ 1
    - ENB: Điều khiển tốc độ động cơ 2

- IN1, IN2: Điều khiển chiều quay động cơ 1
- IN3, IN4: Điều khiển chiều quay động cơ 2

5. **Các thông số khác:**
    - Bảo vệ quá nhiệt: Tích hợp bảo vệ nhiệt trên chip L298N
    - Diode bảo vệ: Có các diode bảo vệ chống lại dòng ngược từ động cơ
    - Tản nhiệt: Tích hợp tản nhiệt để giảm nhiệt độ trong quá trình hoạt động [3]

Động Cơ DC Giảm Tốc Vàng

Động cơ DC giảm tốc Vàng (Yellow DC Gear Motor) là một loại động cơ DC tích hợp hộp số giảm tốc, thường được sử dụng trong các ứng dụng yêu cầu lực xoắn cao và tốc độ quay thấp. Động cơ này phổ biến trong các dự án robot di động, xe điều khiển từ xa, và các hệ thống truyền động cơ bản do tính kinh tế, dễ sử dụng, và hiệu suất đáng tin cậy.

Hình 6. Động cơ DC giảm tốc vàng

Động cơ DC giảm tốc Vàng kết hợp một động cơ DC nhỏ với một hộp số giảm tốc tích hợp. Hộp số giảm tốc giúp giảm tốc độ quay của động cơ và tăng lực xoắn, cho phép động cơ cung cấp sức mạnh cần thiết để di chuyển các tải trọng nặng hoặc vượt qua các trở ngại lớn.

Nguyên lý hoạt động cơ bản của động cơ này dựa trên việc chuyển đổi năng lượng điện từ nguồn DC thành chuyển động quay của trục động cơ. Hộp số giảm tốc thay đổi tỷ lệ quay, biến tốc độ cao và lực xoắn thấp của động cơ thành tốc độ thấp và lực xoắn cao hơn ở trục đầu ra.

1. **Dưới đây là thông số kỹ thuật của nó:**
    - Điện áp hoạt động: 3V - 12V DC.
    - Tốc độ quay (ở 6V):
    - Tỷ lệ giảm tốc 1:48: 200 RPM.
    - Tỷ lệ giảm tốc 1:120: 100 RPM.
    - Lực xoắn (ở 6V):
    - Tỷ lệ giảm tốc 1:48: 1.1 kg.cm.
2. **Giao thức điều khiển và các chân giao tiếp**
    - Dây nguồn (Power): Kết nối với cực dương của nguồn cung cấp điện (thường màu đỏ).
    - Dây nối đất (Ground): Kết nối với cực âm của nguồn cung cấp điện (thường màu đen hoặc xanh dương).

Động Cơ Bước 28BYJ-48-5V và Driver ULN2003

Động cơ bước 28BYJ-48-5V là một loại động cơ bước nhỏ gọn, giá rẻ, thường được sử dụng trong các ứng dụng điện tử tiêu dùng và dự án DIY. Động cơ này hoạt động dựa trên nguyên lý điều khiển từng bước nhỏ, cho phép định vị chính xác và kiểm soát tốc độ quay. Để điều khiển động cơ bước này, người dùng thường sử dụng driver điều khiển ULN2003, một mạch tích hợp có khả năng điều khiển các cuộn dây của động cơ bằng cách chuyển đổi các tín hiệu từ bộ điều khiển thành dòng điện điều khiển.

Hình 7: Động Cơ Bước 28BYJ-48-5V và Driver ULN2003

1. **Dưới đây là thông số thiết kế của cụm thiết bị**
    - Điện áp định mức: 5V DC.
    - Tỷ số truyền: 64:1.
    - Bước góc: 5.625°/64 (sau tỷ số truyền).
    - Số bước mỗi vòng: 2048 bước.
    - Dòng điện định mức: 240 mA.
    - Điện trở cuộn dây: Khoảng 50Ω.
    - Mô-men xoắn tĩnh: ≥ 34.3 mN·m (120 Hz).
    - Kích thước: 28 mm x 28 mm x 25 mm.
    - Khối lượng: Khoảng 30 g. [7]
2. **Driver điều khiển ULN2003**
    - Điện áp điều khiển: 5V - 12V DC.
    - Dòng điện đầu ra: Tối đa 500 mA cho mỗi kênh.
    - Số kênh: 7 (có thể điều khiển 4 cuộn dây của động cơ bước).
    - Bảo vệ chống quá áp: Có (flyback diodes).
    - Giao diện điều khiển: Tín hiệu số từ bộ điều khiển (thường từ vi điều khiển hoặc Arduino).
    - Kích thước: 41 mm x 31 mm. [8]

3. Cấu tạo và nguyên lý hoạt động:

- Động cơ bước 28BYJ-48-5V là một loại động cơ bước đơn cực, có cấu trúc gồm bốn cuộn dây (A, A', B, B') nối với các đầu ra. Khi dòng điện chạy qua các cuộn dây theo thứ tự, nó tạo ra các xung lực từ, đẩy rotor quay theo từng bước. Động cơ bước này có tỷ số truyền 64:1 nhờ hộp số bên trong, cho phép điều khiển vị trí rotor với độ chính xác cao. [7]
- Driver điều khiển ULN2003 là một mạch tích hợp gồm bảy transistor Darlington, cho phép điều khiển các cuộn dây của động cơ bước. Các transistor này chuyển đổi tín hiệu điều khiển từ bộ điều khiển thành dòng điện đủ mạnh để kích hoạt cuộn dây của động cơ. ULN2003 cũng bao gồm các diode bảo vệ để ngăn chặn dòng điện ngược khi cuộn dây ngắt kết nối. [8]

4. Sơ đồ chân kết nối:

Chân	Màu dây	Chức năng
1	Cam	Coil A
2	Vàng	Coil B
3	Hồng	Coil C
4	Xanh	Coil D
5	Đỏ	VCC

Bảng 2. Động cơ bước 28BYJ-48-5V

Chân ULN2003	Chức năng	Kết nối với động cơ
IN1	Điều khiển Coil A	Coil A
IN2	Điều khiển Coil B	Coil B
IN3	Điều khiển Coil C	Coil C
IN4	Điều khiển Coil D	Coil D
COM	Nguồn VCC cho động cơ	VCC
GND	Nối đất	Nối đất

Bảng 3. Driver điều khiển ULN2003

Remote điều khiển

Module thu hồng ngoại + Remote IR1838 là một thiết bị phổ biến trong các dự án điều khiển từ xa sử dụng vi điều khiển như Arduino. Mạch sử dụng cảm biến IR1838 có khả năng nhận tín hiệu hồng ngoại từ các loại remote chuẩn NEC, RC5 hoặc Sony,

thường hoạt động ở tần số 38kHz. Remote đi kèm cho phép truyền các mã lệnh riêng biệt để điều khiển thiết bị từ khoảng cách xa.

Module IR1838 có thiết kế nhỏ gọn, dễ sử dụng, và chỉ cần kết nối đơn giản qua ba chân VCC, GND và OUT. Tín hiệu nhận được có dạng digital, dễ dàng đọc và xử lý qua phần mềm với thư viện IRremote. Thiết bị này thường được dùng trong các ứng dụng như điều khiển robot, bật/tắt thiết bị điện, mở cửa tự động, hoặc kích hoạt các hành vi theo lệnh từ xa.

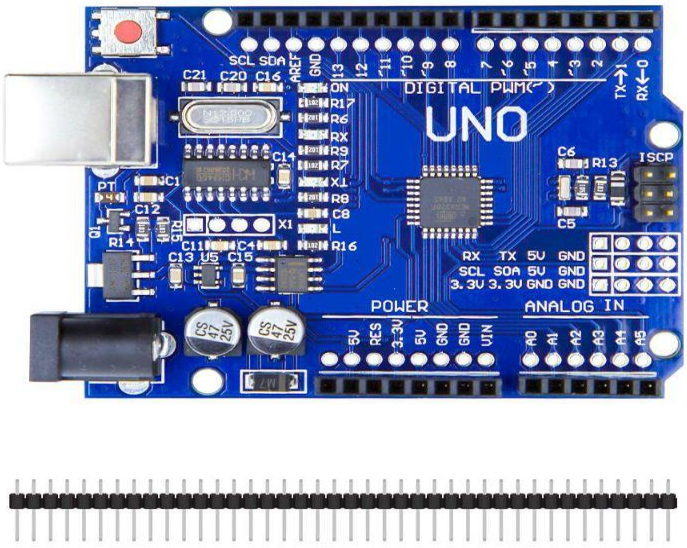
Hình 8. Module thu hồng ngoại và Remote IR 1838

Dưới đây là một vài thông số cơ bản về nó:

1. **Thông số điện:**
    - Điện áp hoạt động: 5VDC
    - Dạng tín hiệu ngõ ra: Digital (mức thấp khi nhận tín hiệu, mức cao khi không có tín hiệu)
    - Dòng tiêu thụ: Rất nhỏ, thường dưới 5mA
2. **Đặc điểm vật lý:**
    - Kích thước cảm biến IR1838: khoảng 5mm x 5mm x 7mm
    - Kích thước remote: khoảng 85mm x 40mm x 6mm
    - Trọng lượng: Nhẹ, dưới 10g

3. **Giao tiếp và điều khiển:**
    - Tần số hoạt động: 38kHz (chuẩn cho các loại remote IR phổ thông)
    - Khoảng cách hoạt động: Từ 5 đến 10 mét tùy điều kiện ánh sáng
    - Góc nhận tín hiệu: Khoảng ±45 độ
    - Giao thức điều khiển: Thường sử dụng chuẩn NEC, nhưng cũng tương thích với một số chuẩn khác (Sony, RC5...)
4. **Chân điều khiển:**
    - OUT: Tín hiệu dữ liệu đầu ra (kết nối đến chân digital của Arduino)
    - VCC: Cấp nguồn 5V
    - GND: Nối đất
5. **Các thông số khác:**
    - Có khả năng lọc nhiễu IR từ môi trường xung quanh
    - Tích hợp mạch giải mã tín hiệu IR
    - Dễ sử dụng trong các ứng dụng điều khiển từ xa bằng hồng ngoại
    - Ứng dụng phổ biến: xe điều khiển từ xa, mở cửa tự động, hệ thống cảnh báo, điều khiển đèn...

Arduino Uno R3



Hình 9. Arduino Uno R3

Arduino Uno R3 là một board vi điều khiển dựa trên chip ATmega328P, được sử dụng rộng rãi trong các dự án điện tử, tự động hóa và giáo dục STEM. Với thiết kế đơn giản, thân thiện với người mới bắt đầu, Uno R3 cung cấp các chân kết nối đa dạng và khả năng giao tiếp mạnh mẽ, là nền tảng lý tưởng để xây dựng các hệ thống nhúng, robot, cảm biến IoT và nhiều ứng dụng khác.

Dưới đây là một vài thông số cơ bản về nó:

1. **Vi điều khiển chính:**
    - Loại vi điều khiển: ATmega328P
    - Tốc độ xung nhịp: 16 MHz
2. **Điện áp :**
    - Điện áp hoạt động (logic): 5V
    - Điện áp cấp nguồn khuyến nghị: 7–12V (qua jack DC)
    - Điện áp đầu vào giới hạn: 6–20V (VIN)
    - Dòng điện tối đa cho mỗi chân I/O: 40 mA
3. **Bộ nhớ:**
    - Flash: 32 KB (trong đó 0.5 KB dành cho bootloader)
    - SRAM: 2 KB
    - EEPROM: 1 KB
4. **Chân I/O :**
    - Tổng chân digital I/O: 14 (trong đó 6 chân hỗ trợ PWM: 3, 5, 6, 9, 10, 11)
    - Chân analog input: 6 (A0–A5)
    - Chân nguồn: GND, 3.3V, 5V, VIN, IOREF
5. **Giao tiếp:**
    - UART (Serial): 1
    - SPI: Chân 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK)
    - I2C: A4 (SDA), A5 (SCL)
    - USB: Cổng USB Type-B (dùng để nạp chương trình và giao tiếp)

2.1.4. Truyền thông

1. **HTTP (Hypertext Transfer Protocol)**

Hypertext Transfer Protocol (HTTP) là một giao thức tầng ứng dụng trong mô hình OSI, được thiết kế để truyền tải các tài liệu siêu văn bản trên World Wide Web.

HTTP hoạt động dựa trên mô hình yêu cầu-phản hồi, nơi máy khách gửi yêu cầu tới máy chủ và nhận về tài liệu, bao gồm cả văn bản, hình ảnh, và video. Với các phương thức chính như GET, POST, PUT, DELETE, HTTP cung cấp cách thức giao tiếp chuẩn hóa cho các ứng dụng web và API. Trong các ứng dụng với Raspberry Pi, HTTP có thể được sử dụng để thiết lập các máy chủ web nhỏ, cho phép điều khiển và giám sát thiết bị từ xa thông qua giao diện web hoặc API RESTful.

2. **SPI (Serial Peripheral Interface)**

Serial Peripheral Interface (SPI) là một giao thức truyền thông đồng bộ, được thiết kế để truyền dữ liệu giữa một thiết bị chính (master) và một hoặc nhiều thiết bị phụ (slave). SPI sử dụng bốn đường tín hiệu chính: MISO (Master In Slave Out), MOSI (Master Out Slave In), SCLK (Serial Clock), và SS (Slave Select), để điều phối việc truyền nhận dữ liệu. Với khả năng truyền dữ liệu song song tốc độ cao, SPI thường được sử dụng trong các hệ thống nhúng để kết nối vi điều khiển với các thiết bị ngoại vi như cảm biến, màn hình, và bộ nhớ. Trên Raspberry Pi, SPI thường được sử dụng để giao tiếp với các màn hình TFT, cảm biến và các module mở
rộng.

