

ĐẠI HỌC BÁCH KHOA HÀ NỘI

# ĐỒ ÁN NGHIÊN CỨU CỦ NHÂN

## Thiết kế hệ thống cảnh báo hỏa hoạn cho chung cư mini

NGUYỄN THẠC HIẾU

Hieu.nt213921@sis.hust.edu.vn

NGUYỄN THỊ TUYẾT TRINH

Trinh.ntt214114@sis.hust.edu.vn

DOÃN ĐĂNG NHẬT

Nhat.dd214031@sis.hust.edu.vn

Ngành Kỹ thuật Điện tử - Viễn thông

Trường Điện – Điện tử

Giảng viên hướng dẫn: PGS. TS. Nguyễn Thúy Anh

Chữ ký của GVHD

KHOA: Điện tử

Giảng viên hướng dẫn: PGS. TS Nguyễn Hữu Trung

Chữ ký của GVHD

KHOA: Kỹ thuật truyền thông

HÀ NỘI, 07/2025



**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**TRƯỜNG ĐIỆN – ĐIỆN TỬ**

**ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP**  
**(DÀNH CHO CÁN BỘ HƯỚNG DẪN)**

Tên đề tài: Thiết kế hệ thống cảnh báo hỏa hoạn cho chung cư mini

Họ tên SV: Nguyễn Thạc Hiếu. MSSV: 20213921

Cán bộ hướng dẫn: PGS.TS Nguyễn Thúy Anh

ST T	Tiêu chí (Điểm tối đa)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	<b>Thái độ làm việc (2,5 điểm)</b>	Nghiêm túc, tích cực và chủ động trong quá trình làm ĐATN  Hoàn thành đầy đủ và đúng tiến độ các nội dung được GVHD giao	
2	<b>Kỹ năng viết quyển ĐATN (2 điểm)</b>	Trình bày đúng mẫu quy định, bỏ cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có cẩn lè, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.  Kỹ năng diễn đạt, phân tích, giải thích, lập luận: Cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.	
3	<b>Nội dung và kết quả đạt được (5 điểm)</b>	Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.  Nội dung và kết quả đạt được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.  Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/tính sáng tạo trong nội dung và kết quả đồ án.	
4	<b>Điểm thành tích (1 điểm)</b>	Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. <b>(1 điểm)</b>  Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. <b>(0,5 điểm)</b>	
		<b>Điểm tổng các tiêu chí:</b>	
		<b>Điểm hướng dẫn:</b>	

**Cán bộ hướng dẫn**  
(Ký và ghi rõ họ tên)



**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**TRƯỜNG ĐIỆN – ĐIỆN TỬ**

**ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP**  
**(DÀNH CHO CÁN BỘ HƯỚNG DẪN)**

Tên đề tài: Thiết kế hệ thống cảnh báo hỏa hoạn cho chung cư mini

Họ tên SV: Nguyễn Thị Tuyết Trinh. MSSV: 20214114

Cán bộ hướng dẫn: PGS.TS Nguyễn Thúy Anh

ST T	Tiêu chí (Điểm tối đa)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	<b>Thái độ làm việc (2,5 điểm)</b>	Nghiêm túc, tích cực và chủ động trong quá trình làm ĐATN  Hoàn thành đầy đủ và đúng tiến độ các nội dung được GVHD giao	
2	<b>Kỹ năng viết quyển ĐATN (2 điểm)</b>	Trình bày đúng mẫu quy định, bỏ cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có cẩn lè, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.  Kỹ năng diễn đạt, phân tích, giải thích, lập luận: Cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.	
3	<b>Nội dung và kết quả đạt được (5 điểm)</b>	Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.  Nội dung và kết quả đạt được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.  Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/tính sáng tạo trong nội dung và kết quả đồ án.	
4	<b>Điểm thành tích (1 điểm)</b>	Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. <b>(1 điểm)</b>  Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. <b>(0,5 điểm)</b>	
		<b>Điểm tổng các tiêu chí:</b>	
		<b>Điểm hướng dẫn:</b>	

**Cán bộ hướng dẫn**  
(Ký và ghi rõ họ tên)



**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**TRƯỜNG ĐIỆN – ĐIỆN TỬ**

**ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP**  
**(DÀNH CHO CÁN BỘ HƯỚNG DẪN)**

Tên đề tài: Thiết kế hệ thống cảnh báo hỏa hoạn cho chung cư mini

Họ tên SV: Doãn Đăng Nhật. MSSV: 20214031

Cán bộ hướng dẫn: PGS.TS Nguyễn Hữu Trung

ST T	Tiêu chí (Điểm tối đa)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	<b>Thái độ làm việc (2,5 điểm)</b>	Nghiêm túc, tích cực và chủ động trong quá trình làm ĐATN  Hoàn thành đầy đủ và đúng tiến độ các nội dung được GVHD giao	
2	<b>Kỹ năng viết quyển ĐATN (2 điểm)</b>	Trình bày đúng mẫu quy định, bỏ cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có cẩn lè, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.  Kỹ năng diễn đạt, phân tích, giải thích, lập luận: Cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.	
3	<b>Nội dung và kết quả đạt được (5 điểm)</b>	Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.  Nội dung và kết quả đạt được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.  Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/tính sáng tạo trong nội dung và kết quả đồ án.	
4	<b>Điểm thành tích (1 điểm)</b>	Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. <b>(1 điểm)</b>  Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. <b>(0,5 điểm)</b>	
		<b>Điểm tổng các tiêu chí:</b>	
		<b>Điểm hướng dẫn:</b>	

**Cán bộ hướng dẫn**  
(Ký và ghi rõ họ tên)



# **ĐẠI HỌC BÁCH KHOA HÀ NỘI**

## **TRƯỜNG ĐIỆN – ĐIỆN TỬ**

# **ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP**

## **(DÀNH CHO CÁN BỘ PHẢN BIÊN)**

Tên đề tài: Thiết kế hệ thống cảnh báo hỏa hoạn cho chung cư mini

Họ tên SV: Nguyễn Thạc Hiếu MSSV: 20213921

Cán bộ phản biện:

ST T	Tiêu chí (Điểm tối đa)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	<b>Trình bày quyển ĐATN (4 điểm)</b>	<p>Đồ án trình bày đúng mẫu quy định, bô cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có cẩn lè, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.</p> <p>Kỹ năng diễn đạt, phân tích, giải thích, lập luận: cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.</p>	
2	<b>Nội dung và kết quả đạt được (5,5 điểm)</b>	<p>Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.</p> <p>Nội dung và kết quả được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.</p> <p>Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/ tính sáng tạo trong nội dung và kết quả đồ án.</p>	
3	<b>Điểm thành tích (1 điểm)</b>	<p>Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ các giải thưởng KH quốc tế/ trong nước từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. <b>(1 điểm)</b></p> <p>Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành. <b>(0,5 điểm)</b></p>	
<b>Điểm tổng các tiêu chí:</b>			
<b>Điểm phản biện:</b>			

## Cán bộ phản biện



**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**TRƯỜNG ĐIỆN – ĐIỆN TỬ**

**ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP**  
**(DÀNH CHO CÁN BỘ PHẢN BIỆN)**

Tên đề tài: Thiết kế hệ thống cảnh báo hỏa hoạn cho chung cư mini

Họ tên SV: Nguyễn Thị Tuyết Trinh                    MSSV: 20214114

Cán bộ phản biện:

ST T	Tiêu chí (Điểm tối da)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	<b>Trình bày quyển ĐATN (4 điểm)</b>	<p>Đồ án trình bày đúng mẫu quy định, bố cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đẽ cập đến trong đồ án, có cẩn lè, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.</p> <p>Kỹ năng diễn đạt, phân tích, giải thích, lập luận: cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.</p>	
2	<b>Nội dung và kết quả đạt được (5,5 điểm)</b>	<p>Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.</p> <p>Nội dung và kết quả đạt được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.</p> <p>Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/ tính sáng tạo trong nội dung và kết quả đồ án.</p>	
3	<b>Điểm thành tích (1 điểm)</b>	<p>Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ các giải thưởng KH quốc tế/ trong nước từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. <b>(1 điểm)</b></p> <p>Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành. <b>(0,5 điểm)</b></p>	
<b>Điểm tổng các tiêu chí:</b>			
<b>Điểm phản biện:</b>			

**Cán bộ phản biện**  
(Ký và ghi rõ họ tên)



**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**TRƯỜNG ĐIỆN – ĐIỆN TỬ**

**ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP**  
**(DÀNH CHO CÁN BỘ PHẢN BIỆN)**

Tên đề tài: Thiết kế hệ thống cảnh báo hỏa hoạn cho chung cư mini

Họ tên SV: Doãn Đăng Nhật                    MSSV: 20214031

Cán bộ phản biện:

ST T	Tiêu chí (Điểm tối da)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	<b>Trình bày quyển ĐATN (4 điểm)</b>	<p>Đồ án trình bày đúng mẫu quy định, bố cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đẽ cập đến trong đồ án, có cẩn lè, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.</p> <p>Kỹ năng diễn đạt, phân tích, giải thích, lập luận: cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.</p>	
2	<b>Nội dung và kết quả đạt được (5,5 điểm)</b>	<p>Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.</p> <p>Nội dung và kết quả đạt được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.</p> <p>Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/ tính sáng tạo trong nội dung và kết quả đồ án.</p>	
3	<b>Điểm thành tích (1 điểm)</b>	<p>Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ các giải thưởng KH quốc tế/ trong nước từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. <b>(1 điểm)</b></p> <p>Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành. <b>(0,5 điểm)</b></p>	
<b>Điểm tổng các tiêu chí:</b>			
<b>Điểm phản biện:</b>			

**Cán bộ phản biện**  
(Ký và ghi rõ họ tên)



## ĐỀ TÀI TỐT NGHIỆP

**Tên đề tài:** Thiết kế hệ thống cảnh báo hỏa hoạn cho chung cư mini.

Họ và tên sinh viên:	Nguyễn Thạc Hiếu	MSSV: 20213921
Họ và tên sinh viên:	Nguyễn Thị Tuyết Trinh	MSSV: 20214114
Họ và tên sinh viên:	Doãn Đăng Nhật	MSSV: 20214031

Giáo viên hướng dẫn  
Ký và ghi rõ họ tên



## LỜI MỞ ĐẦU

### Lời cảm ơn

Nhóm em xin gửi lời cảm ơn chân thành đến các thầy cô trong khoa ngành, những người đã tận tâm giảng dạy và truyền đạt kiến thức quý báu cho em trong suốt quá trình học tập.

Nhóm em đặc biệt biết ơn cô Nguyễn Thúy Anh và thầy Nguyễn Hữu Trung đã tận tình chỉ dẫn và luôn đồng hành cùng nhóm em trong suốt quá trình thực hiện đồ án. Những góp ý sâu sắc và định hướng kịp thời của thầy cô là nguồn động lực lớn giúp nhóm em hoàn thành tốt đề tài này.

Mặc dù đã nỗ lực hoàn thành tốt nhất trong khả năng của mình, nhóm em hiểu rằng đồ án vẫn không tránh khỏi những thiếu sót nhất định. Chúng em mong nhận được những nhận xét và góp ý quý báu từ Quý thầy cô trong Hội đồng bảo vệ để có thể tiếp tục hoàn thiện đề tài và tích lũy thêm kinh nghiệm.

### Tóm tắt nội dung đồ án

Đồ án tập trung thiết kế và xây dựng hệ thống báo cháy kết hợp AI gồm hai thành phần chính. Đầu tiên là phần xử lý trung tâm sử dụng Raspberry Pi 4 tích hợp camera để truyền hình ảnh theo thời gian thực và áp dụng mô hình trí tuệ nhân tạo nhằm nhận diện khói và lửa trong khu vực giám sát. Khi phát hiện nguy cơ cháy, hệ thống sẽ gửi cảnh báo tức thời đến ứng dụng di động, giúp người dùng kịp thời xử lý. Bên cạnh đó hệ thống cho phép người dùng xem video trực tiếp tại khu vực lắp đặt thông qua ứng dụng giúp nắm rõ tình hình, xử lý đạt hiệu quả cao.

Thứ hai là bộ thu thập dữ liệu môi trường sử dụng vi điều khiển ESP32 kết hợp với các cảm biến khói, khí CO và nhiệt độ. Bộ phận này có nhiệm vụ giám sát liên tục các chỉ số môi trường và gửi dữ liệu tới ứng dụng di động thông qua một Server trung gian. Khi giá trị vượt ngưỡng cho phép, ESP32 sẽ kích hoạt còi hú tài chấn để cảnh báo khẩn cấp, người dùng có thể phản ứng lại bằng cách tắt cảnh báo bằng nút bấm trên mạch.

Hệ thống kết hợp giữa công nghệ nhúng, truyền thông không dây và trí tuệ nhân tạo, mang lại khả năng phát hiện cháy sớm và đáng tin cậy. Kết quả kiểm thử cho thấy thiết bị hoạt động ổn định, phản hồi nhanh và phù hợp với các ứng dụng thực tiễn trong giám sát an toàn cháy nổ. Thông qua quá trình thực hiện, nhóm em đã tích lũy được nhiều kiến thức và kinh nghiệm trong việc tích hợp phần cứng, xử lý dữ liệu và triển khai AI vào thực tế.

Sinh viên thực hiện  
Ký và ghi rõ họ tên

Doan Đăng Nhật

Nguyễn Thị Tuyết Trinh

Nguyễn Thạc Hiếu



## LỜI CAM ĐOAN

Nhóm em gồm 3 thành viên là Nguyễn Thạc Hiếu, mã số sinh viên 20213921, sinh viên lớp Điện tử 05, khóa 66, Nguyễn Thị Tuyết Trinh, mã số sinh viên 20214114, sinh viên lớp Điện tử 05, khóa 66 và Doãn Đăng Nhật, mã số sinh viên 20214031, sinh viên lớp Điện tử 05, khóa 66. Giảng viên hướng dẫn là PGS. TS Nguyễn Thúy Anh và PSG. TS Nguyễn Hữu Trung. Nhóm em xin cam đoan rằng toàn bộ nội dung trong đồ án này là kết quả do chính nhóm em thực hiện, không sao chép hay vay mượn từ bất kỳ đồ án nào trước đó. Các số liệu và kết quả được trình bày đều trung thực, phản ánh đúng quá trình nghiên cứu và thực hiện của nhóm. Nhóm em hoàn toàn chịu trách nhiệm về tính xác thực cũng như tính nguyên bản của đồ án. Nếu phát hiện có vi phạm, nhóm em xin chịu mọi hình thức xử lý theo quy định của nhà trường.

