

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**  
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

-----



**ĐỒ ÁN**  
**TỐT NGHIỆP ĐẠI HỌC**

***Đề tài: “Ứng dụng YOLOv10 trong phát hiện cháy  
thời gian thực với hỗ trợ của thiết bị tăng tốc Coral”***

**Người hướng dẫn : TS. HUỖNH TRỌNG THỨA**  
**Sinh viên thực hiện : PHU DỰ THẮNG**  
**Mã số sinh viên : N20DCCN146**  
**Lớp : D20CQCNP02-N**  
**Hệ : Đại học chính quy**

**TP. HỒ CHÍ MINH, NĂM 2024**

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**  
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

---



**ĐỒ ÁN**  
**TỐT NGHIỆP ĐẠI HỌC**

***Đề tài: “Ứng dụng YOLOv10 trong phát hiện cháy  
thời gian thực với hỗ trợ của thiết bị tăng tốc Coral”***

**Người hướng dẫn : TS. HUỖNH TRỌNG THƯA**  
**Sinh viên thực hiện : PHU DỰ THẮNG**  
**Mã số sinh viên : N20DCCN146**  
**Lớp : D20CQCNPM02–N**  
**Hệ : Đại học chính quy**

**TP. HỒ CHÍ MINH, NĂM 2024**

## LỜI CẢM ƠN

Trước tiên, em xin gửi lời tri ân sâu sắc đến quý Thầy Cô tại Học viện Công nghệ Bưu chính Viễn thông. Nhờ sự hướng dẫn tận tình và những bài giảng tâm huyết của Thầy Cô, em đã có cơ hội tiếp nhận một nền tảng kiến thức vững chắc cùng những kỹ năng thiết yếu trong suốt các năm học vừa qua. Qua đó, em đã không chỉ tích lũy được kiến thức mà còn từng bước hoàn thiện bản thân, trưởng thành hơn qua từng trải nghiệm quý báu. Đây thực sự là một hành trang vô giá mà em luôn trân trọng.

Đồng thời, em xin gửi lời cảm ơn chân thành đến Thầy Huỳnh Trọng Thừa – người đã dành thời gian và công sức để tận tâm hướng dẫn và hỗ trợ em trong suốt quá trình nghiên cứu và thực hiện đề tài. Nhờ vào sự chỉ bảo tỉ mỉ cùng những kinh nghiệm quý báu mà Thầy chia sẻ, em đã được học thêm nhiều kiến thức mới, góp phần củng cố nền tảng chuyên môn và có cơ hội phát triển bản thân một cách toàn diện. Thầy không những truyền đạt kiến thức chuyên sâu mà còn khơi gợi cho em tinh thần học hỏi và tác phong làm việc nghiêm túc, một phẩm chất quý báu sẽ luôn là kim chỉ nam giúp em vững bước trong học tập và công việc.

Bên cạnh đó, em nhận thức rõ rằng với vốn kiến thức và kinh nghiệm còn hạn chế, bài làm của em khó tránh khỏi những thiếu sót. Do đó, em rất mong nhận được sự thông cảm cùng những góp ý, nhận xét từ quý Thầy Cô. Những nhận xét này không chỉ giúp em khắc phục các điểm còn hạn chế mà còn là nguồn động lực to lớn để em tiếp tục trau dồi kiến thức và rèn luyện kỹ năng, từ đó hoàn thiện bản thân hơn. Em tin rằng những góp ý của quý Thầy Cô sẽ giúp bản thân nâng cao chất lượng nghiên cứu cũng như có được sự chuẩn bị tốt hơn cho tương lai.

Cuối cùng, em xin gửi tới quý Thầy Cô lời chúc sức khỏe dồi dào và luôn tràn đầy nhiệt huyết để tiếp tục thực hiện sứ mệnh cao cả là truyền đạt tri thức cho các thế hệ mai sau. Những cống hiến của quý Thầy Cô không chỉ giúp em mà còn góp phần nuôi dưỡng và phát triển tài năng cho nhiều thế hệ sinh viên khác. Em chân thành cảm ơn sự hướng dẫn tận tâm của quý Thầy Cô và hy vọng sẽ tiếp tục có cơ hội học hỏi, hợp tác trong các dự án và công việc trong tương lai. Kính chúc Thầy Cô luôn ngập tràn niềm vui và đạt được nhiều thành công trong sự nghiệp giáo dục.

Em xin chân thành cảm ơn!

*TP. Hồ Chí Minh, ngày      tháng      năm*

**SINH VIÊN THỰC HIỆN ĐỀ TÀI**

*(Sinh viên ký và ghi rõ họ tên)*

**PHU DỰ THẮNG**

## MỤC LỤC

LỜI CẢM ƠN .....	i
MỤC LỤC .....	ii
DANH MỤC CÁC HÌNH VẼ.....	iv
DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT.....	vi
LỜI MỞ ĐẦU .....	1
CHƯƠNG 1. MÔ HÌNH YOLOV10.....	2
1. TỔNG QUAN KIẾN TRÚC.....	2
2. TỔNG QUAN HIỆU SUẤT .....	4
2.1. DUNG LƯỢNG BỘ NHỚ.....	4
2.2. TỐC ĐỘ Suy Luận .....	4
2.3. TỔNG KẾT .....	6
3. TẬP DỮ LIỆU .....	7
3.1. PHÂN LOẠI DỮ LIỆU.....	7
3.2. GÁN NHÃN DỮ LIỆU.....	7
3.3. XÂY DỰNG TẬP DỮ LIỆU.....	8
4. TĂNG CƯỜNG TẬP DỮ LIỆU .....	9
4.1. TRIỂN KHAI MÔ HÌNH LÊN ROBOFLOW .....	9
4.2. TÍNH NĂNG AUTO LABEL .....	10
CHƯƠNG 2. THIẾT BỊ NHÚNG .....	13
1. RASPBERRY PI.....	13
2. CAMERA MODULE .....	14
3. CORAL USB ACCELERATOR.....	14
4. KẾT NỐI THIẾT BỊ .....	15
CHƯƠNG 3. ỨNG DỤNG DI ĐỘNG .....	16
1. THIẾT KẾ CHỨC NĂNG.....	16
2. THIẾT KẾ GIAO DIỆN.....	27
CHƯƠNG 4. TRIỂN KHAI THỰC TIỄN.....	32
1. THIẾT KẾ HỆ THỐNG.....	32

1.1. CHI TIẾT THÀNH PHẦN .....	33
1.2. QUY TRÌNH HOẠT ĐỘNG .....	34
2. CÔNG NGHỆ SỬ DỤNG .....	35
2.1. PHẦN MÔ HÌNH YOLOV10 VÀ THIẾT BỊ NHÚNG .....	35
2.2. PHẦN ỨNG DỤNG DI ĐỘNG .....	36
3. TRIỂN KHAI HỆ THỐNG .....	37
3.1. HUẤN LUYỆN MÔ HÌNH .....	37
3.2. CHUYỂN ĐỔI ĐỊNH DẠNG EDGE TPU .....	38
3.3. THỰC NGHIỆM .....	39
KẾT LUẬN, KIẾN NGHỊ .....	41
1. KẾT LUẬN .....	41
2. KHUYẾN NGHỊ .....	41
TÀI LIỆU THAM KHẢO .....	42
PHỤ LỤC .....	43

**DANH MỤC CÁC HÌNH VẼ**

Hình 1.1.1 Tổng quan kiến trúc YOLOv10.....	2
Hình 1.1.2 Biểu đồ so sánh YOLOv10 và các phiên bản khác .....	3
Hình 1.2.1 Biểu đồ tốc độ suy luận YOLOv10 trên CPU .....	5
Hình 1.2.2 Biểu đồ tốc độ suy luận YOLOv10 trên TPU .....	5
Hình 1.2.3 Biểu đồ so sánh tốc độ suy luận YOLOv10 trên CPU và TPU.....	6
Hình 1.3.1 Hình ảnh mã nguồn triển khai mô hình lên Roboflow .....	9
Hình 1.3.2 Hình ảnh minh họa tập dữ liệu được triển khai mô hình.....	10
Hình 1.3.3 Hình ảnh minh họa gắn nhãn tự động .....	11
Hình 1.3.4 Hình ảnh minh họa phê duyệt kết quả gắn nhãn .....	12
Hình 1.3.5 Hình ảnh minh họa tinh chỉnh kết quả gắn nhãn.....	12
Hình 2.1.1 Hình ảnh minh họa thiết bị Raspberry Pi .....	13
Hình 2.2.1 Hình ảnh minh họa thiết bị Camera Module .....	14
Hình 2.3.1 Hình ảnh minh họa thiết bị Coral USB Accelerator.....	14
Hình 3.1.1 Sơ đồ usecase hệ thống.....	16
Hình 3.1.2 Sơ đồ usecase đăng nhập .....	17
Hình 3.1.3 Sơ đồ usecase đăng ký.....	18
Hình 3.1.4 Sơ đồ usecase quên mật khẩu .....	19
Hình 3.1.5 Sơ đồ usecase đổi mật khẩu.....	19
Hình 3.1.6 Sơ đồ usecase xóa tài khoản .....	20
Hình 3.1.7 Sơ đồ usecase đăng xuất.....	21
Hình 3.1.8 Sơ đồ usecase thêm thiết bị .....	22
Hình 3.1.9 Sơ đồ usecase xóa thiết bị.....	22
Hình 3.1.10 Sơ đồ usecase cập nhật thông tin thiết bị .....	23
Hình 3.1.11 Sơ đồ usecase chia sẻ thiết bị .....	24
Hình 3.1.12 Sơ đồ usecase xóa xem thông tin ghi hình thiết bị.....	25
Hình 3.1.13 Sơ đồ usecase xóa tra cứu lịch sử ghi hình thiết bị .....	26
Hình 3.2.1 Hình ảnh minh họa giao diện xác thực tài khoản .....	27
Hình 3.2.2 Hình ảnh minh họa giao diện quản lý tài khoản.....	28
Hình 3.2.3 Hình ảnh minh họa giao diện quản lý thiết bị .....	29
Hình 3.2.4 Hình ảnh minh họa giao diện chia sẻ thiết bị .....	30
Hình 3.2.5 Hình ảnh minh họa giao diện thông tin ghi hình thiết bị.....	31
Hình 4.2.1 Workflow thiết kế hệ thống.....	32
Hình 4.2.2 Quy trình hoạt động của hệ thống .....	34
Hình 4.3.1 Biểu đồ tổng hợp kết quả huấn luyện mô hình.....	37

Hình 4.3.2 Quy trình chuyển đổi định dạng Edge TPU .....	38
Hình 4.3.3 Workflow triển khai hệ thống.....	39
Hình 4.3.4 Hình ảnh minh họa thực nghiệm .....	40

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

<i>Chữ viết tắt</i>	<i>Chữ đầy đủ</i>	<i>Giải thích</i>
AI	Artificial Intelligence	Trí tuệ nhân tạo là lĩnh vực khoa học máy tính nghiên cứu về trí tuệ trên máy móc.
API	Application Programming Interface	Giao diện lập trình ứng dụng, một nền tảng cho phép các ứng dụng tương tác và giao tiếp với nhau.
COCO	Common Objects in Context	Tập dữ liệu dùng tiêu chuẩn dùng để huấn luyện và đánh giá các mô hình nhận diện đối tượng.
CPU	Central Processing Unit	Bộ xử lý trung tâm là đơn vị xử lý chính của máy tính.
CSI	Camera Serial Interface	Giao diện nối tiếp Camera, chuẩn kết nối cho phép truyền tải dữ liệu từ camera đến các thiết bị xử lý.
CSPNet	Cross-Stage Partial Network	Mạng một phần xuyên giai đoạn, mạng nơ-ron đặc biệt giúp giảm mất mát thông tin.
Curl	Client URL	Công cụ cho phép gửi và nhận dữ liệu qua các giao thức mạng mà không cần giao diện đồ họa.
GPU	Graphics Processing Unit	Bộ xử lý chuyên dụng để xử lý đồ họa và tính toán song song.
IoT	Internet of Things	Mạng lưới vạn vật kết nối, một hệ thống cho phép các thiết bị giao tiếp với nhau qua Internet.
mAP	Mean Average Precision	Chỉ số đánh giá hiệu suất mô hình, đo lường độ chính xác trung bình của mô hình nhận diện đối tượng.
PAN	Path Aggregation Network	Mạng tổng hợp đường dẫn, kiến trúc mạng đặc biệt giúp giảm mất mát thông tin.



QR	Quick Response	<i>Loại mã vạch hai chiều cho phép giải mã nhanh hơn so với loại một chiều truyền thống.</i>
SCDD	Spatial–Channel Decoupled Downspampling	<i>Kỹ thuật phân tách thông tin theo không gian – kênh giúp giảm mất mát thông tin.</i>
SDK	Software Development Kit	<i>Bộ công cụ phát triển phần mềm là tập hợp các công cụ và thư viện hỗ trợ việc phát triển ứng dụng.</i>
TPU	Tensor Processing Unit	<i>Bộ xử lý tối ưu cho các tác vụ học sâu và mạng nơ-ron.</i>
USB	Universal Serial Bus	<i>Chuẩn kết nối thông dụng cho các thiết bị ngoại vi.</i>

## LỜI MỞ ĐẦU

Trong những năm gần đây, tình hình cháy nổ tại các khu dân cư, nhà máy và khu rừng ngày càng trở nên phức tạp và nghiêm trọng, gây thiệt hại lớn về người và tài sản. Do đó, việc phát hiện sớm và xử lý kịp thời các vụ cháy trở thành yếu tố then chốt giúp giảm những thiểu thiệt hại này. Nhu cầu nghiên cứu và phát triển các hệ thống cảnh báo cháy sớm, vì vậy, ngày càng trở nên cấp thiết.

Trước bối cảnh đó, sự phát triển vượt bậc của công nghệ nhúng và trí tuệ nhân tạo đã mở ra nhiều tiềm năng cho các hệ thống giám sát tự động hiệu quả cao. Đặc biệt, nhờ sự hỗ trợ của các thiết bị nhúng hiện đại và các mô hình học sâu tiên tiến, chuyên biệt cho việc phát hiện đối tượng như *YOLO*, chúng ta đã có thể xây dựng các hệ thống giám sát nhanh chóng và chính xác hơn bao giờ hết.

Đề tài này được nghiên cứu nhằm mục tiêu xây dựng và phát triển một hệ thống cảnh báo cháy thời gian thực, với cốt lõi là mô hình *YOLOv10* và thiết bị tăng tốc *Coral*. Hệ thống sẽ phân tích và nhận diện các dấu hiệu cháy từ hình ảnh thực tế thu được với hiệu suất cao. Bên cạnh đó, mô hình *YOLOv10* không chỉ đạt được tốc độ xử lý cao khi kết hợp với thiết bị *Coral*, mà còn có thể cải thiện độ chính xác một cách linh hoạt. Nhờ vậy, hệ thống có thể hỗ trợ hiệu quả cho công tác phòng chống cháy nổ.

Trong bài nghiên cứu này, tôi sẽ áp dụng phương pháp thực nghiệm, tập trung vào cải tiến tập dữ liệu để tối ưu hóa mô hình *YOLOv10*, mô phỏng thiết bị ghi hình sử dụng các thiết bị nhúng, và phát triển một ứng dụng di động đáp ứng nhu cầu thực tiễn. Quá trình nghiên cứu sẽ trải qua các giai đoạn từ mô phỏng thiết bị ghi hình và cải tiến tập dữ liệu, đến phát triển mô hình và đánh giá hiệu suất, và cuối cùng là xây dựng ứng dụng di động cùng với cải tiến và tối ưu hóa.

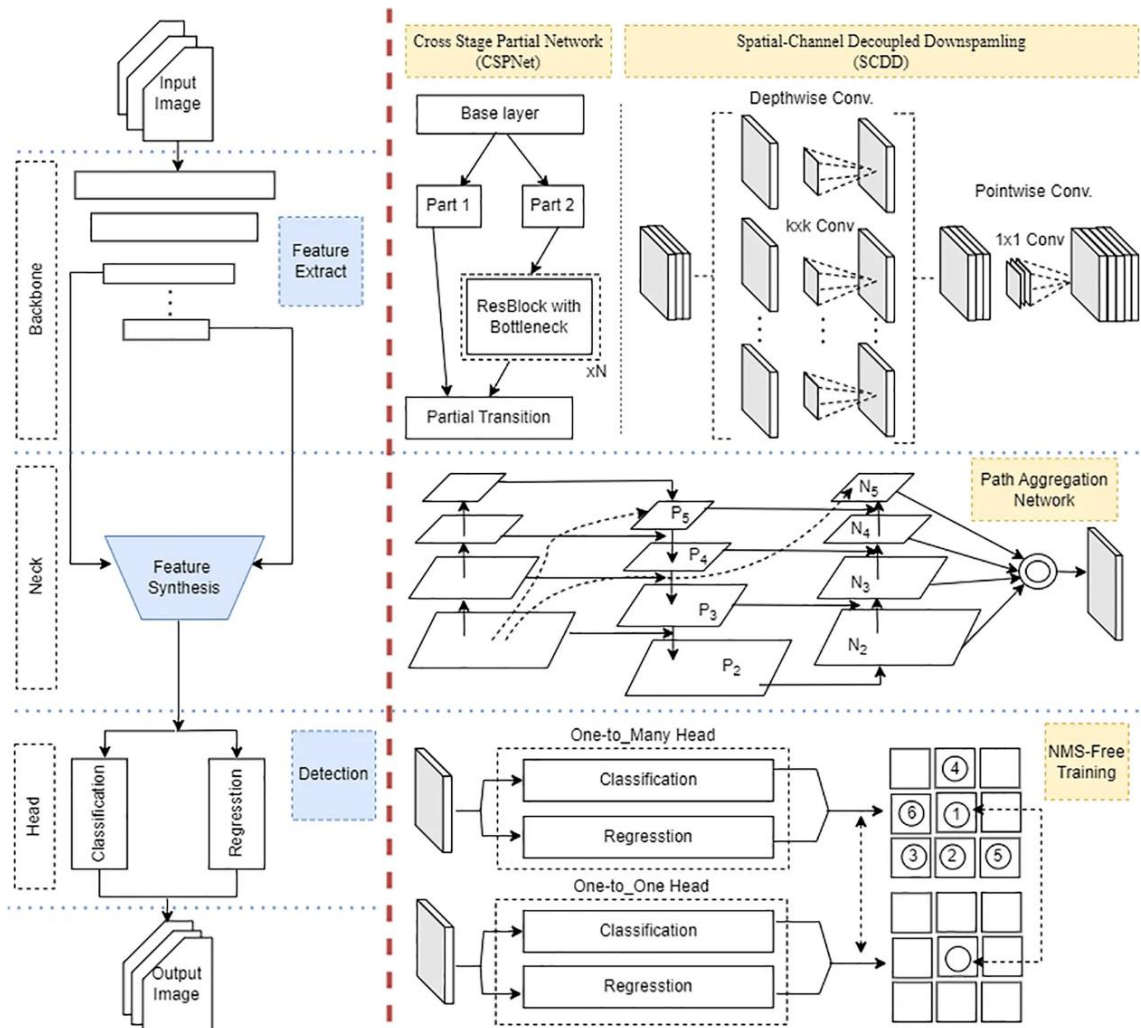
Phạm vi nghiên cứu tập trung vào việc phát hiện các đám cháy theo thời gian thực từ nguồn dữ liệu là hình ảnh thu được từ thiết bị ghi hình, sau đó gửi dữ liệu sự kiện và thông báo cảnh báo đến ứng dụng di động người dùng. Nghiên cứu chú trọng đến việc cải thiện tốc độ xử lý cho mô hình *YOLOv10* thông qua thiết bị tăng tốc *Coral*, đồng thời cải tiến tập dữ liệu huấn luyện nhằm nâng cao tính khả dụng của hệ thống trong các điều kiện thực tế khác nhau.

Báo cáo này trình bày một hệ thống phát hiện cháy thời gian thực hiện đại sử dụng mô hình *YOLOv10* kết hợp với các thiết bị nhúng. Cấu trúc báo cáo bao gồm 4 nội dung chính, lần lượt trình bày chi tiết về các thành phần cấu thành hệ thống và quá trình triển khai thực tiễn. Mỗi phần cung cấp cái nhìn tổng quan về các công nghệ sử dụng và cách thức chúng được tích hợp để xây dựng một hệ thống phát hiện cháy hoàn chỉnh.

- Chương 1: Mô hình *YOLOv10*.
- Chương 2: Thiết bị nhúng.
- Chương 3: Ứng dụng di động.
- Chương 4: Triển khai thực tiễn.