3. **FastAPI**

FastAPI là một framework hiện đại, hiệu suất cao dùng để xây dựng API với Python, được phát triển trên nền tảng ASGI (Asynchronous Server Gateway Interface). FastAPI cho phép xây dựng các RESTful API một cách nhanh chóng và rõ ràng, hỗ trợ tốt các tính năng như xác thực, kiểm tra dữ liệu, và xử lý bất đồng bộ (async/await).
Trong các ứng dụng nhúng như hệ thống giám sát sử dụng Raspberry Pi, FastAPI thường được triển khai để:

- Giao tiếp giữa thiết bị và server backend, như gửi kết quả nhận diện khuôn mặt hoặc cảnh báo an ninh.
- Tạo API điều khiển (mở/đóng cửa từ xa, lấy lịch sử truy cập, hiển thị ảnh).
- Kết nối với hệ thống quản lý trung tâm qua HTTP hoặc WebSocket.

FastAPI còn hỗ trợ tự động sinh tài liệu API thông qua Swagger UI hoặc Redoc, giúp quá trình phát triển và tích hợp dễ dàng hơn.

4. **API**

API là viết tắt của Application Programming Interface, là giao diện cho phép các thành phần phần mềm hoặc thiết bị phần cứng giao tiếp với nhau. Trong hệ thống nhúng

hoặc ứng dụng IoT như đồ án này, API đóng vai trò trung gian giữa thiết bị đầu cuối (như Raspberry Pi) và hệ thống điều khiển hoặc server backend.

Các loại API phổ biến bao gồm:

- RESTful API (qua HTTP): dùng để truyền và nhận dữ liệu bằng các phương thức như GET, POST, PUT, DELETE.
- WebSocket API: dùng cho truyền dữ liệu thời gian thực (real-time).
- Local API: giữa các tiến trình nội bộ hoặc trên cùng một thiết bị.

5. SSH

SSH (viết tắt của Secure Shell) là một giao thức mạng giúp chúng ta, kết nối điều khiển từ xa một thiết bị qua dòng lệnh, giao tiếp an toàn (mã hóa) giữa client (máy bạn) và server (Raspberry Pi).

Ở dự án lần này tôi dùng Termius – một ứng dụng SSH đa nền tảng rất phổ biến để thực hiện SSH gồm các bước sau:

- Bước 1: Cài đặt SSH trên Raspberry Pi (nếu chưa bật)

  Bật bằng câu lệnh sau "sudo raspi-config" khi kết nối thẻ SD card của ras và máy tính. Sau đó vào Interfacing Options > SSH > Enable.

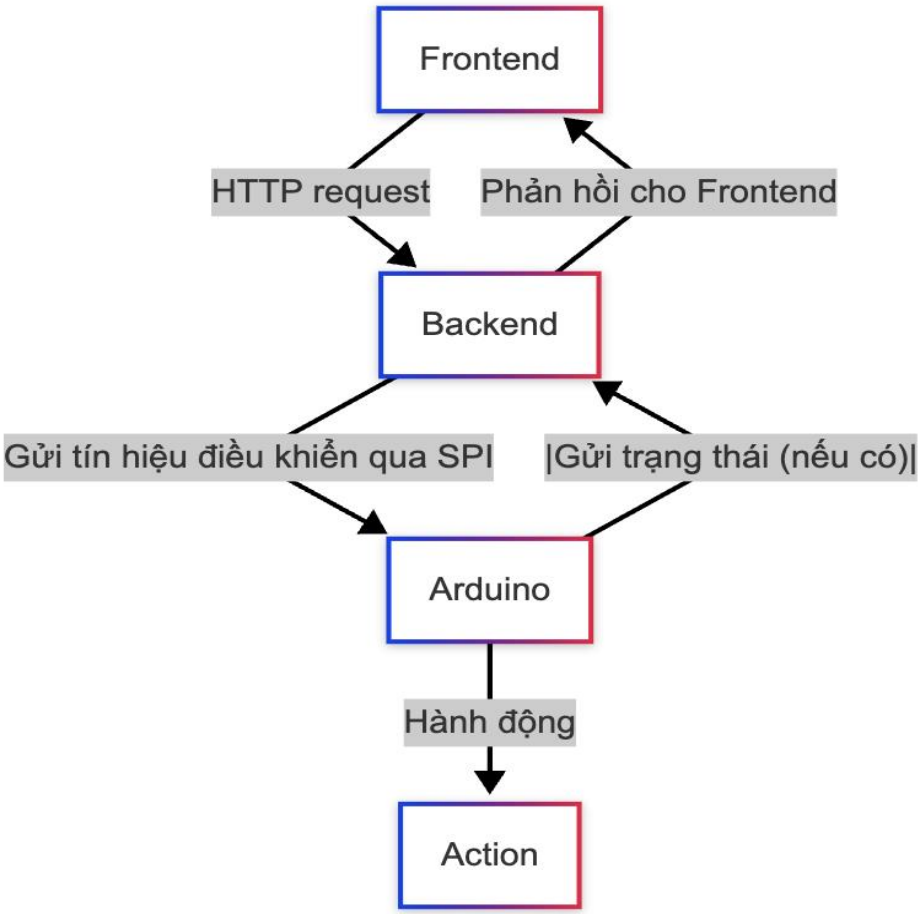
- Bước 2: Kết nối Raspberry Pi vào cùng mạng với thiết bị chạy Termius

  Raspberry Pi cần kết nối cùng mạng Wi-Fi với điện thoại/máy tính chạy Termius. Dùng ứng dụng quét mạng (Fing) để kiểm tra IP của Raspberry Pi.

- Bước 3: Mở Termius và tạo kết nối

  Mở Termius (trên điện thoại hoặc máy tính), bấm "New host", nhập thông tin và tiến hành kết nối.

6. Tóm tắt luồng truyền thông:



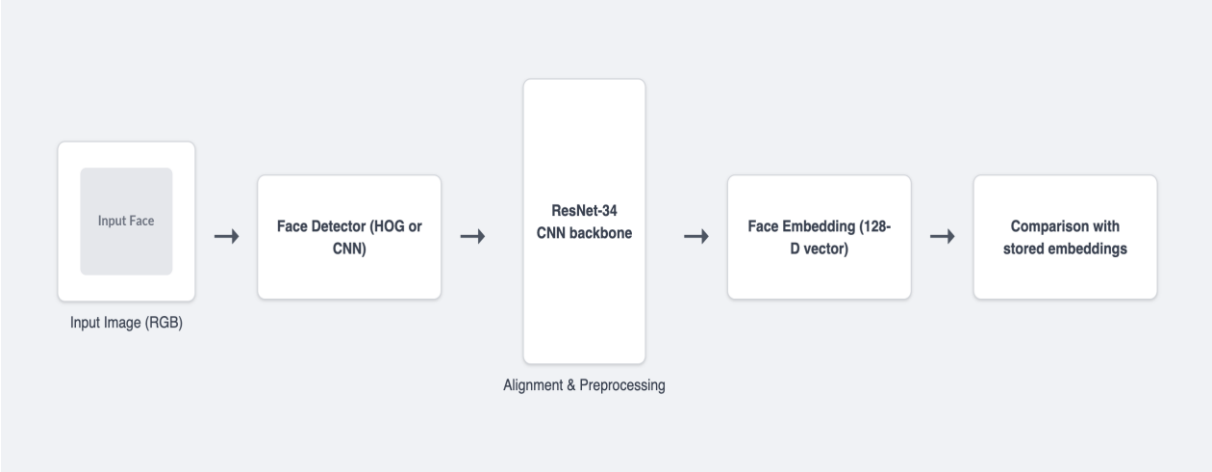
Hình 10. Luồng truyền thông

2.2. Giải pháp AI/KHDL

2.2.1. Giải pháp nhận diện khuôn mặt

Trong đồ án này, nhóm sử dụng thư viện mã nguồn mở Face_recognition một thư viện nổi tiếng trong Python được xây dựng dựa trên mô hình deep learning ResNet được huấn luyện sẵn bởi dlib. Thư viện này hỗ trợ việc phát hiện khuôn mặt, trích xuất đặc trưng (embedding) và so sánh khuôn mặt với độ chính xác cao, hiệu suất tốt và dễ triển khai trên các thiết bị nhúng như Raspberry Pi.

Quy trình nhận diện khuôn mặt được thực hiện qua các bước sau:



Hình 11. Quy trình nhận diện

- Phát hiện khuôn mặt trong ảnh đầu vào bằng mô hình HOG hoặc CNN.
- Mã hóa khuôn mặt (face encoding): Trích xuất vector 128 chiều đại diện cho đặc trưng khuôn mặt bằng mạng nơ-ron sâu.
- So sánh với dữ liệu khuôn mặt đã lưu: Sử dụng khoảng cách Euclidean giữa các vector để xác định xem khuôn mặt đầu vào có khớp với người nào đã được đăng ký trước đó hay không.
- Nếu khoảng cách nhỏ hơn ngưỡng cho phép (mặc định 0.6), hệ thống xác nhận danh tính và cho phép mở cửa.

Thư viện face_recognition có ưu điểm là dễ tích hợp, không cần huấn luyện lại mô hình và hoạt động hiệu quả với cơ sở dữ liệu nhỏ đến trung bình. Ngoài ra, nhóm kết hợp với thư viện OpenCV để xử lý video trực tiếp từ camera và hiển thị kết quả nhận diện theo thời gian thực.

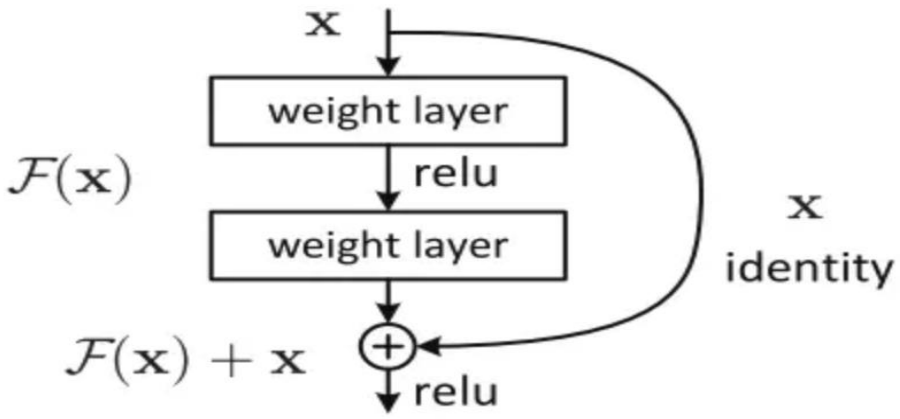
Giải pháp này phù hợp với mục tiêu của hệ thống: nhận diện nhanh, chính xác, dễ triển khai trên thiết bị nhỏ gọn và tiêu thụ ít tài nguyên.

1. Mục đích của mô hình nhận diện khuôn mặt

- Ảnh đầu vào sau khi được phát hiện khuôn mặt và căn chỉnh sẽ được đưa vào ResNet-34.
- Mạng này học các đặc trưng khuôn mặt sâu và chuyển đổi ảnh thành vector embedding 128 chiều (đại diện cho khuôn mặt đó).
- Vector này sau đó được so sánh với các vector khác để xác định người.

2. Giới thiệu mô hình ResNet-34 trong thư viện face_recognition

- Là một mô hình CNN sâu gồm 34 lớp, thuộc họ Residual Networks (ResNet).
- Được giới thiệu bởi Microsoft Research năm 2015 trong bài báo "Deep Residual Learning for Image Recognition".
- Khác biệt chính của ResNet so với các CNN truyền thống là sử dụng residual connections (skip connections) – cho phép thông tin "nhảy qua" một số lớp, giúp huấn luyện các mạng rất sâu mà không bị mất thông tin hay vanishing gradient.
- Khối residual connections (skip connections):



Hình 12. Khối Residual Block

Cách hoạt động chi tiết

1. Đầu vào x đi vào 2 lớp convolution (cùng với Batch Normalization và ReLU).
2. Kết quả đầu ra của 2 lớp conv gọi là F(x) (hàm ánh xạ mà block muốn học).
3. Đầu vào ban đầu x được cộng trực tiếp vào F(x), tức là y = F(x) + x.
4. Kết quả cộng y được đưa qua hàm kích hoạt ReLU (hoặc một số trường hợp khác).

Ý nghĩa của phép cộng F(x) + x

- Cho phép mạng học được phần sai số (residual) so với đầu vào ban đầu.

- Nếu block không học gì (ví dụ F(x) ≈ 0), thì output y ≈ x thông tin không bị mất đi.
- Việc này giúp gradient trong quá trình huấn luyện có thể truyền trực tiếp qua các block, tránh hiện tượng biến mất gradient.
- Khi đi qua nhiều lớp, nếu không có phép cộng này, thông tin gốc có thể bị biến dạng hoặc mất dần.
- Với phép cộng, thông tin đầu vào luôn được bảo toàn trong quá trình lan truyền, tránh mất mát quan trọng.

Kiến trúc mô hình ResNet-34

Block	Layer Configuration	Output Size
Conv1	7×7, 64 filters, stride 2	112×112×64
MaxPool	3×3, stride 2	56×56×64
Conv2_x	3×3 × 3 residual blocks	56×56×64
Conv3_x	3×3 × 4 residual blocks	28×28×128
Conv4_x	3×3 × 6 residual blocks	14×14×256
Conv5_x	3×3 × 3 residual blocks	7×7×512
AvgPool	Global average pooling	1×1×512
FC	Fully connected layer	128-d vector

Bảng 4. Mô hình Resnet-34

Backbone trong mô hình học sâu (deep learning):

- Là phần xương sống của mô hình – một mạng nơ-ron chính (thường là CNN) dùng để trích xuất đặc trưng (features) từ đầu vào (ảnh, video, ...).
- Ví dụ như trong bài toán nhận diện khuôn mặt, ta không dùng ảnh trực tiếp để so sánh, mà dùng vector đặc trưng được tạo ra bởi backbone.