Hà Nội, ngày 20 tháng 6 năm 2025

Người cam đoan

**Doãn Đăng Nhật Nguyễn Thị Tuyết Trinh Nguyễn Thạc Hiếu**



## MỤC LỤC

<b>LỜI MỞ ĐẦU .....</b>	
Lời cảm ơn .....	
Tóm tắt nội dung đồ án .....	
<b>LỜI CAM ĐOAN .....</b>	
<b>MỤC LỤC.....</b>	
<b>DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT .....</b>	i
<b>DANH MỤC HÌNH VẼ .....</b>	iii
<b>DANH MỤC BẢNG BIỂU .....</b>	vii
<b>CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI.....</b>	1
1.1    Đặt vấn đề .....	1
1.2    Mục tiêu của đề tài .....	3
1.3.1    Phát hiện sớm nguy cơ cháy .....	3
1.3.2    Xử lý dữ liệu tại biên (Edge Computing) .....	4
1.3.3    Tích hợp trí tuệ nhân tạo (AI) để xử lý hình ảnh .....	4
1.3.4    Tạo cơ chế cảnh báo đa tầng và can thiệp thủ công.....	4
1.4    Mục tiêu phi chức năng.....	4
1.5    Phạm vi nghiên cứu.....	5
1.6    Phương pháp nghiên cứu.....	5
1.7    Đối tượng hướng đến .....	6
<b>CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....</b>	7
2.1    Giao thức HTTP .....	7
2.1.1    Giới thiệu về giao thức HTTP.....	7
2.1.2    Vai trò của HTTP .....	7
2.1.3    Sơ đồ hoạt động của HTTP.....	7
2.1.4    Các thành phần chính của HTTP .....	8
2.2    Flutter Framework.....	9
2.2.1    Flutter là gì? .....	9
2.2.2    Lý do lựa chọn Flutter? .....	10
2.2.3    Đặc điểm nổi bật của Flutter .....	10
2.3    Firebase .....	11

2.3.1	Firebase là gì? .....	11
2.3.2	Lịch sử phát triển của Firebase .....	11
2.3.3	Lợi ích của Firebase .....	12
2.3.4	Các dịch vụ của Firebase được sử dụng trong đề tài .....	13
2.3.5	Ưu điểm và hạn chế của Firebase .....	14
2.4	Altium Designer .....	15
2.5	Trí tuệ nhân tạo (Artificial Intelligence) .....	15
2.5.1	Giới thiệu về trí tuệ nhân tạo (AI).....	16
2.5.2	Thị giác máy tính .....	16
2.6	Mô hình YOLOv8 .....	18
2.6.1	Tổng quan về bài toán phát hiện đối tượng .....	18
2.6.2	Giới thiệu về YOLO.....	19
2.6.3	Giới thiệu về YOLOv8.....	20
2.6.4	Đánh giá mô hình YOLO.....	30
2.6.5	Intersection over union (IoU).....	31
2.6.6	Bài toán Non – Max suppression .....	34
2.7	Video streaming .....	36
2.7.1	Tổng quan về video streaming .....	37
2.7.2	Giao thức RTMP (Real – Time Messaging Protocol). .....	37
2.7.3	Giao thức HLS (HTTP Live Streaming).....	39
2.7.4	Gstreamer .....	40
2.7.5	Bộ mã hóa và bộ giải mã H.264.....	41
2.8	Tổng quan về cơ sở dữ liệu .....	43
2.8.1	Giới thiệu về cơ sở dữ liệu .....	43
2.8.2	Hệ quản trị CSDL trong hệ thống .....	43
<b>CHƯƠNG 3. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG.....</b>	<b>46</b>	
3.1	Khảo sát các hệ thống hiện có.....	46
3.1.1	Các hệ thống hiện có.....	46
3.1.2	Đánh giá tổng quan các hệ thống hiện có và đề xuất giải pháp	52
3.2	Phân tích và thiết kế hệ thống .....	53
3.2.1	Phân tích tổng thể.....	53
3.2.2	Sơ đồ khối tổng quát của hệ thống.....	55
3.2.3	Sơ đồ thuật toán tổng quát .....	57
3.2.4	Thiết kế phần mềm hệ thống.....	58

3.2.5	Thiết kế ứng dụng di động .....	77
<b>CHƯƠNG 4. TRIỂN KHAI THỰC HIỆN</b>		<b>79</b>
4.1	Triển khai thực hiện phần cứng.....	79
4.1.1	Các linh kiện được sử dụng .....	79
4.1.2	Thiết kế mạch.....	103
4.2	Tính toán nồng độ khí CO và khói từ cảm biến MQ-7 và MP-2 .....	104
4.2.1	Giới thiệu .....	104
4.2.2	Công thức tính toán nồng độ (ppm).....	104
4.2.3	Hiệu chỉnh và điều chỉnh thực nghiệm .....	105
4.3	Đọc dữ liệu nhiệt độ từ cảm biến DHT11 .....	107
4.3.1	Nguyên lý hoạt động của cảm biến DHT11.....	108
4.3.2	ESP32 nhận và xử lý dữ liệu từ DHT11 .....	108
4.4	Thực hiện huấn luyện mô hình.....	109
4.4.1	Huấn luyện mô hình.....	109
4.4.2	Đánh giá kết quả huấn luyện.....	110
4.5	Triển khai mô hình sang Raspberry Pi 4.....	116
4.5.1	Xử lý tại biên trên Raspberry Pi 4.....	116
4.5.2	Hoạt động của Server.....	119
4.6	Triển khai ứng dụng di động .....	123
4.6.1	Thiết kế giao diện người dùng .....	123
4.7	Kết quả thực nghiệm .....	129
4.7.1	Triển khai thực nghiệm mô hình nhận diện đám cháy.....	129
4.7.2	Triển khai thực nghiệm hệ thống .....	131
<b>CHƯƠNG 5. KẾT LUẬN</b>		<b>148</b>
5.1	Kết luận .....	148
5.2	Hướng phát triển của đồ án trong tương lai .....	148
<b>TÀI LIỆU THAM KHẢO</b>		
<b>PHỤ LỤC</b>		



## DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

<b>Viết tắt</b>	<b>Tiếng Anh</b>	<b>Tiếng Việt</b>
<b>A</b>		
AI	Artificial Intelligence	Trí tuệ nhân tạo
ADC	Analog-to-Digital Converter	Bộ chuyển đổi tín hiệu tương tự sang số
<b>B</b>		
BLE	Bluetooth Low Energy	Chuẩn Bluetooth năng lượng thấp
<b>C</b>		
CNN	Convolution Neural Network	Mạng thần kinh tích chập
CO	Carbon Monoxide	Khí CO (gây ngộ độc nồng độ cao)
CPU	Central Processing Unit	Bộ xử lý trung tâm
<b>D</b>		
DHT11	Digital Humidity and Temperature Sensor 11	Cảm biến nhiệt độ và độ ẩm kỹ thuật số
<b>E</b>		
EMA	Exponential Moving Average	Trung bình cộng mũ (lọc nhiễu dữ liệu)
<b>F</b>		
FCM	Firebase Cloud Messaging	Dịch vụ gửi thông báo đẩy của Firebase
FPS	Frames Per Second	Số khung hình hiển thị mỗi giây
<b>G</b>		
GPIO	General Purpose Input/Output	Chân vào/ra đa năng
GStreamer		Framework xử lý video/audio thời gian thực
<b>H</b>		
HTTP	HyperText Transfer Protocol	Giao thức truyền siêu văn bản
HLS	HTTP Live Streaming	Giao thức truyền phát video dựa trên HTTP
<b>I</b>		
IoU	Intersection over Union	Giao điểm trên liên hiệp
IoT	Internet of Things	Internet vạn vật
<b>J</b>		
JSON	JavaScript Object Notation	Định dạng dữ liệu dạng văn bản
<b>M</b>		

<b>ML</b>	Machine Learning	Học máy
<b>M3U8</b>	UTF – 8 Playlist File	Tệp danh sách phát trong HLS
<b>N</b>		
<b>NCNN</b>	Rate of Climb	Tên framework deep learning tối ưu của Tencent
<b>NMS</b>	Non – Max Suppression	Thuật toán loại bỏ hộp trùng lặp
<b>O</b>		
<b>ONNX</b>	Open Neural Network Exchange	Định dạng trao đổi mạng nơ – ron mở
<b>P</b>		
<b>PCCC</b>		Phòng cháy chữa cháy
<b>PPM</b>	Parts Per Million	Đơn vị đo nồng độ (phần triệu)
<b>PCB</b>	Printed Circuit Board	Mạch in (bảng mạch điện tử)
<b>R</b>		
<b>RGB</b>	Red – Green – Blue	Hệ màu cơ bản dùng trong hiển thị hình ảnh
<b>RAM</b>	Random Access Memory	Bộ nhớ truy cập ngẫu nhiên
<b>RTMP</b>	Real – Time Messaging Protocol	Giao thức truyền phát video thời gian thực
<b>S</b>		
<b>SMC</b>	Sliding Mode Control	Điều khiển trượt
<b>SSD</b>	Single Shot Multibox Detector	Mô hình phát hiện vật thể thời gian thực
<b>T</b>		
<b>TS</b>	Transport Stream	Phân đoạn video định dạng .ts
<b>Y</b>		
<b>YOLO</b>	You Only Look Once	Mô hình nhận diện vật thể thời gian thực



## DANH MỤC HÌNH VẼ

Hình 1-1: Hiện trường một vụ cháy .....	1
Hình 2-1: Sơ đồ hoạt động của HTTP .....	7
Hình 2-2: Cấu trúc của HTTP request message .....	9
Hình 2-3: Minh họa đa nền tảng của Flutter .....	10
Hình 2-4: So sánh mô hình truyền thống với mô hình dùng Firebase .....	11
Hình 2-5: Lịch sử phát triển của Firebase .....	12
Hình 2-6: Các lợi ích của Firebase .....	12
Hình 2-7: Mô tả hoạt động của Firebase Authentication .....	13
Hình 2-8: Mô tả cách thức hoạt động của FCM .....	14
Hình 2-9: Giao diện PCB khi thiết kế bằng Altium .....	15
Hình 2-10: Phân loại hình ảnh .....	17
Hình 2-11: Phát hiện đối tượng .....	17
Hình 2-12: Phân đoạn hình ảnh .....	18
Hình 2-13: So sánh YOLOv8 với các phiên bản khác .....	20
Hình 2-14: So sánh số lượng tham số .....	21
Hình 2-15: So sánh độ trễ .....	21
Hình 2-16: Kiến trúc của YOLOv8 .....	22
Hình 2-17: Biểu diễn thang màu xám .....	24
Hình 2-18: Biểu diễn hình ảnh màu RGB .....	24
Hình 2-19: Kiến trúc CNN .....	25
Hình 2-20: Phép tính ma trận lớp chập .....	25
Hình 2-21: Ví dụ về cách tính feature map .....	26
Hình 2-22: Một số ví dụ về các Kernel .....	26
Hình 2-23: Ví dụ về Padding .....	27
Hình 2-24: Phép nhân ma trận với ảnh màu RGB .....	28
Hình 2-25: Ví dụ về Pooling Layer .....	28
Hình 2-26: Hai phương pháp pooling thông dụng .....	29
Hình 2-27: Lớp kết nối đầy đủ .....	29
Hình 2-28: Quá trình làm phẳng Flattening .....	30
Hình 2-29: Cách tính chỉ số IoU .....	32
Hình 2-30: Sự chồng chéo giữa các box .....	32
Hình 2-31: Trường hợp True Positive .....	34
Hình 2-32: Trường hợp False Positive .....	34
Hình 2-33: Thuật toán Non – max Suppression .....	35
Hình 2-34: Ví dụ về thuật toán NMS .....	35
Hình 2-35: Mô hình tổng quát sử dụng RTMP .....	37

Hình 2-36: Sơ đồ pipeline GStreamer .....	40
Hình 2-37: Bộ mã hóa và giải mã H.264 .....	42
Hình 2-38: Bảng cơ sở dữ liệu sensor_data .....	44
Hình 2-39: Bảng cơ sở dữ liệu detections .....	44
Hình 2-40: Bảng cơ sở dữ liệu fcm_tokens .....	45
Hình 3-1: Thiết bị báo cháy độc lập .....	46
Hình 3-2: Hệ thống báo cháy thường .....	48
Hình 3-3: Hệ thống báo cháy địa chỉ .....	49
Hình 3-4: Hệ thống báo cháy thông minh .....	50
Hình 3-5: Sơ đồ khái quát của hệ thống .....	55
Hình 3-6: Sơ đồ thuật toán tổng quát .....	57
Hình 3-7: Biểu đồ ca sử dụng tổng quan .....	63
Hình 3-8: Sơ đồ hoạt động chức năng Đăng ký tài khoản .....	64
Hình 3-9: Sơ đồ tuần tự chức năng Đăng ký tài khoản .....	65
Hình 3-10: Sơ đồ hoạt động chức năng Đăng nhập .....	66
Hình 3-11: Sơ đồ tuần tự chức năng Đăng nhập .....	67
Hình 3-12: Sơ đồ hoạt động chức năng Đăng xuất .....	68
Hình 3-13: Sơ đồ tuần tự chức năng Đăng xuất .....	68
Hình 3-14: Sơ đồ hoạt động chức năng Quên mật khẩu .....	69
Hình 3-15: Sơ đồ tuần tự chức năng Quên mật khẩu .....	70
Hình 3-16: Sơ đồ hoạt động chức năng Thông báo .....	71
Hình 3-17: Sơ đồ tuần tự chức năng Thông báo .....	71
Hình 3-18: Sơ đồ hoạt động chức năng Quản lý thông tin tài khoản .....	72
Hình 3-19: Sơ đồ tuần tự chức năng Quản lý thông tin tài khoản .....	73
Hình 3-20: Sơ đồ hoạt động chức năng xem dữ liệu đo được từ cảm biến .....	74
Hình 3-21: Sơ đồ tuần tự chức năng Xem dữ liệu đo được từ cảm biến .....	74
Hình 3-22: Sơ đồ hoạt động chức năng Xem video trực tiếp .....	75
Hình 3-23: Sơ đồ tuần tự chức năng Xem video trực tiếp .....	75
Hình 3-24: Sơ đồ hoạt động chức năng Kết nối với cảm biến .....	76
Hình 3-25: Sơ đồ tuần tự chức năng Kết nối với cảm biến .....	77
Hình 4-1: Raspberry Pi 4 .....	80
Hình 4-2: NVIDIA Jetson Nano Developer Kit .....	82
Hình 4-3: BeagleBone Black – Rev C .....	84
Hình 4-4: NVIDIA Jetson Xavier NX .....	86
Hình 4-5: Arduino Portenta H7 .....	89
Hình 4-6: Nguồn 5V 3A .....	93
Hình 4-7: Thẻ microSD 64GB .....	94