## CHƯƠNG 1. MÔ HÌNH YOLOV10

### 1. TỔNG QUAN KIẾN TRÚC



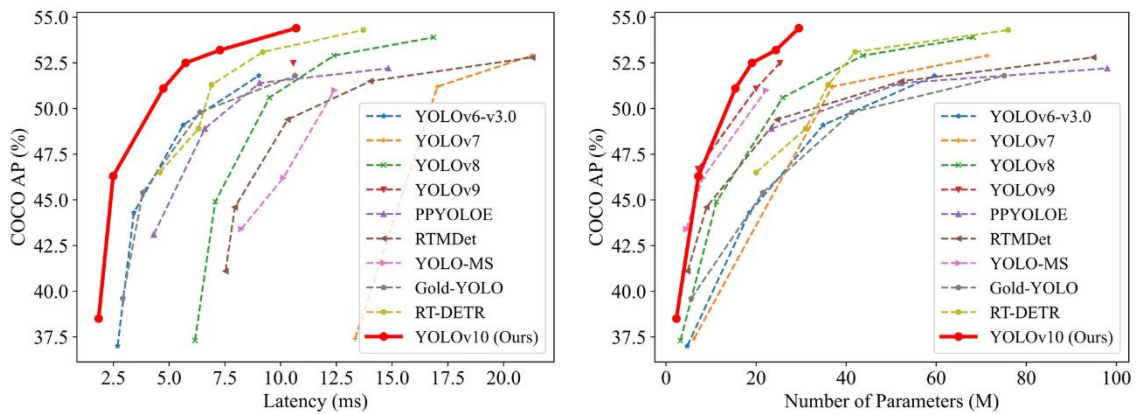
Hình 1.1.1 Tổng quan kiến trúc YOLOv10 [1]

Kiến trúc mô hình *YOLO* đã không ngừng cải tiến qua từng phiên bản để cân bằng hiệu quả giữa độ chính xác và tốc độ. Những thay đổi đáng kể trong các phiên bản mới, đặc biệt là *YOLOv10*, đã giúp mô hình đạt được hiệu suất tối ưu trong việc nhận diện đối tượng theo thời gian thực. Tổng quan, kiến trúc mô hình *YOLO* thế hệ 10 được cấu thành từ 3 thành phần chính gồm:

- **Backbone** là một mạng nơ-ron tích chập có vai trò trích xuất đặc trưng từ ảnh đầu vào, với đầu ra là các đặc trưng ở nhiều mức độ phân giải khác nhau. Trong *YOLOv10*, *backbone* áp dụng kiến trúc *CSPNet* (Cross Stage Partial Network), giúp tăng hiệu quả luồng *gradient* và giảm thiểu dư thừa tính toán, từ đó nâng cao hiệu suất mô hình.
- **Neck** là một thành phần quan trọng trong kiến trúc mạng, có vai trò kết hợp các đặc trưng từ nhiều mức độ phân giải khác nhau và tạo ra đặc trưng tối ưu. Trong *YOLOv10*, thành phần này áp dụng kiến trúc *PAN* (Path Aggregation Network), mạng nơ-ron được thiết kế để hợp nhất hiệu quả các đặc trưng đa tỉ lệ, giúp cải thiện khả năng nhận diện đối tượng ở nhiều kích thước khác nhau.

- **Head** là thành phần chịu trách nhiệm nhận diện đối tượng từ các đặc trưng tổng hợp từ *Neck*, gồm 2 mạng nơ-ron riêng biệt được sử dụng cho từng mục tiêu cụ thể, đầu ra *OneToMany* áp dụng cho quá trình huấn luyện và đầu ra *OneToOne* áp dụng cho quá trình dự đoán, giúp giảm thiểu sự phụ thuộc vào các thuật toán xử lý hậu kỳ và nâng cao khả năng thực thi thời gian thực của mô hình.

Phiên bản *YOLOv10* đã được kiểm tra kỹ lưỡng trên các bộ dữ liệu tiêu chuẩn như *COCO*, thể hiện hiệu suất và hiệu quả vượt trội. Mô hình này đạt được những kết quả xuất sắc trên nhiều phiên bản khác nhau, cho thấy những cải tiến đáng kể về độ trễ và độ chính xác so với các phiên bản trước cũng như các mô hình phát hiện đối tượng đương thời.



Hình 1.1.2 Biểu đồ so sánh *YOLOv10* và các phiên bản khác [2]


Tổng quan, *YOLOv10* cho thấy sự cân bằng xuất sắc giữa độ chính xác, độ trễ và kích thước mô hình, vượt trội hơn nhiều mô hình khác về tốc độ và hiệu quả. Với khả năng duy trì độ chính xác cao và độ trễ thấp cùng số lượng tham số ít, *YOLOv10* trở thành sự lựa chọn lý tưởng cho các tác vụ phát hiện đối tượng thời gian thực, đặc biệt trong các ứng dụng yêu cầu xử lý nhanh và có tài nguyên hạn chế.

- **Độ trễ (ms):** *YOLOv10* là một trong những mô hình nhanh nhất, đạt được chỉ số *COCO AP* rất cao (~ 52%) với độ trễ thấp hơn nhiều so với các đối thủ như *YOLOv9* và *RT-DETR*. Điều này cho thấy *YOLOv10* đặc biệt hiệu quả trong các ứng dụng yêu cầu xử lý thời gian thực.
- **Số lượng tham số (M):** *YOLOv10* có thiết kế tối ưu với ít tham số hơn so với các phiên bản nổi bật như *YOLOv9* và *RT-DETR*, nhưng vẫn có thể duy trì hoặc cải thiện độ chính xác hiệu quả. Điều này giúp giảm đáng kể chi phí tính toán, hỗ trợ cho việc triển khai trên các thiết bị có tài nguyên hạn chế.

Tổng kết, phiên bản *YOLOv10* thiết lập một tiêu chuẩn mới trong phát hiện đối tượng thời gian thực bằng cách khắc phục những điểm yếu của các phiên bản *YOLO* trước đó, đồng thời tích hợp các chiến lược thiết kế sáng tạo. Khả năng cung cấp độ chính xác cao với chi phí tính toán thấp đáng kể khiến phiên bản này trở thành lựa chọn lý tưởng cho nhiều ứng dụng thực tế.

## 2. TỔNG QUAN HIỆU SUẤT

### 2.1. DUNG LƯỢNG BỘ NHỚ

 Môi trường thực thi


- Dịch vụ Google Colab.
- Hệ điều hành Linux dùng CPU (Intel Xeon 2.20GHz).
- Python 3.10.12 dùng Ultralytics 8.3.25 và PyTorch 2.5.0.

Phiên bản mô hình gốc	Độ chính xác gốc (%)	Thời gian chuyển đổi (s)	Dung lượng (MB)	
			PyTorch (CPU)	TFLite Edge (TPU)
<i>yolov10n</i>	38.5	313.8	5.58	3.32
<i>yolov10s</i>	46.3	495.9	15.8	9.66
<i>yolov10m</i>	51.1	872.2	32	17.8
<i>yolov10b</i>	52.5	1168.9	39.7	21.2
<i>yolov10l</i>	53.2	1436.1	49.9	26.3
<i>yolov10x</i>	54.4	1933	61.4	32.1

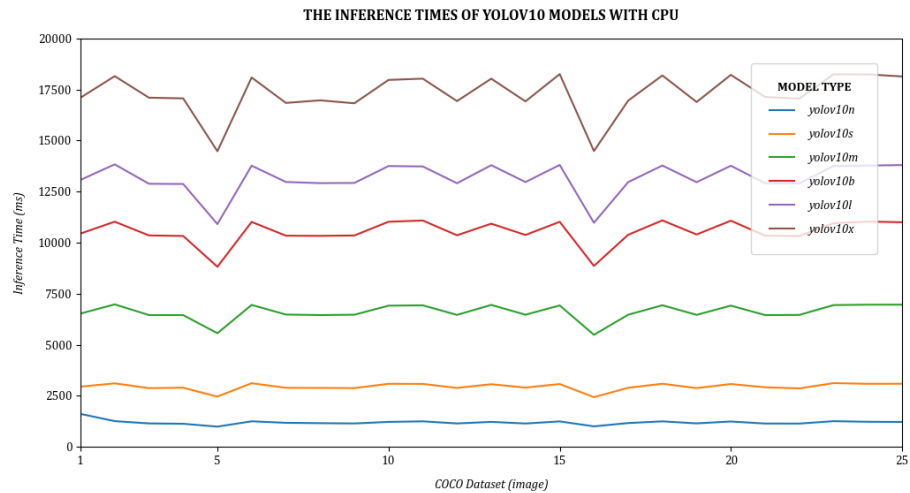
Thời gian chuyển đổi định dạng và dung lượng bộ nhớ chiếm dụng của các mô hình *YOLOv10* tăng dần từ phiên bản *N* đến *X*. Điều này cho thấy mô hình có kích thước lớn và phức tạp hơn sẽ cần nhiều tài nguyên hơn, thời gian xử lý cũng như bộ nhớ sử dụng cũng gia tăng đáng kể để mô hình có thể chuyển đổi và hoạt động ổn định. Sự gia tăng này diễn ra bất kể mô hình được triển khai để vận hành trên thiết bị sử dụng *CPU* hay *TPU*.

Đối với tất cả các phiên bản mô hình *YOLOv10*, dung lượng bộ nhớ chiếm dụng ở định dạng *TPU* luôn nhỏ hơn đáng kể so với *CPU*. Điều này cho thấy định dạng mô hình tối ưu cho *TPU* đã được áp dụng các kỹ thuật tối ưu hóa đặc biệt để mô hình hoạt động hiệu quả trên phần cứng *Edge*. Nhờ vậy, mô hình ở định dạng *TPU* không những sử dụng ít không gian bộ nhớ hơn mà còn nâng cao hiệu suất xử lý của mô hình.

### 2.2. TỐC ĐỘ SUY LUẬN

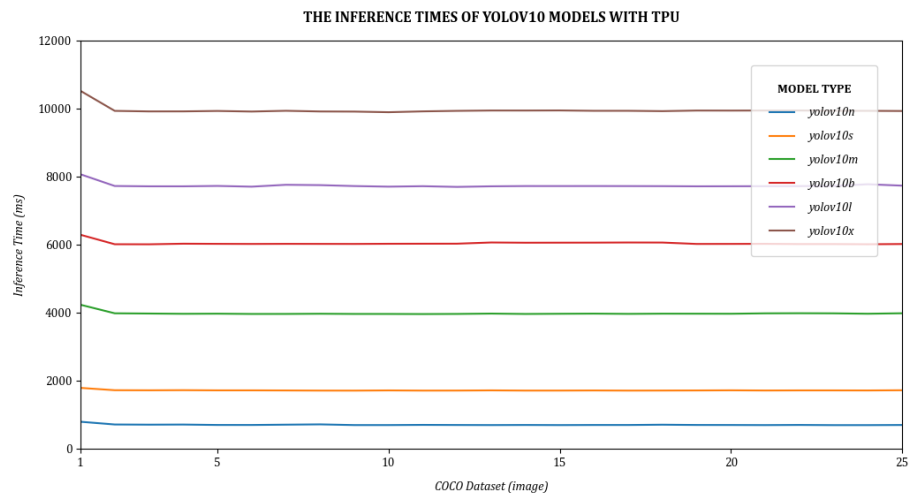
 Môi trường thực thi

- Thiết bị Raspberry Pi 4 Model B và Coral USB Accelerator.
- Hệ điều hành Raspberry Pi (Debian Bookworm).
- Python 3.11.2 dùng Ultralytics 8.3.25 và PyTorch 2.5.1.

**a) Hoạt động trên CPU (Raspberry Pi CPU)****Hình 1.2.1 Biểu đồ tốc độ suy luận YOLOv10 trên CPU**

Biểu đồ cho thấy rằng thời gian suy luận của các mô hình YOLOv10 trên CPU tăng dần theo độ phức tạp của mô hình.

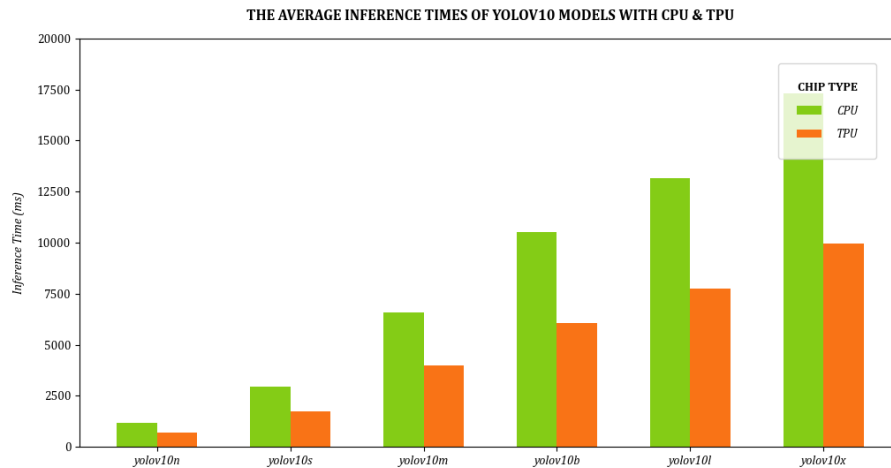
- Phiên bản *yolov10n* có tốc độ suy luận nhanh nhất ( $< 2500\text{ ms}$ ), trong khi *yolov10x* chậm nhất ( $> 17500\text{ ms}$ ).
- Tốc độ suy luận của mô hình dao động nhẹ ( $\pm 1000\text{ ms}$ ) và tăng dần theo độ phức tạp của mô hình.

**b) Hoạt động trên TPU (Coral USB Accelerator TPU)****Hình 1.2.2 Biểu đồ tốc độ suy luận YOLOv10 trên TPU**

Biểu đồ cho thấy rằng thời gian suy luận của các mô hình YOLOv10 trên TPU tăng dần theo độ phức tạp của mô hình.

- Phiên bản *yolov10n* có tốc độ suy luận nhanh nhất ( $< 2000\text{ ms}$ ), trong khi *yolov10x* lâu nhất ( $> 10000\text{ ms}$ ).
- Tốc độ suy luận của mô hình đạt được sự ổn định ( $\pm 50\text{ ms}$ ) ở mọi độ phức tạp của mô hình.

### c) So sánh phiên bản trên CPU và TPU



Hình 1.2.3 Biểu đồ so sánh tốc độ suy luận YOLOv10 trên CPU và TPU

Biểu đồ cho thấy rằng TPU cải thiện đáng kể thời gian suy luận cho các mô hình YOLOv10 so với CPU.

- Đối với cả CPU và TPU, thời gian suy luận của mô hình luôn tăng dần từ phiên bản *yolov10n* đến *yolov10x*, phản ánh đúng tính chất của độ phức tạp và kích thước của mô hình.
- TPU luôn có thời gian suy luận thấp hơn đáng kể so với CPU ở mọi phiên bản mô hình. Sự khác biệt càng lớn khi độ phức tạp của mô hình càng lớn, cho thấy hiệu suất xử lý vượt trội của TPU.
- Các phiên bản nhẹ như *yolov10n* và *yolov10s* có thời gian suy luận rất thấp trên cả CPU và TPU, là lựa chọn phù hợp cho các ứng dụng yêu cầu nhận diện thời gian thực.

## 2.3. TỔNG KẾT

Tổng quan, sự khác biệt về dung lượng bộ nhớ và tốc độ suy luận của mô hình YOLOv10 giữa CPU và TPU là rất rõ rệt. TPU được tối ưu hóa để xử lý các phép toán *tensor* nhanh hơn, trong khi CPU mặc dù linh hoạt nhưng lại không tối ưu cho các phép toán phức tạp này.

Các phiên bản nhẹ của mô hình YOLOv10 như *N* và *S* có dung lượng ít và thời gian suy luận nhanh, phù hợp cho các tác vụ yêu cầu ít bộ nhớ và tốc độ cao. Ngược lại, các phiên bản nặng như *M*, *L*, và *X* tuy cần nhiều bộ nhớ và thời gian xử lý hơn nhưng mang lại độ chính xác vượt trội, phù hợp cho các tác vụ yêu cầu độ chính xác cao.

Kết luận, việc lựa chọn phiên bản mô hình YOLOv10 phù hợp cần được cân nhắc kỹ lưỡng giữa khả năng của phần cứng và các yêu cầu đặc thù của ứng dụng. Điều này giúp đảm bảo sự tối ưu về bộ nhớ và hiệu suất, cũng như khả năng xử lý khi triển khai mô hình trong thực tế.



### 3. TẬP DỮ LIỆU

Trong bài toán dự báo cháy sử dụng công nghệ *AI*, dữ liệu huấn luyện giữ vai trò đặc biệt quan trọng. Đối với các mô hình nhận diện đối tượng đặc thù như *YOLO*, hình ảnh không chỉ cần độ chính xác cao mà còn phải đảm bảo sự đa dạng để nâng cao khả năng phát hiện của mô hình trong nhiều điều kiện thực tiễn khác nhau. Ngoài ra, tập dữ liệu cần tuân theo các tiêu chuẩn nhất định nhằm đảm bảo tính tương thích và độ hiệu quả trong quá trình huấn luyện với các nền tảng công nghệ khác.

#### 3.1. PHÂN LOẠI DỮ LIỆU

- **Hình ảnh có lửa** gồm các hình ảnh thể hiện rõ sự hiện diện của ngọn lửa với sự đa dạng về hình dạng, màu sắc, kích thước và độ sáng trong nhiều bối cảnh khác nhau (ngoài trời, trong nhà, ban ngày, ban đêm, v.v.). Điều này giúp mô hình nhận diện chính xác các dấu hiệu cháy.
- **Hình ảnh có khói** gồm các hình ảnh thể hiện rõ sự hiện diện của khói với sự đa dạng về hình dạng, màu sắc, kích thước và độ dày đặc trong nhiều bối cảnh khác nhau. Điều này giúp mô hình phát hiện sớm dấu hiệu cháy, ngay cả khi ngọn lửa chưa xuất hiện.
- **Hình ảnh khác** gồm các hình ảnh ngoài sự hiện diện của khói và lửa, sẽ có thêm sự xuất hiện của các yếu tố gây nhầm lẫn như ánh đèn, ánh sáng phản chiếu, v.v. Điều này giúp mô hình phân biệt giữa các dấu hiệu cháy thực và các nguồn sáng không liên quan.

#### 3.2. GÁN NHÃN DỮ LIỆU

	<i>Trường</i>	<i>Giá trị</i>	<i>Mô tả</i>
<i>Nhãn</i>	<code>&lt;class_index&gt;</code>	Số nguyên	Chỉ mục của nhãn trong tập tin cấu hình tập dữ liệu.
<i>Hộp giới hạn</i>	<code>&lt;x_coordinate&gt;</code>	[0, 1]	Tọa độ tương đối trên trục x của điểm trung tâm.
	<code>&lt;y_coordinate&gt;</code>	[0, 1]	Tọa độ tương đối trên trục y của điểm trung tâm.
	<code>&lt;width&gt;</code>	[0, 1]	Chiều rộng tương đối của hộp giới hạn trên hình ảnh.
	<code>&lt;height&gt;</code>	[0, 1]	Chiều cao tương đối của hộp giới hạn trên hình ảnh.

- **Nhãn** là một giá trị đại diện để mô tả đối tượng cụ thể trong bối cảnh bài toán mà mô hình phải nhận diện trong phạm vi của hộp giới hạn, thường được biểu diễn bằng số hoặc chuỗi ký tự như *fire* (lửa), *smoke* (khói) và *other* (yếu tố gây nhiễu), v.v.



- **Hộp giới hạn** là một tập giá trị để xác định phạm vi tồn tại của khu vực chứa đối tượng trong hình ảnh. Định dạng của tập giá trị có thể được biểu diễn khác nhau và được trực quan hóa bằng hình chữ nhật bao quanh khu vực chứa đối tượng cần nhận diện.

### 3.3. XÂY DỰNG TẬP DỮ LIỆU

Tổng quan, tập dữ liệu huấn luyện mô hình *YOLOv10* gồm những hình ảnh được chọn lọc phù hợp cho quá trình nhận diện các dấu hiệu cháy. Bên cạnh đó, việc sử dụng 3 nhãn sẽ giúp mô hình giảm thiểu báo động giả một cách chủ động, từ đó nâng cao tính khả dụng của mô hình trong thực tế.

- **Tiền xử lý dữ liệu** gồm những thao tác biến đổi dữ liệu như cân bằng màu sắc, chuẩn hóa kích thước, v.v. và những kỹ thuật tăng cường dữ liệu như xoay và lật, thay đổi độ sáng, v.v. Điều này giúp cải thiện chất lượng và tạo ra sự đa dạng cho tập dữ liệu, giúp mô hình học tốt hơn.
- **Tăng cường dữ liệu** là quá trình tạo ra các phiên bản mới từ dữ liệu gốc bằng cách áp dụng các phép biến đổi khác nhau. Mục tiêu của tăng cường dữ liệu là làm phong phú tập dữ liệu huấn luyện, giúp mô hình tổng quát hóa tốt hơn và giảm nguy cơ bị quá khớp.

Trong quá trình tiền xử lý, tập dữ liệu được áp dụng các kỹ thuật tăng cường dữ liệu như gây nhiễu, xoay ảnh, v.v. Những kỹ thuật biến đổi này giúp tăng độ phong phú và đa dạng của dữ liệu, giúp mô hình nâng cao khả năng nhận diện trong nhiều tình huống thực tế khác nhau.


- **Phân chia tập dữ liệu** là quá trình chia tập dữ liệu gốc thành 3 phần, mỗi phần phục vụ cho mục tiêu cụ thể trong quá trình huấn luyện mô hình. Có nhiều phương pháp phân chia khác nhau, việc lựa chọn phương pháp phù hợp phụ thuộc vào đặc điểm của dữ liệu và mục tiêu bài toán.
  - **Tập huấn luyện** là tập dữ liệu con chiếm tỉ lệ lớn nhất, được sử dụng để mô hình học các đặc trưng của đối tượng cần nhận diện.
  - **Tập kiểm tra** là tập dữ liệu con chiếm tỉ lệ nhỏ, được sử dụng để đánh giá khả năng học của mô hình trong quá trình huấn luyện.
  - **Tập kiểm thử** là tập dữ liệu con chiếm tỉ lệ nhỏ nhất, được sử dụng để đánh giá hiệu suất của mô hình sau quá trình huấn luyện.

	Số lượng	Tỷ lệ
<i>Tập huấn luyện</i>	15,626	0.74
<i>Tập kiểm tra</i>	3578	0.17
<i>Tập kiểm thử</i>	1914	0.09
<b>Tổng</b>	21,118	

## 4. TĂNG CƯỜNG TẬP DỮ LIỆU

Trong bối cảnh phát triển mạnh mẽ của công nghệ AI, các công cụ tự động hóa quy trình gán nhãn dữ liệu được phát triển một cách nhanh chóng. Tính năng *Auto Label* của nền tảng *Roboflow* tận dụng các tiến bộ này để tự động hóa việc gán nhãn cho các đối tượng trong hình ảnh. Công nghệ học sâu cho phép *Auto Label* nhận diện và phân loại đối tượng trong hình ảnh một cách chính xác, qua đó giảm thiểu sự can thiệp của con người và tiết kiệm đáng kể thời gian so với phương pháp gán nhãn thủ công truyền thống.