CNN backbone trong mô hình học sâu

- Trong học sâu, backbone là mạng nơ-ron chính (thường là CNN) dùng để trích xuất đặc trưng từ dữ liệu đầu vào như ảnh hoặc video.

- Trong bài toán nhận diện khuôn mặt, ảnh không được sử dụng trực tiếp để so sánh, mà được chuyển thành vector đặc trưng thông qua backbone.
- Với face_recognition, CNN backbone là ResNet-34, đầu vào là ảnh khuôn mặt (cỡ 112×112 hoặc 150×150), đầu ra là vector 128 chiều đại diện cho khuôn mặt.

2.2.2. Giải pháp nhận diện người

Trong hệ thống gara thông minh, mô hình phát hiện người giúp xác định sự hiện diện của người trong vùng quan sát, hỗ trợ các chức năng như cảnh báo an ninh hoặc điều khiển thiết bị tự động. Để đáp ứng yêu cầu nhẹ, chính xác và xử lý thời gian thực, nhóm lựa chọn sử dụng YOLOv8n (nano) – phiên bản tối ưu hóa của YOLOv8 dành cho thiết bị nhúng như Raspberry Pi.

1. Mục đích của mô hình nhận diện người

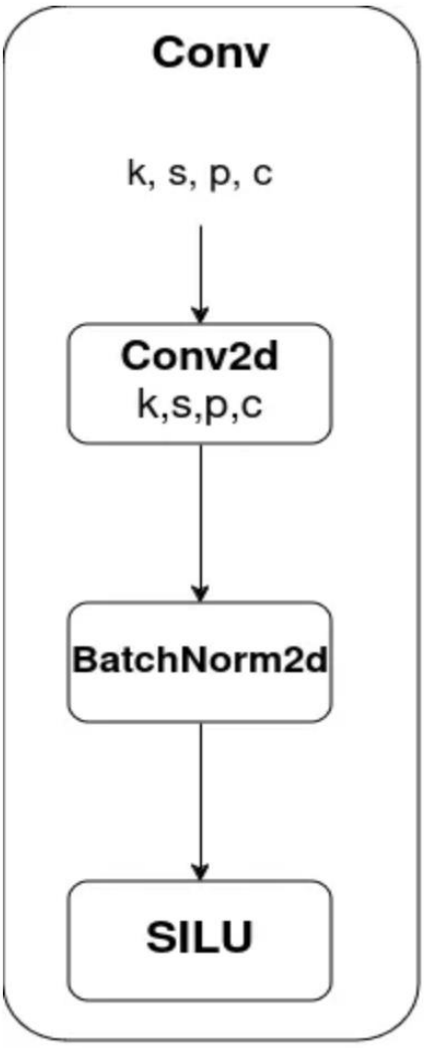
- Phát hiện người trong khung hình từ camera theo thời gian thực.
- Trả về tọa độ các bounding box chứa người và độ tin cậy.
- Làm đầu vào cho các logic xử lý tiếp theo như bật đèn, phát cảnh báo, …

2. Giới thiệu mô hình YOLOv8n

- YOLO (You Only Look Once) là dòng mô hình phát hiện đối tượng một bước (one-stage), nổi bật nhờ tốc độ nhanh và độ chính xác cao.
- YOLOv8n là phiên bản nhẹ nhất của YOLOv8, phù hợp với các thiết bị tài nguyên thấp.
- Mô hình bao gồm 3 phần chính:
  - Backbone: CSPDarknet-nano chịu trách nhiệm trích xuất đặc trưng từ ảnh đầu vào. YOLOv8 sử dụng C2f block và Bottleneck để tăng hiệu quả và giảm tham số.
  - Neck: PAN (Path Aggregation Network) – kết hợp và khuếch đại các đặc trưng đa cấp.
  - Head: dự đoán trực tiếp tọa độ hộp giới hạn**,** lớp đối tượng, và độ tin cậy (confidence score**)** mà không cần anchor box, giúp đơn giản hóa xử lý hậu kỳ và tăng tốc độ.

Các khối được sử dụng trong YOLOv8n

a) Khối tích chập (Conv Block)



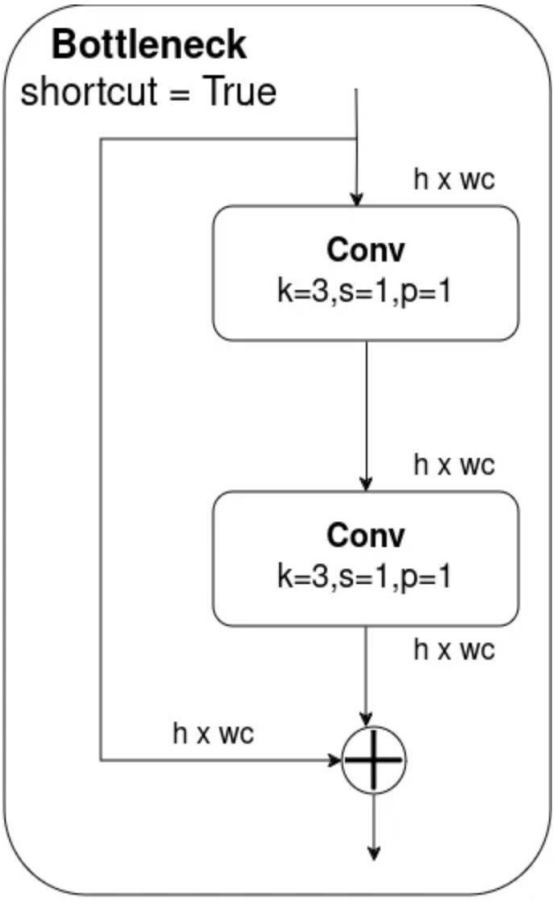
Hình 13. Khối tích chập Conv

Khối cơ bản trong kiến trúc gồm lớp Conv2d, BatchNorm2d và hàm kích hoạt SiLU:

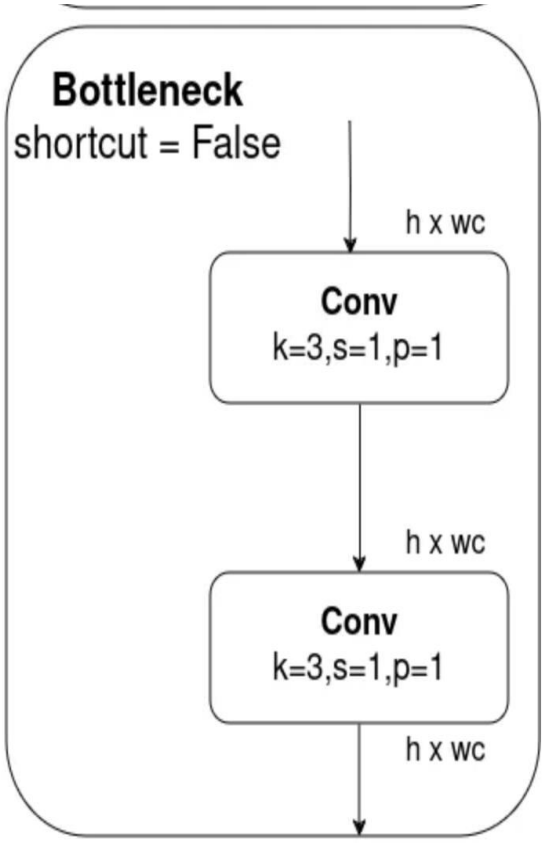
- **Conv2d:** Lớp tích chập 2 chiều trượt bộ lọc (kernel) trên ảnh đầu vào để trích xuất đặc trưng.
  - *k*: số bộ lọc, xác định độ sâu đầu ra.
  - *s*: bước trượt, ảnh hưởng kích thước đầu ra.
  - *p*: padding, thêm viền để giữ kích thước không gian.
  - *c*: số kênh đầu vào (ví dụ ảnh RGB có c=3).
- **BatchNorm2d:** Chuẩn hóa đầu ra của lớp Conv2d theo từng batch, giúp ổn định và tăng tốc độ huấn luyện bằng cách duy trì giá trị đầu ra trong phạm vi phù hợp.

- **Hàm kích hoạt SiLU (Swish):** SiLU(x) = x * sigmoid(x) (sigmoid(x) = 1 / (1 + e^(-x))). SiLU tạo gradient mượt mà, giúp cải thiện hiệu quả huấn luyện và giảm vấn đề gradient biến mất.

b) Khối nút thắt cổ chai(Bottleneck)



Hình 14. Nút thắc cổ chai mở



Hình 15. Nút thắc cổ chai đóng

Khối nút thắt cổ chai (Bottleneck Block) là một khối gồm hai lớp Conv nối tiếp và tùy chọn kết nối phím tắt (shortcut).

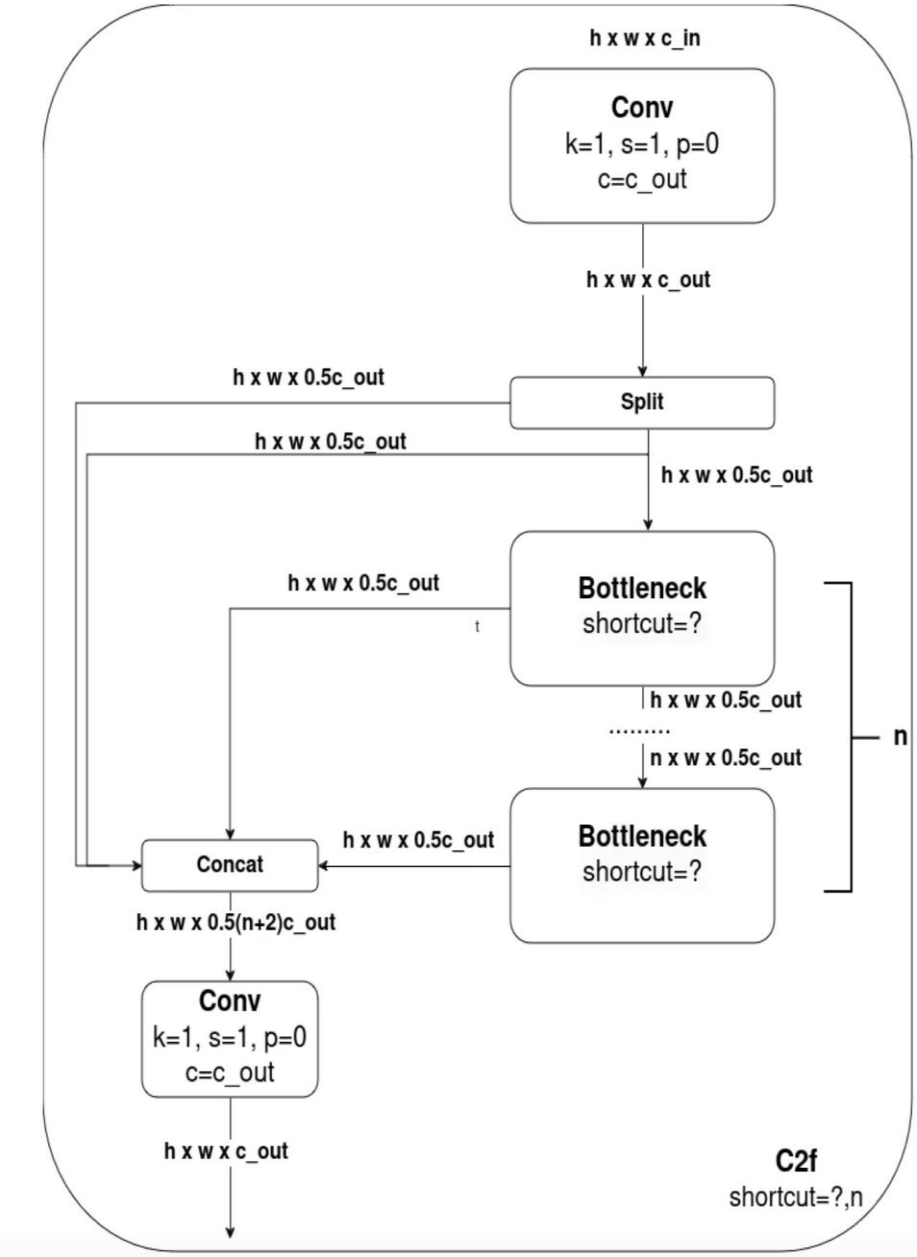
- Nếu shortcut = True, đầu vào sẽ được cộng trực tiếp vào đầu ra (residual connection).
- Nếu shortcut = False, đầu vào đi qua đủ hai lớp Conv mà không cộng thêm.

Kết nối phím tắt (Shortcut Connection) là một loại kết nối bỏ qua một hoặc nhiều lớp, giúp gradient truyền ngược dễ dàng hơn. Cơ chế này giúp:

- Giảm hiện tượng gradient biến mất,
- Cho phép học ánh xạ danh tính dễ dàng hơn,
- Tăng hiệu quả huấn luyện trong các mạng sâu.