Hình 4-8: Camera OV5647 .....	94
Hình 4-9: ESP32 DEV KIT 30 chân.....	97
Hình 4-10: Cảm biến DHT11.....	98
Hình 4-11: Cảm biến MQ-7 .....	100
Hình 4-12: Cảm biến MP-2.....	101
Hình 4-13: Còi báo 5V .....	101
Hình 4-14: Nguồn 5V 2A.....	102
Hình 4-15: Sơ đồ nguyên lý mạch cảm biến.....	103
Hình 4-16: Hình ảnh 3D mạch PCB .....	104
Hình 4-17: Đồ thị mối quan hệ giữa $Rs/R0$ và nồng độ khí (ppm).....	106
Hình 4-18: Đồ thị mối quan hệ giữa VRL và nồng độ khí (ppm) .....	107
Hình 4-19: Sơ đồ kết nối DHT11 với vi điều khiển.....	107
Hình 4-20: Sơ đồ timing giao tiếp của DHT11 .....	108
Hình 4-21: Cách gán nhãn cho ảnh .....	109
Hình 4-22: Dataset sử dụng.....	110
Hình 4-23: Kết quả sau khi train .....	110
Hình 4-24: Confusion Matrix .....	111
Hình 4-25: Confusion Matrix Normalized .....	112
Hình 4-26: F1-Confidence Curve.....	112
Hình 4-27: Precision-Confidence Curve .....	113
Hình 4-28: Precision-Recall Curve .....	113
Hình 4-29: Recall-Confidence Curve .....	114
Hình 4-30: Kết quả chung .....	114
Hình 4-31: train_batch1 .....	115
Hình 4-32: train_batch0 .....	115
Hình 4-33: val_batch0_pred.....	116
Hình 4-34: val_batch0_labels .....	116
Hình 4-35: Cảnh báo hỏa hoạn trên app .....	121
Hình 4-36: Dữ liệu từ cảm biến lưu vào database.....	121
Hình 4-37: Giao diện xem video trực tiếp trên app .....	122
Hình 4-38: Giao diện đăng nhập .....	124
Hình 4-39: Các file .dart điều khiển logic của phần mềm .....	124
Hình 4-40: Giao diện đăng nhập .....	125
Hình 4-41: Giao diện đăng ký tài khoản .....	126
Hình 4-42: Giao diện quên mật khẩu .....	126
Hình 4-43: Giao diện màn hình chính.....	127
Hình 4-44: Giao diện quản lý thông tin tài khoản.....	127

Hình 4-45: Giao diện kết nối với cảm biến.....	128
Hình 4-46: Giao diện xem cảnh báo .....	128
Hình 4-47: Giao diện xem video trực tiếp .....	129
Hình 4-48: Nhận diện lửa.....	130
Hình 4-49: Nhận diện khói.....	130
Hình 4-50: Hình ảnh thực tế bộ cảm biến .....	144
Hình 4-51: Hình ảnh thực tế phần detect .....	144
Hình 4-52: Kết quả đo công suất tiêu thụ của bộ cảm biến .....	146
Hình 4-53: Kết quả đo công suất tiêu thụ của Raspberry Pi 4 kết nối camera ..	146

## DANH MỤC BẢNG BIỂU

Bảng 1-1: Một số vụ cháy gây thiệt hại lớn .....	2
Bảng 2-1: Một số phương thức HTTP request thường dùng .....	9
Bảng 2-2: Bảng đánh giá và so sánh các phiên bản của mô hình YOLOv8 .....	21
Bảng 3-1: Bảng ưu/nhược điểm của đầu báo cháy độc lập.....	47
Bảng 3-2: Ưu/nhược điểm của hệ thống báo cháy thường .....	48
Bảng 3-3: Bảng yêu cầu dữ liệu .....	58
Bảng 3-4: Bảng yêu cầu về người dùng .....	59
Bảng 3-5: Bảng yêu cầu chức năng.....	60
Bảng 4-1: Bảng so sánh các loại máy tính Nhúng .....	91
Bảng 4-2: So sánh các loại vi điều khiển .....	95
Bảng 4-3: Bảng so sánh các cảm biến nhiệt độ, độ ẩm.....	98
Bảng 4-4: Bảng so sánh các cảm biến khí CO .....	99
Bảng 4-5: Bảng hiệu chỉnh giá trị R0 cho cảm biến .....	105
Bảng 4-6: Bảng mô tả các bước kiểm thử chức năng Đăng ký tài khoản.....	131
Bảng 4-7: Bảng kết quả kiểm thử chức năng đăng ký .....	131
Bảng 4-8: Bảng mô tả các bước kiểm thử chức năng Đăng nhập.....	132
Bảng 4-9: Bảng kết quả kiểm thử chức năng Đăng nhập .....	132
Bảng 4-10: Bảng mô tả các bước kiểm thử chức năng Quên mật khẩu.....	133
Bảng 4-11: Bảng kết quả kiểm thử chức năng Quên mật khẩu .....	133
Bảng 4-12: Bảng mô tả các bước kiểm thử chức năng xem dữ liệu từ cảm biến .....	134
Bảng 4-13: Bảng kết quả kiểm thử chức năng xem dữ liệu từ cảm biến .....	134
Bảng 4-14: Bảng mô tả các bước kiểm thử chức năng xem camera giám sát ...	135
Bảng 4-15: Bảng kết quả kiểm thử chức năng xem camera giám sát.....	135
Bảng 4-16: Bảng mô tả các bước kiểm thử chức năng còi báo .....	136
Bảng 4-17: Bảng kết quả kiểm thử chức năng còi báo .....	136
Bảng 4-18: Bảng kết quả thực nghiệm TPR .....	138
Bảng 4-19: Bảng kết quả thực nghiệm FAR & FPR.....	140
Bảng 4-20: Bảng kết quả thực nghiệm khả năng chịu lỗi .....	141



## CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

Chương này trình bày các mục đích và yêu cầu của đề tài, đồng thời mô tả phạm vi và phương pháp nghiên cứu được áp dụng để thực hiện đề tài.

### 1.1 Đặt vấn đề

Hỏa hoạn là một trong những hiểm họa có thể gây thiệt hại nghiêm trọng đến tính mạng, tài sản và môi trường. Hàng năm nước ta xảy ra hàng nghìn vụ cháy, cướp đi sinh mạng của rất nhiều người và gây ra tổn thất nặng nề về mặt kinh tế. Đặc biệt là tại các đô thị đông dân cư và có cơ sở hạ tầng phức tạp như thủ đô Hà Nội thì nguy cơ hỏa hoạn luôn hiện hữu và có thể bùng phát bất cứ lúc nào nếu không có biện pháp phòng ngừa và ứng phó kịp thời.



Hình 1-1: Hiện trường một vụ cháy

Theo thông tin của Cục Cảnh sát Phòng cháy, chữa cháy và cứu nạn cứu hộ thì trong năm 2024 cả nước đã ghi nhận 4.112 vụ cháy, làm chết 100 người và bị thương 89 người, thiệt hại về tài sản ước tính khoảng 657,45 tỷ đồng và 637,08 ha rừng. Trong số này, 48 vụ cháy lớn và 51 vụ cháy gây thiệt hại nghiêm trọng về người, khiến 99 người chết và 89 người bị thương. So sánh với năm 2023, số vụ cháy tăng 13,67% ( $4.112/3.550$  vụ), nhưng số người chết giảm 50% ( $100/150$  người) và số người bị thương giảm 28,09% ( $89/114$  người), tuy nhiên thiệt hại về tài sản lại tăng 37,73%.

Cũng theo thống kê của Cục Cảnh sát Phòng cháy, chữa cháy và cứu nạn cứu hộ về tình hình cháy 6 tháng đầu năm 2024, Cháy chủ yếu xảy ra tại khu vực thành thị với 1.343 vụ (chiếm 60,4%), nông thôn xảy ra 879 vụ (chiếm 39,6%). Trong đó cháy tại loại hình nhà dân vẫn chiếm tỷ lệ cao nhất với 37% (823 vụ); vụ cháy kho, cơ sở sản xuất, kinh doanh chiếm 12,2% (271 vụ). Các loại hình khác, mỗi loại hình đều chiếm tỷ lệ dưới 10%.

Theo thống kê, 1.873 vụ cháy (chiếm 74,83% trong số 2.503 vụ đã điều tra) có nguyên nhân xuất phát từ sự cố hệ thống, thiết bị điện. Đây là lí do chính khiến tình trạng cháy nổ gia tăng ở khu vực thành thị, nơi có tốc độ đô thị hóa và sản xuất nhanh. Trong số này, các vụ cháy nhà ở kết hợp kinh doanh chiếm 41,58%, là khu vực đáng báo động nhất.

Bảng 1-1: Một số vụ cháy gây thiệt hại lớn

STT	Địa điểm	Nguyên nhân gây cháy	Hậu quả	Nguyên nhân tử vong
1	29/70 phố Khương Hạ, quận Thanh Xuân, Hà Nội	Do chập mạch điện trên đường dây dẫn điện tại khu vực bình ắc quy thuộc phần đầu xe mô tô sử dụng động cơ xăng.	56 người tử vong, 37 người bị thương.	Phàn lớn do ngạt khói hoặc bị thương do nhảy khỏi tòa nhà từ tầng cao.
2	Ngõ 119 Trung Kính, phường Trung Hòa, quận Cầu Giấy, Hà Nội	Do chập mạch điện trên đường dây dẫn điện tại khu vực đầu xe máy điện. Sau đó, làm cháy lớp vỏ cách điện và cháy lan ra các xe máy xung quanh.	14 người tử vong	Ngạt khí độc
3	Số 207 phố Định Công Hạ, phường Định Công, quận Hoàng Mai	Vẫn đang trong quá trình điều tra, theo nhận định ban đầu là chập điện tại phòng ngủ ở tầng 4 sau đó lan ra các tầng khác.	4 người tử vong	Ngạt khí độc

Theo những thông tin nhóm em tổng hợp được từ báo cáo điều tra của các cơ quan chức năng thì có thể tóm gọn lại nguyên nhân phổ biến dẫn đến các vụ cháy thương tâm như sau:

- Sự cố về điện:** Đây là một trong những nguyên nhân chủ yếu gây cháy, xuất phát từ tình trạng chập mạch, dây dẫn hư hỏng, thiết bị xuống cấp. Dù đây là nguyên nhân chủ yếu gây cháy nhưng lại rất khó phát hiện và kiểm soát.
- Bất cẩn trong sinh hoạt:** Nhiều vụ cháy xuất phát từ những hành vi thiếu thận trọng của con người, chẳng hạn như đốt rác không kiểm soát, hút thuốc gần vật dễ cháy hoặc sử dụng bếp lửa, các thiết bị trong nhà bếp không an toàn.

- **Hệ thống phòng cháy chữa cháy chưa đám bão:** Đây không phải là nguyên nhân gây ra hỏa hoạn nhưng lại là nguyên nhân khiến đám cháy khó kiểm soát, dẫn đến hậu quả nghiêm trọng. Những hạn chế thường gặp gồm thiếu thiết bị phòng cháy chữa cháy, thiết bị cảnh báo hỏng, hệ thống chữa cháy không đủ công suất, ...
- **Hệ thống thoát hiểm:** Những vụ cháy gây hậu quả to lớn về người thì nguyên nhân chính gây tử vong lại không phải là bong, cháy mà là ngạt khí độc. Nguyên nhân chủ yếu là do lối thoát hiểm không có hoặc bị chặn, không đạt tiêu chuẩn an toàn PCCC.

Trước thực trạng này, việc ứng dụng công nghệ để cải thiện hệ thống cảnh báo và ngăn ngừa hỏa hoạn là điều cấp thiết. Là những người trực tiếp sinh sống trong chung cư mini, nhóm sinh viên lớp Điện tử 05 – K66 chúng em nhận thức rõ mức độ nguy hiểm của hỏa hoạn và đã lựa chọn đề tài " Thiết kế hệ thống cảnh báo hỏa hoạn cho chung cư mini". Đề tài này được xây dựng dựa trên những kiến thức đã học tại Đại học Bách khoa Hà Nội, cũng như kiến thức trong quá trình phát triển bản thân, với mục tiêu phát triển hệ thống giám sát, cảnh báo sớm và tự động hóa các giải pháp ứng phó khi có sự cố cháy xảy ra.

## 1.2 Mục tiêu của đề tài

Phát triển và nghiên cứu hệ thống cảnh báo hỏa hoạn cho chung cư mini, tích hợp cảm biến khói và xử lý hình ảnh từ camera tại biên (Edge Computing) cùng với trí tuệ nhân tạo, nhằm phát hiện sớm dấu hiệu hỏa hoạn, tối ưu hóa hiệu năng và chi phí triển khai để hướng đến ứng dụng thực tế và thương mại hóa.

Mục tiêu cụ thể:

- Tìm hiểu và nghiên cứu hệ thống cảnh báo hỏa hoạn.
- Tối ưu hóa hiệu năng và chi phí triển khai.
- Ứng dụng thực tiễn và kiểm nghiệm hệ thống.