### 4.1. TRIỂN KHAI MÔ HÌNH LÊN ROBOFLOW

 *Môi trường thực thi*

- Mỗi Dataset trên Roboflow chỉ liên kết với 1 mô hình.
- Mô hình YOLOv10 dùng Ultralytics  $\geq 8.0.196$ .
- Python  $\geq 3.6$  dùng Roboflow  $\geq 1.0.1$ .



```
from roboflow import Roboflow

rf = Roboflow("api_key")
project = rf.workspace("workspace").project("project")
version = project.version("version")
version.deploy("model_type", "model_path", "file_name")
```

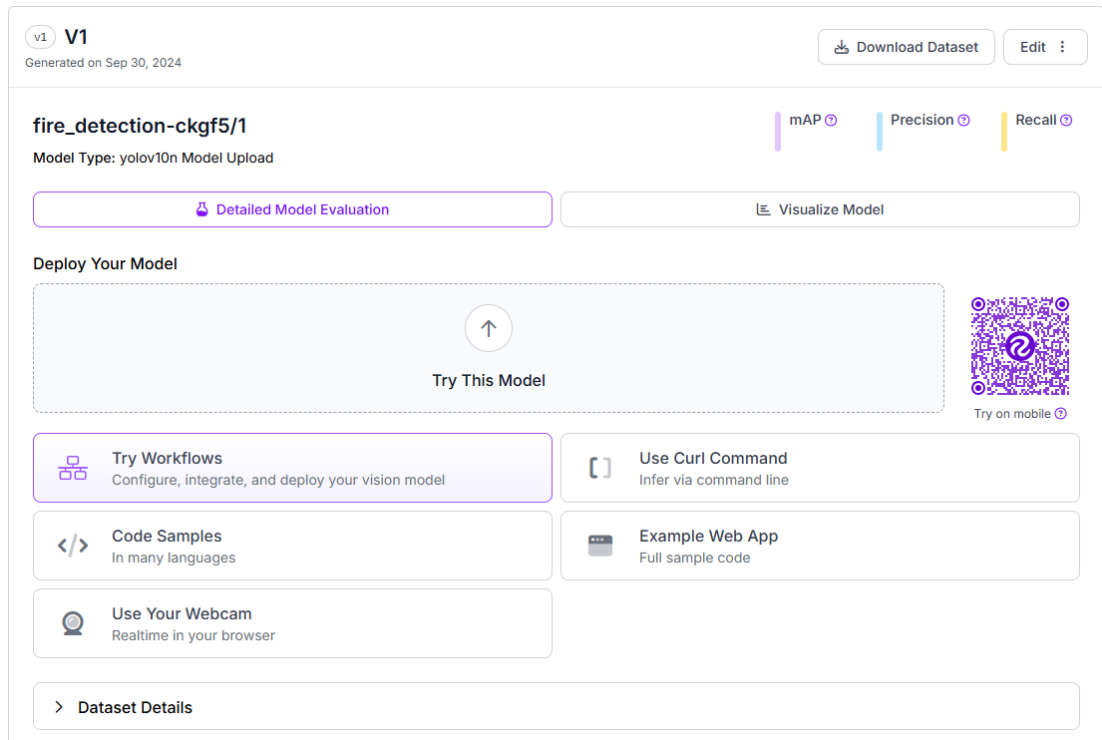
Hình 1.3.1 Hình ảnh mã nguồn triển khai mô hình lên Roboflow

*Roboflow* hỗ trợ triển khai mô hình YOLO được huấn luyện dùng thư viện *Ultralytics* lên nền tảng trực tuyến thông qua thư viện *Python* cùng tên. Mã nguồn *Python* trên thực hiện tính năng này.

- **api\_key:** Thông tin khóa API để truy cập vào tài nguyên trên Roboflow.
- **workspace:** Thông tin tên Workspace nơi lưu trữ các Project.
- **project:** Thông tin tên Project nơi lưu trữ các Dataset.
- **version:** Thông tin phiên bản Dataset.
- **model\_type:** Thông tin loại mô hình triển khai.
- **model\_path & file\_name:** Thông tin đường dẫn đến tệp tin mô hình.

Tổng kết, tính năng triển khai mô hình lên *Roboflow* giúp tích hợp dễ dàng các mô hình học sâu như YOLO vào các hệ thống thực tế. Quá trình này bao gồm việc huấn luyện mô hình trên một phiên bản Dataset cụ thể, sau đó triển khai mô hình lên nền tảng *Roboflow* để thực hiện dự đoán. Nhờ các thư viện *Python*, điều này đã có thể được thực hiện một cách nhanh chóng và hiệu quả.

## 4.2. TÍNH NĂNG AUTO LABEL

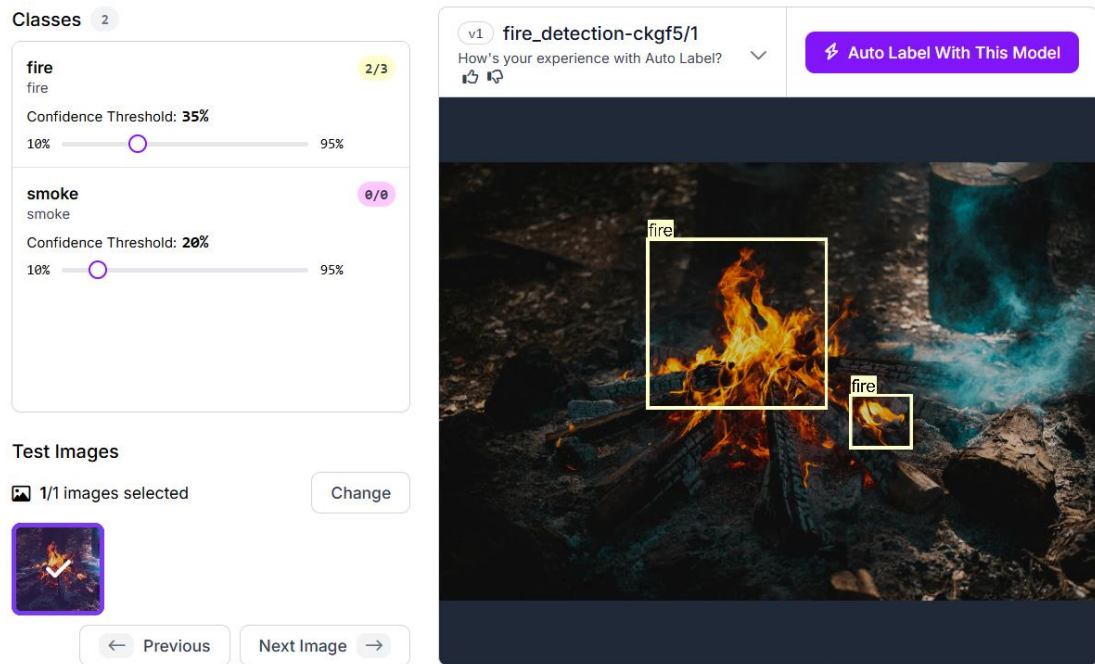


Hình 1.3.2 Hình ảnh minh họa tập dữ liệu được triển khai mô hình

- **Evaluation Metrics:** Roboflow cung cấp các thông số đánh giá quan trọng của mô hình như *mAP*, *Precision* và *Recall*, giúp đánh giá trực quan hiệu suất của mô hình trong việc phân loại và nhận diện các đối tượng.
- **Detailed Model Evaluation:** Roboflow cung cấp những phân tích chuyên sâu hơn về hiệu suất của mô hình. Điều này giúp xác định các điểm yếu cần cải thiện, hỗ trợ tối ưu hóa mô hình hiệu quả hơn.
- **Visualize Model:** Roboflow hỗ trợ hiển thị kết quả dự đoán của mô hình trực quan trên dữ liệu mới hoặc tập dữ liệu đã có. Điều này giúp đánh giá cách mô hình nhận diện các đối tượng trong các tình huống thực tế.
- **Try on Mobile:** Roboflow cung cấp mã QR giúp dễ dàng kiểm tra mô hình ngay cả thiết bị di động, giúp tăng cường trải nghiệm thực tế và đánh giá tính hiệu quả của mô hình khi triển khai trên các thiết bị khác nhau.
- **Deployment Options:** Roboflow cung cấp đa dạng các phương thức triển khai mô hình. Điều này giúp tích hợp mô hình nhanh chóng và linh hoạt vào nhiều môi trường ứng dụng khác nhau.
  - **Try Workflow** giúp cấu hình và triển khai mô hình trên đa dạng các nền tảng như *Server*, *IoT* và *Cloud*, v.v. giúp mô hình hoạt động linh hoạt và hiệu quả trên các môi trường khác nhau.
  - **Use Curl Command** cung cấp cách sử dụng mô hình thông qua giao diện dòng lệnh với công cụ *Curl*, thích hợp cho các ứng dụng yêu cầu giao diện lập trình nhanh và đơn giản.

- **Code Samples & Example Web App** cung cấp mã nguồn sẵn có cho nhiều ngôn ngữ lập trình, giúp tích hợp mô hình vào hệ thống và triển khai ứng dụng web một cách nhanh chóng và dễ dàng.
- **Use Your Webcam: Roboflow** hỗ trợ thử nghiệm mô hình ngay trong trình duyệt thông qua webcam, giúp kiểm tra nhanh chóng khả năng nhận diện của mô hình trong thời gian thực.

#### a) Tinh chỉnh hiệu suất dự đoán

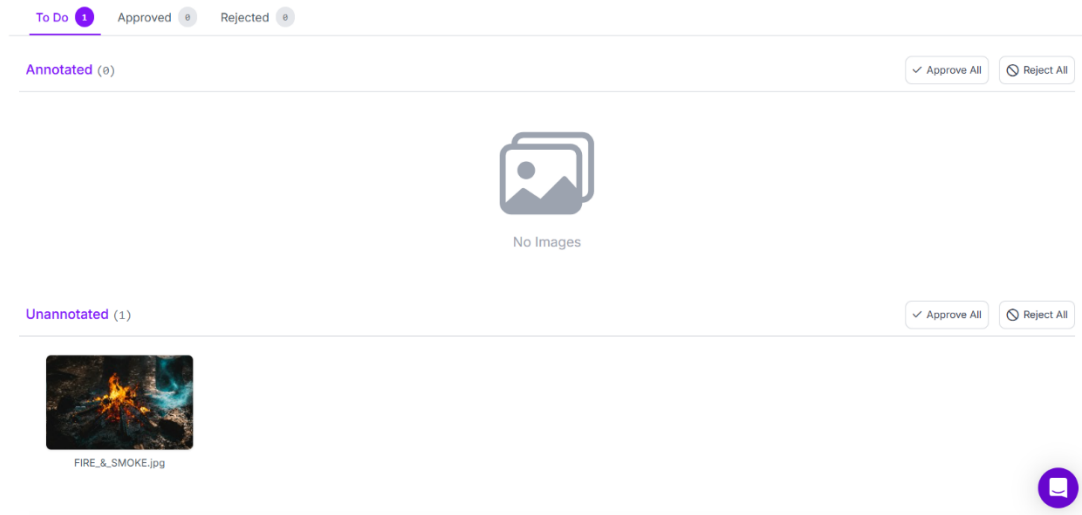


Hình 1.3.3 Hình ảnh minh họa gán nhãn tự động

- **Kiểm thử hình ảnh:** Roboflow cho phép lựa chọn các hình ảnh trong tập dữ liệu để tiến hành kiểm thử với giao diện trực quan, giúp nhanh chóng đánh giá tổng quan về hiệu suất của mô hình.
- **Hiển thị thông tin dự đoán:** Roboflow hỗ trợ hiển thị thông tin các nhãn và số lượng đối tượng được phát hiện tương ứng, giúp dễ dàng điều chỉnh ngưỡng tin cậy để kiểm soát kết quả dự đoán.
- **Điều chỉnh ngưỡng tin cậy:** Roboflow cho phép thiết lập ngưỡng tin cậy riêng biệt cho từng nhãn, từ đó mô hình chỉ giữ lại các dự đoán đạt ngưỡng chỉ định, giúp kiểm soát độ nhạy của mô hình.

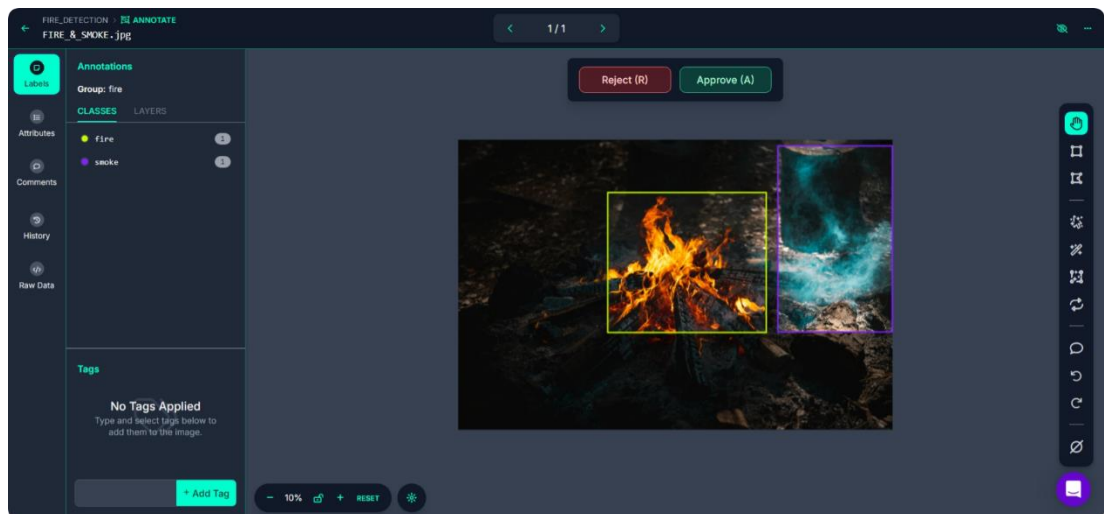
Tổng quan, các tính năng của Roboflow đóng vai trò quan trọng trong việc tối ưu hóa và điều chỉnh mô hình. Kiểm thử hình ảnh giúp đánh giá nhanh chóng hiệu suất mô hình, trong khi hiển thị thông tin dự đoán cung cấp cái nhìn rõ ràng về kết quả. Đồng thời, tính năng điều chỉnh ngưỡng tin cậy giúp kiểm soát linh hoạt độ nhạy của mô hình, từ đó giảm thiểu sai sót. Những tính năng này kết hợp cùng nhau tạo ra một quy trình gán nhãn hiệu quả, góp phần nâng cao hiệu suất và tính ổn định của mô hình trong suốt quá trình phát triển.

### b) *Tinh chỉnh kết quả dự đoán*



Hình 1.3.4 Hình ảnh minh họa phê duyệt kết quả gán nhãn

- **Chấp nhận/Từ chối:** Roboflow cho phép quản lý và kiểm duyệt hình ảnh trong quá trình gán nhãn dữ liệu, đảm bảo chỉ những hình ảnh đạt yêu cầu chất lượng mới được thêm vào tập dữ liệu.



Hình 1.3.5 Hình ảnh minh họa tinh chỉnh kết quả gán nhãn

- **Tinh chỉnh kết quả:** Roboflow cho phép điều chỉnh thủ công kết quả gán nhãn bởi tính năng *Auto Label*. Sau khi kiểm tra, ta có thể chọn phê duyệt hoặc từ chối kết quả để đảm bảo hình ảnh thỏa mãn yêu cầu.

Tổng quan, tính năng phê duyệt và tinh chỉnh kết quả gán nhãn giúp đảm bảo chất lượng dữ liệu đầu vào, từ đó hỗ trợ hiệu quả cho quá trình huấn luyện mô hình. Những tính năng này cung cấp sự linh hoạt trong việc kiểm soát chặt chẽ quá trình xây dựng tập dữ liệu. Qua đó, quy trình phát triển mô hình trở nên tối ưu hơn, giúp giảm thiểu sai sót và cải thiện tính ổn định của kết quả dự đoán, góp phần tạo ra những mô hình đáng tin cậy, không chỉ hoạt động hiệu quả mà còn đáp ứng sự đa dạng của môi trường thực tế.

## CHƯƠNG 2. THIẾT BỊ NHÚNG

*Raspberry Pi*, *Camera Module* và *Coral USB Accelerator* là các thiết bị nhúng thiết yếu trong hệ thống cảnh báo cháy thời gian thực. *Raspberry Pi* đóng vai trò là thiết bị xử lý trung tâm, thực thi các tác vụ tính toán và xử lý dữ liệu. *Camera Module* là thiết bị ghi hình, ghi lại hình ảnh môi trường xung quanh. *Coral USB Accelerator* giúp tăng tốc quá trình suy luận của mô hình, giảm độ trễ và nâng cao hiệu suất xử lý. Tổng quan, sự phối hợp giữa 3 thiết bị này tạo nên một hệ thống mạnh mẽ đáp ứng tính thời gian thực.

### 1. RASPBERRY PI



Hình 2.1.1 Hình ảnh minh họa thiết bị Raspberry Pi [3]

*Raspberry Pi* là một máy tính nhúng nhỏ gọn và linh hoạt, được sử dụng trong các dự án IoT và ứng dụng nhúng. Với khả năng xử lý mạnh mẽ, hỗ trợ đa dạng cổng kết nối và tính năng kết nối không dây, *Raspberry Pi* phù hợp với vai trò là bộ điều khiển trung tâm trong các hệ thống tự động, giúp thu thập, xử lý dữ liệu và thực hiện các tác vụ học máy với hiệu suất cao.

#### a) Thông số kỹ thuật [3]

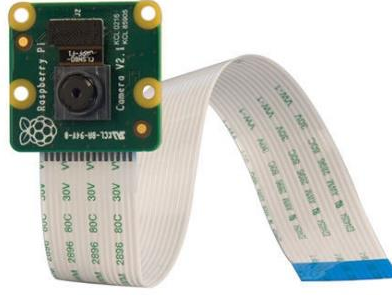
- **MODEL:** *Raspberry Pi 4 Model B.*
- **PROCESSOR:** *Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz.*
- **MEMORY:** *1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM.*
- **CONNECTIVITY:** *2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE, Gigabit Ethernet.*
- **DIMENSIONS:** *85 mm × 56 mm × 17 mm.*
- **PORTS:** *2 USB 3.0 & 2.0, 2 MIPI DSI & CSI, 2 micro-HDMI.*
- **SD CARD SUPPORT:** *Micro-SD card.*
- **PYTHON LIBRARY SUPPORT:** *RPi.GPIO, gpiozero, smbus2, ...*

#### b) Lý do lựa chọn

*Raspberry Pi* có thiết kế nhỏ gọn, hiệu năng mạnh mẽ và khả năng mở rộng linh hoạt, có thể đáp ứng các yêu cầu của ứng dụng nhúng ở nhiều quy mô khác nhau. Ngoài ra, thiết bị này cũng tương thích tốt với các thiết bị khác như *Camera Module* và *Coral USB Accelerator*, giúp triển khai các hệ thống IoT một cách dễ dàng và hiệu quả.



## 2. CAMERA MODULE



Hình 2.2.1 Hình ảnh minh họa thiết bị Camera Module [4]

Camera Module là thiết bị ghi hình nhỏ gọn, được thiết kế để dễ dàng tích hợp với Raspberry Pi. Thiết bị này phù hợp với vai trò thu nhận hình ảnh từ môi trường xung quanh trong thời gian thực, cung cấp chất lượng hình ảnh sắc nét và hoạt động ổn định. Với khả năng chụp ảnh chi tiết và liên tục, Camera Module là nguồn cung cấp dữ liệu lý tưởng cho các hệ thống phân tích hình ảnh và nhận diện đối tượng.

### a) Thông số kỹ thuật [4]

- **MODEL:** Raspberry Pi Camera Module V2 8MP.
- **LENS:** Fixed focal length.
- **SENSOR RESOLUTION:** 8 megapixels (3280 × 2464).
- **VIDEO SUPPORT:** 1080p30, 726p60, 640 × 480p90.
- **CONNECTIVITY:** 15 cm ribbon cable to CSI port.
- **DIMENSIONS:** 25 mm × 23 mm × 9 mm.
- **WEIGHTS:** 3 g.
- **RASPBERRY PI SUPPORT:** All versions.
- **PYTHON LIBRARY SUPPORT:** PiCamera, PiCamera2.

### b) Lý do lựa chọn

Camera Module được thiết kế nhỏ gọn và tối ưu cho dòng thiết bị Raspberry Pi, giúp việc tích hợp trở nên đơn giản và nhanh chóng. Với khả năng hoạt động ổn định ngay cả trong điều kiện ánh sáng yếu, thiết bị này đảm bảo cung cấp chất lượng hình ảnh vượt trội trong nhiều điều kiện môi trường khác nhau. Điều này mang lại sự linh hoạt và hiệu quả cho nhiều loại hệ thống thực tế.

## 3. CORAL USB ACCELERATOR



Hình 2.3.1 Hình ảnh minh họa thiết bị Coral USB Accelerator [5]

*Coral USB Accelerator* là một thiết bị hỗ trợ tăng tốc phần cứng, sử dụng công nghệ *Edge TPU* để tăng cường tốc độ suy luận của các mô hình AI. Với khả năng xử lý nhanh chóng các tác vụ suy luận và tiêu thụ năng lượng thấp, thiết bị này phù hợp cho các ứng dụng nhúng yêu cầu tính chất thời gian thực, là giải pháp lý tưởng cho các hệ thống yêu cầu khả năng phân tích và phản hồi nhanh.