Gradient biến mất (Vanishing Gradient) là hiện tượng gradient trở nên rất nhỏ khi lan truyền ngược qua nhiều lớp, làm cho các lớp đầu khó học. Shortcut giúp khắc phục vấn đề này bằng cách duy trì dòng gradient mạnh hơn trong quá trình huấn luyện.

c) Khối C2f



Hình 16. Khối C2f

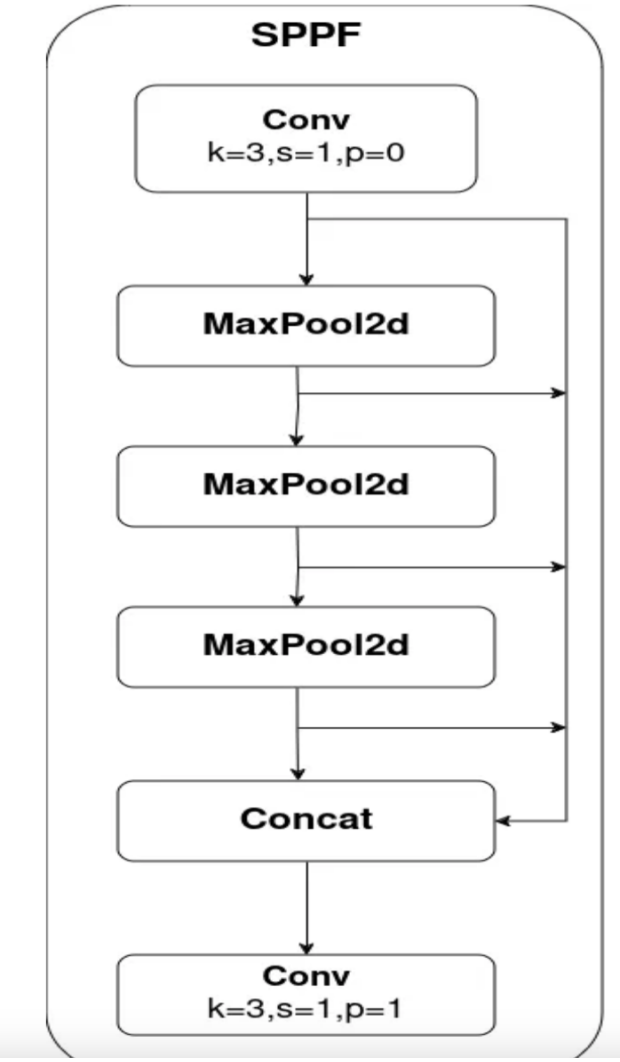
Khối C2f (Cross-Stage Partial with Bottlenecks) là một kiến trúc bao gồm:

- Một lớp Conv2d đầu vào tạo ra bản đồ đặc trưng.
- Bản đồ này được chia thành hai phần:
  - Một phần được đưa qua chuỗi các khối Bottleneck
    - Bottleneck Block là một khối gồm nhiều lớp Conv2D được sắp xếp theo một cách đặc biệt để:
    - Nén (giảm) số lượng kênh trung gian → giảm chi phí tính toán.
    - Sau đó mở rộng lại → giữ lại thông tin quan trọng.

- Có thể kết hợp với kết nối tắt (shortcut/residual connection).
  - Phần còn lại đi thẳng đến khối Concat.
- Số lượng khối Bottleneck được điều khiển bởi tham số depth_multiple của mô hình.
- Cuối cùng, đầu ra từ các khối Bottleneck và phần đặc trưng còn lại được nối lại (concatenated) rồi đưa qua một lớp Conv để tạo đầu ra khối C2f.

Cấu trúc này giúp giảm chi phí tính toán, tận dụng kết nối tắt, đồng thời duy trì khả năng học các đặc trưng sâu.

d) Khối SPPF



Hình 17. Khối SPPF

Khối SPPF (Spatial Pyramid Pooling - Fast) gồm:

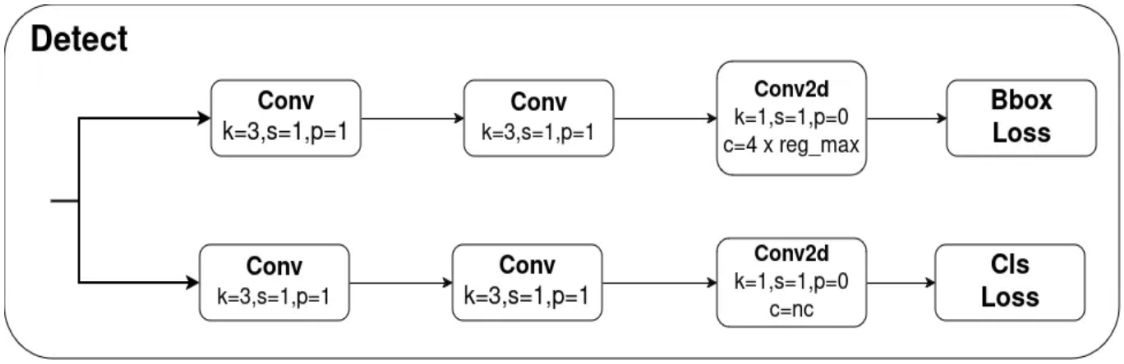
- Một lớp Conv2d đầu vào,

- Theo sau là ba lớp MaxPool2d liên tiếp (thường với kernel cố định, ví dụ 5x5),
- Các đầu ra từ mỗi lớp MaxPool2d được nối (concatenate) với đầu ra ban đầu,
- Sau đó được đưa qua một lớp Conv cuối để sinh ra đặc trưng đầu ra.

Ý tưởng chính của SPP là trích xuất đặc trưng đa tỷ lệ bằng cách gộp thông tin từ nhiều vùng không gian, giúp mạng xử lý hiệu quả hình ảnh với kích thước và tỷ lệ đối tượng khác nhau. Tuy nhiên, SPPF đơn giản hóa SPP bằng cách chỉ dùng một kích thước kernel cố định, giúp giảm chi phí tính toán mà vẫn giữ được hiệu quả nhận dạng.

MaxPool2d là lớp gộp tối đa 2D, dùng để giảm kích thước không gian và giữ lại đặc trưng nổi bật nhất trong mỗi vùng. Việc giảm mẫu này giúp giảm độ phức tạp và tăng tính khái quát của mô hình.

a) Khối phát hiện



Hình 18. Khối phát hiện

Khối Detect (Phát hiện) chịu trách nhiệm dự đoán vị trí và phân loại đối tượng. Không giống các phiên bản trước, YOLOv8 sử dụng kiến trúc không neo (anchor-free), trong đó mô hình dự đoán trực tiếp tâm đối tượng và kích thước hộp giới hạn thay vì lệch so với các anchor box cố định. Cách tiếp cận này giúp giảm số lượng dự đoán cần thiết, tăng tốc độ xử lý hậu kỳ và cải thiện độ chính xác.

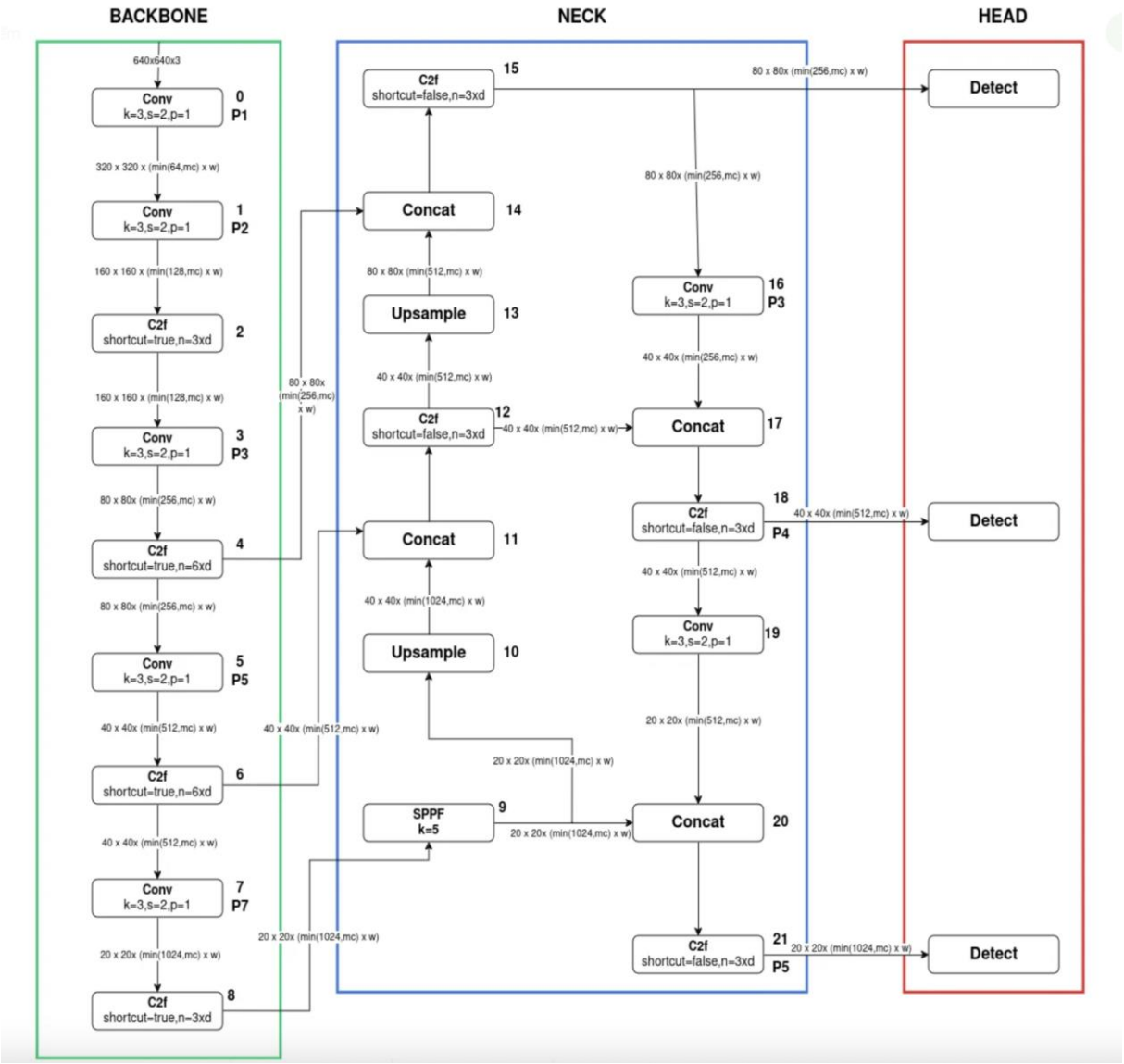
Khối Detect gồm hai nhánh (track):

- Nhánh 1: Dự đoán hộp giới hạn (Bounding Box),
- Nhánh 2: Dự đoán phân lớp (Class Prediction).

Mỗi nhánh gồm hai lớp tích chập (Conv) và kết thúc bằng một lớp Conv2d duy nhất để tạo ra đầu ra tương ứng:

- Bounding Box Loss ở nhánh 1,
- Class Loss ở nhánh 2.

Sơ đồ tổng thể



Hình 19. Kiến trúc tổng thể của Yolov8n

a) Phần Backbone

Khối 0 – Lớp tích chập đầu tiên

Hình ảnh đầu vào có kích thước 640 x 640 x 3 được đưa vào một lớp tích chập với kích thước kernel là 3, bước tiến là 2 và đệm là 1. Việc sử dụng bước tiến bằng 2 giúp giảm một nửa độ phân giải không gian của ảnh, tức là kích thước đầu ra của bản đồ đặc trưng là 320 x 320.

Số lượng kênh đầu ra được tính theo công thức:

min(64, max_channels) × width_multiple

Trong đó:

- 64 là số kênh cơ sở.
- max_channels là số kênh cực đại tùy thuộc vào cấu hình mô hình.
- width_multiple là hệ số tỉ lệ chiều rộng của mô hình (ví dụ: 0.25 với YOLOv8-n).

Ví dụ: với YOLOv8-n (max_channels = 1024, width_multiple = 0.25), số kênh đầu ra là 16.

Khối 2 – Khối C2f (Cross-Stage Partial with Feature Fusion)

Khối C2f bao gồm một lớp tích chập theo sau là các khối Bottleneck. Nó có hai tham số chính:

- shortcut: tham số boolean quy định có sử dụng kết nối tắt (residual) trong các khối bottleneck hay không.
- n: số lượng khối bottleneck được sử dụng bên trong, tính bằng công thức:

n = round(3 × depth_multiple)

Trong đó depth_multiple là hệ số tỉ lệ độ sâu của mô hình. Ví dụ, với YOLOv8-n (depth_multiple = 0.33), ta có n = round(3 × 0.33) ≈ 1.

Khối C2f có đặc điểm là không làm thay đổi độ phân giải không gian và số kênh đầu ra nếu không có lớp tích chập bổ sung nào.

Khối 9 – Khối SPFF (Spatial Pyramid Pooling - Fast)

Đây là khối cuối cùng của Backbone và đóng vai trò quan trọng trong việc tạo ra các đặc trưng giàu thông tin ở nhiều tỷ lệ khác nhau.

Cấu trúc của khối SPFF gồm:

- Một lớp tích chập đầu vào.
- Ba lớp MaxPool2d liên tiếp với cùng kích thước kernel nhưng không làm thay đổi kích thước ảnh.
- Các bản đồ đặc trưng đầu ra từ các lớp MaxPool2d được nối lại với nhau (concatenate).
- Sau đó, kết quả được đưa qua một lớp tích chập cuối cùng để hợp nhất thông tin.

Ý tưởng chính của khối SPFF là thu thập thông tin không gian ở nhiều cấp độ mà không làm mất đi đặc trưng quan trọng. Điều này giúp mô hình xử lý hiệu quả các đối tượng

có kích thước khác nhau trong ảnh. So với phương pháp SPP truyền thống, SPFF đơn giản hơn và ít tốn chi phí tính toán hơn, do chỉ sử dụng một loại kernel pooling cố định thay vì nhiều mức pooling khác nhau.

b) Phần Neck

Neck trong YOLOv8 sử dụng kiến trúc PAN (Path Aggregation Network) nhằm mục đích kết hợp và khuếch đại đặc trưng đa cấp từ các tầng khác nhau của Backbone. Đây là bước trung gian giúp kết nối thông tin từ phần trích xuất đặc trưng (Backbone) đến phần dự đoán (Head), từ đó cải thiện chất lượng phát hiện đối tượng.