## 1.3 Mục tiêu chức năng

### 1.3.1 Phát hiện sớm nguy cơ cháy

- **Thông qua cảm biến:** Hệ thống tích hợp các cảm biến khói được lắp đặt tại các vị trí trọng yếu trong không gian chung cư mini. Các cảm biến này liên tục giám sát nồng độ hạt bụi mịn và khí cháy. Mọi biến đổi bất thường sẽ được ghi nhận và đánh giá nhằm phát hiện sớm khả năng xảy ra hỏa hoạn.
- **Thông qua trí tuệ nhân tạo (AI):** Dữ liệu thu thập từ cảm biến, cùng với hình ảnh hoặc video từ camera, sẽ được xử lý bởi các thuật toán học máy. Hệ thống AI được huấn luyện để nhận diện các đặc điểm và biểu hiện của hỏa hoạn trong nhiều điều kiện môi trường khác nhau, từ đó đưa ra các cảnh báo chính xác và kịp thời.

### 1.3.2 Xử lý dữ liệu tại biên (Edge Computing)

- **Xử lý dữ liệu tại biên:** Dữ liệu thu thập từ cảm biến và camera được phân tích ngay tại tầng, giúp giảm thiểu độ trễ so với khi truyền tải thông tin đến hệ thống trung tâm. Điều này cho phép hệ thống phản ứng nhanh chóng trong tình huống khẩn cấp.
- **Tính độc lập và độ tin cậy cao:** Mỗi đơn vị xử lý hoạt động độc lập, hạn chế việc cả hệ thống bị ảnh hưởng khi có một đơn vị gặp sự cố, giúp nâng cao khả năng chịu lỗi và đảm bảo sự ổn định của toàn hệ thống.

### 1.3.3 Tích hợp trí tuệ nhân tạo (AI) để xử lý hình ảnh

- Nghiên cứu, ứng dụng các thuật toán AI để phân tích hình ảnh từ camera và nhận diện dấu hiệu cháy, từ đó xác định mức độ nghiêm trọng của tình huống.
- Kết hợp dữ liệu từ cảm biến khói và hình ảnh để đưa ra quyết định cảnh báo chính xác.

### 1.3.4 Tạo cơ chế cảnh báo đa tầng và can thiệp thủ công

- **Tổng hợp dữ liệu thu được:** Hệ thống sẽ liên tục thu thập và phân tích dữ liệu từ cảm biến và hình ảnh/video từ camera xử lý bởi AI. Việc tích hợp này cho phép hệ thống đánh giá chính xác tình trạng cháy dựa trên nhiều nguồn dữ liệu khác nhau.
- **Cảnh báo nội bộ và kích hoạt sơ tán:** Ngay khi phát hiện dấu hiệu cháy qua cảm biến hoặc khi hình ảnh được AI nhận diện có đặc trưng của đám cháy, hệ thống sẽ kích hoạt cảnh báo nội bộ tại phòng bằng cách bật đèn báo. Sau một khoảng thời gian, nếu không nhận được phản hồi từ người dùng và tình trạng nguy hiểm được xác định rõ ràng, hệ thống sẽ tự động gửi cảnh báo chi tiết qua ứng dụng di động đến chủ căn hộ và ban quản lý chung cư. Điều này sẽ giúp người dùng kịp thời kiểm soát và can thiệp khi cần thiết.
- **Kích hoạt đèn thoát hiểm:** Trong trường hợp xảy ra cháy, các đèn thoát hiểm được bố trí tại các vị trí chiến lược (gần cửa ra vào và lối thoát hiểm) sẽ tự động bật sáng. Điều này chỉ dẫn con đường an toàn, giúp mọi người nhanh chóng định hướng và sơ tán khỏi khu vực nguy hiểm.

## 1.4 Mục tiêu phi chức năng

- **Tính bảo mật:** Đảm bảo chỉ những người dùng được ủy quyền mới có thể đăng nhập vào ứng dụng giám sát để xem hình ảnh và video từ camera. Hệ thống sẽ triển khai các cơ chế xác thực đơn nhằm ngăn chặn truy cập trái phép, mà không yêu cầu bảo vệ phức tạp dữ liệu cá nhân.

- **Độ tin cậy và ổn định:** Hệ thống phải luôn duy trì hoạt động liên tục trong mọi điều kiện. Có thể nghiên cứu, tích hợp các giải pháp dự phòng, theo dõi trạng thái liên tục và cơ chế tự phục hồi khi gặp sự cố để đảm bảo hiệu suất ổn định và tính sẵn sàng cao của toàn bộ hệ thống.
- **Khả năng mở rộng:** Với thiết kế theo mô hình phân mảnh, hệ thống có thể dễ dàng mở rộng hoặc bổ sung thêm các loại cảm biến và tính năng mới mà không ảnh hưởng đến hoạt động hiện tại. Điều này tạo điều kiện thuận lợi cho việc nâng cấp hệ thống khi có yêu cầu phát triển trong tương lai.
- **Dễ sử dụng:** Giao diện của hệ thống phải thân thiện, trực quan và dễ thao tác, cho phép người dùng ở mọi trình độ có thể nhanh chóng làm quen và sử dụng hiệu quả.
- **Hiệu suất:** Hệ thống phải đảm bảo tốc độ phản hồi nhanh chóng với độ trễ ở mức tối thiểu, nhất là trong các tình huống khẩn cấp.

## 1.5 Phạm vi nghiên cứu

Đề tài tập trung vào việc nghiên cứu và triển khai thử nghiệm một hệ thống nhà thông minh phòng cháy chữa cháy sử dụng mạng không dây. Nội dung của đề tài bao gồm các kiến thức sau:

- **Phân tích và thiết kế hệ thống nhúng:** Xác định các yêu cầu cho hệ thống, nghiên cứu các công nghệ và giải pháp hiện có, từ đó chọn ra phương án, giải pháp phù hợp nhất cho hệ thống.
- **Lập trình cho vi điều khiển:** Tìm hiểu và sử dụng các bộ công cụ phát triển phần mềm cho vi điều khiển để lập trình các chức năng như nhận diện đám cháy và điều khiển các thiết bị nhúng. Triển khai chương trình phần mềm trên vi điều khiển để đảm bảo tính ổn định và hiệu suất của hệ thống.
- **Phân tích thiết kế hướng đối tượng:** Sử dụng phương pháp phân tích thiết kế hướng đối tượng để xác định và tổ chức các đối tượng, lớp, và mối quan hệ trong hệ thống. Xây dựng các biểu đồ lớp, biểu đồ tuần tự, và biểu đồ hoạt động để mô tả cấu trúc và hoạt động của hệ thống.
- **Trí tuệ nhân tạo:** Ứng dụng trí tuệ nhân tạo để phân tích dữ liệu từ camera, phát hiện sớm dấu hiệu cháy.

## 1.6 Phương pháp nghiên cứu

- Tổng hợp lại kiến thức đã học trong chương trình đào tạo tại trường.
- Tìm kiếm, nghiên cứu các bài báo khoa học, tạp chí, luận văn, và báo cáo kỹ thuật liên quan đến đề tài.

- Tham khảo các tài liệu liên quan đến công cụ được sử dụng cho hệ thống.
- Nghiên cứu về các tiêu chuẩn về phòng cháy chữa cháy của các tổ chức uy tín. Khảo sát các giải pháp hiện có trên thị trường.
- Tham khảo ý kiến trên các diễn đàn.
- Hỗ trợ từ giảng viên hướng dẫn.

## 1.7 Đối tượng hướng đến

- **Chung cư mini:** Hệ thống được thiết kế phù hợp cho các khu chung cư mini hoặc nhà trọ nhiều tầng nhằm hạn chế mối nguy hại về hỏa hoạn đối với cư dân xuống mức thấp nhất.
- **Hộ gia đình cá nhân:** Các hộ gia đình có thể mua và lắp đặt hệ thống này tại nhà. Với khả năng phát hiện sớm nguy cơ hỏa hoạn và cảnh báo tự động, sản phẩm giúp tăng cường an toàn, đặc biệt khi không có người ở nhà hoặc vào ban đêm, từ đó bảo vệ an toàn cho tất cả các thành viên trong gia đình.
- **Nhà xưởng:** Với việc xử lý dữ liệu tại biên, hệ thống này cũng phù hợp để sử dụng tại các nhà xưởng. Tuy nhiên, trong một môi trường khác, điều kiện khác sẽ cần một số điều chỉnh và cải tiến.

## CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Chương này trình bày các kiến thức nền tảng phục vụ cho quá trình xây dựng và triển khai hệ thống cảnh báo hỏa hoạn. Các nội dung bao gồm các giao thức truyền thông phổ biến như HTTP, các công nghệ IoT, cảm biến môi trường và các phương pháp phát hiện cháy bằng thị giác máy tính. Việc hiểu rõ cơ sở lý thuyết sẽ giúp đảm bảo tính đúng đắn và hiệu quả cho các lựa chọn kỹ thuật trong quá trình thiết kế hệ thống.

### 2.1 Giao thức HTTP

#### 2.1.1 Giới thiệu về giao thức HTTP

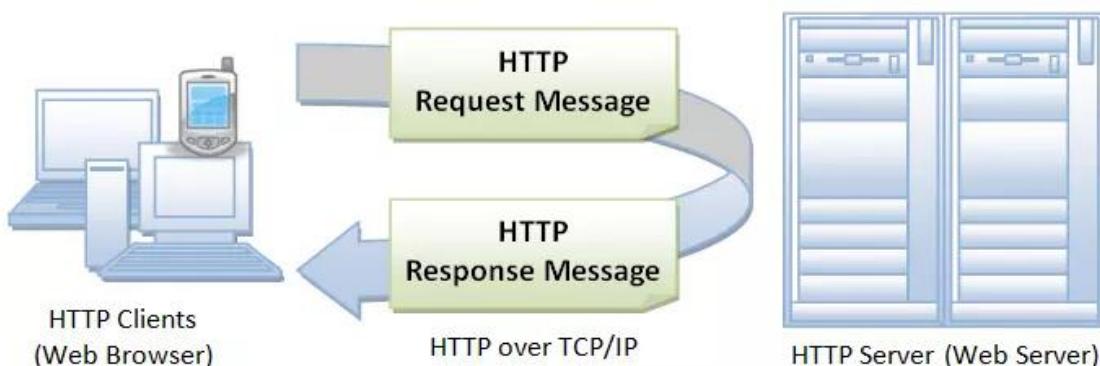
HTTP viết tắt của HyperText Transfer Protocol – Giao thức truyền tải siêu văn bản. Đây là một giao thức truyền thông được sử dụng phổ biến trên internet để truyền tải dữ liệu giữa client (trình duyệt, ứng dụng) và server (máy chủ web). Đây là một giao thức ứng dụng của bộ giao thức TCP/IP.

#### 2.1.2 Vai trò của HTTP

HTTP là nền tảng cho việc truy cập web, cho phép:

- Client yêu cầu dữ liệu từ server.
- Server phản hồi dữ liệu để client hiển thị hoặc xử lý.

#### 2.1.3 Sơ đồ hoạt động của HTTP



Hình 2-1: Sơ đồ hoạt động của HTTP

HTTP hoạt động dựa trên mô hình Client – Server. Trong mô hình này, các máy tính của người dùng sẽ đóng vai trò làm máy khách (Client). Sau một thao tác nào đó của người dùng, các máy khách sẽ gửi yêu cầu đến máy chủ (Server) và chờ đợi câu trả lời từ máy chủ.

HTTP là một stateless protocol. Hay nói cách khác, request hiện tại không biết những gì đã hoàn thành trong request trước đó.

HTTP cho phép tạo các yêu cầu gửi và nhận các kiểu dữ liệu, do đó cho phép xây dựng hệ thống độc lập với dữ liệu được truyền giao.

Trong HTTP, Client sẽ sử dụng URL để xác định và truy cập chính xác tài nguyên cần thiết từ Server. URL (Uniform Resource Locator) là địa chỉ duy nhất của một tài nguyên trên internet. URL có cấu trúc như sau:

**Protocol://hostname:port/path-and-file-name**

Trong một URL sẽ có 4 thành phần:

- **Protocol:** Giao thức tầng ứng dụng được sử dụng bởi Client và Server.
- **Hostname:** Tên DNS domain.
- **Port:** Cổng TCP để server lắng nghe request từ client.
- **Path-and-file-name:** Tên và vị trí của tài nguyên yêu cầu.

#### 2.1.4 Các thành phần chính của HTTP

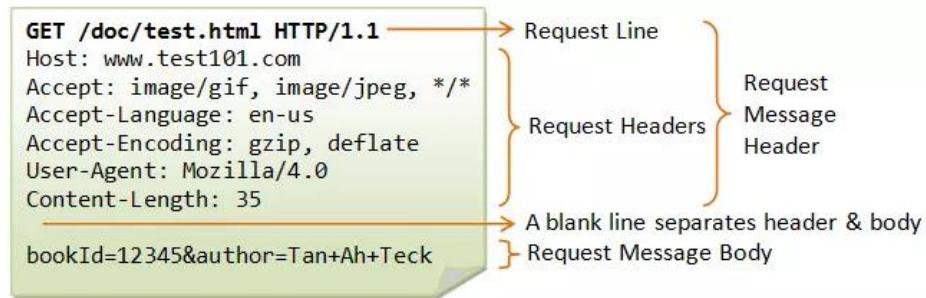
HTTP – Requests: HTTP Request Method: Là phương thức để chỉ ra hành động mong muốn được thực hiện trên tài nguyên đã xác định.

Cấu trúc của HTTP request:

- Một Request – line = Phương thức + URL – Request + Phiên bản HTTP. Giao thức HTTP định nghĩa một tập các giao thức GET, POST, HEAD, PUT ... Client có thể sử dụng một trong các phương thức đó để gửi request lên Server.
- Có thể có hoặc không có các trường header.
- Một dòng trống để đánh dấu sự kết thúc của các trường header.
- Request Header Fields: Các trường header cho phép Client truyền thông tin bổ sung về yêu cầu, và về chính Client, đến Server. Một số trường: Accept-Charset, Accept-Encoding, Accept-Language, Authorization, Expect, From, Host, ...
- Tùy chọn một thông điệp.