**a) Thông số kỹ thuật [5]**

- **MODEL:** *Coral USB Accelerator.*
- **ML ACCELERATOR:** *Google Edge TPU coprocessor.*
- **PERFORMANCE:** *4 TOPS (int8); 2 TOPS per watt.*
- **CONNECTIVITY:** *USB 3.0 port.*
- **DIMENSIONS:** *65 mm × 30mm × 8 mm.*
- **RASPBERRY PI SUPPORT:** *All versions, but only tested Raspberry Pi 3 Model B+ and Raspberry Pi 4.*
- **PYTHON LIBRARY SUPPORT:** *PyCoral, Edge TPU Silva, Ultralytics.*

**b) Lý do lựa chọn**

*Coral USB Accelerator* có thiết kế nhỏ gọn và tích hợp dễ dàng với *Raspberry Pi* thông qua cổng *USB*. Nhờ sử dụng *Edge TPU*, thiết bị này giúp cải thiện tốc độ suy luận của mô hình AI đáng kể so với *CPU* của *Raspberry Pi*. Ngoài ra, *Coral USB Accelerator* tiêu thụ điện năng thấp, giúp tiết kiệm năng lượng trong khi vẫn duy trì hiệu suất cao trong suốt quá trình hoạt động.

#### **4. KẾT NỐI THIẾT BỊ**

Hệ thống cảnh báo cháy thời gian thực được thiết kế dựa trên sự phối hợp chặt chẽ giữa các thiết bị nhúng *Raspberry Pi*, *Camera Module* và *Coral USB Accelerator*. Các thiết bị này đều hỗ trợ kết nối với nhau một cách trực tiếp và hiệu quả, nhờ vào các cổng kết nối phù hợp trên *Raspberry Pi*.

○ **Kết nối phần cứng**

*Trên thiết bị Raspberry Pi, ta thiết lập các kết nối sau:*

- *Coral USB Accelerator: USB 3.0 port.*
- *Camera Module: MIPI CSI port.*

○ **Kết nối phần mềm**

*Trên mã nguồn Python, ta cài đặt các thư viện sau:*

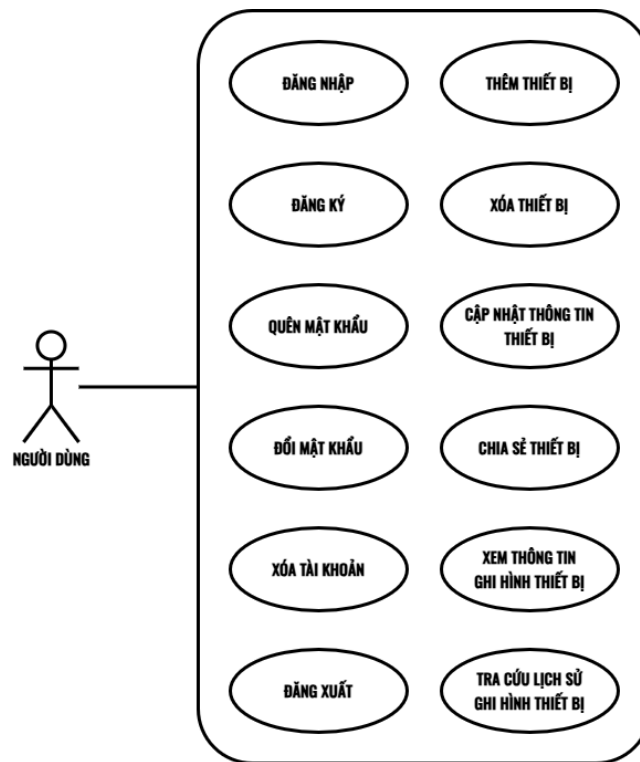
- *Coral USB Accelerator: Ultralytics.*
- *Camera Module: PiCamera2.*

Tổng kết, việc kết nối giữa phần cứng và phần mềm trong hệ thống được thiết lập một cách trực quan và dễ dàng nhờ sự hỗ trợ của *Raspberry Pi* cùng các thư viện *Python*. Điều này giúp đảm bảo tốc độ truyền tải dữ liệu nhanh, tối ưu hóa hiệu suất của từng thiết bị tích hợp và giảm thiểu độ trễ trong quá trình xử lý.



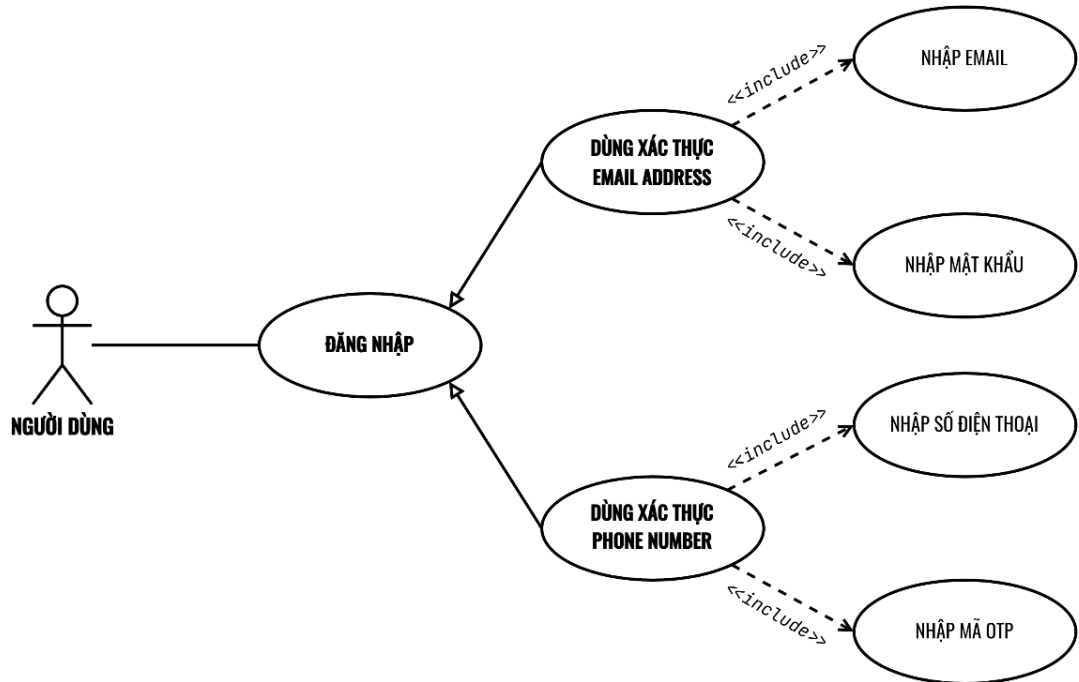
## CHƯƠNG 3. ỨNG DỤNG DI ĐỘNG

### 1. THIẾT KẾ CHỨC NĂNG



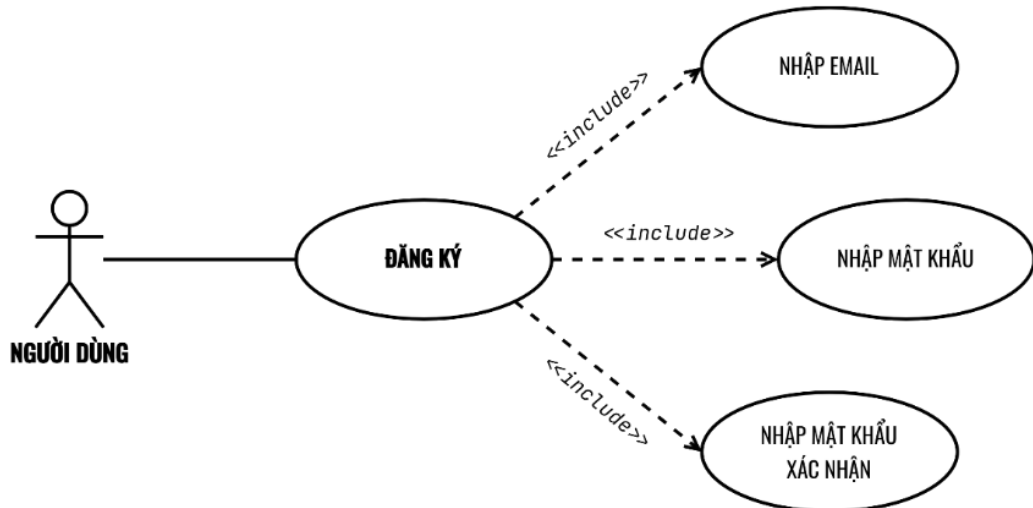
Hình 3.1.1 Sơ đồ usecase hệ thống

- a) **Chức năng về xác thực người dùng** giúp đảm bảo bảo mật hệ thống và quản lý quyền truy cập người dùng an toàn.
- Đăng nhập.
  - Đăng ký.
  - Quên mật khẩu.
  - Đổi mật khẩu.
  - Xóa tài khoản.
  - Đăng xuất.
- b) **Chức năng về quản lý thiết bị** giúp người dùng quản lý các thiết bị giám sát, bao gồm quản lý thông tin và chia sẻ thiết bị.
- Thêm thiết bị.
  - Xóa thiết bị.
  - Cập nhật thông tin thiết bị.
  - Chia sẻ thiết bị.
- c) **Chức năng về quản lý thông tin ghi hình** giúp người dùng theo dõi trực tiếp và tra cứu lịch sử ghi hình dễ dàng.
- Xem thông tin ghi hình thiết bị.
  - Tra cứu lịch sử ghi hình thiết bị.

○ **Đăng nhập**

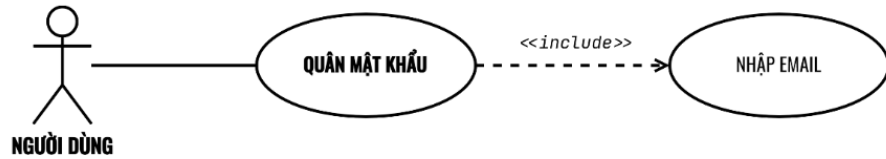
Hình 3.1.2 Sơ đồ usecase đăng nhập

Tên usecase	Đăng nhập
Điều kiện trước	<ul style="list-style-type: none"> <li>▪ Người dùng truy cập giao diện đăng nhập.</li> <li>▪ Người dùng đã có tài khoản hợp lệ.</li> </ul>
Điều kiện tối thiểu	Người dùng chỉ có thể đăng nhập theo phương thức quy định của hệ thống.
Điều kiện sau	<input checked="" type="checkbox"/> Người dùng đăng nhập thành công và chuyển hướng đến giao diện chính. <input checked="" type="checkbox"/> Hệ thống hiển thị thông báo lỗi chi tiết.
<b>Chuỗi sự kiện chính</b> <ol style="list-style-type: none"> <li>1. Người dùng mở ứng dụng di động.</li> <li>2. Hệ thống hiển thị giao diện đăng nhập.</li> <li>3. Người dùng chọn phương thức đăng nhập.</li> <li>4. Người dùng nhập thông tin đăng nhập.</li> <li>5. Hệ thống kiểm tra tính hợp lệ của thông tin.</li> <li>6. Hệ thống xác thực thông tin tài khoản.</li> <li>7. Hệ thống ghi lại hoạt động và cấp phiên làm việc.</li> <li>8. Người dùng được chuyển hướng đến giao diện chính.</li> <li>9. Kết thúc usecase.</li> </ol>	

○ **Đăng ký**

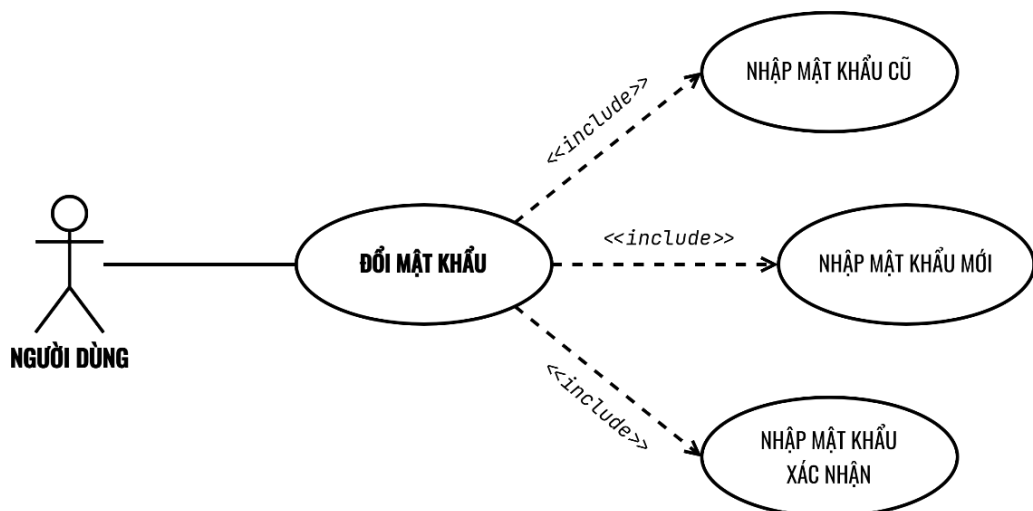
Hình 3.1.3 Sơ đồ usecase đăng ký

Tên usecase	Đăng ký
Điều kiện trước	Người dùng truy cập giao diện đăng nhập theo phương thức xác thực với Email Address.
Điều kiện tối thiểu	Hệ thống tạo tài khoản tương ứng với thông tin đăng ký từ người dùng.
Điều kiện sau	<input checked="" type="checkbox"/> Hệ thống tạo tài khoản mới an toàn và thực thi đăng nhập tự động vào tài khoản đó. <input checked="" type="checkbox"/> Hệ thống hiển thị thông báo lỗi chi tiết.
<p><b>Chuỗi sự kiện chính</b></p> <ol style="list-style-type: none"> <li>1. Người dùng mở ứng dụng di động.</li> <li>2. Hệ thống hiển thị giao diện đăng nhập.</li> <li>3. Người dùng chọn phương thức xác thực với Email Address.</li> <li>4. Người dùng chọn chức năng “Đăng ký tài khoản”.</li> <li>5. Hệ thống hiển thị hộp thoại đăng ký tài khoản.</li> <li>6. Người dùng nhập thông tin đăng ký.</li> <li>7. Hệ thống kiểm tra tính hợp lệ của thông tin.</li> <li>8. Hệ thống xác thực thông tin tài khoản.</li> <li>9. Hệ thống tạo tài khoản mới.</li> <li>10. Hệ thống thực thi đăng nhập tự động vào tài khoản được tạo.</li> <li>11. Hệ thống ghi lại hoạt động và cấp phiên làm việc.</li> <li>12. Người dùng được chuyển hướng đến giao diện chính.</li> <li>13. Kết thúc usecase.</li> </ol>	

○ **Quên mật khẩu**

Hình 3.1.4 Sơ đồ usecase quên mật khẩu

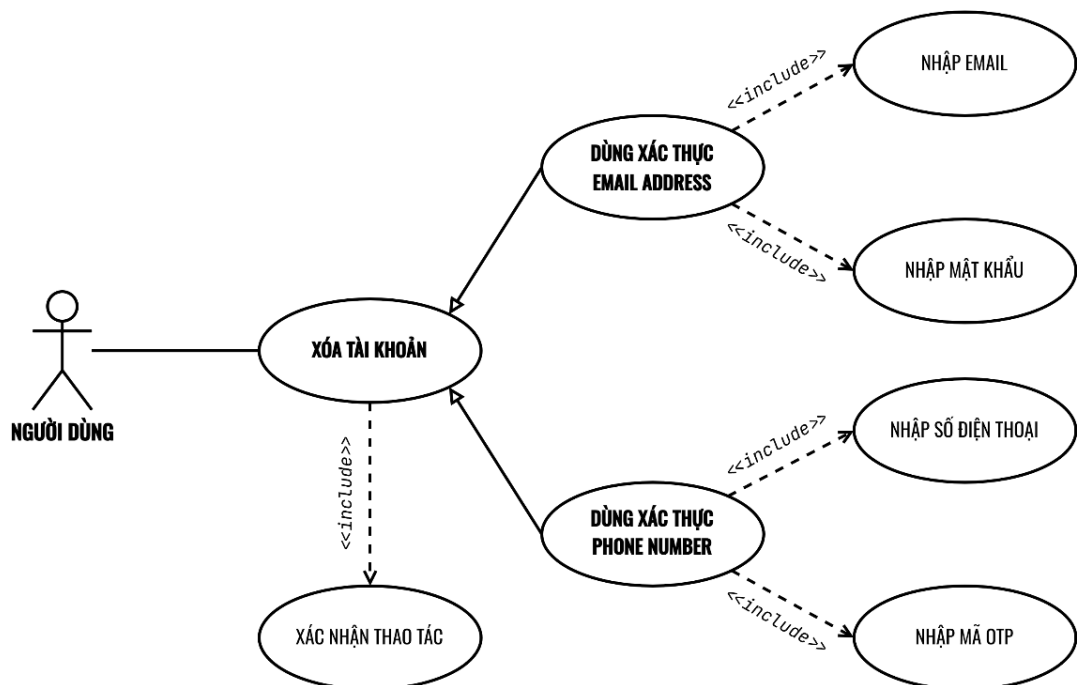
Tên usecase	Quên mật khẩu
Điều kiện trước	Người dùng truy cập giao diện đăng nhập theo phương thức xác thực với Email Address.
Điều kiện tối thiểu	Hệ thống gửi email xác thực đến địa chỉ email chỉ định.
Điều kiện sau	<input checked="" type="checkbox"/> Email xác thực được gửi đến người dùng. <input checked="" type="checkbox"/> Hệ thống hiển thị thông báo lỗi chi tiết.
Chuỗi sự kiện chính <ol style="list-style-type: none"> <li>1. Người dùng truy cập giao diện đăng nhập.</li> <li>2. Người dùng chọn phương thức xác thực với Email Address.</li> <li>3. Người dùng chọn chức năng “Quên mật khẩu”.</li> <li>4. Hệ thống hiển thị hộp thoại quên mật khẩu.</li> <li>5. Người dùng nhập thông tin quên mật khẩu.</li> <li>6. Hệ thống kiểm tra tính hợp lệ của thông tin.</li> <li>7. Hệ thống xác thực thông tin tài khoản.</li> <li>8. Hệ thống gửi email xác thực đến địa chỉ email chỉ định.</li> <li>9. Kết thúc usecase.</li> </ol>	

○ **Đổi mật khẩu**

Hình 3.1.5 Sơ đồ usecase đổi mật khẩu

Tên usecase	Đổi mật khẩu
Điều kiện trước	Người dùng đã đăng nhập vào hệ thống.
Điều kiện tối thiểu	Hệ thống cập nhật tài khoản người dùng an toàn.
Điều kiện sau	<input checked="" type="checkbox"/> Tài khoản được cập nhật mật khẩu an toàn và thông báo thực thi thành công. <input checked="" type="checkbox"/> Hệ thống hiển thị thông báo lỗi chi tiết.
<p><b>Chuỗi sự kiện chính</b></p> <ol style="list-style-type: none"> <li>1. Người dùng truy cập giao diện quản lý tài khoản.</li> <li>2. Người dùng chọn chức năng “Đổi mật khẩu”.</li> <li>3. Hệ thống hiển thị hộp thoại đổi mật khẩu.</li> <li>4. Người dùng nhập thông tin đổi mật khẩu.</li> <li>5. Hệ thống kiểm tra tính hợp lệ của thông tin.</li> <li>6. Hệ thống xác thực thông tin tài khoản.</li> <li>7. Hệ thống cập nhật thông tin mật khẩu trong cơ sở dữ liệu và hiển thị thông báo thành công.</li> <li>8. Kết thúc usecase.</li> </ol>	

○ **Xóa tài khoản**

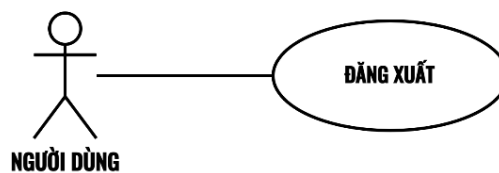


Hình 3.1.6 Sơ đồ usecase xóa tài khoản

Tên usecase	Xóa tài khoản
Điều kiện trước	Người dùng đã đăng nhập vào hệ thống.