- Chức năng chính của Neck là lấy mẫu lại bản đồ đặc trưng và kết hợp đặc trưng từ nhiều tầng của Backbone, giúp mô hình học được thông tin từ cả các vật thể lớn và nhỏ trong ảnh.
- Lớp lấy mẫu (Upsample) trong Neck có vai trò tăng gấp đôi kích thước bản đồ đặc trưng mà không làm thay đổi số lượng kênh. Điều này giúp đồng bộ độ phân giải với các tầng nông hơn để dễ dàng nối kết.
- Khối Concat thực hiện việc kết hợp các đặc trưng từ nhiều tầng lại với nhau. Khối này đơn giản chỉ nối các bản đồ đặc trưng theo chiều kênh, giữ nguyên độ phân giải không gian.

Neck giúp mô hình tận dụng thông tin ngữ nghĩa từ các tầng sâu và thông tin chi tiết từ các tầng nông, qua đó tăng cường hiệu suất phát hiện vật thể ở nhiều tỷ lệ khác nhau.

c) Phần Head

Head là phần cuối cùng trong kiến trúc YOLOv8, có nhiệm vụ dự đoán các hộp giới hạn (bounding boxes), phân lớp đối tượng, và tính toán độ tin cậy (confidence score) cho từng dự đoán.

Không giống như các phiên bản YOLO trước đó sử dụng các anchor box, YOLOv8 là một mô hình không sử dụng anchor (anchor-free). Điều này có nghĩa là mô hình sẽ dự đoán trực tiếp tâm của đối tượng, giúp giảm độ phức tạp hậu xử lý, tăng tốc suy luận và giảm số lượng hộp dự đoán dư thừa.

Phần Head của YOLOv8 bao gồm ba khối phát hiện chính, mỗi khối chuyên xử lý các đối tượng ở một tỷ lệ cụ thể:

- Khối phát hiện đầu tiên chuyên phát hiện các đối tượng nhỏ, lấy đầu vào từ khối C2f trong Khối 15.

- Khối phát hiện thứ hai chuyên phát hiện các đối tượng trung bình, lấy đầu vào từ khối C2f trong Khối 18.
- Khối phát hiện thứ ba chuyên phát hiện các đối tượng lớn, lấy đầu vào từ khối C2f trong Khối 21.

Mỗi khối phát hiện bao gồm hai nhánh riêng biệt:

- Một nhánh dùng để dự đoán tọa độ hộp giới hạn
- Một nhánh dùng để dự đoán lớp đối tượng

3. Cách hoạt động

1. Ảnh đầu vào từ camera được resize và chuẩn hóa.
2. Backbone trích xuất đặc trưng không gian từ ảnh.
3. Neck kết hợp thông tin từ nhiều tầng đặc trưng (feature pyramid).
4. Head đầu ra bounding box, class (người), và confidence.
5. Hộp được lọc bằng ngưỡng confidence và Non-Max Suppression (NMS).

4. Ưu điểm

- Nhẹ và nhanh, dễ triển khai trên Raspberry Pi.
- Khả năng phát hiện người tốt trong nhiều điều kiện ánh sáng và góc nhìn.
- Dễ dàng huấn luyện lại với dữ liệu riêng nếu cần tăng độ chính xác.

2.2.3. Giải pháp nhận diện giọng nói

Trong hệ thống, để nhận diện và xử lý giọng nói, nhóm sử dụng mô hình nhận dạng giọng nói Vosk – là một thư viện mã nguồn mở hỗ trợ nhiều ngôn ngữ, hoạt động tốt trên các thiết bị hiệu năng thấp như Raspberry Pi.

1. Mục đích

Xây dựng hệ thống điều khiển thiết bị thông minh thông qua giọng nói – cho phép người dùng ra lệnh một cách thuận tiện, không chạm, tăng tính hiện đại cho hệ thống.

2. Giới thiệu mô hình

Hệ thống sử dụng Vosk – một mô hình nhận dạng giọng nói mã nguồn mở, đã được huấn luyện sẵn. Mô hình có khả năng nhận dạng giọng nói và chuyển đổi thành văn bản (Speech-to-Text) hoàn toàn offline, tương thích với nhiều thiết bị như PC, Raspberry Pi.

- Mô hình sử dụng: vosk-model-small-vi-0.3 (cho tiếng Việt).
- Không cần huấn luyện lại, chỉ cần tải về và sử dụng.

3. Cách hoạt động

1. Người dùng nói vào micro được kết nối với Raspberry Pi.

2. Âm thanh được thu và xử lý bởi thư viện sounddevice.
3. Mô hình Vosk chuyển âm thanh sang văn bản theo thời gian thực.
4. Backend kiểm tra nội dung văn bản:
    - Nếu là "mở cửa" → gửi tín hiệu đến Arduino để mở cửa.
    - Nếu là "đóng cửa" → gửi tín hiệu đóng cửa.
5. Phản hồi lại người dùng (có thể thêm loa, đèn LED...).

Toàn bộ quá trình chạy hoàn toàn trên thiết bị cục bộ, không cần internet.

4. Ưu điểm

- Không cần huấn luyện: Mô hình có sẵn, dễ dùng.
- Chạy offline: Không phụ thuộc mạng.
- Nhẹ, hiệu quả: Phù hợp với thiết bị như Raspberry Pi.
- Tốc độ phản hồi nhanh, dễ tích hợp Python, FastAPI, …

5. Nhược điểm

- Độ chính xác phụ thuộc môi trường: Nhiễu, tiếng ồn có thể gây lỗi nhận dạng.
- Chỉ nhận dạng được ngôn ngữ đã hỗ trợ (Tiếng Việt, Anh...).
- Không nhận dạng được giọng nói phức tạp hoặc đối thoại dài.
- Không tự học thêm nếu không tích hợp mô hình nâng cao.
- Nhận diện không tốt với chất giọng địa phương.

2.3. Giải pháp phần mềm

2.3.1. Phát triển bài toán

1. Bài toán đặt ra

Hệ thống cần phát hiện và kiểm soát ra vào nhà xe thông minh dựa trên ba yếu tố:

- Nhận diện khuôn mặt để xác thực người dùng được phép vào.
- Phát hiện người trong khu vực để xe để giám sát và cảnh báo xâm nhập trái phép.
- Nhận diện giọng nói để mở/đóng cửa hoặc thực hiện một số lệnh cơ bản.

Mỗi bài toán đều yêu cầu tích hợp mô hình trí tuệ nhân tạo (AI) với phần mềm backend, frontend, và điều khiển thiết bị thông qua vi điều khiển.

2. Mục tiêu phát triển

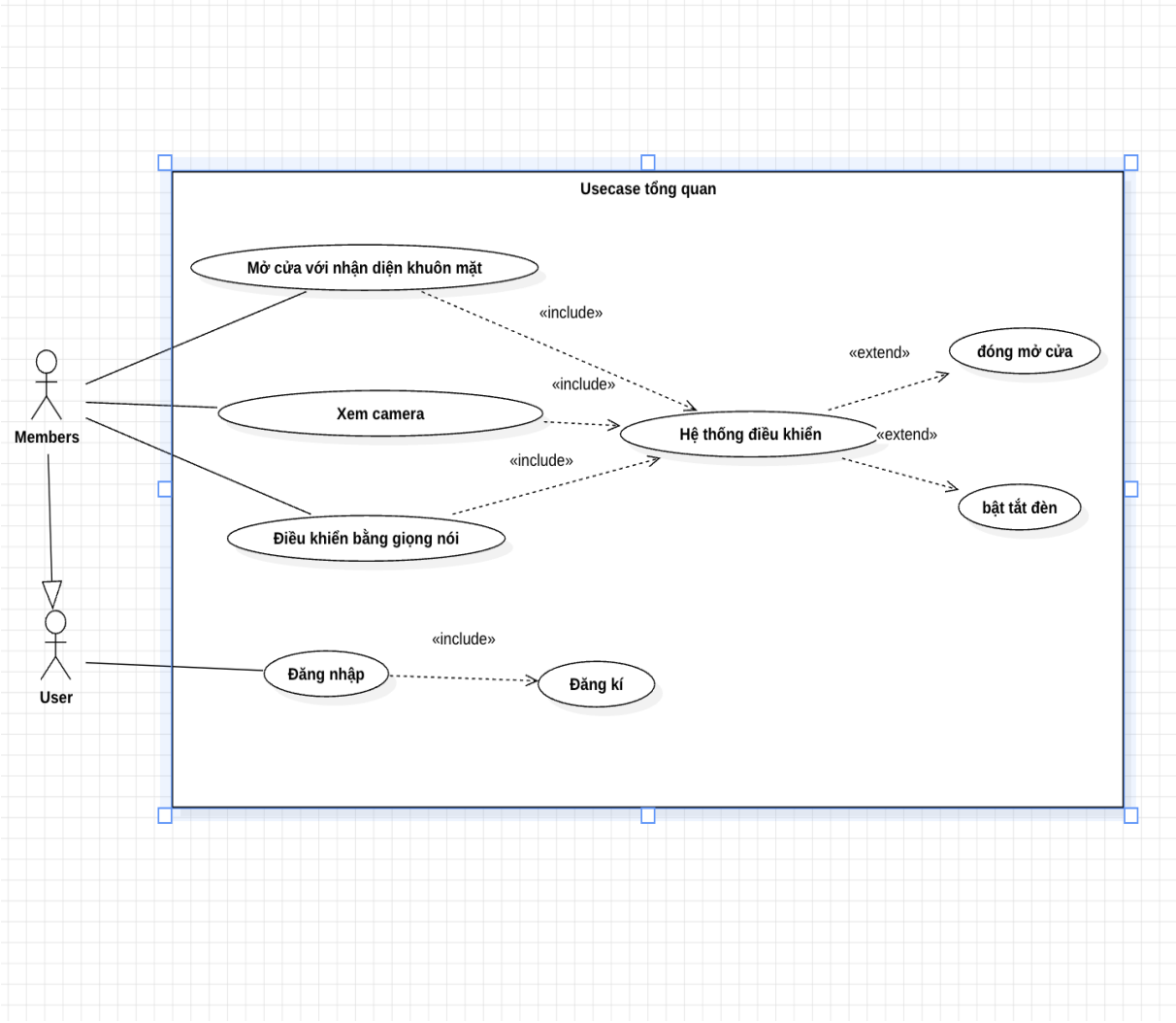
- Xây dựng phần mềm đảm nhiệm vai trò:

- Giao tiếp với người dùng thông qua giao diện web (frontend).
- Xử lý yêu cầu và điều khiển AI, thiết bị (backend).
- Tích hợp các mô hình nhận diện AI.
- Điều khiển phần cứng thông qua Raspberry Pi và Arduino.

2.3.2. Công nghệ sử dụng.

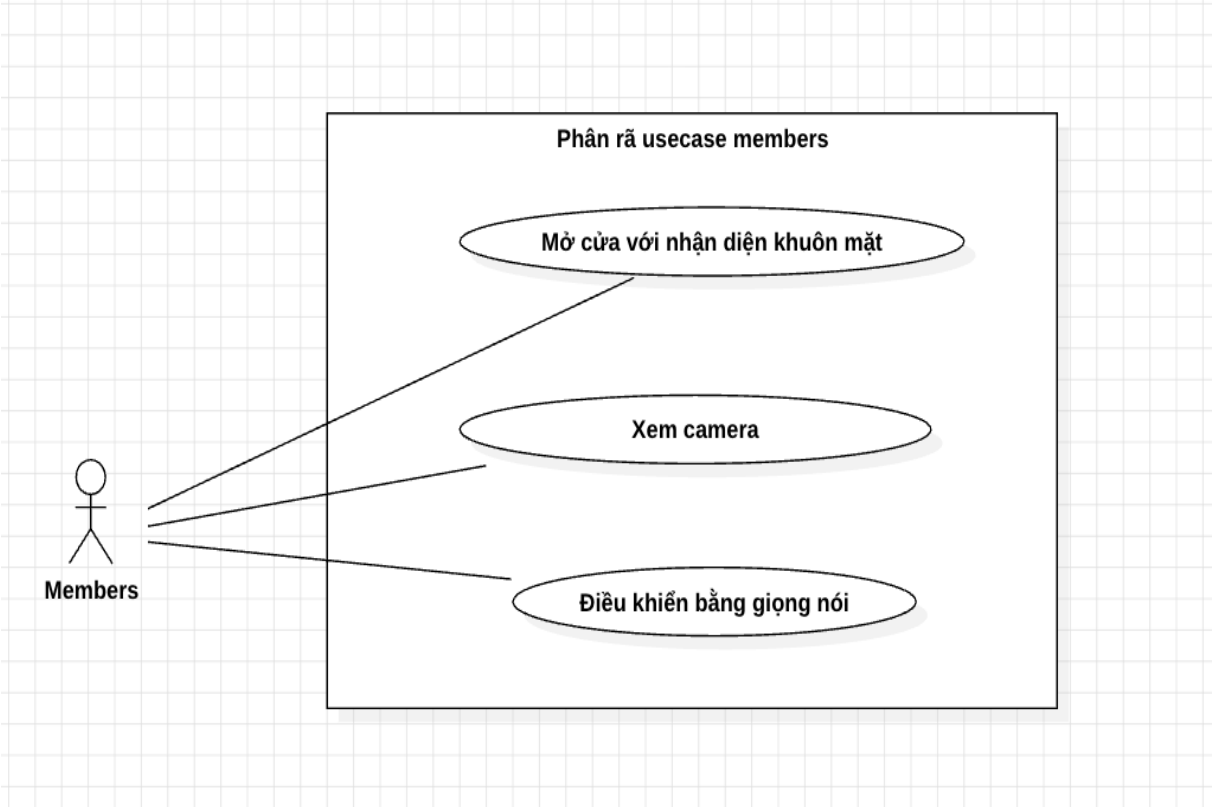
- **Ngôn ngữ chính**: Python.
- **Framework backend**: FastAPI.
- **Frontend**: HTML/CSS/JS/REACT/TYPESCRIPS hoặc dùng template có sẵn.
- **AI Model**: Face_recognition, YOLOv8n, Vosk.
- **Giao tiếp phần cứng**: HTTP request và serial với Arduino.