Khi Request đến Server, Server thực hiện một trong 3 hành động sau:

- Server phân tích Request nhận được, maps yêu cầu với tập tin trong tập tài liệu của Server, và trả lại tập tin yêu cầu Client.
- Server phân tích request nhận được, maps yêu cầu vào một chương trình trên Server, thực thi chương trình và trả lại kết quả của chương trình đó.
- Request từ client không thể đáp ứng, server trả lại thông báo lỗi.



Hình 2-2: Cấu trúc của HTTP request message

Giao thức HTTP định nghĩa một tập các phương thức request, Client có thể sử dụng một trong các phương thức này để tạo request tới HTTP Server, dưới đây liệt kê một số phương thức phổ biến.

Một số phương thức HTTP Request thường dùng:

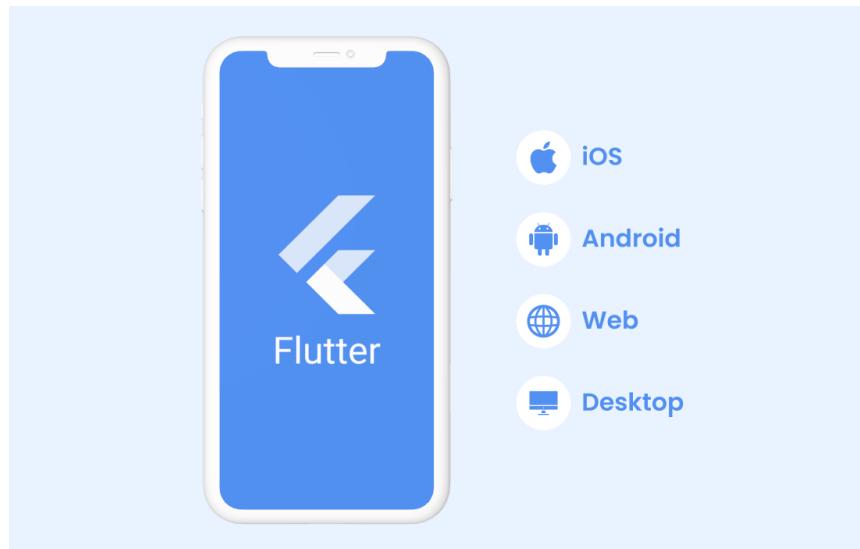
Bảng 2-1: Một số phương thức HTTP request thường dùng

Method	Hoạt động
<b>GET</b>	Được sử dụng để lấy thông tin từ Server.
<b>POST</b>	Yêu cầu máy chủ chấp nhận thực thể được đính kèm trong request được xác định bởi URI.
<b>PUT</b>	Nếu URI có đề cập đến một tài nguyên đã có, nó sẽ bị sửa đổi, nếu URI không trỏ đến một tài nguyên hiện có, thì máy chủ có thể tạo ra tài nguyên với URI đó.
<b>DELETE</b>	Xóa bỏ tất cả các đại diện của tài nguyên được chỉ định bởi URI
<b>PATCH</b>	Áp dụng cho việc sửa đổi một phần của tài nguyên được xác định

## 2.2 Flutter Framework

### 2.2.1 Flutter là gì?

Flutter là một framework mã nguồn mở do Google phát triển, cho phép lập trình viên xây dựng ứng dụng giao diện người dùng (UI) đẹp và hiệu năng cao cho nhiều nền tảng như Android, iOS,... chỉ từ một codebase duy nhất. Flutter sử dụng ngôn ngữ lập trình Dart, được thiết kế tối ưu cho việc xây dựng giao diện nhanh chóng, dễ bảo trì.



Hình 2-3: Minh họa đa nền tảng của Flutter

### 2.2.2 Lý do lựa chọn Flutter?

Nhóm em lựa chọn Flutter để xây dựng app cho đề tài này vì những lí do sau:

- **Phát triển nhanh:** Flutter hỗ trợ tính năng Hot Reload giúp người dùng nhanh chóng thay đổi ngay lập tức mà không cần phải build lại ứng dụng.
- **Đa nền tảng:** Viết một lần, triển khai được trên nhiều nền tảng, tiết kiệm thời gian và công sức so với việc phát triển ứng dụng riêng biệt cho từng nền tảng.
- **Giao diện đẹp và có thể tùy biến:** Flutter cung cấp nhiều Widget chuẩn theo Material Design (Android) và Cupertino (iOS), dễ dàng tùy chỉnh giao diện theo nhu cầu.
- **Cộng đồng lớn mạnh và tài liệu đầy đủ,** hỗ trợ tốt cho sinh viên trong quá trình học và phát triển.

### 2.2.3 Đặc điểm nổi bật của Flutter

- **Fast Development:** Flutter có một tính năng vô cùng mạnh mẽ và đặc trưng đó chính là Hot Reload. Tính năng này cho phép người dùng xem ngay kết quả thay đổi trong giao diện hoặc logic chỉ sau vài mili giây mà không cần phải build lại toàn bộ ứng dụng. Ví dụ khi người dùng thay đổi một đoạn mã Dart, chỉ cần nhấn nút Hot Reload (hoặc dùng lệnh trong terminal) thì Flutter sẽ ngay lập tức chèn đoạn mã mới đó vào nhưng vẫn giữ nguyên trạng thái hiện tại của app, giao diện sẽ cập nhật ngay tức thì mà không cần khởi động lại, mất thời gian chờ như các framework khác.
- **Expressive and Flexible UI:** Flutter cho phép xây dựng giao diện người dùng (UI) đẹp, sinh động và linh hoạt nhờ vào bộ widget phong phú và

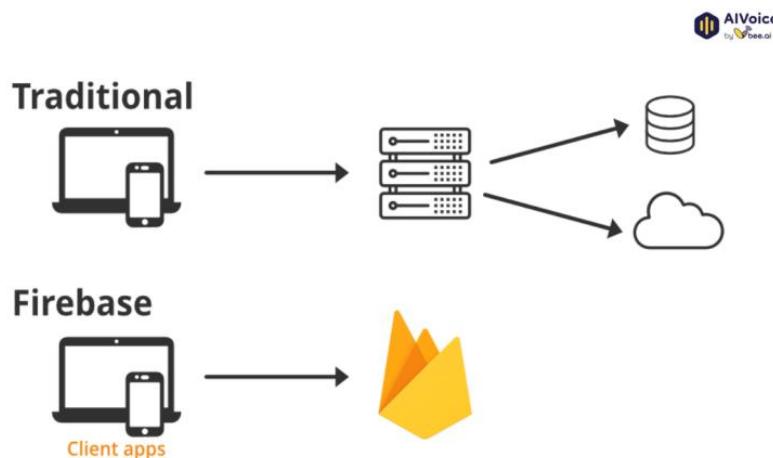
tùy biến cao, cho phép người dùng dễ dàng thiết kế giao diện hiện tại, trực quan, giàu hiệu ứng. Ví dụ như Material Design Widgets là giao diện theo chuẩn Google dành cho Android, Cupertino widgets là giao diện giống iOS như NavigationBar, Switch, Picker,...

- **Native Performance (Hiệu năng như ứng dụng gốc):** Ứng dụng được viết bằng Flutter chạy mượt mà, nhanh và tối ưu gần giống như app viết bằng ngôn ngữ native (ví dụ như Java/Kotlin cho Android và Swift cho iOS). Flutter đạt được điều này vì Flutter sử dụng Engine đồ họa riêng biệt, không cần “cầu nối” đến native UI của hệ điều hành, giao diện được vẽ trực tiếp lên màn hình với tốc độ cao.

## 2.3 Firebase

### 2.3.1 Firebase là gì?

Firebase là một nền tảng phát triển ứng dụng di động và web được Google phát triển, cung cấp các công cụ cùng dịch vụ giúp lập trình viên xây dựng, phát triển hay vận hành ứng dụng một cách nhanh chóng, hiệu quả. Với Firebase, bạn có thể dễ dàng quản lý cơ sở dữ liệu, xác thực người dùng, lưu trữ tệp tin, gửi thông báo đẩy và phân tích dữ liệu mà không cần phải tự xây dựng các hệ thống phức tạp.



Hình 2-4: So sánh mô hình truyền thống với mô hình dùng Firebase

### 2.3.2 Lịch sử phát triển của Firebase

Firebase bắt đầu hành trình của mình vào năm 2011, được thành lập bởi James Tamplin và Andrew Lee, với sản phẩm cốt lõi là Realtime Database. Dịch vụ này mang tính đột phá, cho phép đồng bộ hóa dữ liệu giữa các thiết bị một cách nhanh chóng, giải quyết bài toán lớn về đồng bộ dữ liệu trong thời gian thực.



Hình 2-5: Lịch sử phát triển của Firebase

Đến năm 2014, Google đã mua lại và tích hợp Firebase sâu hơn vào hệ sinh thái Google Cloud, bổ sung hàng loạt các dịch vụ quan trọng như xác thực người dùng, lưu trữ đám mây, máy chủ đám mây, phân tích hiệu suất và phân tích người dùng. Đặc biệt, Firebase đang được tích hợp với các công nghệ trí tuệ nhân tạo (AI) và học máy (ML) của Google, nhằm mang lại trải nghiệm phát triển ứng dụng thông minh hơn.

### 2.3.3 Lợi ích của Firebase

Firebase mang lại nhiều lợi ích, thúc đẩy nhanh quá trình phát triển ứng dụng và tối ưu hóa chi phí.



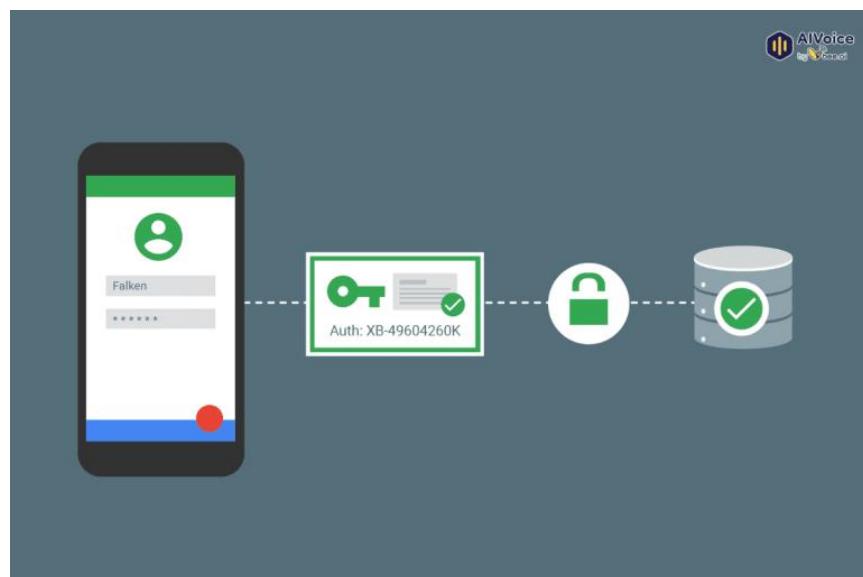
Hình 2-6: Các lợi ích của Firebase

- **Phát triển nhanh chóng:** Firebase cung cấp một loạt các dịch vụ được xây dựng sẵn, bao gồm cơ sở dữ liệu (Realtime Database và Firestore), xác thực người dùng (Authentication), lưu trữ tệp (Cloud Storage), và nhiều dịch vụ khác. Điều này giúp người dùng tiết kiệm thời gian đáng kể, không cần phải xây dựng các tính năng cốt lõi từ đầu mà chỉ cần tập trung vào việc phát triển các tính năng độc đáo cho ứng dụng của mình.
- **Đồng bộ dữ liệu thời gian thực:** Với cơ sở dữ liệu thời gian thực của mình, Firebase sẽ cập nhật dữ liệu ngay lập tức trên các thiết bị kết nối, mang lại trải nghiệm mượt mà và liền mạch.
- **Tích hợp đa nền tảng:** Firebase hỗ trợ phát triển ứng dụng trên nhiều nền tảng, bao gồm Android và iOS. Cho phép người dùng sử dụng cùng một mã nguồn để triển khai ứng dụng trên nhiều thiết bị, giúp tiết kiệm thời gian và công sức.
- **Tính bảo mật cao:** Firebase được xây dựng dựa trên nền tảng bảo mật của Google, đảm bảo an toàn cho dữ liệu người dùng. Các tiêu chuẩn bảo mật nghiêm ngặt giúp bảo vệ ứng dụng khỏi các mối đe dọa tiềm tàng.
- **Chi phí linh hoạt:** Firebase cung cấp nhiều gói dịch vụ với mức giá khác nhau, bao gồm cả gói miễn phí với chức năng cơ bản.

### 2.3.4 Các dịch vụ của Firebase được sử dụng trong đề tài

#### 2.3.4.1. Firebase Authentication – Xác thực người dùng

Firebase Authentication là một dịch vụ mạnh mẽ của Firebase, giúp đơn giản hóa quá trình quản lý và xác thực danh tính người dùng trong ứng dụng.



Hình 2-7: Mô tả hoạt động của Firebase Authentication

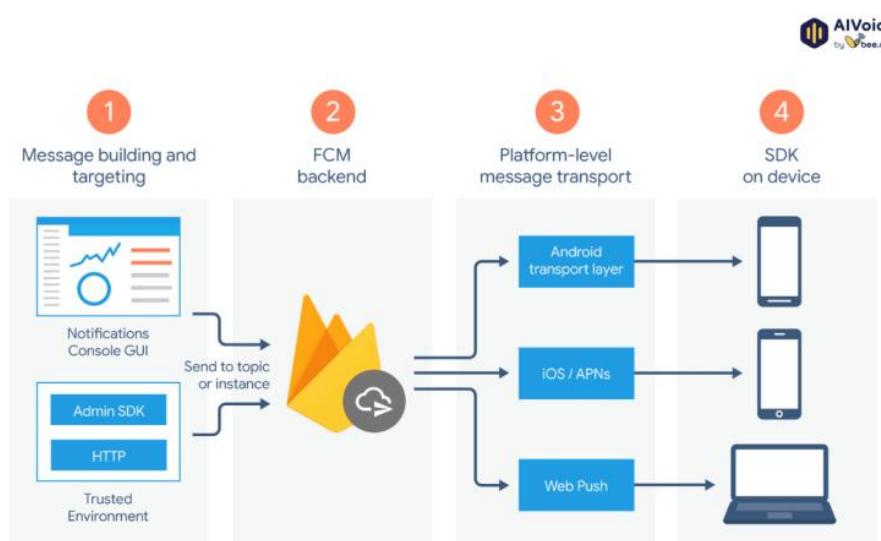
Dịch vụ này cho phép quản lý thông tin người dùng dễ dàng bao gồm việc tạo, cập nhật và xóa tài khoản. Bên cạnh đó cũng hỗ trợ nhiều phương thức đăng nhập phổ biến như Email/Password, Google, Facebook, Apple, Phone number, ...