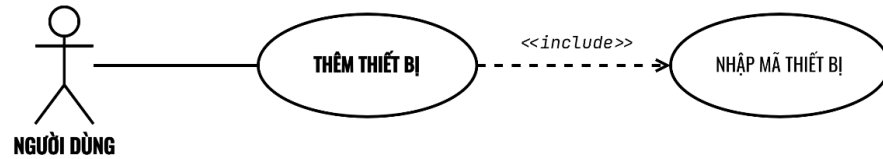
Điều kiện tối thiểu	Thông tin tài khoản người dùng được xóa khỏi hệ thống.
Điều kiện sau	<input checked="" type="checkbox"/> Tài khoản được xóa khỏi hệ thống an toàn; Hệ thống thông báo thực thi thành công. <input checked="" type="checkbox"/> Hệ thống hiển thị thông báo lỗi chi tiết.
<b>Chuỗi sự kiện chính</b> <ol style="list-style-type: none"> <li>1. Người dùng truy cập giao diện quản lý tài khoản.</li> <li>2. Người dùng chọn chức năng “Xóa tài khoản”.</li> <li>3. Hệ thống hiển thị hộp thoại xóa tài khoản theo phương xác thực đăng ký.</li> <li>4. Người dùng nhập thông tin xóa tài khoản.</li> <li>5. Hệ thống kiểm tra tính hợp lệ của thông tin.</li> <li>6. Hệ thống xác thực thông tin tài khoản.</li> <li>7. Hệ thống xóa tài khoản người dùng.</li> <li>8. Hệ thống hủy bỏ phiên làm việc của người dùng.</li> <li>9. Người dùng được chuyển hướng đến giao diện đăng nhập.</li> <li>10. Kết thúc usecase.</li> </ol>	

○ **Đăng xuất**



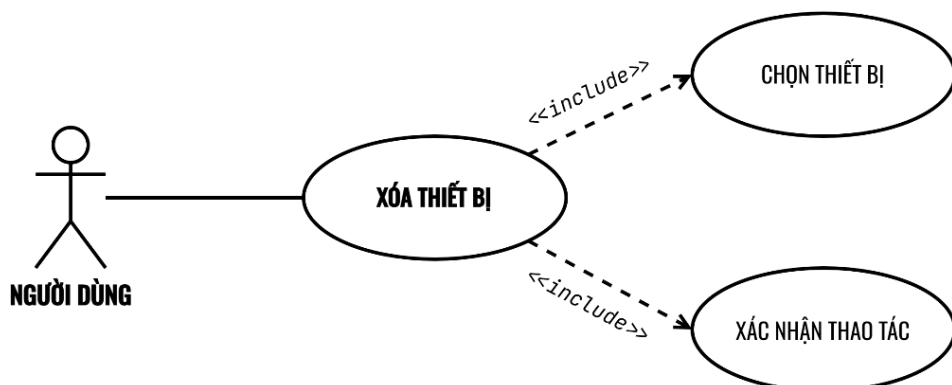
Hình 3.1.7 Sơ đồ usecase đăng xuất

Tên usecase	Đăng xuất
Điều kiện trước	Người dùng đã đăng nhập vào hệ thống.
Điều kiện tối thiểu	Người dùng được đăng xuất an toàn.
Điều kiện sau	Người dùng chuyển hướng đến giao diện đăng nhập.
<b>Chuỗi sự kiện chính</b> <ol style="list-style-type: none"> <li>1. Người dùng truy cập giao diện quản lý tài khoản.</li> <li>2. Người dùng chọn chức năng “Đăng xuất”.</li> <li>3. Hệ thống hiển thị giao diện đăng xuất (nếu có).</li> <li>4. Hệ thống tiến hành đăng xuất tài khoản người dùng.</li> <li>5. Hệ thống hủy bỏ phiên làm việc của người dùng.</li> <li>6. Người dùng được chuyển hướng đến giao diện đăng nhập.</li> <li>7. Kết thúc usecase.</li> </ol>	

○ **Thêm thiết bị**

Hình 3.1.8 Sơ đồ usecase thêm thiết bị

Tên usecase	Thêm thiết bị
Điều kiện trước	Người dùng đã đăng nhập vào hệ thống.
Điều kiện tối thiểu	Thông tin thiết bị được lưu trữ vào hệ thống.
Điều kiện sau	<input checked="" type="checkbox"/> Thiết bị được liên kết với tài khoản người dùng an toàn; Hệ thống thông báo thực thi thành công. <input checked="" type="checkbox"/> Hệ thống hiển thị thông báo lỗi chi tiết.
Chuỗi sự kiện chính <ol style="list-style-type: none"> <li>1. Người dùng truy cập giao diện quản lý thiết bị.</li> <li>2. Người dùng chọn chức năng “Thêm thiết bị”.</li> <li>3. Hệ thống hiển thị hộp thoại thêm thiết bị.</li> <li>4. Người dùng nhập thông tin thêm thiết bị.</li> <li>5. Hệ thống kiểm tra tính hợp lệ của thông tin.</li> <li>6. Hệ thống xác thực thông tin thiết bị.</li> <li>7. Thiết bị được liên kết với tài khoản người dùng.</li> <li>8. Hệ thống cập nhật lại giao diện.</li> <li>9. Kết thúc usecase.</li> </ol>	

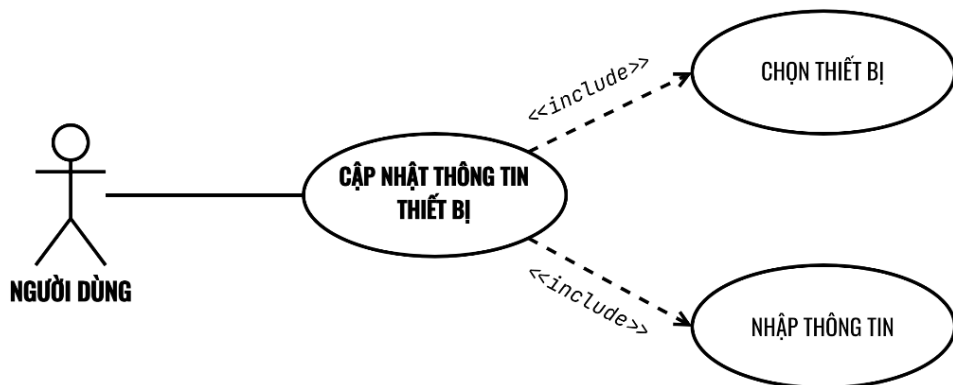
○ **Xóa thiết bị**

Hình 3.1.9 Sơ đồ usecase xóa thiết bị

Tên usecase	Xóa thiết bị
Điều kiện trước	<ul style="list-style-type: none"> <li>▪ Người dùng đã đăng nhập vào hệ thống.</li> </ul>

	<ul style="list-style-type: none"> <li>Thiết bị được liên kết với tài khoản người dùng.</li> </ul>
Điều kiện tối thiểu	Thiết bị được hủy liên kết với tài khoản người dùng.
Điều kiện sau	<ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Thiết bị được hủy liên kết với tài khoản người dùng an toàn; Hệ thống thông báo thực thi thành công.</li> <li><input checked="" type="checkbox"/> Hệ thống hiển thị thông báo lỗi chi tiết.</li> </ul>
<p><b>Chuỗi sự kiện chính</b></p> <ol style="list-style-type: none"> <li>Người dùng truy cập giao diện quản lý thiết bị.</li> <li>Người dùng chọn thiết bị.</li> <li>Người dùng chọn chức năng “Xóa thiết bị”.</li> <li>Hệ thống hiển thị hộp thoại xác nhận thao tác.</li> <li>Người dùng xác nhận thao tác.</li> <li>Thiết bị được liên kết với tài khoản người dùng.</li> <li>Hệ thống cập nhật lại giao diện.</li> <li>Kết thúc usecase.</li> </ol>	

○ **Cập nhật thông tin thiết bị**



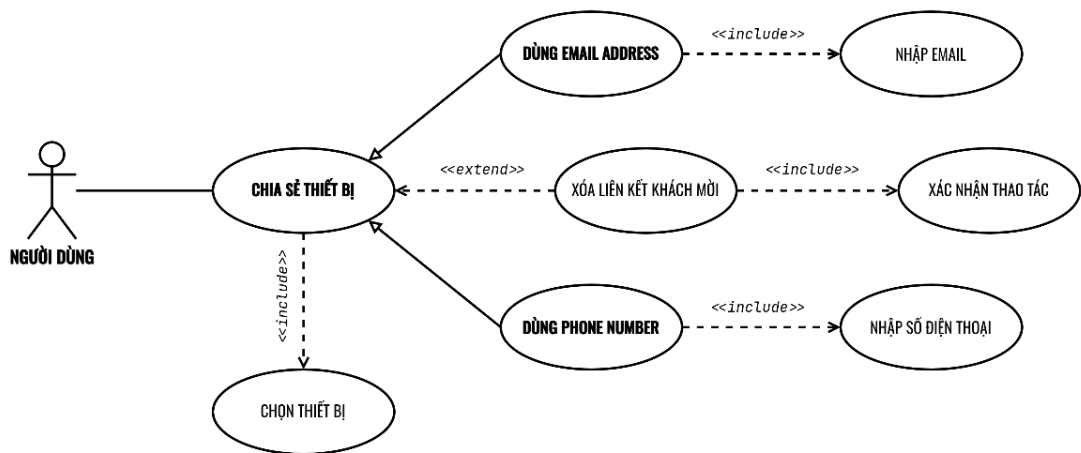
Hình 3.1.10 Sơ đồ usecase cập nhật thông tin thiết bị

Tên usecase	Cập nhật thông tin thiết bị
Điều kiện trước	<ul style="list-style-type: none"> <li>Người dùng đã đăng nhập vào hệ thống.</li> <li>Thiết bị được liên kết với tài khoản người dùng.</li> </ul>
Điều kiện tối thiểu	<ul style="list-style-type: none"> <li>Thông tin thiết bị được cập nhật lên hệ thống.</li> <li>Hệ thống chỉ cho phép tài khoản thêm thiết bị thực hiện cập nhật thông tin thiết bị.</li> </ul>
Điều kiện sau	<ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Thiết bị được cập nhật thông tin an toàn; Hệ thống thông báo thực thi thành công.</li> <li><input checked="" type="checkbox"/> Hệ thống hiển thị thông báo lỗi chi tiết.</li> </ul>



*Chuỗi sự kiện chính*

1. Người dùng truy cập giao diện quản lý thiết bị.
2. Người dùng chọn thiết bị.
3. Người dùng chọn chức năng “Cập nhật thông tin thiết bị”.
4. Hệ thống hiển thị hộp thoại cập nhật thông tin.
5. Người dùng nhập thông tin cập nhật.
6. Thiết bị được cập nhật thông tin.
7. Hệ thống cập nhật lại giao diện.
8. Kết thúc usecase.

○ **Chia sẻ thiết bị**

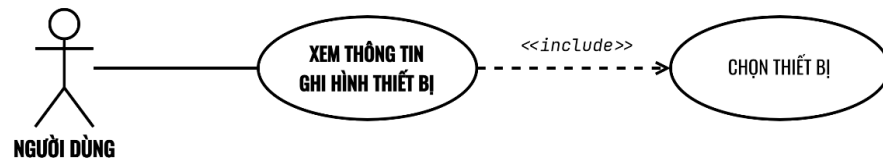
Hình 3.1.11 Sơ đồ usecase chia sẻ thiết bị

Tên usecase	Chia sẻ thiết bị
Điều kiện trước	<ul style="list-style-type: none"> <li>▪ Người dùng đã đăng nhập vào hệ thống.</li> <li>▪ Thiết bị được liên kết với tài khoản người dùng.</li> </ul>
Điều kiện tối thiểu	<ul style="list-style-type: none"> <li>▪ Người dùng chỉ có thể chia sẻ thiết bị theo phương thức quy định của hệ thống.</li> <li>▪ Hệ thống chỉ cho phép tài khoản thêm thiết bị thực hiện xóa liên kết của tài khoản được chia sẻ.</li> </ul>
Điều kiện sau	Hệ thống gửi lời mời đến tài khoản chỉ định; Hệ thống xóa liên kết với tài khoản chỉ định.
<i>Chuỗi sự kiện chính</i> <ol style="list-style-type: none"> <li>1. Người dùng truy cập giao diện quản lý thiết bị.</li> <li>2. Người dùng chọn thiết bị.</li> <li>3. Người dùng chọn chức năng “Chia sẻ thiết bị”.</li> <li>4. Người dùng thực hiện chức năng chỉ định.</li> </ol>	

- Người dùng thực hiện chia sẻ thiết bị.
  - Người dùng chọn phương thức xác thực.
  - Người dùng nhập thông tin tài khoản.
  - Hệ thống kiểm tra tính hợp lệ của thông tin.
  - Hệ thống xác thực thông tin thiết bị và tài khoản.
  - Hệ thống gửi lời mời đến tài khoản chỉ định.
- Người dùng thực hiện hủy chia sẻ thiết bị.
  - Người dùng chọn tài khoản liên kết.
  - Người dùng chọn chức năng “Hủy chia sẻ thiết bị”.
  - Hệ thống hiển thị hộp thoại xác nhận thao tác.
  - Người dùng xác nhận thao tác.
  - Hệ thống xác thực thông tin thiết bị và tài khoản chỉ định.
  - Hệ thống hủy liên kết giữa thiết bị và tài khoản chỉ định.
  - Hệ thống cập nhật lại giao diện.

##### 5. Kết thúc usecase.

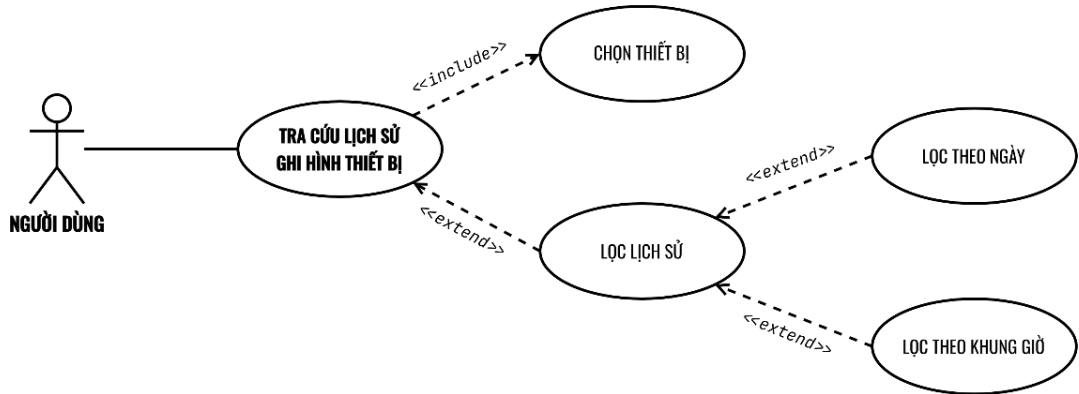
##### ○ Xem thông tin ghi hình thiết bị



Hình 3.1.12 Sơ đồ usecase xóa xem thông tin ghi hình thiết bị

Tên usecase	Xem thông tin ghi hình thiết bị
Điều kiện trước	<ul style="list-style-type: none"> <li>▪ Người dùng đã đăng nhập vào hệ thống.</li> <li>▪ Thiết bị được liên kết với tài khoản người dùng.</li> </ul>
Điều kiện tối thiểu	Hệ thống hiển thị thông tin ghi hình thiết bị.
Điều kiện sau	Hệ thống hiển thị thông tin ghi hình thiết bị chính xác.
Chuỗi sự kiện chính <ol style="list-style-type: none"> <li>1. Người dùng truy cập giao diện quản lý thiết bị.</li> <li>2. Người dùng chọn thiết bị.</li> <li>3. Người dùng truy cập giao diện thông tin ghi hình thiết bị.</li> <li>4. Hệ thống hiển thị thông tin ghi hình thiết bị.</li> <li>5. Hệ thống cập nhật thông tin ghi hình mỗi khi có thay đổi.</li> <li>6. Kết thúc usecase.</li> </ol>	

○ *Tra cứu lịch sử ghi hình thiết bị*

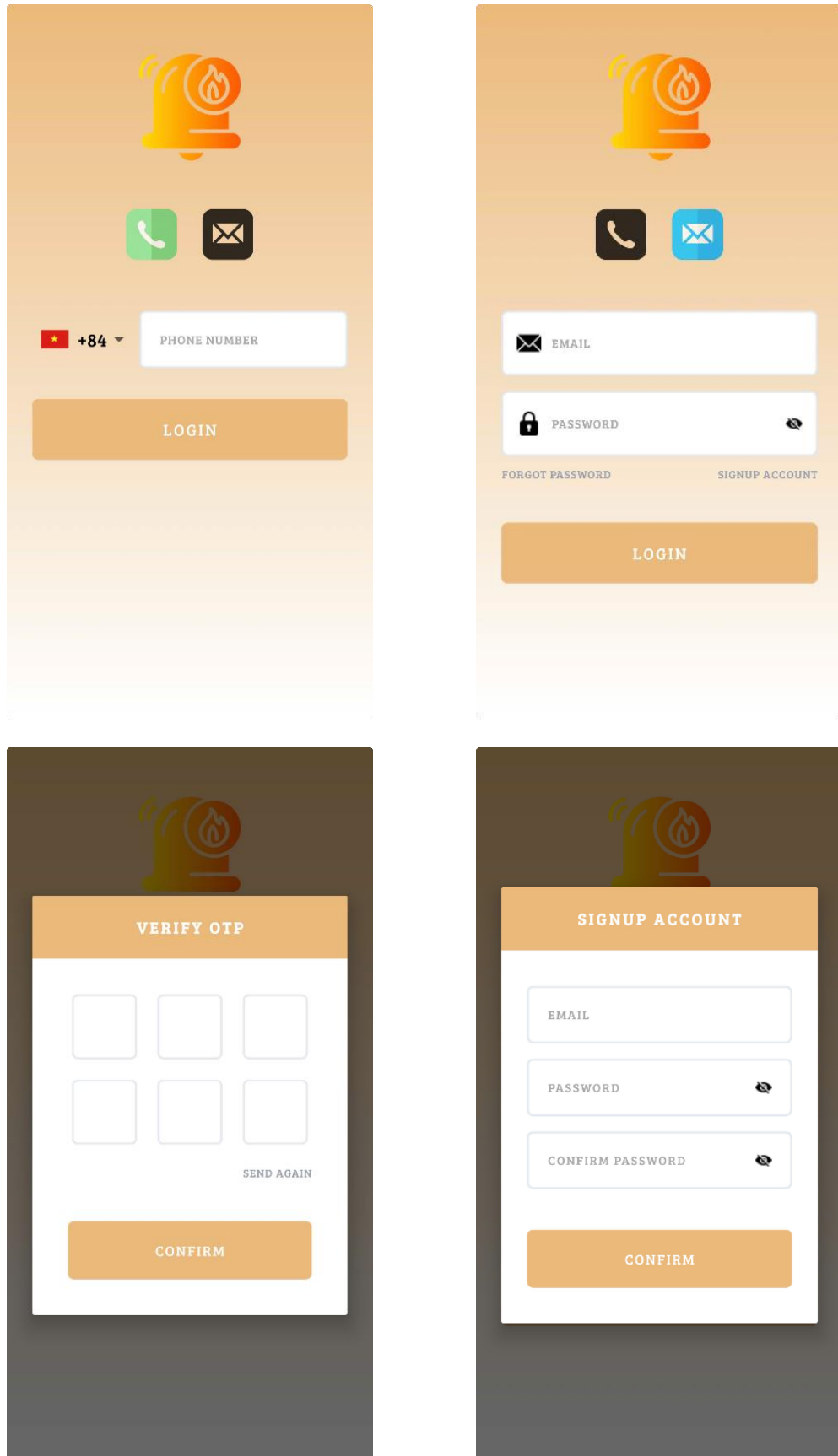


Hình 3.1.13 Sơ đồ usecase xóa tra cứu lịch sử ghi hình thiết bị

Tên usecase	Tra cứu lịch sử ghi hình thiết bị
Điều kiện trước	<ul style="list-style-type: none"> <li>▪ Người dùng đã đăng nhập vào hệ thống.</li> <li>▪ Thiết bị được liên kết với tài khoản người dùng.</li> </ul>
Điều kiện tối thiểu	Hệ thống hiển thị lịch sử ghi hình thiết bị theo điều kiện lọc chỉ định.
Điều kiện sau	Hệ thống hiển thị lịch sử ghi hình thiết bị chính xác.
<p><i>Chuỗi sự kiện chính</i></p> <ol style="list-style-type: none"> <li>1. Người dùng truy cập giao diện quản lý thiết bị.</li> <li>2. Người dùng chọn thiết bị.</li> <li>3. Người dùng truy cập giao diện lịch sử ghi hình thiết bị.</li> <li>4. Hệ thống hiển thị lịch sử ghi hình thiết bị.</li> <li>5. Người dùng chọn chức năng “Lọc lịch sử”.               <ul style="list-style-type: none"> <li>○ Người dùng thực hiện lọc theo ngày.                   <ul style="list-style-type: none"> <li>▪ Hệ thống hiển thị hộp thoại chọn ngày.</li> <li>▪ Người dùng chọn ngày.</li> </ul> </li> <li>○ Người dùng thực hiện lọc theo khung giờ.                   <ul style="list-style-type: none"> <li>▪ Hệ thống hiển thị hộp thoại chọn giờ.</li> <li>▪ Người dùng chọn giờ.</li> </ul> </li> </ul> </li> <li>6. Hệ thống cập nhật lại giao diện.</li> <li>7. Kết thúc usecase.</li> </ol>	