2.3.3. Biểu đồ usecase hệ thống.



Hình 20. Sơ đồ usecase tổng quan

- Phân rã usecase members



Hình 21.Sơ đồ phân rã usecase Members

Use case: Xem camera			
Mô tả:	Người sử dụng theo dõi tình hình thông qua camera		
Người dùng:	Chủ nhà		
Sự kiện kích hoạt	Click vào item để vào hệ thống		
Điều kiện tiên quyết	Camera phải được bật và kết nối với hệ thống		
Luồng sự kiện chính	**STT**	**Thực hiện bởi**	**Hành động**
	1	Người sử dụng	Chọn chức năng theo dõi camera
	2	Hệ thống	Hiển thị ra dữ liệu mà camera ghi lại được truyền cho chủ nhà
Luồng sự kiện phụ:	5.1	Hệ thống	Thông báo lỗi: Đường truyền không ổn định
	6.1	Hệ thống	Thông báo lỗi: Mất kết nối
Điều kiện sau:	Hiển thị cho người dùng xem		

Bảng 5. Phân rã usecase xem camera

Use case: Mở cửa với nhận diện khuôn mặt

Mô tả:	Người sử dụng quét mặt bằng camera để mở cửa		
Người dùng:	Chủ nhà		
Sự kiện kích hoạt	Click vào item để vào hệ thống		
Điều kiện tiên quyết	Camera thiết bị hoạt động tốt		
Luồng sự kiện chính	**STT**	**Thực hiện bởi**	**Hành động**
	1	Người sử dụng	Chọn chức năng mở cửa
	2	Hệ thống	Hiển thị ra dữ liệu mà camera ghi lại được truyền cho ardunio để mở cửa
Luồng sự kiện phụ:	5.1	Hệ thống	Thông báo lỗi: Đường truyền không ổn định
	6.1	Hệ thống	Thông báo lỗi: Mất kết nối
Điều kiện sau:	Mở cửa		

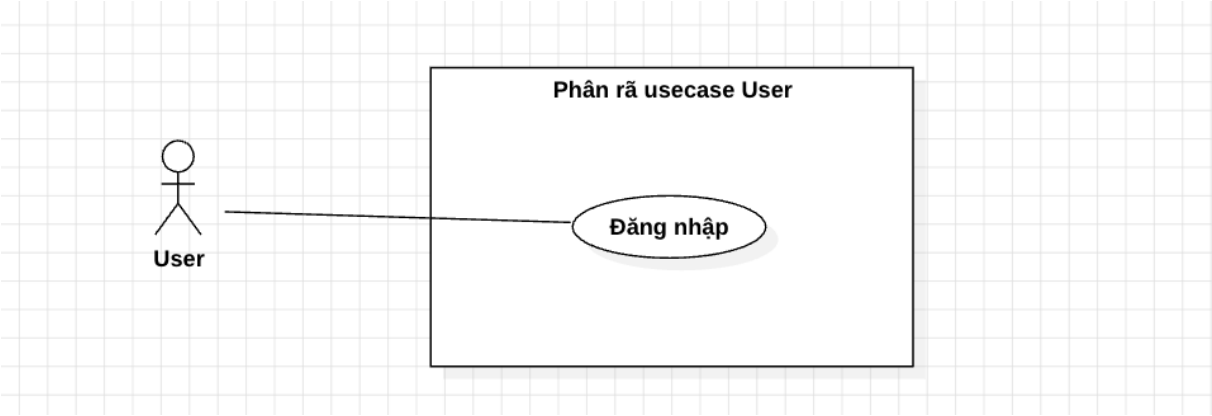
Bảng 6. Phân rã usecase mở cửa với nhận diện khuôn mặt

Use case: Điều khiển bằng giọng nói			
Mô tả:	Người sử dụng nói để thực hiện các thao tác mở, đóng cửa …		
Người dùng:	Chủ nhà		
Sự kiện kích hoạt	Click vào item để vào hệ thống		
Điều kiện tiên quyết	Thiết bị sử dụng thu âm được		
Luồng sự kiện chính	**STT**	**Thực hiện bởi**	**Hành động**
	1	Người sử dụng	Chọn chức năng điều khiển bằng giọng nói
	2	Hệ thống	Hiển thị ra dữ liệu thu âm ghi lại được truyền cho ardunio
Luồng sự kiện phụ:	5.1	Hệ thống	Thông báo lỗi: Đường truyền không ổn định
	6.1	Hệ thống	Thông báo lỗi: Mất kết nối

Điều kiện sau:	Các thiết bị hoạt động

Bảng 7. Phân rã usecase điều khiển bằng giọng nói

- Phân rã usecase user



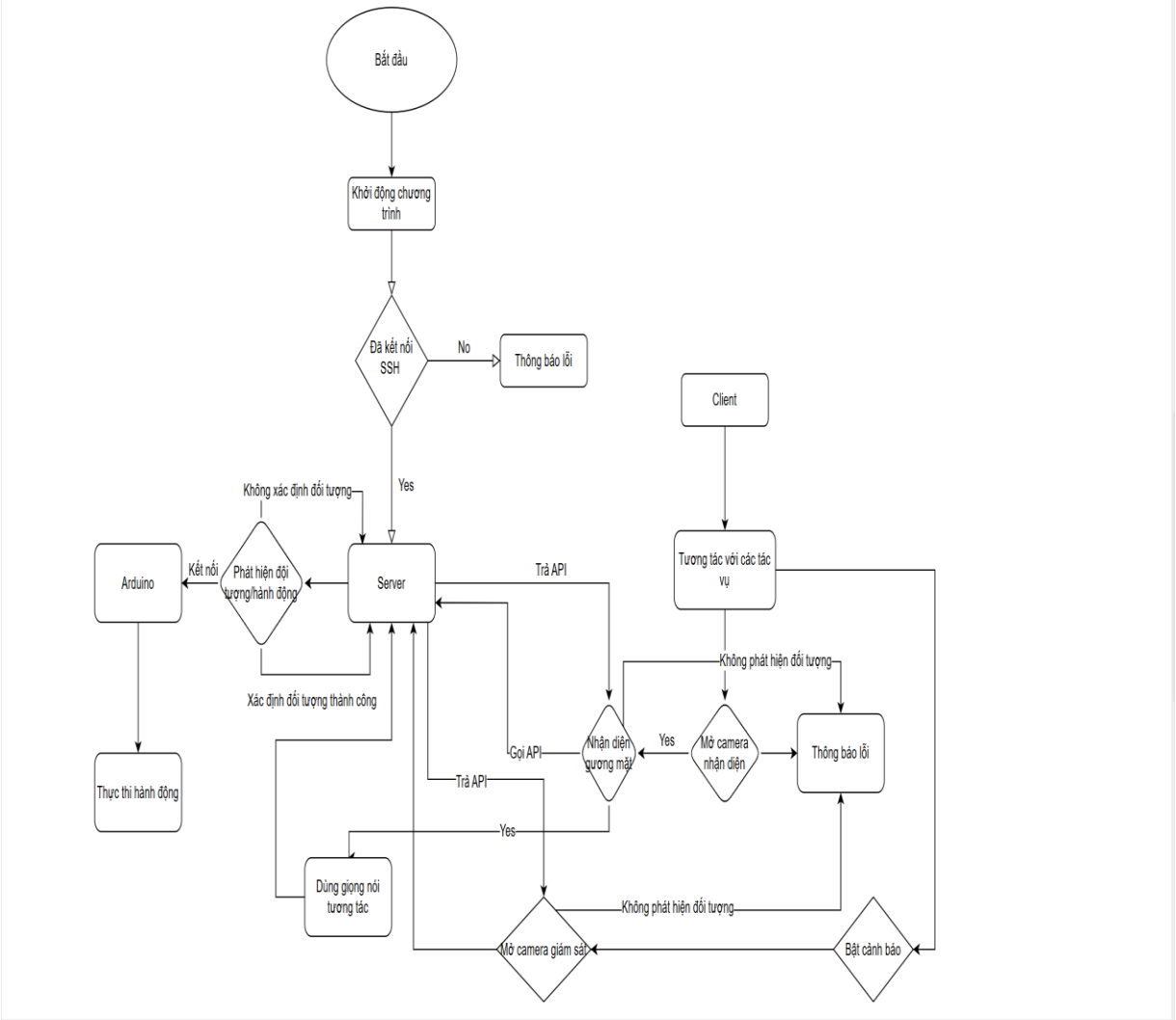
Hình 22. Sơ đồ phân rã usecase User

Use case: Đăng nhập			
Mô tả:	Người sử dụng đăng nhập		
Người dùng:	Người sử dụng		
Sự kiện kích hoạt	Người dùng click vào nút hoặc liên kết "Đăng nhập" trên giao diện hệ thống.		
Điều kiện tiên quyết	Người dùng đã có tài khoản hợp lệ được cấp trong hệ thống.		
Luồng sự kiện chính	**STT**	**Thực hiện bởi**	**Hành động**
	1	Người sử dụng	Nhập tên đăng nhập và mật khẩu vào form đăng nhập.
	2	Hệ thống	Kiểm tra thông tin đăng nhập có hợp lệ không.
Luồng sự kiện phụ:	5.1	Hệ thống	Nếu hợp lệ, chuyển hướng người dùng đến giao diện chính của hệ thống.
	6.1	Hệ thống	Báo lỗi
Điều kiện sau:	- Người dùng truy cập thành công vào hệ thống (nếu đăng nhập đúng).		

	- Hoặc nhận cảnh báo / yêu cầu xác minh (nếu sai nhiều lần hoặc đăng nhập bất thường).

Bảng 8. Phân rã usecase đăng nhập

2.3.4. Sơ đồ khối hệ thống



Hình 23. Sơ đồ khối hệ thống

3. Kết quả

3.1. Nhận diện khuôn mặt

3.1.1. Tập dữ liệu

Tập dữ liệu huấn luyện gốc (của mô hình gốc trong face_recognition)

- Tên mô hình gốc: dlib_face_recognition_resnet_model_v1
- Kiến trúc: ResNet-34
- Đầu ra: Mỗi khuôn mặt được ánh xạ thành một vector đặc trưng 128 chiều.
- Tập dữ liệu huấn luyện gốc được huấn luyện bởi dlib trên tập dữ liệu kết hợp từ:
  - VGGFace2: Tập dữ liệu lớn chứa hàng triệu khuôn mặt trong điều kiện ánh sáng, biểu cảm, góc nhìn khác nhau.
  - LFW (Labeled Faces in the Wild): Gồm ~13,000 ảnh khuôn mặt từ internet, gắn nhãn theo danh tính thật.
  - CASIA-WebFace (có thể đã được dùng): Một bộ dữ liệu khuôn mặt phổ biến khác trong nghiên cứu.
- Các ảnh đều đã được canh chỉnh (alignment) trước khi huấn luyện.

Hệ thống sử dụng dữ liệu huấn luyện đã được tích hợp sẵn trong mô hình face_recognition, do đó không cần huấn luyện lại mô hình nhận diện khuôn mặt. Tuy nhiên, để nhận diện được một người cụ thể, cần cung cấp ảnh mẫu của người đó cho hệ thống để trích xuất đặc trưng (vector embedding).

Cấu trúc thư mục dữ liệu người dùng, ví dụ:

```
datasets/
├── persona/
│   ├── 1.jpg
│   ├── 2.jpg
│   ├── ...
├── personb/
│   ├── 1.jpg
│   ├── 2.jpg
│   ├── ...
```

Trong đó:

- Mỗi thư mục con như persona, personb, ... tương ứng với một người cụ thể.
- Mỗi thư mục chứa từ 1 đến 10 ảnh chân dung của người đó, được chụp trong điều kiện ánh sáng hợp lý, gương mặt rõ nét và nhìn chính diện.