#### 2.3.4.2. *Firestore Database*

Đây là một cơ sở dữ liệu được lưu trữ trên đám mây, nổi bật với khả năng đồng bộ dữ liệu giữa các thiết bị ngay lập tức. Đây là một giải pháp phù hợp với các ứng dụng yêu cầu cập nhật dữ liệu liên tục và đồng thời trên nhiều thiết bị.

#### 2.3.4.3. *Firebase Cloud Messaging (FCM) – Gửi thông báo đầy*

Firebase Cloud Messaging (FCM) là một dịch vụ nhắn tin đa nền tảng cho phép bạn gửi tin nhắn và thông báo một cách đáng tin cậy mà không mất phí.



Hình 2-8: Mô tả cách thức hoạt động của FCM

Dịch vụ này cho phép gửi thông báo đến người dùng trên các nền tảng Android, iOS và Web. Trong đề tài, dịch vụ này sẽ được sử dụng để gửi thông báo đến người dùng khi phát hiện nguy cơ hỏa hoạn.

### 2.3.5 **Ưu điểm và hạn chế của Firebase**

#### 2.3.5.1. *Ưu điểm*

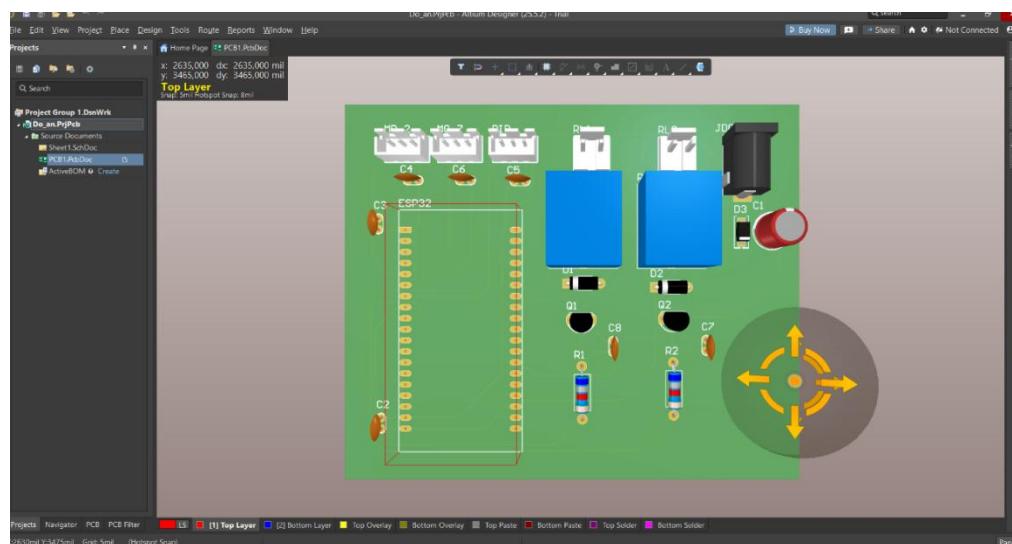
- Giảm thời gian và công sức xây dựng các dịch vụ cơ bản, cốt lõi.
- Dễ dàng tích hợp với Google Cloud, mở rộng khả năng của ứng dụng.
- Hỗ trợ đa nền tảng.

### 2.3.5.2. Hạn chế

- Giới hạn truy vấn với Firestore, tốn kém khi mở rộng.
- Không phù hợp với ứng dụng có dữ liệu quan hệ phức tạp.
- Sử dụng Firebase có thể dẫn đến sự phụ thuộc và Google, gây khó khăn khi chuyển đổi sang nền tảng khác.

## 2.4 Altium Designer

Trong đề tài này, để thiết kế mạch in PCB cho bộ cảm biến, nhóm em đã sử dụng Altium Designer. Altium Designer là một phần mềm thiết kế mạch in (PCB – Printed Circuit Board) chuyên nghiệp, được sử dụng rộng rãi trong công nghiệp điện – điện tử. Phần mềm này hỗ trợ thiết kế sơ đồ nguyên lý (schematic) và thiết kế mạch in (layout) với độ chính xác cao, đồng thời tích hợp nhiều công cụ kiểm tra, mô phỏng và xuất file sản xuất (Gerber).



Hình 2-9: Giao diện PCB khi thiết kế bằng Altium

Nhóm em lựa chọn Altium Designer vì:

- Giao diện trực quan, dễ sử dụng.
- Thiết kế chính xác: Hỗ trợ định tuyến tự động với độ chính xác cao.
- Kiểm tra lỗi mạnh mẽ: Tự động phát hiện lỗi ngắn mạch sai, sai kết nối, lỗi khoảng cách, ...
- Xuất file Gerber chuyên nghiệp.
- Tích hợp Schematic – PCB – 3D: Dễ dàng kiểm tra mối liên hệ giữa sơ đồ mạch và mạch in thực tế, có thể xem mô hình 3D.

## 2.5 Trí tuệ nhân tạo (Artificial Intelligence)

### 2.5.1 Giới thiệu về trí tuệ nhân tạo (AI)

Trí tuệ nhân tạo là một lĩnh vực trong khoa học máy tính nhằm xây dựng các hệ thống có khả năng mô phỏng trí tuệ con người như học tập, suy luận, giải quyết vấn đề, nhận thức, đưa ra quyết định. Ngày nay AI đang được ứng dụng rất rộng rãi trong nhiều lĩnh vực như y tế, giao thông, giáo dục, tài chính quản lý,... và đặc biệt là thị giác máy tính (Computer Vision).

Một số nhánh chính của AI:

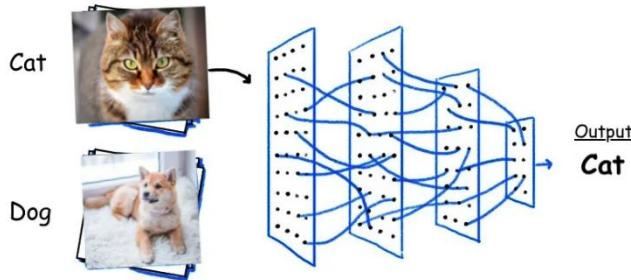
- **Machine Learning (Học máy):** nghiên cứu về các thuật toán giúp máy tính học từ dữ liệu mẫu để cải thiện bản thân chúng. ML có thể tự dự đoán hoặc đưa ra quyết định mà không cần được lập trình cụ thể.
- **Deep Learning (Học sâu):** hoạt động chủ yếu dựa trên mạng thần kinh nhân tạo để bắt chước khả năng tư duy và suy nghĩ của bộ não con người. Gọi là học sâu vì nó bao gồm nhiều lớp đơn vị tính toán neurons tạo thành mạng neural phức tạp.
- **Computer Vision (Thị giác máy tính):** nghiên cứu cách để máy tính có thể “nhìn” và “hiểu” được thế giới xung quanh thông qua hình ảnh và video. Mục đích để giúp máy tính phân tích, xử lý thông tin từ hình ảnh và video như con người.

### 2.5.2 Thị giác máy tính

Ngày nay, thị giác máy tính đang ngày càng phát triển giúp nâng cấp cuộc sống của con người. Những đổi mới trong phần cứng và những tiến bộ trong các thuật toán đang tạo ra cơ hội cho những mục tiêu tưởng chừng như không thể đạt được. Tất cả các hệ thống máy móc trong cuộc sống đang ngày càng thông minh và mang lại hiệu suất vô cùng tốt cho thấy được lĩnh vực AI nói chung, cụ thể là thị giác máy tính nói riêng chiếm lĩnh vị trí rất quan trọng để đưa công nghệ kĩ thuật nâng lên tầm cao mới.

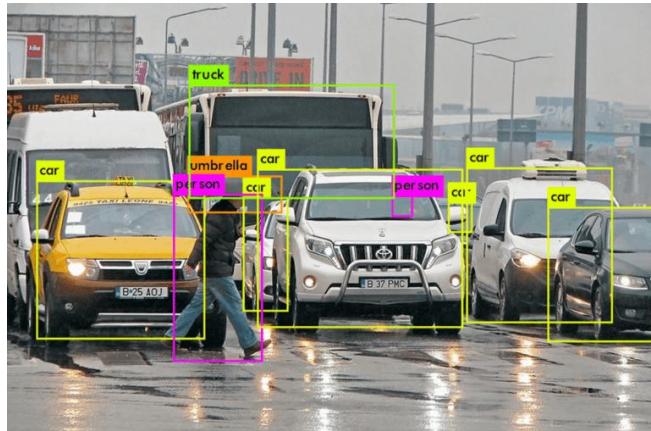
Trong thị giác máy tính, cũng có nhiều các nhánh khác, nhưng có các kỹ thuật cốt lõi sau:

- **Phân loại hình ảnh (Image Classification):** là quá trình kiểm tra hình ảnh và dán nhãn phân loại trên nội dung. Để phân loại được thì máy đã được huấn luyện qua mô hình với tập dữ liệu (dataset) trước đó. Ví dụ, trong ảnh có thể phân loại được đó là chó hay mèo hay thú khác với độ chính xác cao.



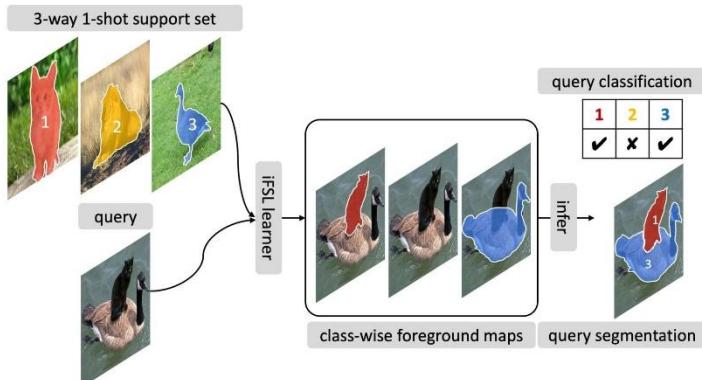
Hình 2-10: Phân loại hình ảnh

- **Phát hiện đối tượng (Object Detection):** là quá trình xác định và phân loại các đối tượng có trong ảnh hoặc video, đồng thời xác định vị trí của chúng thông qua các bounding box (hộp giới hạn).



Hình 2-11: Phát hiện đối tượng

- **Phân đoạn hình ảnh (Image Segmentation):** là quá trình phân chia ảnh thành các vùng nhỏ có ý nghĩa – mỗi vùng chứa các điểm ảnh (pixel) cùng loại, ví dụ như cùng thuộc đối tượng như người, xe, cây, con vật,... Hay dễ hiểu hơn đó kỹ thuật nâng cao hơn phát hiện đối tượng khi các bounding box bao quanh viền các đối tượng trong ảnh.



Hình 2-12: Phân đoạn hình ảnh

- **Theo dõi đối tượng (Object Tracking):** đây là bài toán theo dõi một hoặc nhiều đối tượng trong một chuỗi video. Mục tiêu là nhận diện và xác định vị trí của các đối tượng được quan tâm trong mỗi khung hình và liên kết chúng qua các khung để theo dõi chuyển động của chúng theo thời gian.

Thị giác máy tính sử dụng nhiều thuật toán khác nhau để thực hiện các nhiệm vụ trên. Một số thuật toán tiêu biểu phải kể đến như CNN cho phân loại ảnh, Faster R-CNN, SSD và đặc biệt là YOLO cho phát hiện đối tượng thời gian thực. Với đề tài của nhóm em là phát hiện và nhận diện lửa và khói trong thời gian thực, chúng em chọn sử dụng thuật toán YOLO vì có tốc độ nhanh, độ chính xác cao phù hợp với yêu cầu của hệ thống.

## 2.6 Mô hình YOLOv8

### 2.6.1 Tổng quan về bài toán phát hiện đối tượng

Trong những năm trở lại đây, object detection (phát hiện đối tượng) là một trong những đề tài rất hot của deep learning bởi khả năng ứng dụng cao, dữ liệu dễ chuẩn bị và ứng dụng rất nhiều. Các thuật toán mới của object detection như YOLO (You only look once), SSD (Single shot multibox detector) có tốc độ khá nhanh và độ chính xác cao nên giúp cho object detection có thể thực hiện được các tác vụ dường như là real-time, thậm chí là nhanh hơn so với con người mà độ chính xác không giảm. Các mô hình cũng trở nên nhẹ hơn nên có thể hoạt động trên các thiết bị IoT để tạo nên các thiết bị thông minh.

Object detection – bài toán phát hiện đối tượng: là một công nghệ máy tính liên quan đến thị giác máy tính và xử lý hình ảnh nhằm phát hiện các trường hợp của đối tượng của một lớp nhất định trong hình ảnh hoặc video kỹ thuật số. Các thuật toán Object detection có ứng dụng rất đa dạng như: Đếm số lượng vật thể, thanh toán tiền tại quầy hàng, chấm công tự động, phát hiện vật thể nguy hiểm như súng, dao, .... và rất nhiều các ứng dụng khác. Có thể nói dường bất kì lĩnh vực nào cũng đều có thể ứng dụng được object detection.