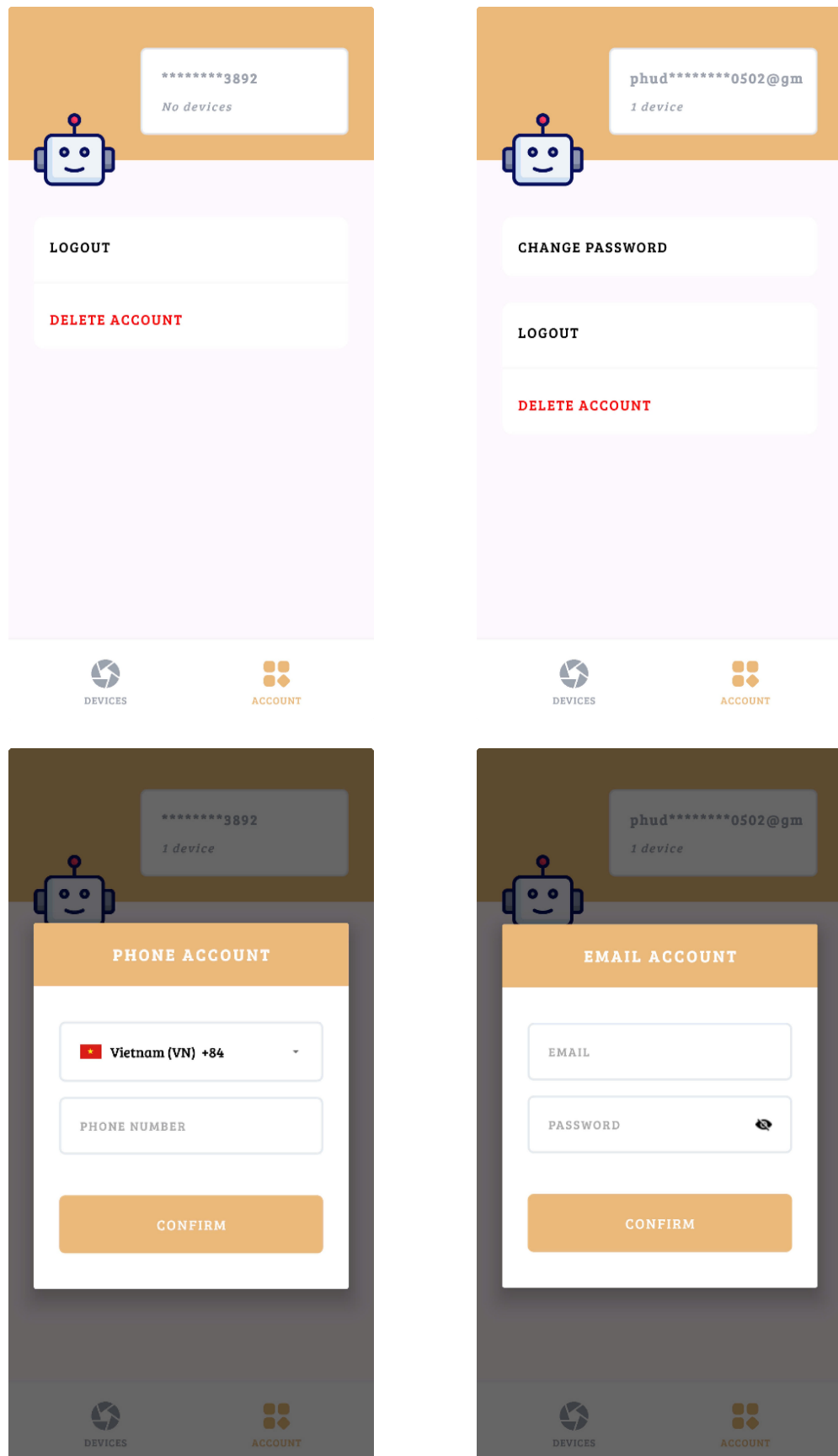
## 2. THIẾT KẾ GIAO DIỆN

### 1) Xác thực người dùng



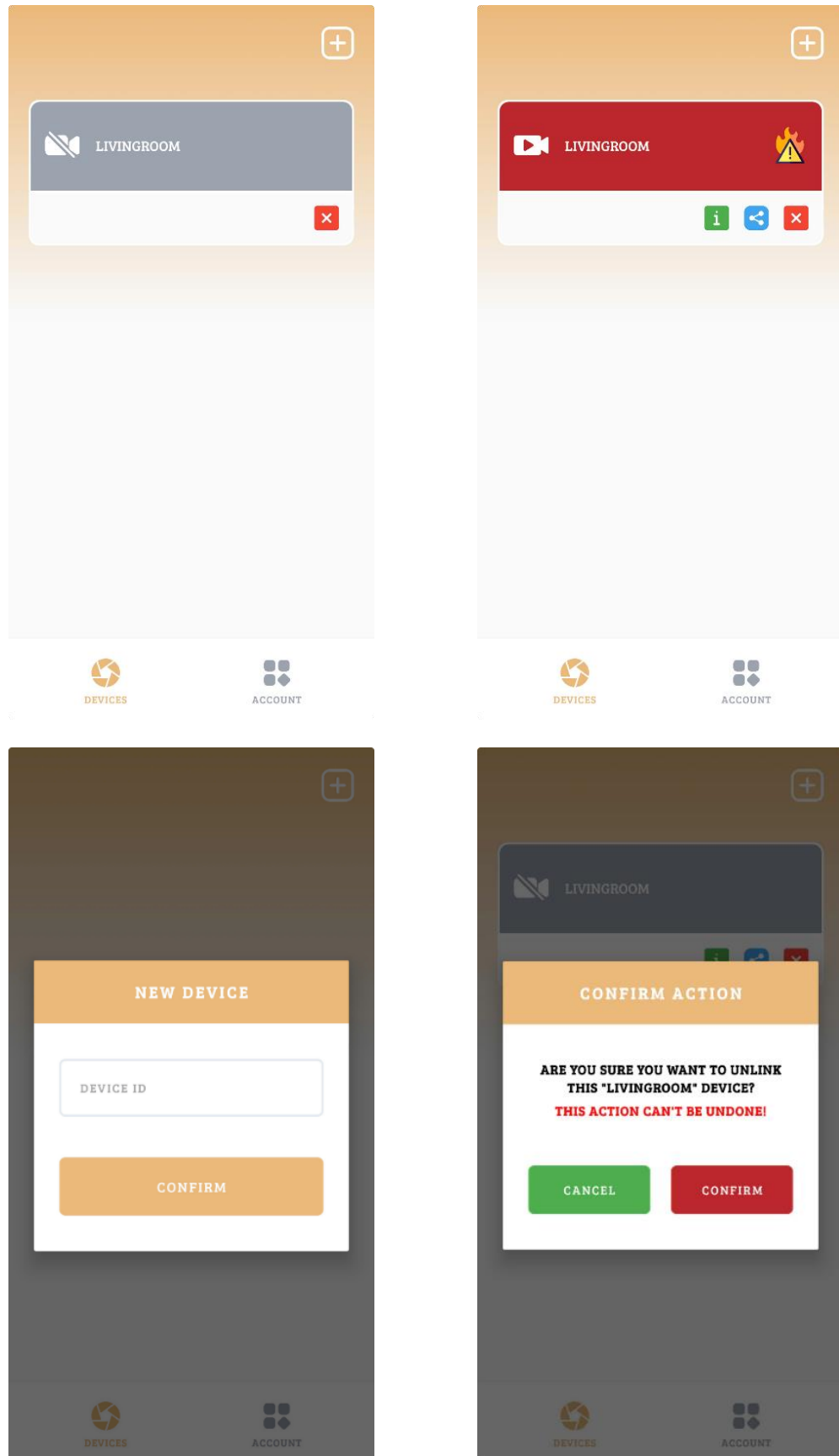
Hình 3.2.1 Hình ảnh minh họa giao diện xác thực tài khoản

## 2) Quản lý tài khoản



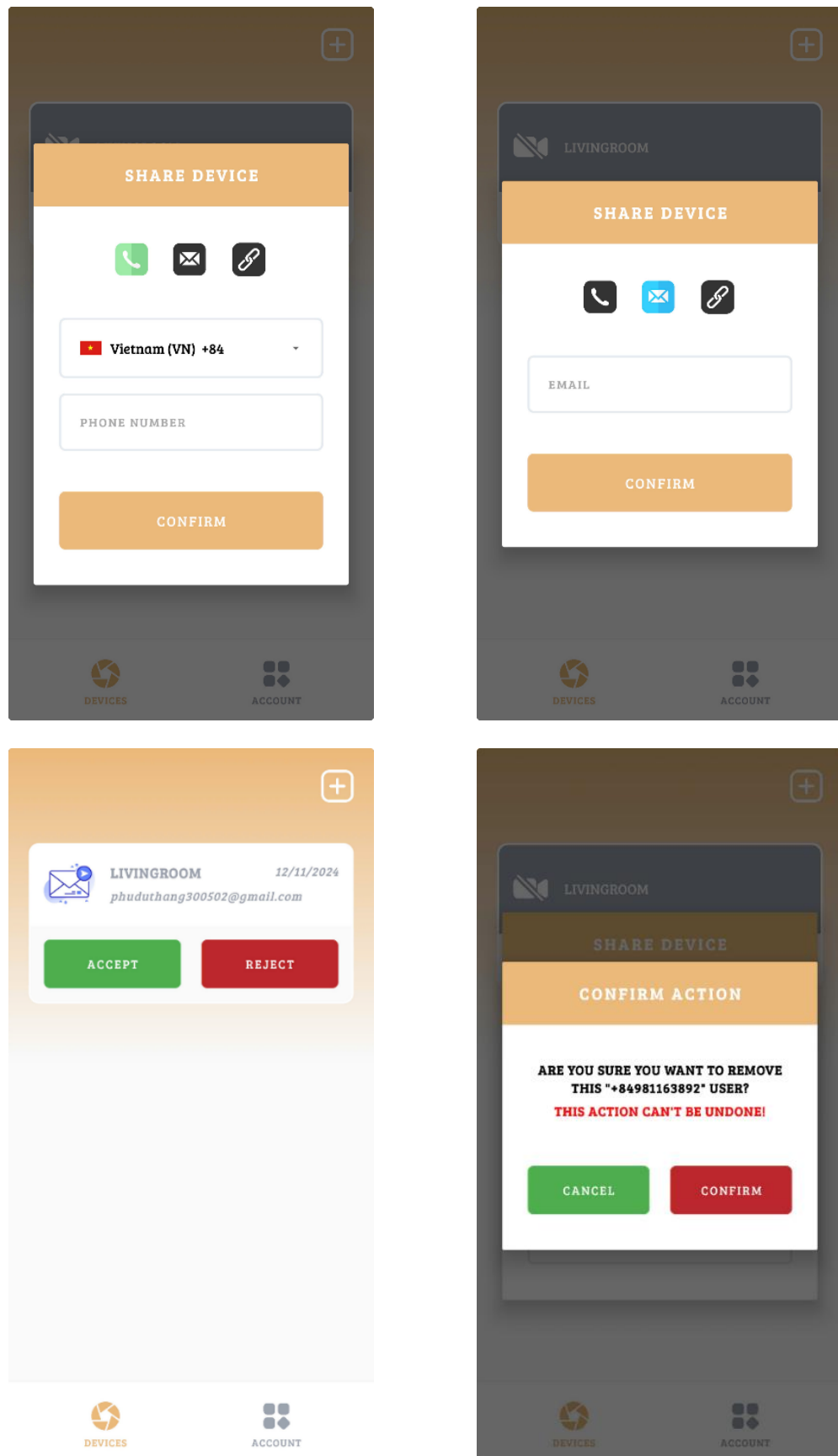
Hình 3.2.2 Hình ảnh minh họa giao diện quản lý tài khoản

### 3) Quản lý thiết bị

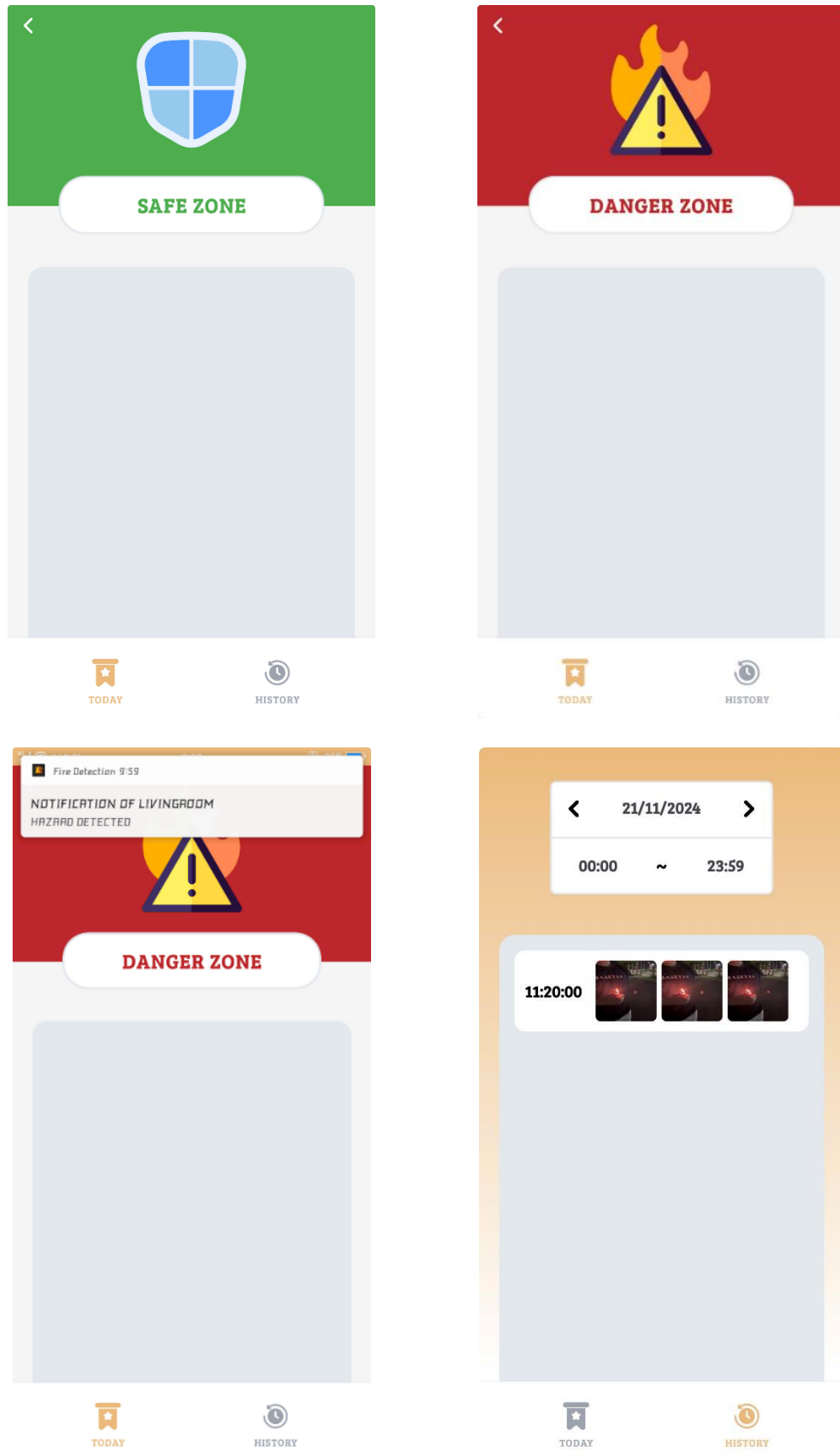


Hình 3.2.3 Hình ảnh minh họa giao diện quản lý thiết bị

#### 4) Chia sẻ thiết bị



Hình 3.2.4 Hình ảnh minh họa giao diện chia sẻ thiết bị

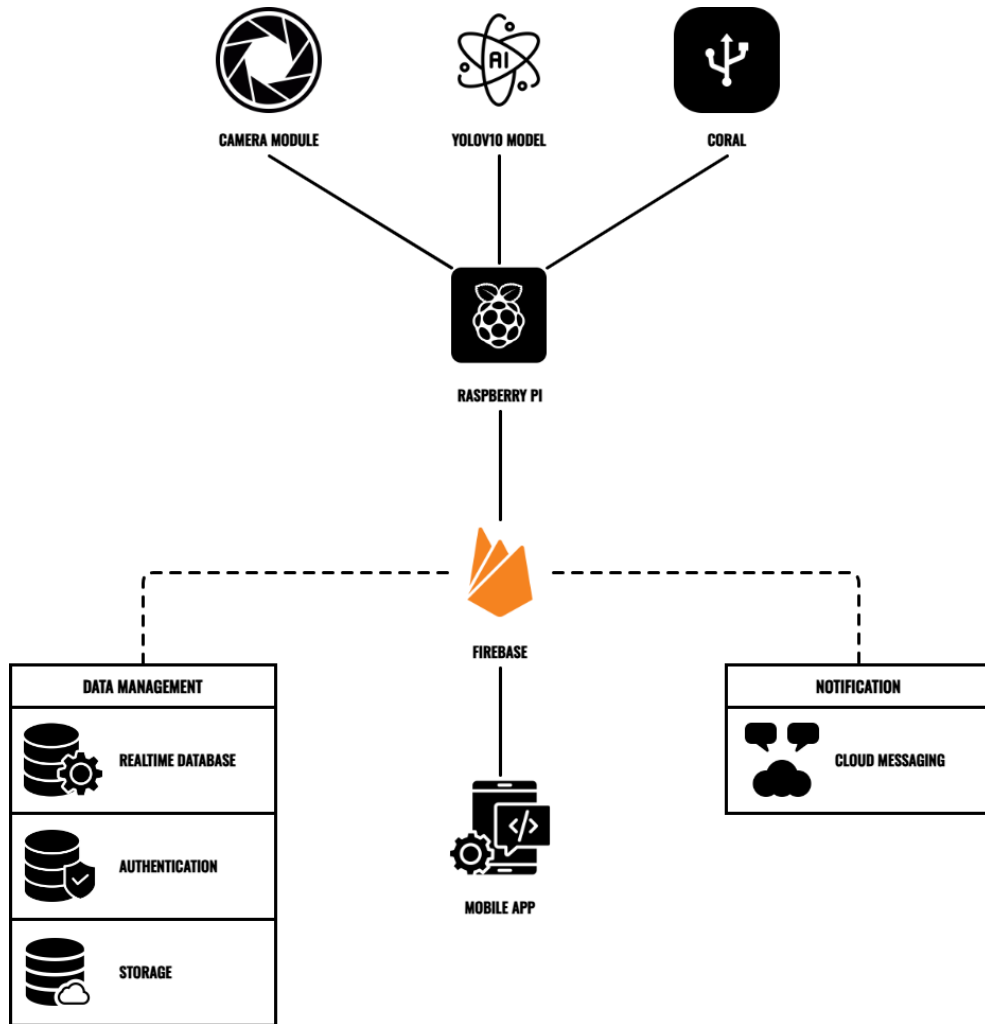
**5) Thông tin ghi hình thiết bị**

Hình 3.2.5 Hình ảnh minh họa giao diện thông tin ghi hình thiết bị



## CHƯƠNG 4. TRIỂN KHAI THỰC TIỄN

### 1. THIẾT KẾ HỆ THỐNG



Hình 4.2.1 Workflow thiết kế hệ thống

Hệ thống phát hiện cháy thời gian thực được phát triển dựa trên mô hình học sâu *YOLOv10*, triển khai trên nền tảng nhúng với *Raspberry Pi* là thiết bị xử lý trung tâm và *Coral USB Accelerator* hỗ trợ tăng tốc độ xử lý cho mô hình. Bên cạnh đó, *Camera Module* thu nhận hình ảnh và truyền tải đến *Raspberry Pi*, nơi dữ liệu được phân tích và các tác vụ hệ thống được thực thi.

Khi phát hiện sự hiện diện của đám cháy, *Raspberry Pi* sẽ gửi dữ liệu sự kiện đến *Firebase*, nơi lưu trữ và quản lý thông tin hệ thống. Đồng thời, nó cũng gửi một thông báo tức thì đến ứng dụng di động của người dùng thông qua dịch vụ *Firebase Cloud Messaging*, giúp họ nhanh chóng tiếp cận thông tin cảnh báo, từ đó kịp thời đưa ra các quyết định cần thiết.

Tổng quan, hệ thống phát hiện cháy thời gian thực này tận dụng sức mạnh của những công nghệ hiện đại như *YOLO*, *Firebase*, *Raspberry Pi* và *Coral* để nâng cao hiệu quả trong việc phát hiện và cảnh báo cháy. Nhờ đó, hệ thống đảm bảo thông tin sẽ được truyền tải kịp thời đến người dùng, góp phần cải thiện khả năng quản lý các tình huống khẩn cấp và bảo vệ an toàn cho các khu vực trọng yếu.

## 1.1. CHI TIẾT THÀNH PHẦN

### ○ CAMERA MODULE

*Camera Module* là thiết bị ghi hình thời gian thực, được thiết kế để hoạt động tối ưu với các dòng thiết bị *Raspberry Pi*. Kết nối trực tiếp với *Raspberry Pi* thông qua cổng *CSI*, *Camera Module* truyền tải dữ liệu hình ảnh một cách nhanh chóng và liên tục, giúp hệ thống có thể phát hiện các sự kiện kịp thời và đưa ra cảnh báo ngay khi cần thiết.

### ○ YOLOV10 MODEL

*YOLOv10* là mô hình phát hiện đối tượng dựa trên học sâu, được tối ưu hóa để nhận diện các dấu hiệu của đám cháy, bao gồm lửa và khói. Kiến trúc của *YOLO* cho phép mô hình xử lý dữ liệu nhanh chóng, đồng thời vẫn có thể cải thiện độ chính xác một cách hiệu quả. Trong hệ thống, *YOLOv10* sẽ phân tích hình ảnh đầu vào để phát hiện các dấu hiệu cháy, hỗ trợ cho các tác vụ tiếp theo trong quy trình cảnh báo cháy.

### ○ CORAL

*Coral USB Accelerator* là thiết bị tăng tốc phần cứng, hỗ trợ xử lý hiệu quả các tác vụ học sâu, giúp giảm thiểu tài nguyên cần dùng khi thực thi mô hình *YOLOv10* trên *Raspberry Pi*. Thiết bị này cung cấp khả năng xử lý dữ liệu hình ảnh nhanh chóng, tối ưu hóa thời gian suy luận của mô hình để đảm bảo các cảnh báo cháy được phát hiện và xử lý kịp thời.

### ○ RASPBERRY PI

*Raspberry Pi* là thiết bị xử lý trung tâm của hệ thống, nhận dữ liệu đầu vào từ *Camera Module* và xử lý đầu ra với mô hình *YOLOv10* cùng sự hỗ trợ của *Coral*. Khi phát hiện nguy cơ cháy, *Raspberry Pi* sẽ gửi dữ liệu sự kiện cùng cảnh báo đến *Mobile App* thông qua *Firebase*, và kích hoạt hệ thống báo động khi cần thiết.