3.1.2. Huấn luyện mô hình

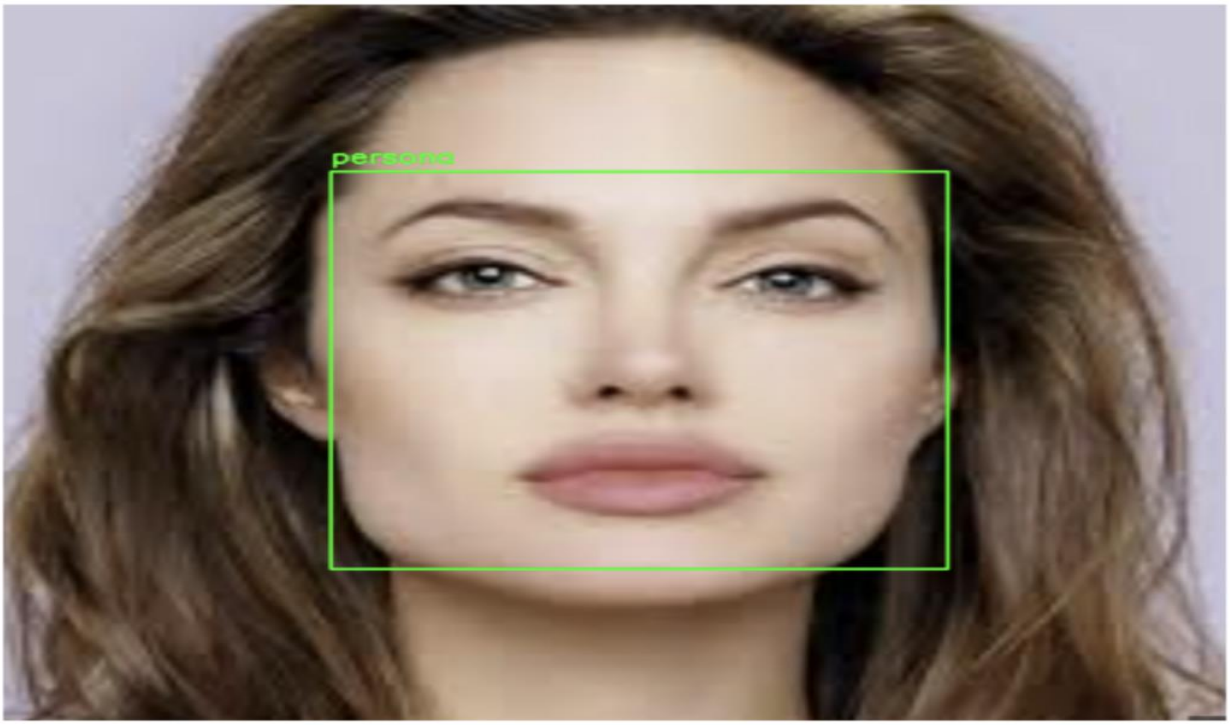
Quy trình sử dụng tập dữ liệu:

1. **Tạo thư mục dữ liệu:**
   - Mỗi người cần nhận diện sẽ có 1 thư mục riêng chứa các ảnh mẫu.
   - Có thể đặt tên thư mục theo mã định danh, tên người, …
2. **Tiền xử lý và mã hóa:**
   - Mỗi ảnh sẽ được đưa qua các bước:
     - Phát hiện khuôn mặt (HOG hoặc CNN).
     - Căn chỉnh và trích xuất đặc trưng bằng mô hình ResNet-34 → tạo vector đặc trưng 128 chiều.
   - Các vector đặc trưng này sẽ được lưu vào bộ nhớ hoặc file .pkl để sử dụng cho nhận diện sau này.
3. **Nhận diện thời gian thực:**
   - Khi có ảnh đầu vào từ camera, hệ thống cũng trích xuất vector đặc trưng.
   - So sánh với các vector đã lưu bằng khoảng cách Euclidean.
   - Nếu khoảng cách nhỏ hơn ngưỡng (thường là 0.6), xác định là người đã đăng ký.
4. **Lưu ý về "huấn luyện" trong thư viện face_recognition:**
   - Không giống các mô hình truyền thống cần train từ đầu.
   - Việc thêm người mới bằng cách thêm ảnh + mã hóa và lưu vector đặc trưng → "huấn luyện" rất nhanh và dễ mở rộng.
   - Không cần GPU, không cần tái huấn luyện toàn bộ.

3.1.3. Kết quả nhận diện

Sau khi tích hợp thư viện face_recognition vào hệ thống và thử nghiệm với các hình ảnh khuôn mặt của người dùng đã đăng ký, mô hình cho kết quả nhận diện như sau:

Hình 24. Ảnh nhận diện người không thành công



Hình 25. Ảnh nhận diện người thành công

Đặc điểm nổi bật:

- Hệ thống có thể nhận diện khuôn mặt chính xác > 90% nếu người dùng cung cấp từ 3–5 ảnh mẫu rõ nét.
- Thời gian xử lý trung bình từ ảnh đầu vào đến khi có kết quả thấp.

- Hoạt động ổn định trong thời gian thực khi tích hợp với camera (sử dụng thư viện OpenCV).
- Nếu không khớp với người dùng nào trong cơ sở dữ liệu, hệ thống hiển thị nhãn "Unknown" và không mở cửa.

3.2. Nhận diện người

3.2.1. Tập dữ liệu

Dữ liệu huấn luyện ban đầu:

- Mô hình YOLOv8n được huấn luyện sẵn trên tập COCO dataset (2017) – gồm hơn 118.000 ảnh và 80 lớp đối tượng, trong đó có lớp "person". Điều này giúp mô hình nhận diện người rất tốt trong các tình huống phổ biến.

Fine-tune với dữ liệu thực tế

- Để tăng độ chính xác và phù hợp hơn với bối cảnh giám sát thực tế tại gara (ánh sáng yếu, góc nhìn cố định từ camera), nhóm đã thực hiện huấn luyện tinh chỉnh (fine-tune) lại mô hình trên tập dữ liệu thu thập riêng:
  - Tổng số ảnh: 1.529 ảnh chứa người trong điều kiện ánh sáng yếu (ban đêm hoặc vùng tối).
  - Tỉ lệ chia dữ liệu:
    - 70% (1.070 ảnh) để huấn luyện (training)
    - 20% (306 ảnh) để xác thực (validation)
    - 10% (153 ảnh) để kiểm tra (testing)
  - Định dạng nhãn: Nhãn được gán theo định dạng YOLO (class x_center y_center width height).
  - Công cụ gán nhãn: Roboflow.
  - Lớp duy nhất: person (người), phù hợp với yêu cầu phát hiện người trong khu vực để xe.

Lý do fine-tune

Mặc dù YOLOv8n đã nhận diện người khá tốt, nhưng trong điều kiện ánh sáng yếu hoặc camera có độ phân giải thấp, độ chính xác có thể giảm. Vì vậy:

- Fine-tune giúp mô hình học thêm các đặc trưng riêng của môi trường mục tiêu.
- Tăng độ nhạy với các vùng có ánh sáng yếu, giảm false negative.
- Cải thiện hiệu quả khi triển khai thực tế trên camera giám sát.

Kết quả sau fine-tune

- Mô hình sau fine-tune nhận diện người tốt hơn trong điều kiện ánh sáng thấp.
- Độ chính xác tăng rõ rệt khi thử nghiệm với video thực tế từ camera tại gara.

3.2.2. Huấn luyện mô hình

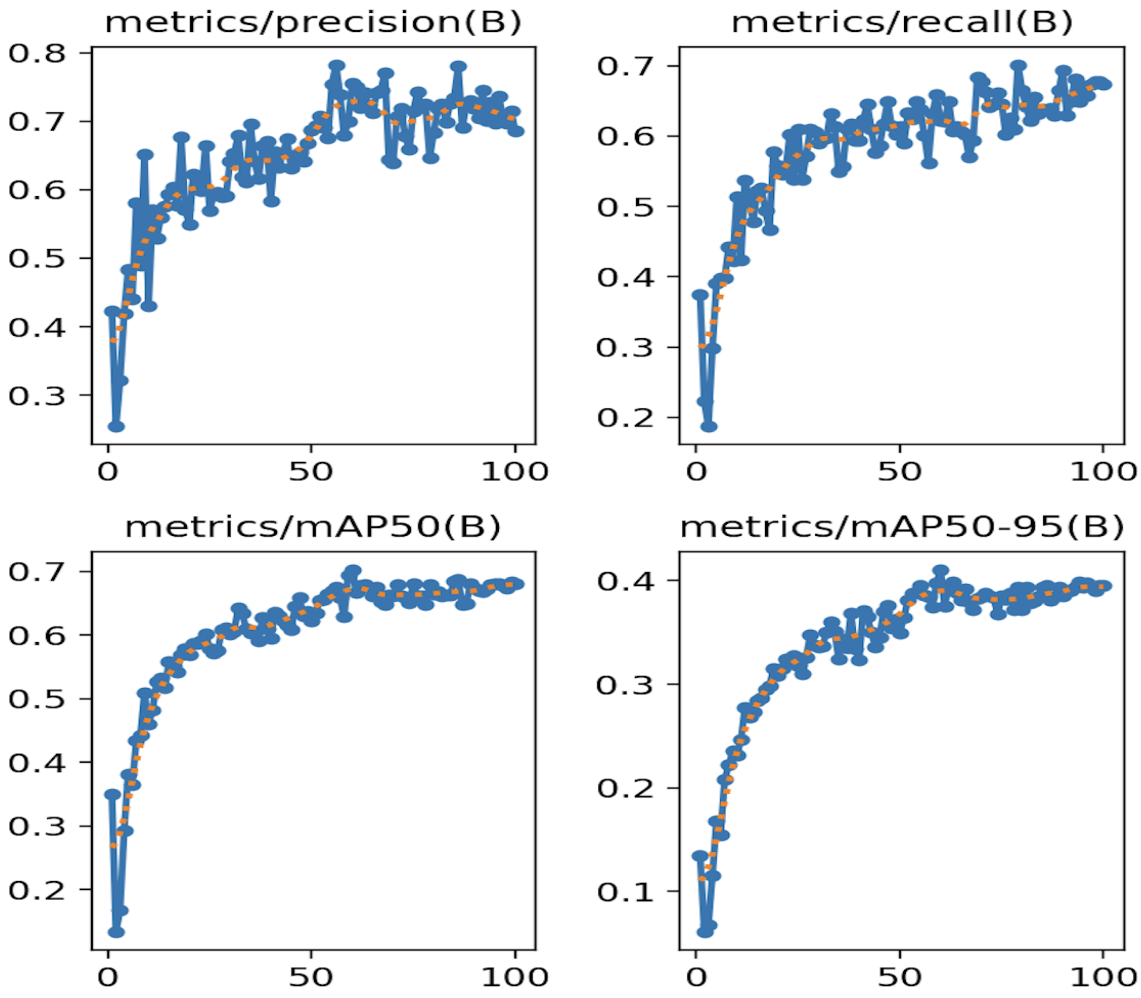
Mô hình YOLOv8n được sử dụng dưới dạng mô hình đã huấn luyện sẵn (pre-trained) trên COCO dataset để tận dụng khả năng nhận diện người cơ bản. Để tối ưu hóa hiệu suất cho môi trường thực tế tại gara với điều kiện ánh sáng yếu, nhóm tiến hành fine-tune mô hình trên bộ dữ liệu người thu thập riêng. Quá trình huấn luyện được thực hiện như sau:

- Chuẩn bị dữ liệu: Dữ liệu được chia thành ba tập: 70% ảnh huấn luyện, 20% ảnh xác thực, 10% ảnh kiểm tra. Nhãn đối tượng được định dạng theo chuẩn YOLO.
- Cấu hình huấn luyện:
    - Sử dụng kiến trúc YOLOv8n với trọng số pre-trained.
    - Thiết lập learning rate thích hợp để mô hình học hiệu quả trên dữ liệu mới mà không làm mất đi các đặc trưng đã học.
    - Số epoch phù hợp (ví dụ 50-100 epochs) để đạt độ chính xác tối ưu.
    - Batch size và các tham số kỹ thuật khác tùy theo khả năng phần cứng.
- Kỹ thuật tăng cường dữ liệu (Data augmentation): Áp dụng các kỹ thuật như xoay ảnh, thay đổi độ sáng, zoom, cắt xén để làm phong phú dữ liệu và tăng khả năng tổng quát của mô hình.
- Theo dõi và đánh giá:
    - Sử dụng tập xác thực để điều chỉnh hyperparameters và tránh hiện tượng overfitting.
    - Đánh giá kết quả trên tập kiểm tra với các chỉ số như Precision, Recall, mAP để đảm bảo mô hình hoạt động tốt.

3.2.3. Kết quả nhận diện

Sau khi fine-tune mô hình YOLOv8n với tập dữ liệu gồm 1529 ảnh chụp người trong điều kiện ánh sáng yếu (tương tự môi trường giám sát thực tế), mô hình đã cho kết quả khả quan như sau:

- **Độ chính xác (Precision):** ~0.8
- **Độ nhạy (Recall):** ~0.7
- **mAP@0.5 (Mean Average Precision ở IoU = 0.5):** ~0.7
- **mAP@0.5:0.95:** ~0.42



Hình 26. Kết quả mô hình YOLOv8n

Một số quan sát thực tế:

- Mô hình nhận diện người tốt trong điều kiện ánh sáng yếu, góc quay từ trên cao hoặc xa, phù hợp với ứng dụng trong giám sát an ninh.
- Trong một số trường hợp đặc biệt như người quay lưng hoặc bị che khuất một phần, mô hình vẫn cho ra kết quả nhận diện đúng với confidence trên 0.6.
- Tốc độ xử lý đạt khoảng 15 FPS trên Raspberry Pi 4, đủ để xử lý theo thời gian gần thực tế.

Hình 27. Kết quả nhận diện từ YOLOv8n

Như vậy, việc fine-tune lại mô hình YOLOv8n giúp tăng hiệu quả nhận diện người trong môi trường cụ thể của hệ thống, đồng thời tối ưu cho phần cứng có hạn như Raspberry Pi.

3.3. Nhận diện giọng nói

3.3.1. Tập dữ liệu

Trong bài toán nhận diện giọng nói, nhóm sử dụng mô hình Vosk đã được huấn luyện sẵn, do đó không cần tự xây dựng hay huấn luyện lại tập dữ liệu.

Cụ thể:

- Vosk là một thư viện mã nguồn mở hỗ trợ nhận diện giọng nói (speech-to-text) hoạt động ngoại tuyến (offline).
- Mô hình nhóm sử dụng là vosk-model-vn-0.4– đây là mô hình tiếng Việt kích thước nhỏ, phù hợp để chạy trên các thiết bị như Raspberry Pi.
- Mô hình này đã được huấn luyện sẵn trên các tập dữ liệu tiếng Anh như:
    - LibriSpeech – một trong những bộ dữ liệu lớn và phổ biến trong lĩnh vực nhận dạng giọng nói, gồm các đoạn đọc sách từ người nói thật.