Các mô hình phát hiện đối tượng thường được đào tạo để phát hiện sự hiện diện của các đối tượng cụ thể trong hình ảnh, video hoặc hoạt động thời gian thực. Bài toán có input là ảnh màu và output là vị trí của đối tượng trong ảnh. Bài toán được chia ra làm 2 giai đoạn với 2 bài toán nhỏ là xác định các bounding box quanh đối tượng (object localization) và với mỗi bounding box thì xác định xem đó là đối tượng gì và bao nhiêu phần trăm chắc chắn (image classification).

Chính vì tính ứng dụng rất cao, dễ chuẩn bị dữ liệu và huấn luyện mô hình đơn giản nên nhóm chúng em sẽ sử dụng thuật toán object detection state-of-art, đó là YOLO. Sau đây, nhóm em xin trình bày một số đặc điểm về YOLO.

### 2.6.2 Giới thiệu về YOLO

You look only once (YOLO) là thuật toán phát hiện đối tượng (object detection) được giới thiệu vào năm 2015 trong một bài nghiên cứu của Joseph Redmon, Santosh Divvala, Ross Girshick và Ali Farhadi. Kiến trúc của YOLO là một cuộc cách mạng quan trọng trong không gian object detection theo thời gian thực, vượt qua thuật toán tiền nhiệm của nó là Mạng nơron tích chập dựa trên vùng (RCNN).

YOLO là một thuật toán thị giác máy tính được sử dụng để phát hiện đối tượng trong thời gian thực và có thể xử lý hình ảnh trong vài mini giây. Đây là một mô hình mạng CNN cho việc phát hiện, nhận dạng và phân loại đối tượng.

**Lịch sử phát triển của YOLO:** Từ năm 2015 đến nay, YOLO đã trải qua một số lần lặp và cải tiến, và phiên bản mới nhất là YOLOv12 được phát hành vào 20-02-2025. Cho đến nay, YOLO đã được phát triển thành nhiều phiên bản : YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, YOLOv8, YOLOv9, YOLOv10, YOLOv11 và YOLOv12. Mỗi phiên bản đều được xây dựng dựa trên phiên bản trước với các tính năng nâng cao như độ chính xác cải thiện hơn, xử lý nhanh hơn và xử lý tốt hơn các đối tượng nhỏ.

Hiện nay, YOLO được ứng dụng trong rất nhiều lĩnh vực quan trọng trong đời sống, đặc biệt phải kể đến các lĩnh vực:

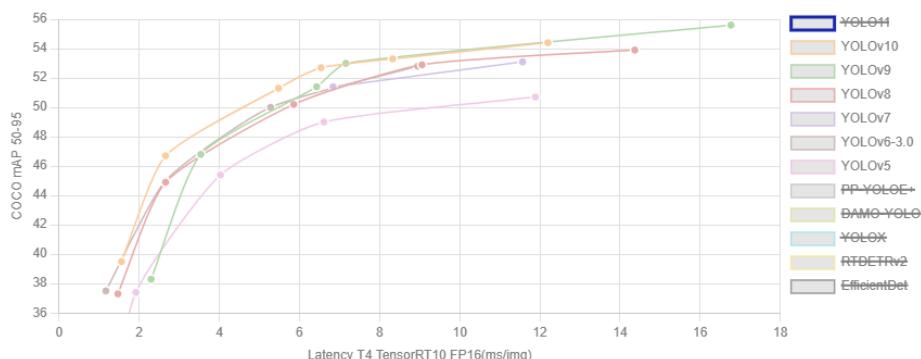
- **Trong lĩnh vực y tế:** YOLO được sử dụng để phát hiện ung thư, phân đoạn da và xác định thuốc giúp cải thiện độ chính xác chuẩn đoán và quy trình điều trị hiệu quả hơn.
- **Trong nông nghiệp:** YOLO được sử dụng để phát hiện và phân loại cây trồng, sâu bệnh, hỗ trợ kỹ thuật nông nghiệp chính xác và tự động hóa quy trình canh tác.
- **Trong viễn thám:** được sử dụng để phát hiện và phân loại đối tượng trong ảnh vệ tinh và trên không, hỗ trợ lập bản đồ sử dụng đất, quy hoạch và giám sát môi trường.

- **Trong hệ thống an ninh:** đã tích hợp các mô hình YOLO để giám sát và phân tích nguồn cấp dữ liệu video thời gian thực, cho phép phát hiện nhanh các hoạt động đáng ngờ, giãn cách xã hội và phát hiện khẩu trang. Các mô hình cũng đã được áp dụng trong kiểm tra bìa mặt để phát hiện khuyết tật và bất thường, tăng cường kiểm soát chất lượng quy trình sản xuất.
- **Trong giao thông:** các mô hình YOLO được sử dụng cho các tác vụ như phát hiện biển số xe và nhận dạng biển báo giao thông.
- **Trong tự nhiên:** được sử dụng trong việc phát hiện và giám sát động vật hoang dã để xác định các loài có nguy cơ tuyệt chủng để bảo tồn sinh học và quản lý hệ sinh thái.

YOLO cũng được điều chỉnh cho các nhiệm vụ phát hiện khuôn mặt trong hệ thống sinh trắc học, bảo mật, nhận dạng khuôn mặt và ứng dụng robot, object detection từ drones.

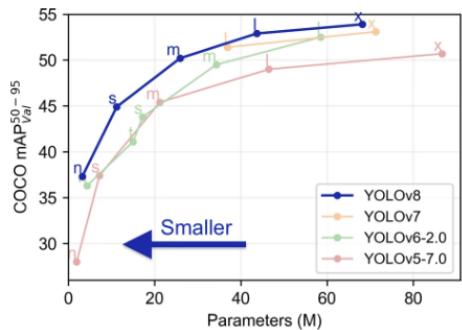
### 2.6.3 Giới thiệu về YOLOv8

YOLO đã cung cấp nhiều versions khác nhau và ngày càng cho kết quả tốt, xong nhóm em vẫn quyết định lựa chọn YOLOv8 trong project vì đây là phiên bản được phát hành từ sớm (2013) nên rất ổn định, đã được kiểm chứng rộng rãi và có nhiều tài liệu, tutorial hỗ trợ. Các version mới vẫn đang trong giai đoạn phát triển nên dễ phát sinh lỗi đặc biệt khi dùng custom dataset. YOLOv8 còn thêm việc hỗ trợ xuất mô hình sang nhiều định dạng như ONNX, NCNN phù hợp với yêu cầu triển khai trên các thiết bị nhúng đặc biệt là trên Raspberry Pi 4.

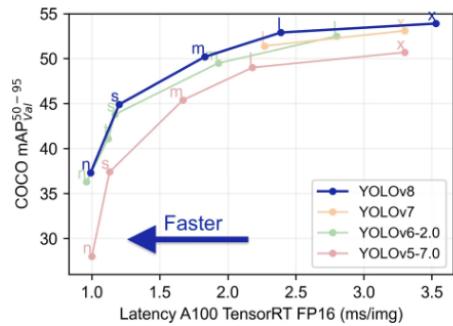


Hình 2-13: So sánh YOLOv8 với các phiên bản khác

Để đánh giá YOLOv8 so với các versions khác về số lượng tham số (Parameters) và độ trễ xử lý (Latency). Chúng em có sưu tầm được hai ảnh so sánh sau:



Hình 2-14: So sánh số lượng tham số



Hình 2-15: So sánh độ trễ

Nhận xét thấy rằng, khi so sánh trên tập dữ liệu COCO, biểu đồ thứ nhất cho thấy YOLOv8 đạt độ chính xác cao hơn so với các phiên bản trước ở cùng mức độ phức tạp mô hình, tức là ít tham số hơn nhưng vẫn hiệu quả hơn. Biểu đồ thứ hai cho thấy YOLOv8 cũng có tốc độ xử lý nhanh hơn khi chạy trên GPU A100 với TensorRT FP16. Như vậy, YOLOv8 vượt trội cả về độ chính xác và tốc độ, chứng minh là phiên bản tối ưu hơn cho các ứng dụng cần cân bằng giữa hiệu năng và độ trễ.

YOLOv8 cũng có nhiều dòng khác nhau: n-nano, s-small, m-medium, l-large và x-extra large. Thông số hiệu năng được mô tả trong bảng sau:

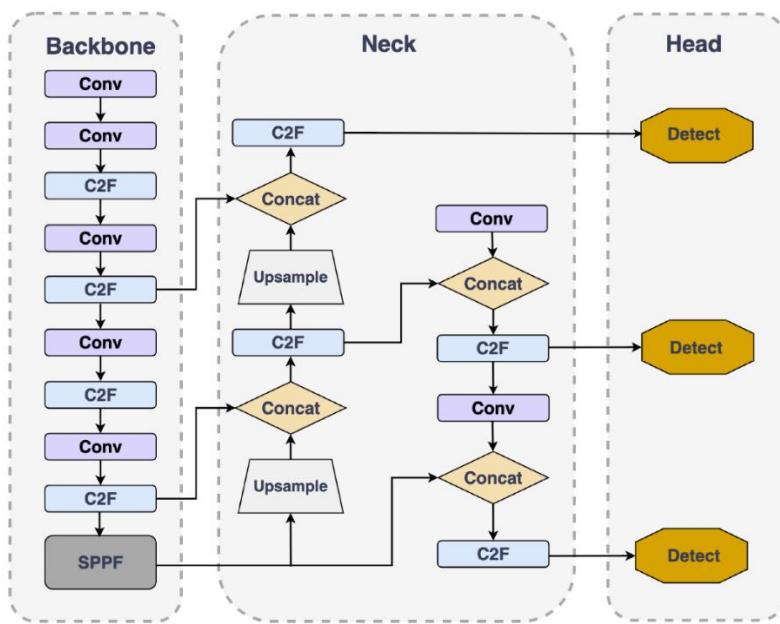
Bảng 2-2: Bảng đánh giá và so sánh các phiên bản của mô hình YOLOv8

Model	Size (pixels)	mAP <sub>50-95</sub>	Speed CPU ONNX	Speed A100 TensorRT	Parames (M)	FLOPS (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Từ bảng so sánh trên, nhận xét thấy YOLOv8n có kích thước mô hình nhỏ nhất với 3.2 triệu tham số và tốc độ xử lý nhanh nhất trên CPU, trong khi vẫn giữ được độ chính xác tương đối (mAPval = 37.3 %). Đây là lựa chọn phù hợp cho các thiết bị có tài nguyên hạn chế như Raspberry Pi4 vốn có hiệu năng CPU và RAM thấp hơn so với máy tính thông thường. Vì vậy, nhóm em đã lựa chọn YOLOv8n để huấn luyện và triển khai trên Pi4 trong dự án này. Mục tiêu là đạt được sự cân bằng giữa hiệu quả tính toán và độ chính xác, đảm bảo mô hình có thể chạy thời gian thực mà không gây quá tải cho hệ thống.

### 2.6.3.1. Kiến trúc mô hình YOLOv8

YOLOv8 tổng thể vẫn có 3 phần chính gồm Backbone, Neck và Head như kiến trúc của các version cũ những đã loại bỏ, cải tiến nhiều giúp tối ưu hơn, nhẹ hơn những vẫn giữ được độ chính xác cao. YOLOv8 sử dụng đường trực tương tự như YOLOv5 với một số thay đổi trên lớp CSP. Module C2f (nút cỗ chai một phần giao đoạn chéo với 2 tích chập) kết hợp các tính năng cấp cao với thông tin ngữ cảnh để cải thiện độ chính xác khi phát hiện.



Hình 2-16: Kiến trúc của YOLOv8

- **Backbone (Xương sống):** chịu trách nhiệm trích xuất các tính năng hữu ích từ hình ảnh input. Nó thường là một CNN đã được trained một task phân loại hình ảnh số lượng lớn. Backbone nắm bắt các tính năng phân cấp ở các quy mô khác nhau, với các tính năng thấp hơn được trích xuất trong các lớp trước đó và các tính năng cấp cao hơn được trích xuất trong các lớp sâu hơn. YOLOv8 đang sử dụng xương sống CSPDarknet53 với các kết nối để trích xuất đặc trưng của hình ảnh và áp dụng thuật toán nhận dạng đối tượng YOLOv8 trên các đặc trưng đó.
- **Neck (Cỗ):** Cỗ là một thành phần trung gian kết nối xương sống với đầu. Nó tổng hợp và tinh chỉnh các đặc điểm được trích xuất bởi xương sống, thường tập trung vào việc nâng cao thông tin không gian và ngữ nghĩa trên các quy mô khác nhau. Cỗ có thể bao gồm các lớp tích chập bổ sung, mạng kim tự tháp tính năng (FPN), hoặc các cơ chế khác để cải thiện sự biểu diễn của các đặc điểm. Thay vì mạng kim tự tháp tính năng (FPN) truyền thống, YOLOv8 sử dụng mô – đun C2f mới để kết hợp các tính năng cấp cao với thông tin ngữ cảnh để cải thiện độ chính xác khi phát hiện.

- **Head (Đầu):** là thành phần cuối cùng của mô hình phát hiện vật thể, nó chịu trách nhiệm đưa ra dự đoán dựa trên các tính năng được cung cấp bởi backbone và neck. YOLOv8 sử dụng nhiều mô – đun ở phần này, thường bao gồm một hoặc nhiều mạng con dành riêng cho nhiệm vụ phân loại, localization và gán đây là phân đoạn thực thể và ước tính dáng. Head xử lý các đặc điểm mà cổ cung cấp từ đó tạo ra các dự đoán cho từng đối tượng. Cuối cùng, một bước xử lý hậu kì, ví dụ NMS, lọc ra các dự đoán chồng chéo và chỉ giữ lại những phát hiện đáng tin cậy nhất.

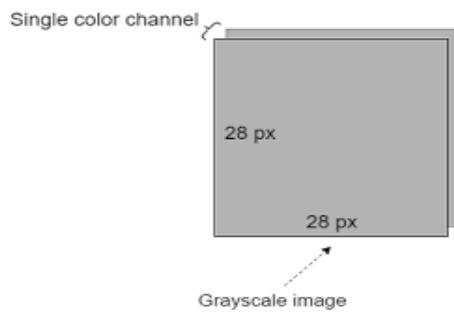
YOLOv8 sử dụng mô hình không neo với đầu tách rời để xử lý độc lập các tác vụ phân loại và hồi quy. Thiết kế này cho phép mỗi nhánh tập trung vào nhiệm vụ của mình và cải thiện độ chính xác tổng thể của mô hình. Trong lớp đầu ra của mô hình YOLOv8, sử dụng hàm Sigmoid làm hàm kích hoạt, đại diện cho xác suất hộp giưới hạn chứa một đối tượng. Nó sử dụng hàm softmax cho xác suất lớp, đại diện cho xác suất của các đối tượng thuộc về mỗi lớp có thể. YOLOv8 sử dụng những hàm mất mát là CIoU và DFL cho bounding box và entropy chéo để mất phân loại.