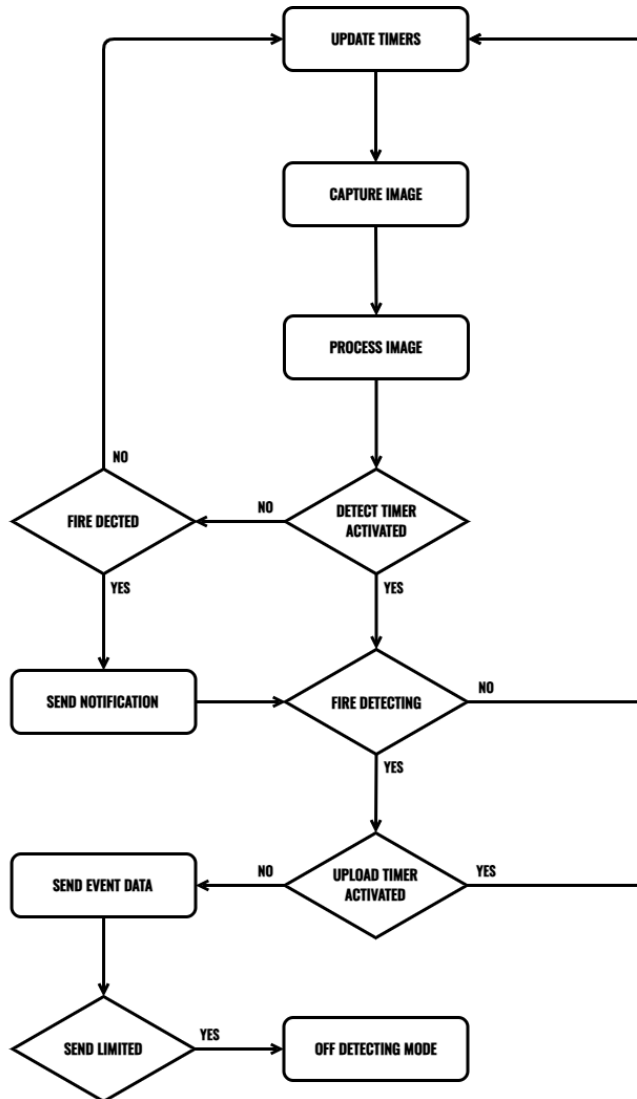
### ○ FIREBASE

*Firebase* là nền tảng phát triển ứng dụng do *Google* cung cấp, đóng vai trò là kho lưu trữ dữ liệu đám mây và hạ tầng cho hệ thống. Khi phát hiện cháy, thông tin sự kiện sẽ được lưu trữ và quản lý trên các dịch vụ của *Firebase*. Nhờ đó, *Firebase* đảm bảo dữ liệu luôn được cập nhật liên tục và hỗ trợ giao tiếp hiệu quả giữa *Raspberry Pi* và *Mobile App*.

### ○ MOBILE APP

*Mobile App* là giao diện được phát triển cho người dùng, giúp hiển thị dữ liệu về các sự kiện được ghi nhận một cách trực quan. Người dùng có thể nhận thông báo tức thì, quản lý thiết bị và tra cứu lịch sử cảnh báo. Ứng dụng này cho phép người dùng giám sát tình hình từ xa và thực hiện các hành động kịp thời khi cần thiết.

## 1.2. QUY TRÌNH HOẠT ĐỘNG



Hình 4.2.2 Quy trình hoạt động của hệ thống

- **UPDATE TIMERS:** Hệ thống cập nhật các bộ đếm thời gian để giãn cách giữa các lần thực thi tác vụ, tránh thực thi liên tục.
- **DETECT/UPLOAD TIMER ACTIVATED & FIRE DETECTED/DETECTING:** Những trạng thái cục bộ để kiểm soát các tác vụ, tránh thực thi liên tục.
- **CAPTURE IMAGE:** Hệ thống thực hiện trích xuất hình ảnh từ thiết bị ghi hình *Camera Module* để tạo dữ liệu đầu vào.
- **PROCESS IMAGE:** Hệ thống thực hiện nạp dữ liệu thu được vào mô hình học sâu *YOLOv10* để nhận diện các dấu hiệu cháy.
- **SEND NOTIFICATION:** Hệ thống lập tức gửi cảnh báo đến ứng dụng di động người dùng khi dấu hiệu cháy được xác nhận.
- **SEND EVENT DATA:** Hệ thống liên tục truyền dữ liệu sự kiện ghi nhận được lên *Firebase* để lưu trữ khi dấu hiệu cháy được xác nhận.
- **OFF DETECTING MODE:** Hệ thống tắt chế độ “nhận diện” khi lượng dữ liệu gửi đi đạt mức chỉ định để tiết kiệm tài nguyên hệ thống.

## 2. CÔNG NGHỆ SỬ DỤNG

Tổng quan, sự hỗ trợ linh hoạt và mạnh mẽ của các công nghệ đã tạo điều kiện thuận lợi cho việc phát triển và triển khai hệ thống. Các thư viện không chỉ cung cấp những công cụ thiết yếu để phân tích và xử lý tác vụ hiệu quả mà còn giúp nâng cao tính liên kết giữa các phần ứng dụng trong hệ thống. Sự kết hợp này cho phép phát triển các giải pháp tối ưu và đáp ứng linh hoạt yêu cầu thực tế. Qua đó, đề tài không chỉ đạt được mục tiêu kỹ thuật mà còn mở ra hướng đi mới cho các ứng dụng trong lĩnh vực thị giác máy tính và nhúng, mang lại giá trị thiết thực cho người dùng.

### 2.1. PHẦN MÔ HÌNH YOLOV10 VÀ THIẾT BỊ NHÚNG

#### 1) Ngôn ngữ lập trình Python

*Python* là một ngôn ngữ lập trình bậc cao, ra đời vào năm 1991 tại Hà Lan do *Guido van Rossum* phát triển. Ngôn ngữ này nổi bật với thiết kế ưu việt: dễ đọc, dễ học và dễ nhớ. Cú pháp sáng sủa, cấu trúc rõ ràng, cùng tính linh hoạt giúp *Python* trở thành lựa chọn lý tưởng cho việc phát triển và bảo trì phần mềm.

##### Lý do lựa chọn:

- Ngôn ngữ lập trình cho phần mô hình *YOLOv10* và thiết bị nhúng.
- Cung cấp nhiều thư viện cho lập trình nhúng.
- Giúp lập trình trên *Raspberry Pi*, *Camera Module* và *Coral*.

#### 2) Thư viện *Firebase Admin*

*Firebase Admin* là một thư viện mã nguồn mở mạnh mẽ do *Google* phát triển, hỗ trợ *Python* cùng nhiều ngôn ngữ lập trình khác. Thư viện này giúp dễ dàng tích hợp và quản lý các dịch vụ *Firebase*, đồng thời cung cấp nhiều tính năng quan trọng cho hạ tầng ứng dụng, và là lựa chọn phổ biến cho các hệ thống máy chủ, từ ứng dụng di động đến *web* và thiết bị nhúng.

##### Lý do lựa chọn:

- Hỗ trợ trên ngôn ngữ *Python*.
- Được nền tảng *Firebase* hỗ trợ.
- Giúp tương tác với các dịch vụ *Firebase*.

#### 3) Thư viện *Ultralytics*

*Ultralytics* là một thư viện mã nguồn mở dựa trên *Python*, nổi tiếng với mô hình học sâu *YOLO* tiên tiến, chuyên biệt cho việc phát hiện và nhận diện đối tượng trong hình ảnh. Với thiết kế dễ sử dụng và hiệu quả cao, *Ultralytics* cung cấp nhiều công cụ mạnh mẽ để huấn luyện, đánh giá và triển khai mô hình *YOLO*. Qua đó, thư viện này được ứng dụng rộng rãi trong đa dạng lĩnh vực, bao gồm an ninh, y tế, công nghiệp và nhiều ngành khác.

##### Lý do lựa chọn:

- Hỗ trợ trên ngôn ngữ *Python*.
- Được nền tảng *Roboflow* hỗ trợ.
- Giúp huấn luyện và triển khai mô hình *YOLOv10*.

#### 4) Thư viện *PiCamera2*

*PiCamera2* là thư viện mã nguồn mở mạnh mẽ hỗ trợ cho *Python*, được phát triển bởi *Raspberry Pi* để tương tác với các camera tương thích với dòng thiết bị *Raspberry Pi* một cách linh hoạt và hiệu quả. Là phiên bản nâng cấp của thư viện *PiCamera* trước đó, *PiCamera2* được cải tiến để tận dụng tối đa hiệu năng các dòng *Camera Module* hiện đại, qua đó hỗ trợ hiệu quả cho tác vụ ghi hình thời gian thực.

##### Lý do lựa chọn:

- Hỗ trợ trên ngôn ngữ *Python*.
- Được nền tảng *Raspberry Pi* hỗ trợ.
- Giúp thu thập dữ liệu hình ảnh từ môi trường.

## 2.2. PHẦN ỨNG DỤNG DI ĐỘNG

### 1) Ngôn ngữ lập trình *Java*

*Java* là một ngôn ngữ lập trình bậc cao, được giới thiệu lần đầu vào năm 1995 bởi *Sun Microsystems*. Với triết lý "*Write Once, Run Anywhere*", *Java* cho phép phát triển các ứng dụng hoạt động độc lập với nền tảng phần cứng hoặc hệ điều hành. Nhờ tính ổn định và khả năng mở rộng linh hoạt, *Java* trở thành lựa chọn hàng đầu trong phát triển ứng dụng doanh nghiệp, ứng dụng web, và đặc biệt là ứng dụng di động trên hệ điều hành *Android*.

##### Lý do lựa chọn:

- Ngôn ngữ lập trình cho phần ứng dụng di động.
- Cung cấp nhiều thư viện cho lập trình ứng dụng di động.
- Giúp lập trình ứng dụng di động trên nền tảng *Android*.

### 2) Bộ công cụ phát triển phần mềm *Firebase*

*Firebase SDK* là bộ công cụ phát triển phần mềm do *Google* phát triển, hỗ trợ xây dựng và phát triển ứng dụng di động và web nhanh chóng và hiệu quả. Bộ công cụ này cung cấp nhiều dịch vụ và thư viện mạnh mẽ, giúp tích hợp và quản lý các dịch vụ *Firebase* một cách dễ dàng. Qua đó, người phát triển có thể tận dụng các dịch vụ mạnh mẽ để cải thiện toàn diện về hiệu suất và bảo mật cho ứng dụng.

##### Lý do lựa chọn:

- Hỗ trợ trên ngôn ngữ *Java*.
- Được nền tảng *Firebase* hỗ trợ.
- Giúp tương tác với các dịch vụ *Firebase*.

### 3. TRIỂN KHAI HỆ THỐNG

#### 3.1. HUẤN LUYỆN MÔ HÌNH

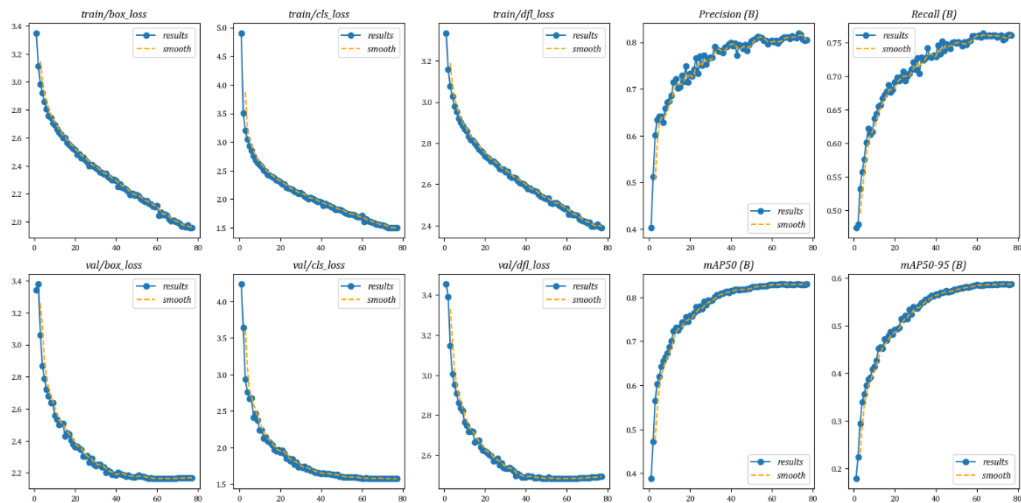
🌐 *Môi trường thực thi*

- Dịch vụ Google Colab.
- Hệ điều hành Linux dùng GPU (Tesla T4, 15102 MB).
- Python 3.10.12 dùng Ultralytics 8.3.39 và PyTorch 2.5.1.

##### a) Thông số huấn luyện

- **MODEL/EPOCH/BATCH** – yolov10s/100/32: Huấn luyện mô hình YOLOv10 phiên bản S trong 100 vòng, mỗi vòng dùng 32 mẫu.
- **PATIENCE** – 5: Kết thúc quá trình huấn luyện sớm nếu kết quả không được cải thiện trong 5 vòng huấn luyện liên tiếp.
- **SAVE PERIOD** – 10: Lưu trạng thái mô hình mỗi 10 vòng huấn luyện để tiếp tục huấn luyện khi không có đủ tài nguyên.
- **OPTIMIZER** – SGD: Huấn luyện sử dụng thuật toán tối ưu *Stochastic Gradient Descent* với các thiết lập cụ thể như sau:
  - **LEARNING RATE** – 0.01: Tốc độ học giúp mô hình học từ từ hoặc nhanh chóng tùy theo giá trị được đặt.
  - **MOMENTUM** – 0.937: Giá trị động lượng giúp giảm thiểu sự dao động và tăng sự ổn định trong quá trình tối ưu.

##### b) Kết quả huấn luyện

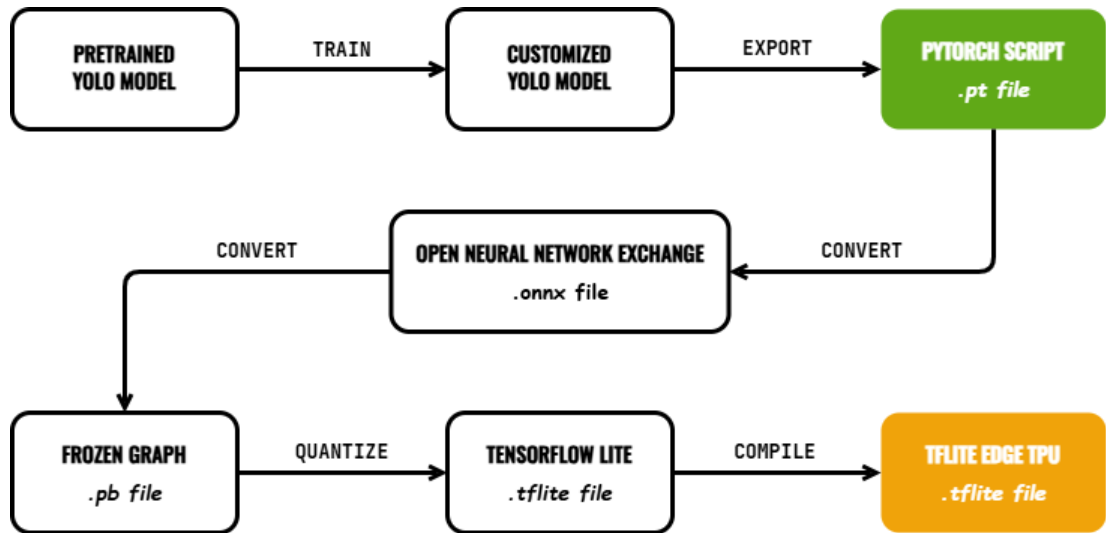


Hình 4.3.1 Biểu đồ tổng hợp kết quả huấn luyện mô hình

- Các giá trị mất mát giảm dần, trong khi các giá trị chính xác tăng dần và ổn định ở cả 2 giai đoạn huấn luyện và kiểm tra. Điều này cho thấy mô hình đã học và tối ưu hóa tốt trên tập dữ liệu.
- Mô hình tăng rất chậm ở giai đoạn sau, dừng sớm quá trình huấn luyện ở vòng 77 và có xu hướng hội tụ ở  $mAP50 \approx 82\%$ . Mô hình đạt được trung bình  $mAP50 \approx 83\%$  và  $mAP50-95 \approx 59\%$ .

### 3.2. CHUYỂN ĐỔI ĐỊNH DẠNG EDGE TPU

Quá trình chuyển đổi mô hình sang định dạng *TensorFlow Edge TPU* giúp tăng tốc và tối ưu hóa các tác vụ học máy. Công nghệ này phù hợp với các ứng dụng có hạn chế về năng lượng, tài nguyên tính toán và kết nối. *Edge TPU* là bộ tăng tốc phần cứng của *Google*, được phát triển nhằm cải thiện hiệu suất của các mô hình *TensorFlow Lite* trên các dòng thiết bị *Edge*. Hình ảnh minh họa dưới đây trình bày quy trình thực hiện điều này.



Hình 4.3.2 Quy trình chuyển đổi định dạng Edge TPU

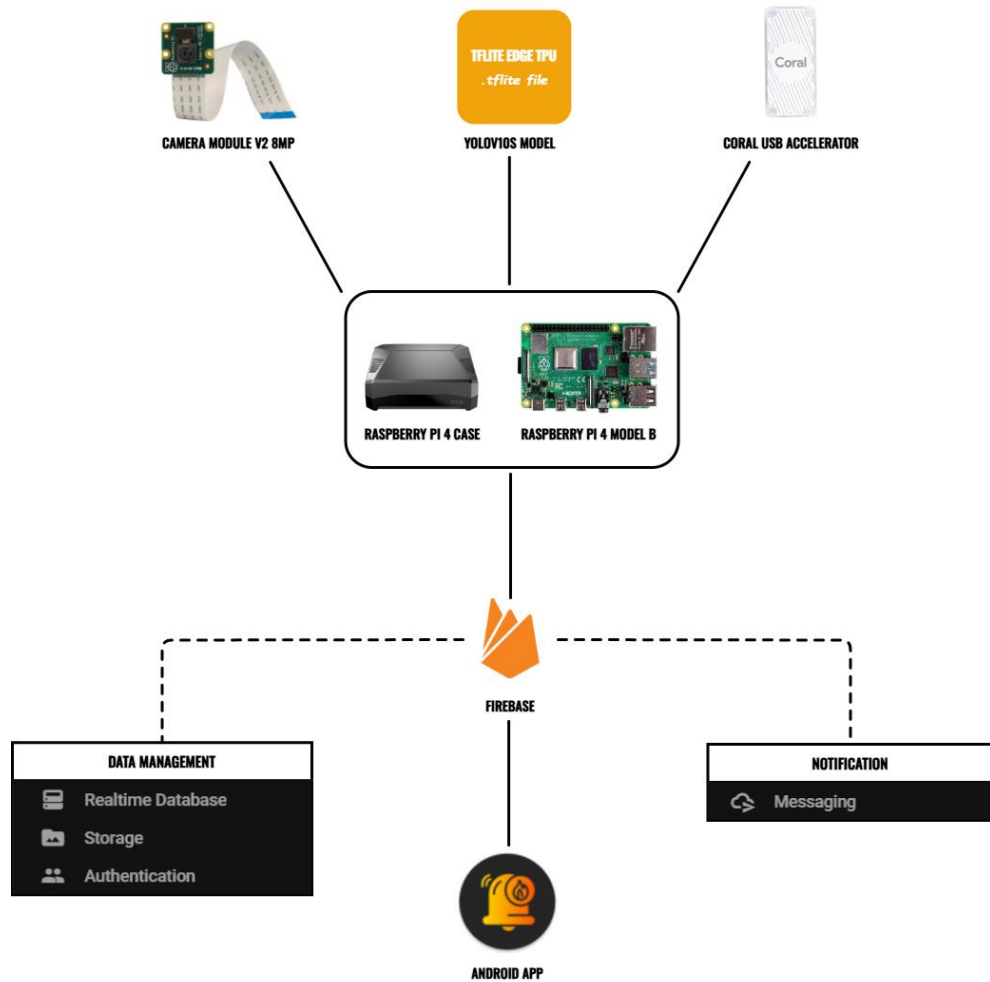
Quá trình chuyển đổi định dạng *TFLite Edge TPU* thực chất là quá trình tối ưu hóa, biên dịch thông qua *Edge TPU Compiler*, định dạng *TensorFlow Lite*.

- **PyTorch Script** là định dạng tiêu chuẩn của *framework PyTorch*, được kết xuất khi huấn luyện mô hình *YOLO* với thư viện *Ultralytics*.
- **Open Neural Network Exchange** là định dạng chuẩn mở, giúp chuyển đổi mô hình giữa các *framework* khác nhau.
- **Protocol Buffers** là định dạng tiêu chuẩn của *framework TensorFlow*, là nền tảng chính được tối ưu để triển khai mô hình học sâu.
- **TensorFlow Lite** là phiên bản nhẹ được tối ưu hóa để triển khai trên các thiết bị di động và nhúng của *framework TensorFlow*.
- **TensorFlow Lite Edge TPU** là phiên bản *TensorFlow Lite* được biên dịch để hoạt động trên các phần cứng chuyên dụng dòng *Edge*.

*Edge TPU* chỉ hoạt động với các mô hình đã được lượng tử hóa. Quá trình lượng tử hóa giúp mô hình trở nên nhỏ gọn và nhanh hơn mà không ảnh hưởng nhiều đến độ chính xác. Điều này rất lý tưởng cho các thiết bị tính toán *Edge* có tài nguyên hạn chế, cho phép ứng dụng phản hồi nhanh hơn nhờ giảm độ trễ và xử lý dữ liệu trực tiếp cục bộ, mà không phụ thuộc vào công nghệ đám mây. Bên cạnh đó, việc xử lý cục bộ cũng giúp bảo vệ quyền riêng tư và bảo mật của người dùng vì dữ liệu không gửi lên máy chủ từ xa.