- Common Voice – bộ dữ liệu mã nguồn mở do Mozilla phát triển, gồm hàng ngàn giờ giọng nói từ nhiều người với các chất giọng khác nhau.

- Các tập dữ liệu huấn luyện này giúp mô hình có thể nhận diện tốt các lệnh cơ bản, phát âm chuẩn, trong điều kiện tương đối yên tĩnh.

3.3.2. Huấn luyện mô hình

- Không sử dụng tập dữ liệu tự thu.
- Mô hình đã được huấn luyện trước trên dữ liệu lớn như LibriSpeech và Common Voice.
- Phù hợp cho các lệnh đơn giản như: "bật đèn", "tắt đèn", "mở cửa", "đóng cửa" khi người dùng phát âm rõ ràng bằng tiếng Anh.

3.3.3. Kết quả nhận diện

Mô hình nhận diện giọng nói được sử dụng trong hệ thống là VOSK – một giải pháp nhận dạng tiếng nói mã nguồn mở, hỗ trợ tiếng Việt và hoạt động trực tiếp trên thiết bị mà không cần kết nối Internet. Tuy nhiên, trong quá trình thử nghiệm thực tế, mô hình gặp một số khó khăn dẫn đến sai số trong nhận diện.

Kết quả đạt được:

- Trong môi trường yên tĩnh, mô hình nhận diện khá tốt các câu lệnh cơ bản như: "mở cửa", "tắt đèn", "bật đèn", với độ chính xác khá.
- Khi có nhiễu nền nhẹ (quạt máy, tiếng người nói xa), độ chính xác giảm.
- Trong môi trường có tiếng ồn lớn hoặc giọng nói không rõ ràng, kết quả nhận diện bị sai lệch hoặc không phản hồi, độ chính xác chỉ còn khoảng.

Các yếu tố gây ảnh hưởng đến chất lượng nhận diện:

- Nhiễu âm thanh xung quanh.
- Chất lượng micro thấp, bắt âm không rõ, khiến mô hình không thể trích xuất đặc trưng chính xác.
- Giọng nói không chuẩn, ngắt quãng hoặc quá nhỏ.

4. Kết luận

4.1. Đánh giá

Đồ án xây dựng hệ thống gara xe thông minh với Raspberry Pi làm trung tâm điều khiển, kết hợp các mô hình AI nhận diện khuôn mặt, nhận diện người và nhận diện giọng nói đã đạt được các mục tiêu đề ra. Hệ thống nhận diện khuôn mặt hoạt động hiệu quả với độ chính xác cao trong điều kiện ánh sáng hợp lý nhờ sử dụng thư viện face_recognition và mô hình ResNet-34. Mô hình nhận diện người YOLOv8n được fine-tune phù hợp với môi trường ánh sáng yếu, đảm bảo phát hiện chính xác người trong khu vực giám sát. Mô hình vosk hỗ trợ nhận diện giọng nói để thực hiện các tác vụ cơ bản. Giao diện web thân thiện, dễ sử dụng giúp người dùng có thể quản lý và kiểm soát gara thuận tiện. Tổng thể, sản phẩm vận hành ổn định, đáp ứng tốt các yêu cầu về tính năng và hiệu năng trên phần cứng Raspberry Pi.

4.2. Hướng phát triển

4.2.1. Nhận diện gương mặt

- **Nâng cao độ chính xác và khả năng nhận diện trong điều kiện khó khăn**: Hiện tại, mô hình face_recognition sử dụng ResNet-34 hoạt động tốt trong điều kiện ánh sáng hợp lý và góc chụp thuận lợi. Tuy nhiên, để áp dụng trong môi trường thực tế phức tạp như gara xe với ánh sáng thay đổi, góc nhìn đa dạng, cần cải thiện bằng cách áp dụng các kỹ thuật tăng cường dữ liệu (data augmentation) chuyên sâu, đồng thời thử nghiệm các mô hình CNN hiện đại như ArcFace, FaceNet hoặc sử dụng các bộ tiền xử lý nâng cao để cải thiện chất lượng ảnh đầu vào.
- **Mở rộng tính năng nhận diện**: Bên cạnh nhận diện danh tính, có thể mở rộng để nhận diện cảm xúc, tuổi tác, giới tính, giúp hệ thống có thể cung cấp thông tin chi tiết hơn phục vụ các mục đích bảo mật hoặc tương tác thông minh.
- **Tích hợp và tối ưu trên thiết bị edge**: Nghiên cứu tối ưu mô hình cho Raspberry Pi hoặc các thiết bị nhúng khác nhằm giảm độ trễ, tiết kiệm năng lượng, tăng tốc độ xử lý để phù hợp hơn với ứng dụng thời gian thực.

4.2.2. Nhận diện người.

- **Cải thiện khả năng nhận diện trong môi trường đa người và phức tạp**: Mô hình YOLOv8n hiện đã được fine-tune cho điều kiện ánh sáng yếu, tuy nhiên khi

có nhiều người xuất hiện đồng thời hoặc trong điều kiện ánh sáng thay đổi phức tạp, độ chính xác và tốc độ phát hiện có thể bị giảm. Có thể sử dụng các phiên bản YOLO lớn hơn (YOLOv8m, YOLOv8l) kết hợp thuật toán theo dõi đối tượng như SORT hoặc Deep SORT để vừa nhận diện chính xác vừa theo dõi liên tục các cá thể trong khung hình.

- **Phát hiện hành vi bất thường và cảnh báo tự động**: Mở rộng hệ thống bằng cách áp dụng các mô hình phân tích hành vi, phát hiện xâm nhập trái phép, hoặc các trường hợp nguy hiểm (như đột nhập, chạy nhảy trong gara) để nâng cao tính an ninh và an toàn.
- **Nhận diện đa đối tượng liên quan gara**: Ngoài con người, có thể phát triển mô hình nhận diện các đối tượng khác như xe máy, ô tô, vật dụng nguy hiểm để tạo thành hệ thống quản lý và tự động hóa toàn diện cho gara.
- **Tích hợp thêm cảm biến và các thiết bị IoT**: Kết hợp nhận diện người với dữ liệu từ các cảm biến chuyển động, cảm biến nhiệt, hoặc thiết bị báo động để tạo thành hệ thống giám sát thông minh, phản ứng nhanh với các tình huống bất thường.

4.2.3. Nhận diện giọng nói

Mặc dù mô hình VOSK đã hỗ trợ hệ thống nhận diện giọng nói với độ chính xác tương đối ổn định trong điều kiện lý tưởng, nhưng để nâng cao khả năng sử dụng trong thực tế và cải thiện độ tin cậy, hệ thống cần được phát triển thêm theo các hướng sau:

1. Tinh chỉnh mô hình theo giọng nói cụ thể

- Thu thập dữ liệu giọng nói từ chính người dùng dự kiến sử dụng hệ thống (ví dụ: chủ nhà, nhân viên).
- Tinh chỉnh (fine-tune) mô hình nhận dạng với tập dữ liệu nhỏ này để tăng độ chính xác với giọng quen thuộc, giảm sai số.

2. Khử nhiễu âm thanh đầu vào (Noise Reduction)

- Áp dụng các kỹ thuật tiền xử lý âm thanh như:
  - Bộ lọc thông dải (band-pass filter)
  - Mạng học sâu khử nhiễu (Deep Noise Suppression)
- Giúp nâng cao chất lượng âm thanh trước khi đưa vào mô hình nhận dạng.

3. Kết hợp mô hình ngữ cảnh (Contextual Model)

- Áp dụng logic theo ngữ cảnh để giới hạn tập từ vựng có thể xảy ra, ví dụ:

- Nếu người dùng đang đứng gần cửa → ưu tiên nhận lệnh "mở cửa", "khóa cửa"
- Nếu trong phòng tối → ưu tiên nhận lệnh "bật đèn"

- Kết hợp với confidence score để bỏ qua kết quả mơ hồ.

4. Sử dụng mô hình hiện đại hơn

- Thay thế hoặc kết hợp với các mô hình tiên tiến hơn như:
  - **Whisper** của OpenAI (có độ chính xác cao, hỗ trợ nhiều ngôn ngữ, robust với nhiễu)
  - **Mozilla DeepSpeech** hoặc **Wav2Vec 2.0** (open source, có thể fine-tune dễ dàng)

4.2.4. Phần cứng

- **Nâng cấp thiết bị trung tâm xử lý**: Hiện hệ thống sử dụng Raspberry Pi (phiên bản hiện tại) có thể gặp giới hạn về tốc độ xử lý và khả năng đa nhiệm khi chạy đồng thời nhiều mô hình AI và xử lý video thời gian thực. Có thể nâng cấp lên các phiên bản Raspberry Pi mới hơn như Raspberry Pi 4 hoặc Raspberry Pi 5 với CPU mạnh hơn, RAM lớn hơn để tăng hiệu năng. Ngoài ra, có thể cân nhắc sử dụng các máy tính nhúng mạnh mẽ hơn như NVIDIA Jetson Nano, Jetson Xavier NX để tận dụng khả năng tăng tốc phần cứng cho mô hình AI.
- **Hệ thống lưu trữ và mạng**: Cải thiện bộ nhớ lưu trữ bằng ổ SSD thay vì thẻ nhớ SD để tăng tốc độ đọc ghi, nâng cao độ ổn định. Tối ưu kết nối mạng nội bộ hoặc kết nối Internet để đảm bảo truyền dữ liệu nhanh và ổn định, hỗ trợ cập nhật và giám sát từ xa.
- **Hệ thống cấp điện và dự phòng**: Đảm bảo nguồn cấp điện ổn định, sử dụng nguồn có chất lượng tốt hoặc bộ lưu điện (UPS) để tránh mất dữ liệu và hệ thống ngừng hoạt động khi bị mất điện đột ngột. Cân nhắc sử dụng nguồn dự phòng tự động chuyển đổi để hệ thống hoạt động liên tục.
- **Camera và thiết bị ngoại vi**: Nâng cấp camera giám sát có độ phân giải cao hơn, hỗ trợ hồng ngoại để hoạt động tốt trong điều kiện ánh sáng yếu. Kết hợp thêm các cảm biến chuyển động, cảm biến nhiệt để tăng cường khả năng phát hiện và phản hồi kịp thời.
- **Tích hợp các module mở rộng**: Sử dụng các module chuyên dụng như module AI Accelerator (Google Coral TPU, Intel Movidius) giúp tăng tốc độ xử lý AI

mà không cần thay đổi toàn bộ phần cứng. Điều này giúp tiết kiệm chi phí mà vẫn nâng cao hiệu suất.

4.3. Chi phí thực hiện

Mã Nhóm Linh Kiện	Tên Nhóm Linh Kiện	Số Tiền
1	Dây cắm mạch (Jumper wire)	140.000
2	Breadboard	23.000
3	Động cơ DC, L298N, Linh kiện phục vụ lắp cửa trượt	450.000
4	Rasperry Pi 4 4GB + Adapter chuyển đổi	3.250.000
5	Động cơ bước + Driver	74.000
6	Đèn Led	20.000
7	Động cơ Servo	50.000
8	Bìa mica, bìa cứng, gỗ ép	500.000
9	Các linh kiện phục vụ lắp ráp	485.000
TỔNG TIỀN		4.992.000

Bảng 9. Tổng tiền

5. Danh mục tài liệu tham khảo

Tiếng Việt

[1] Dương Văn Thiện (2005), API và Nodejs, nhà xuất bản Khoa học và Tổng hợp thành phố Hồ Chí Minh.

[2] Nguyễn Ngọc Bình Phương(2016), các giải pháp lập trình React .

[3] Học Máy và Ứng Dụng(2020) Nguyễn Nhật Quang (NXB Đại học Quốc gia TP.HCM)

[4] "Trí tuệ nhân tạo: Một cách tiếp cận hiện đại"(2015)

[5] The Picamera2 Library(2022)

[6] Học viện AI Việt Nam (AI Academy Vietnam)

Tiếng Anh

[7] Li, S. Z., & Jain, A. K. (Eds.). (2011). Handbook of Face Recognition, Computer Science, England.

[8] CRC Press (2020), “Deep Learning for Facial Recognition”, University of Kuopio, Finland.

[9] Stoyan Stefanov(2018), "React Up & Running".

Internet

1.https://www.raspberrypi.com/documentation/computers/camera_software.html#libcamera-and-libcamera-apps

2.https://viblo.asia/p/restful-api-la-gi-1Je5EDJ4lnL.

3.https://www.youtube.com/watch?v=in4n27bCTg&t=829s&ab_channel=H%E1%BB%8FiD%C3%A2nIT

4.https://www.youtube.com/watch?v=N8GhaR7K3tI&t=138s&ab_channel=F8Official

5. https://phamdinhkhanh.github.io/2020/03/12/faceNetAlgorithm.html

6. https://www.youtube.com/watch?v=3TUlJrRJUeM&t=328s

7. https://www.youtube.com/watch?v=LNwODJXcvt4

8. https://www.youtube.com/watch?v=m9fH9OWn8YM&t=1772s

9. https://abintimilsina.medium.com/yolov8-architecture-explained-a5e90a560ce5

10. https://www.youtube.com/watch?v=K3sKya6up98

11. https://www.youtube.com/watch?v=FY_uX9rAABA&t=425s

12. https://docs.ultralytics.com/vi/models/yolov8/#how-do-i-train-a-yolov8-model

13. http://arduino.vn/bai-viet/288-infrared-remote-control-dieu-khien-bang-hong-ngoai-voi-arduino

14. http://arduino.vn/bai-viet/893-cach-dung-module-dieu-khien-dong-co-l298n-cau-h-de-dieu-khien-dong-co-dc

15. http://arduino.vn/bai-viet/77-bai-2-cach-lam-den-led-nhap-nhay-theo-yeu-cau

16. https://fastapi.tiangolo.com/

17. https://react.dev/