### 3.3. THỰC NGHIỆM

#### a) Workflow triển khai hệ thống



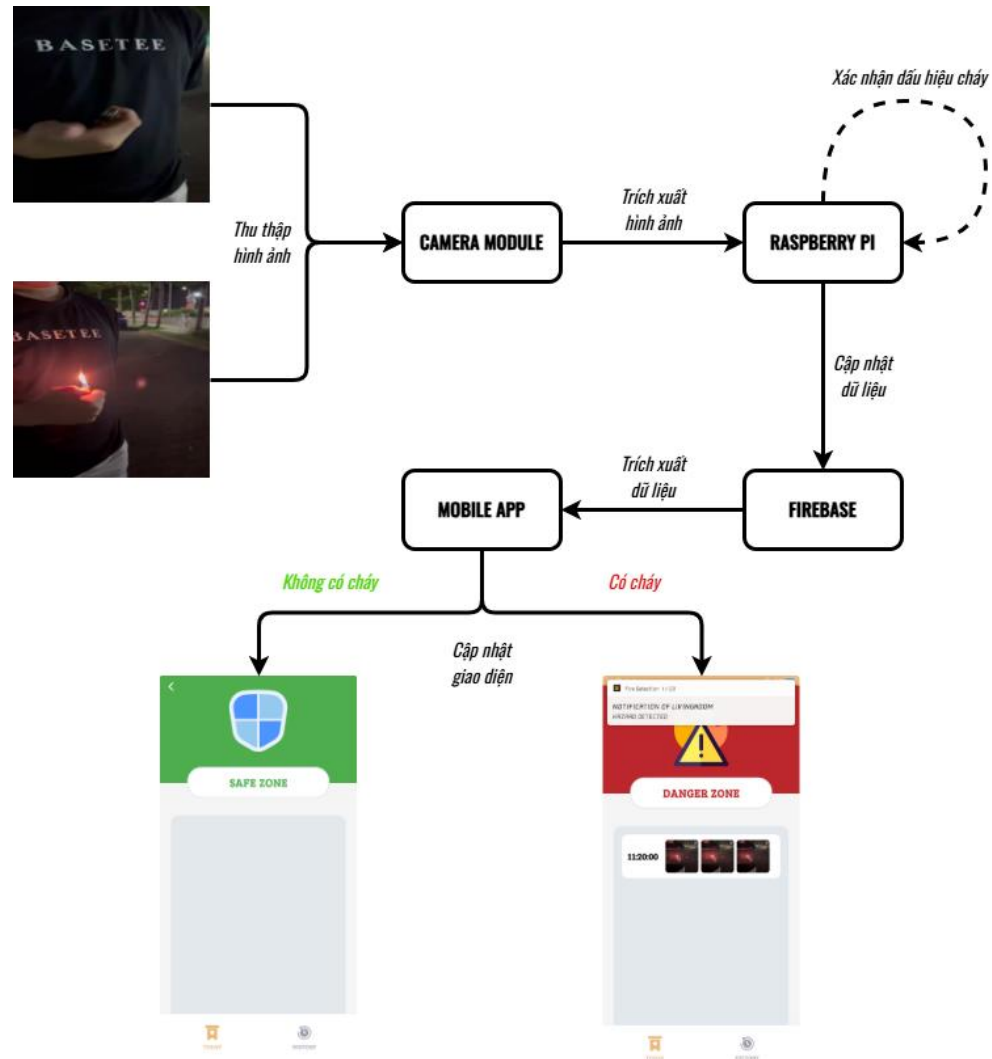
Hình 4.3.3 Workflow triển khai hệ thống

Tổng quan, hệ thống cảnh báo cháy thời gian thực được triển khai dựa trên các phần cứng hiệu năng cao, bao gồm *Raspberry Pi 4 Model B* làm thiết bị xử lý trung tâm, tích hợp *Camera Module V2* để thu thập dữ liệu hình ảnh và *Coral USB Accelerator* để tăng tốc độ suy luận cho mô hình *YOLOv10* phiên bản *S* sử dụng định dạng *Edge TPU*.

Ngoài ra, hệ thống cũng cung cấp một ứng dụng di động trên nền tảng *Android*, cho phép người dùng quản lý và chia sẻ thiết bị, theo dõi trực quan và nhận thông báo cảnh báo từ xa. Ứng dụng này kết nối với phần cứng thông qua các dịch vụ nền tảng *Firebase*, đóng vai trò trung gian trong việc lưu trữ – truyền tải dữ liệu và gửi thông báo thời gian thực.

Tổng kết, sự kết hợp giữa các phần cứng hiệu năng cao và mô hình học sâu mạnh mẽ đảm bảo quy trình cảnh báo cháy diễn ra trong thời gian thực một cách nhanh chóng và chính xác. Đồng thời, ứng dụng di động được phát triển nhằm tăng cường tính thực tiễn cho hệ thống, minh chứng cho khả năng tích hợp linh hoạt với các nền tảng công nghệ hiện đại khác.



b) *Kết quả thực nghiệm*

Hình 4.3.4 Hình ảnh minh họa thực nghiệm

Quy trình hoạt động của hệ thống được thiết kế nhằm đảm bảo phát hiện các dấu hiệu cháy một cách nhanh chóng và chính xác. Tổng quan, quy trình thực nghiệm diễn ra như sau:

**1) CAMERA MODULE**

- Thu thập hình ảnh.

**2) RASPBERRY PI**

- Trích xuất hình ảnh.
- Xác nhận dấu hiệu cháy.
- Cập nhật dữ liệu lên Firebase.

**3) FIREBASE**

- Lưu trữ dữ liệu từ Raspberry Pi và Mobile App.

**4) MOBILE APP**

- Trích xuất dữ liệu từ Firebase.
- Cập nhật giao diện.

## KẾT LUẬN, KIẾN NGHỊ

### 1. KẾT LUẬN

Đề tài "Ứng dụng *YOLOv10* trong phát hiện cháy thời gian thực với hỗ trợ của thiết bị tăng tốc *Coral*" giới thiệu một giải pháp toàn diện gồm huấn luyện mô hình *YOLOv10*, triển khai trên thiết bị nhúng *Raspberry Pi*, *Camera Module* và *Coral*, cùng ứng dụng di động hỗ trợ người dùng. Sự kết hợp giữa *YOLOv10* và *Coral* đảm bảo tính thời gian thực, trong khi ứng dụng di động nâng cao tính thực tiễn. Tổng quan, đề tài sử dụng *YOLOv10* phiên bản *S* và các kết quả đạt được đã chứng minh tính khả thi của hệ thống.

#### ○ Độ chính xác

	<i>Precision</i>	<i>Recall</i>	<i>mAP<sub>50</sub></i>	<i>mAP<sub>50-95</sub></i>
<i>all</i>	0.809	0.761	0.83	0.588
<i>fire</i>	0.848	0.841	0.911	0.666
<i>smoke</i>	0.857	0.82	0.896	0.64
<i>other</i>	0.721	0.623	0.682	0.459

#### ○ Độ hiệu quả

	Bộ nhớ (MB)	Tốc độ (ms)
<i>CPU</i>	15.7	4043.1
<i>TPU</i>	9.28	1698.1

### 2. KHUYẾN NGHỊ

#### ○ Tối ưu hóa mô hình *YOLOv10*

Mô hình *YOLOv10* đã được cải tiến đáng kể về tập dữ liệu và kỹ thuật huấn luyện so với phiên bản trước đó, nhưng vẫn còn một số điểm yếu, đặc biệt khi nhận diện trong các điều kiện môi trường phức tạp. Để mô hình tối ưu hơn, ta cần tập trung cải tiến chất lượng tập dữ liệu và áp dụng các kỹ thuật huấn luyện nâng cao để giúp mô hình học tốt hơn. Đồng thời, việc theo dõi và điều chỉnh trong quá trình huấn luyện để tránh *overfitting* là cần thiết, giúp nâng cao tính khả dụng của mô hình.

#### ○ Mở rộng ứng dụng di động

Ứng dụng di động được phát triển nhằm nâng cao tính thực tiễn của hệ thống so với phiên bản trước đó. Dù đã đáp ứng được các tính năng thiết yếu, ứng dụng vẫn còn nhiều khía cạnh phát triển. Ngoài các tính năng cơ bản, ứng dụng có thể mở rộng với các tính năng nâng cao như đa ngôn ngữ và đa nền tảng giúp tiếp cận người dùng rộng rãi hơn, quản lý thông tin cá nhân giúp cá nhân hóa trải nghiệm người dùng, và phát trực tiếp – thông báo đa kênh giúp việc theo dõi trở nên dễ dàng hơn.

**TÀI LIỆU THAM KHẢO**

- [1]. T.T. Huynh, H.T. Nguyen, and D.T. Phu, “Enhancing Fire Detection Performance Based on Fine-Tuned YOLOv10,” *Comput. Mater. Contin.*, vol. 81, no. 2, pp. 2281-2298, 2024. <https://doi.org/10.32604/cmc.2024.057954>.
- [2]. Glenn Jocher, Qiu Jing, Ayush Chaurasia, “YOLOv10: Real-Time End-to-End Object Detection,” *Ultralytics*. [Online]. Available: <https://docs.ultralytics.com/models/yolov10/>. [Accessed 15 10 2024].
- [3]. “Raspberry Pi 4 Tech Specs,” *Raspberry Pi*. [Online]. Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/>. [Accessed 25 10 2024].
- [4]. “Raspberry Pi Camera Module V2 – 8MP,”. [Online]. Available: <https://raspberrypi.vn/san-pham/raspberry-pi-camera-module-v2-8mp>. [Accessed 25 10 2024].
- [5]. “USB Accelerator Datasheet,” *Coral*. [Online]. Available: <https://coral.ai/docs/accelerator/datasheet/>. [Accessed 25 10 2024].
- [6]. Catargiu Constantin, *FireSmokeDataset*, Roboflow. [Online]. Available: <https://universe.roboflow.com/catargiuconstantin/firesmokedataset>. [Accessed 15 10 2024].

PHỤ LỤC

A. BẢNG SỐ LIỆU

1. BIỂU ĐỒ SO SÁNH YOLOV10 VÀ CÁC PHIÊN BẢN KHÁC

	Params (M)	mAP <sub>50-95</sub> (%)	Latency (ms)
YOLOv6-3.0-N	4.7	37	2.69
Gold-YOLO-N	5.6	39.6	2.92
YOLOv8-N	3.2	37.3	6.16
YOLOv10-N	2.3	39.5	1.84
YOLOv6-3.0-S	18.5	44.3	3.42
Gold-YOLO-S	21.5	45.4	3.82
YOLOv8-S	11.2	44.9	7.07
YOLOv10-S	7.2	46.8	2.49
RT-DETR-R18	20	46.5	4.58
YOLOv6-3.0-M	34.9	49.1	5.63
Gold-YOLO-M	41.3	49.8	6.38
YOLOv8-M	25.9	50.6	9.5
YOLOv10-M	15.4	51.3	4.74
YOLOv6-3.0-L	59.6	51.8	9.02
Gold-YOLO-L	75.1	51.8	10.65
YOLOv8-L	43.7	52.9	12.39
RT-DETR-R50	42	53.1	9.2
YOLOv10-L	24.4	53.4	7.28
YOLOv8-X	68.2	53.9	16.86
RT-DETR-R101	76	54.3	13.71
YOLOv10-X	29.5	54.4	10.7

**2. BIỂU ĐỒ TỐC ĐỘ SUY LUẬN YOLOV10 TRÊN CPU**

	N	S	M	B	L	X
1	1606.53	2943.19	6524.75	10447.13	13075.46	17104.41
2	1252.39	3104.31	6974.99	11027.17	13836.69	18159.74
3	1141.25	2869.53	6452.15	10353.73	12885.58	17104.46
4	1123.81	2890.96	6448.78	10328.38	12874.43	17070.94
5	983.51	2455.35	5561.29	8817.69	10910.18	14479.04
6	1244.4	3109.84	6948.73	11015.62	13771.9	18094.19
7	1170.84	2886.11	6476.29	10341.51	12973.89	16848.16
8	1152.75	2879.74	6450.87	10332.69	12916.23	16974.27
9	1138.68	2870.54	6467.58	10352	12925.03	16830.13
10	1215.47	3081.92	6911.44	11023.75	13754.68	17975.17
11	1241.97	3077.54	6927.73	11084.8	13733.98	18036.52
12	1140.35	2880.15	6460.4	10361.92	12911.96	16938
13	1216.21	3062.88	6948.24	10929.73	13795.91	18035.57
14	1137.13	2895.75	6468.19	10376.11	12969.57	16923.44
15	1239.26	3072.51	6923.13	11021.06	13808.36	18260.11
16	997.27	2427.58	5483.22	8859.59	10974.49	14490.21
17	1159.28	2886.72	6467.52	10386.27	12966.32	16959.16
18	1241.19	3085.83	6932.31	11090.71	13779.91	18194.67
19	1134.87	2870.7	6461	10400.62	12962.24	16891.69
20	1235.22	3072.14	6914.34	11075.51	13769.63	18221.88
21	1134.87	2911.27	6449.76	10341.89	12911.16	17141.45
22	1133.28	2857.54	6457.91	10326.63	12901.26	17057.61
23	1250.45	3115.31	6939.22	10948.03	13760.3	18250.74
24	1220.79	3081.48	6958.99	11032.61	13778.12	18247.1
25	1212.39	3085.27	6961.34	11001.36	13802.27	18145.25

**3. BIỂU ĐỒ TỐC ĐỘ Suy LUẬN YOLOV10 TRÊN TPU**

	N	S	M	B	L	X
1	792.31	1788.32	4232.7	6290.35	8068.95	10522.23
2	712.22	1719.95	3981.3	6012.15	7724.76	9933.74
3	707.19	1717.2	3974.55	6010.46	7715.19	9917.76
4	710.39	1721.01	3966.68	6027.9	7714.95	9918.74
5	698.32	1715.49	3970.32	6023.81	7726.59	9931.18
6	697.91	1715.6	3962.18	6020.95	7704.85	9914.04
7	707.07	1712.61	3962.97	6023.81	7760.11	9936.12
8	715.53	1708.77	3968.46	6022.5	7750.33	9916.26
9	695.03	1707.91	3962.7	6021.17	7722.84	9912.75
10	695.03	1713.8	3962.15	6025.78	7705.39	9894.15
11	700.57	1708.55	3959.25	6027.46	7718.9	9919.08
12	697.29	1709.59	3962.95	6028.48	7697.89	9936.14
13	694.97	1715.03	3973.09	6064.19	7715.57	9944.15
14	697.88	1708.91	3962.11	6057.12	7722.96	9943.7
15	694.51	1709.35	3967.7	6058.75	7722.54	9946.29
16	697.64	1712.55	3972	6060	7723.25	9935.41
17	697.4	1708.44	3964.62	6063.9	7722.01	9935.68
18	706.1	1710.34	3970.7	6062.26	7720.02	9925.62
19	698.08	1712.79	3969.74	6020.86	7714.96	9943.14
20	697.09	1716.74	3968.21	6021.88	7716.91	9941.89
21	694.31	1711.44	3980.47	6024.34	7718.84	9944.11
22	698.85	1714.33	3983.47	6016.84	7720.87	9944.88
23	693.2	1714.21	3980.46	6016.98	7726.47	9936.58
24	693.56	1712.63	3968.64	6011.48	7775.89	9934.8
25	697.4	1718.47	3982.32	6018.76	7734.98	9929.95

**4. BIỂU ĐỒ SO SÁNH TỐC ĐỘ Suy LUẬN YOLOV10 TRÊN CPU & TPU**

	N	S	M	B	L	X
CPU	1189.32	2938.97	6598.81	10531.06	13149.98	17297.36
TPU	703.59	1716.16	3980.39	6041.29	7737.84	9954.26

**B. MÃ NGUỒN****1. MÔ HÌNH YOLOV10**

- *Tải tập dữ liệu từ Roboflow*

```
from roboflow import Roboflow

rf = Roboflow(api_key="api_key")
project = rf.workspace("catargiuconstantin").project("firesmokedataset")
version = project.version(1)
dataset = version.download("yolov9", location="/content/dataset")
```

- *Tải mô hình huấn luyện sẵn từ Ultralytics*

```
!wget -P '/content/pretrained' 'https://github.com/ultralytics/assets/releases/download/v8.2.0/yolov10s.pt'
```

- *Huấn luyện mô hình trong 10 vòng đầu tiên*

```
!yolo task=detect mode=train \
data='/content/dataset/data.yaml' \
model='/content/pretrained/yolov10s.pt' \
epochs=100 batch=32 patience=5 save_period=10 plots=True \
optimizer='SGD' warmup_epochs=0
```

- *Huấn luyện mô hình mỗi 10 vòng tiếp theo*

```
!yolo task=detect mode=train \
data='/content/dataset/data.yaml' \
model='/content/checkpoint.pt' resume=True
```

- *Chuyển đổi định dạng Edge TPU*

```
!yolo export model='/content/model.pt' format=edgetpu
```

## 2. THIẾT BỊ NHÚNG

### ○ Nhận diện dấu hiệu cháy

```
def detect_image(self, image):
    result = self.model.predict(source=image, conf=0.5, verbose=False)[0]
    boxes = result.boxes.xyxy
    state = False

    if len(boxes) > 0:
        clses = result.boxes.cls
        cls_names = result.names

        for box, cls in zip(boxes, clses):
            cls_name = cls_names[int(cls)]

            if cls_name != "other":
                state = True
                t, l, r, b = map(int, box.tolist())
                color = (40, 40, 190) if cls_name == "fire" else (210, 210, 210)
                # DRAW BOUNDING BOX.
                cv2.rectangle(image, (t, l), (r, b), color, 2)
                # DRAW CLASS NAME.
                cv2.putText(image, cls_name, (t, l - 10), CLS_FONT, 0.65, color)

    return state
```

### ○ Ghi nhận dữ liệu sự kiện khi phát hiện dấu hiệu cháy

```
def handle_signal(self, state):
    if state or self.state != state:
        self.detect_timer.activate()
        self.state = state
        self.firebase.set_value(state, "detect")

    if state:
        self.is_detecting = True
        self.timestamp = datetime.now().strftime(r"%Y-%m-%d/%H:%M:%S")
        Thread(target=self.firebase.send_message).start()
```

### ○ Lưu trữ dữ liệu sự kiện vào Firebase

```
def handle_upload(self, image):
    if self.upload_quantity <= 3:
        if not self.upload_timer.is_active:
            _, image_encoded = cv2.imencode(".jpg", image)
            image_bytes = BytesIO(image_encoded.tobytes())
            args = image_bytes, self.timestamp, self.upload_quantity

            Thread(target=self.firebase.upload_image, args=args).start()
            self.upload_timer.activate()
            self.upload_quantity += 1
    else:
        self.is_detecting = False
        self.timestamp, self.upload_quantity = None, 1
```



### 3. ỨNG DỤNG DI ĐỘNG

- *Lưu trữ thông tin thiết bị và tài khoản người dùng vào Firebase*

```
private void loadUserData() {
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    if (user != null) {
        USER_UID = user.getId();
        String userPhone = user.getPhoneNumber(), userEmail = user.getEmail();
        USER_ID = userPhone != null && !userPhone.isEmpty() ? userPhone : userEmail;

        FirebaseMessaging.getInstance().getToken().addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                FirebaseDatabase database = FirebaseDatabase.getInstance();
                // TOKEN.
                String tokenPath = String.format("users/%s/token", USER_UID);
                DatabaseReference tokenRef = database.getReference(tokenPath);
                tokenRef.setValue(task.getResult());
                // IDENTIFIER.
                String identifierPath = String.format("users/%s/identifier", USER_UID);
                DatabaseReference identifierRef = database.getReference(identifierPath);
                identifierRef.setValue(USER_ID);
            }
        });
    }
}
```

- *Nhận thông báo cảnh báo từ Firebase*

```
@Override
public void onMessageReceived(@NonNull RemoteMessage message) {
    super.onMessageReceived(message);
    // SHOW NOTIFICATION.
    Map<String, String> messageData = message.getData();
    String title = messageData.get("title");
    String body = messageData.get("body");
    String deviceId = messageData.get("device_id");
    String deviceName = messageData.get("device_name");
    this.showNotification(title, body, deviceId, deviceName);
}
```

- *Cập nhật giao diện khi trạng thái “nhận diện” thay đổi*

```
String detectedPath = String.format("devices/%s/detect", DeviceActivity.DEVICE_ID);
DatabaseReference detectRef = database.getReference(detectedPath);
ValueEventListener detectListener = detectRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        Boolean detected = snapshot.getValue(Boolean.class);
        changeLayout(detected != null && detected);
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
    }
});
```

○ *Cập nhật hình ảnh cháy khi dữ liệu sự kiện được tải lên Firebase*

```
// LAST CAPTURE EVENT.
ValueEventListener lastCapturedListener = capturedRef.limitToLast(1)
    .addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            if (snapshot.exists()) {
                DataSnapshot lastCapture = snapshot.getChildren().iterator().next();
                updateHistory(lastCapture);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
```

```
private void updateHistory(DataSnapshot lastCapture) {
    // GET ITEM.
    String timestamp = lastCapture.getKey();
    List<String> listURL = new ArrayList<>();
    lastCapture.getChildren().forEach(dataCapture ->
        listURL.add(dataCapture.getValue(String.class)));
    HistoryItem item = new HistoryItem(timestamp, listURL);
    // GET DATA.
    List<HistoryItem> listItem = new ArrayList<>(this.historyRecyclerAdapter.getOriginalData());
    if (!listItem.isEmpty() && this.lastCaptureTimestamp.equals(timestamp)) listItem.remove(0);
    else this.lastCaptureTimestamp = timestamp;
    listItem.add(0, item);
    this.historyRecyclerAdapter.loadOriginalData(listItem);
}
```

○ *Lọc dữ liệu sự kiện được ghi nhận*

```
private void filterHistory() {
    // ENABLE LOADING.
    this.loadingView.setVisibility(View.VISIBLE);
    // GET DATA.
    String startTime = String.format("%s:00", this.tvStartTime.getText().toString());
    String limitTime = String.format("%s:59", this.tvLimitTime.getText().toString());
    List<HistoryItem> filteredData = this.historyRecyclerAdapter.getOriginalData().stream()
        .filter(item -> {
            String timestamp = item.getTimestamp();
            return timestamp.compareTo(startTime) >= 0 && timestamp.compareTo(limitTime) <= 0;
        }).collect(Collectors.toList());
    this.historyRecyclerAdapter.loadFilteredData(filteredData);
    // DISABLE LOADING.
    this.loadingView.setVisibility(View.GONE);
}
```