#### *2.6.3.2. Tổng quan về Convolution Neural Network*

Để có thể phát hiện được đối tượng trong một ảnh, YOLO chia hình ảnh thành một lưới và gán từng ô lưới để dự đoán các đối tượng trong khu vực của nó. YOLO không chỉ vẽ các bounding box xung quanh các đối tượng mà còn cho biết độ tự tin của mỗi bounding box để biết được độ chắc chắn của đối tượng. YOLO sử dụng mạng thần kinh tích chập (Convolution Neural Network CNN) – một mô hình thông minh giúp thuật toán hiểu và xử lý hình ảnh.

CNN (Convolution neural network - mạng thần kinh tích chập): Đây là một lớp các mạng thần kinh sâu trích xuất các tính năng từ hình ảnh, được đưa ra dưới dạng đầu vào, để thực hiện các tác vụ cụ thể như phân loại hình ảnh, nhận dạng khuôn mặt và hệ thống hình ảnh ngữ nghĩa. CNN có một hoặc nhiều lớp chập để trích xuất tính năng đơn giản, thực hiện hoạt động tích chập (nghĩa là nhân của một tập hợp các trọng số với đầu vào) trong khi vẫn giữ được các tính năng quan trọng (thông tin không gian và thời gian) mà không có sự giám sát của con người.

CNN hoạt động được với cả hình ảnh màu và ảnh xám. Hai thang màu này biểu diễn khác nhau trong mạng nơ-ron. CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy  $H \times W \times D$  ( $H$ : Chiều cao,  $W$ : Chiều rộng,  $D$ : Độ dày).

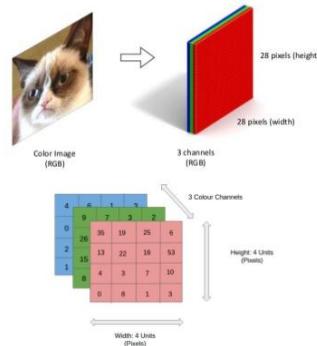


Hình 2-17: Biểu diễn thang màu xám

Với ảnh xám chỉ có một màu, vì thế được biểu diễn dưới dạng vector  $H \times W \times 1$  hoặc chỉ cần  $H \times W$ . Mỗi pixel trong ảnh xám chỉ cần biểu diễn bằng một giá trị nguyên trong khoảng từ  $[0, 255]$  thay vì (red, green, blue) như ảnh màu.

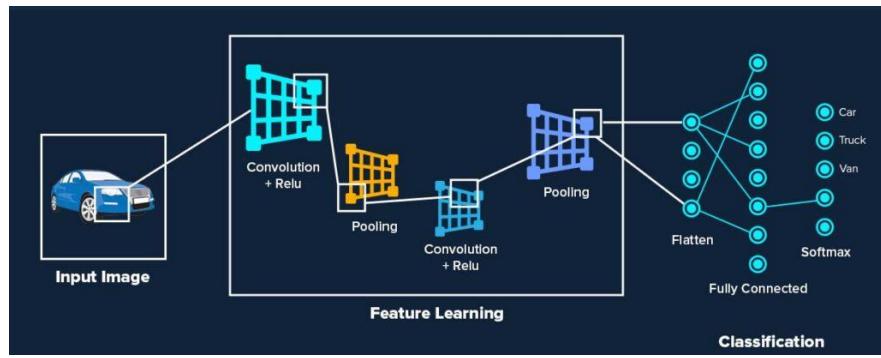
Đối với ảnh màu RGB, có 3 thang màu (red, green, blue) vì thế hình ảnh được biểu diễn dưới dạng tensor 3 chiều kích thước  $H \times W \times D$  với 3 ma trận màu red, green, blue chồng lên nhau. Ví dụ biểu diễn ảnh màu kích thước  $28 \times 28$  dưới dạng tensor  $28 \times 28 \times 3$ .

color image is 3rd-order tensor



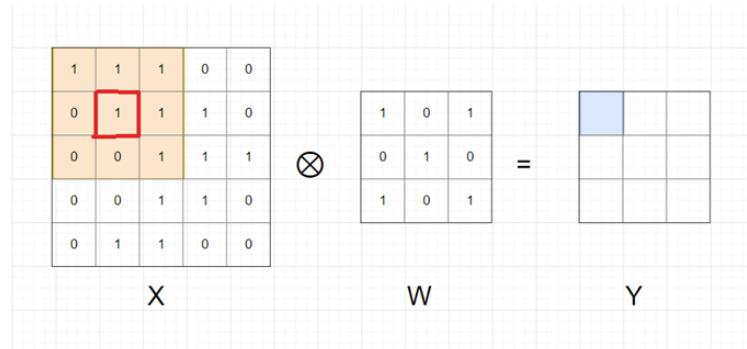
Hình 2-18: Biểu diễn hình ảnh màu RGB

Kiến trúc CNN là một kiến trúc tương đối khó, có nhiều lớp bổ sung khác nhau cho kiến trúc CNN. CNN được xây dựng bằng cách chồng nhiều lớp tích chập (Convolutional Layers), xen kẽ với lớp pooling (Pooling Layers) và kết thúc bằng lớp kết nối đầy đủ (Fully Connected Layers), giúp xử lý và phân loại hình ảnh. Ngoài ra, các hàm kích hoạt phi tuyến (như ReLU, Tanh) được sử dụng để tăng khả năng biểu diễn của mô hình. Mỗi lớp trong CNN đảm nhận một nhiệm vụ cụ thể, từ trích xuất đặc trưng đến giảm kích thước và cuối cùng là phân loại hình ảnh. Về cơ bản, CNN sẽ làm 2 việc chủ yếu : học các đặc trưng và phân loại, mỗi việc sẽ có các lớp khác nhau đảm nhiệm. Cụ thể, để học các đặc trưng sẽ gồm các lớp chập (convolution) + ReLU và lớp gộp (pooling). Còn để phân loại sẽ gồm các lớp Flatten, Fully Connected, Softmax Sau đây em xin đi chi tiết vào từng lớp để trình bày về những đặc điểm, cách hoạt động của từng lớp.



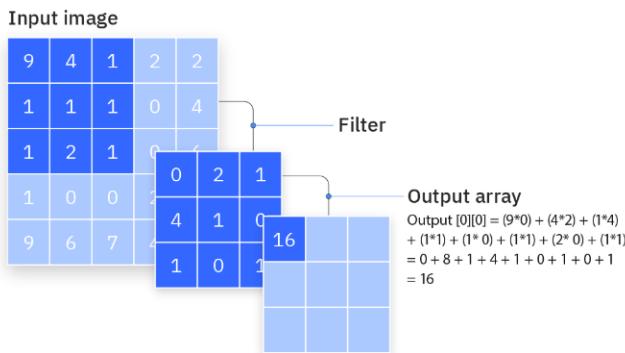
Hình 2-19: Kiến trúc CNN

- **Convolution layer (Lớp chập):** Lớp chập là lớp đầu tiên và là lớp cốt lõi để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh sử dụng các ô vuông nhỏ của dữ liệu đầu vào. Nó là một phép toán có 2 đầu vào là một ma trận hình ảnh và một bộ lọc (kernel). Ở lớp chập, thuật toán được thực hiện như sau :



Hình 2-20: Phép tính ma trận lớp chập

Ví dụ: Ma trận  $5 \times 5$  (ma trận  $X$ ) sẽ nhân với ma trận bộ lọc (kernel)  $3 \times 3$  (ma trận  $W$ ) để tại ra được một ma trận mới gọi là Feature map (ma trận  $Y$ ). Định nghĩa kernel (ma trận bộ lọc) là một ma trận vuông kích thước  $k^* k$  trong đó  $k$  là số lẻ.  $k$  có thể bằng  $1, 3, 5, 7, 9, \dots$  Ví dụ kernel kích thước  $3*3$ . Với mỗi phần tử  $x_{i,j}$  trong ma trận  $X$  lấy ra một ma trận có kích thước bằng kích thước của kernel  $W$  có phần tử  $x_{i,j}$  làm trung tâm (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận  $A$ . Sau đó tính tổng các phần tử của phép tính element-wise của ma trận  $A$  và ma trận  $W$ , rồi viết vào feature map  $Y$ . Ví dụ về cách tính toán cho  $Y$  cụ thể :



Hình 2-21: Ví dụ về cách tính feature map

- Sự kết hợp của một hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Nhìn vào ảnh dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các bộ lọc khác nhau.

Operation	Kernel $\omega$	Image result $g(x,y)$
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Hình 2-22: Một số ví dụ về các Kernel

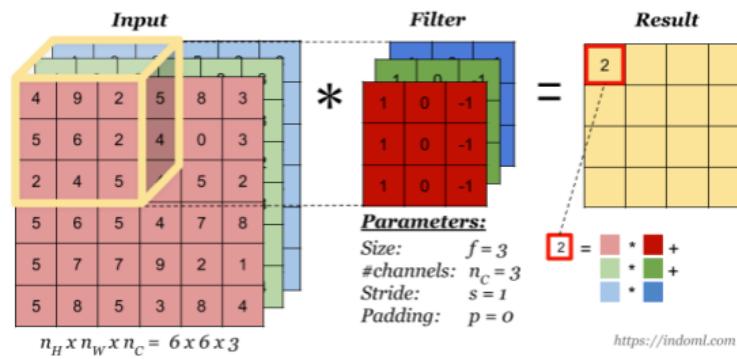
- Ở lớp này sẽ chịu ảnh hưởng của các tham số gồm: số lượng bộ lọc, kích thước của ma trận bộ lọc, Stride (bước nhảy) và Padding (đường viền).

- **Bước nhảy (Stride):** bước nhảy để xác định khoảng cách di chuyển của kernel trên ảnh đầu vào theo cả chiều từ trái sang phải và từ trên xuống dưới, thường dùng để làm giảm kích thước ma trận sau phép tính convolution. Stride ảnh hưởng đến độ nét của bức ảnh, làm giảm kích thước đầu vào xuống. Bước nhảy càng cao thì độ nét càng giảm xuống.
- **Đường viền (Padding):** Như ở trên thì mỗi lần thực hiện phép tính convolution xong thì kích thước ma trận Y đều nhỏ hơn X. Tuy nhiên giờ ta muốn ma trận Y thu được có kích thước bằng ma trận X. Tìm cách giải quyết cho các phần tử ở viền bằng cách thêm giá trị 0 ở viền ngoài ma trận X. Lúc này ma trận Y thu được sẽ có kích thước bằng ma trận X ban đầu. Phép tính này gọi là convolution với padding=1. Padding=k nghĩa là thêm k vector 0 vào mỗi phía (trên, dưới, trái, phải) của ma trận. Sau khi áp dụng padding, thu được kích thước feature map bằng với kích thước ảnh đầu vào.

0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	1	1	0	0	0
0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0

Hình 2-23: Ví dụ về Padding

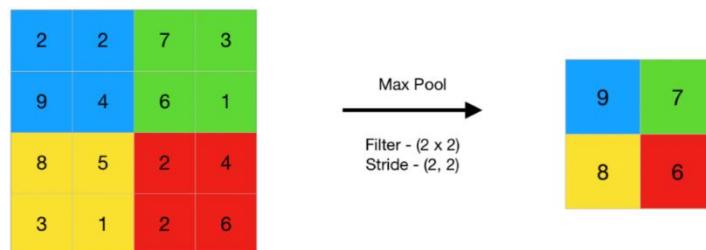
- Sau mỗi phép chập để hoàn thiện hơn, CNN thường gán một giá trị độ lệch b vào feature map và áp dụng hàm kích hoạt ReLU (Rectified Linear Unit) để loại bỏ giá trị âm, tăng tính phi tuyến và giúp mô hình học hiệu quả hơn.
- Các phần giới thiệu ở trên để giới thiệu cho ảnh xám, còn với các ảnh RGB, bộ lọc sẽ cần phải có 3 kênh để thực hiện tính toán cho mỗi kênh màu của hình ảnh. Tuy nhiên ma trận đầu ra sẽ là ma trận 2 chiều. Với ảnh màu, lớp chập sẽ được thực hiện cho từng kênh sau đó cộng kết quả đầu ra của tất cả các kênh để thu được ma trận đầu ra.



Hình 2-24: Phép nhân ma trận với ảnh màu RGB

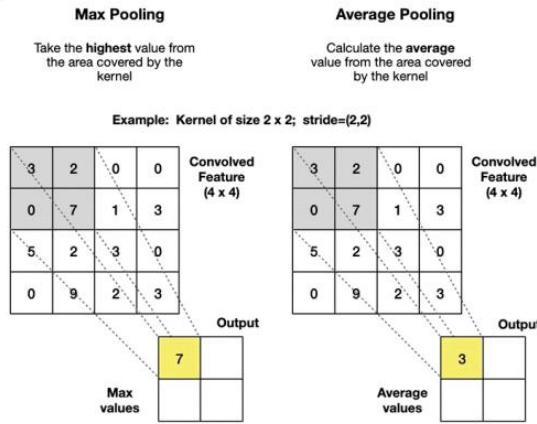
Tuy nhiên, để cải thiện và lưu trữ nhiều thông tin khác nhau của dữ liệu ở ma trận đầu ra, ta có thể sử dụng nhiều bộ lọc khác nhau. Khi đó, ma trận đầu ra sẽ là một ma trận 3 chiều với số kênh bằng số bộ lọc được sử dụng.

- Mặc dù CNN có thể hoạt động chỉ với các lớp chập, nhưng để tăng tốc tính toán thì sẽ sử dụng thêm một lớp tiếp theo đó là lớp gộp (Pooling layer):



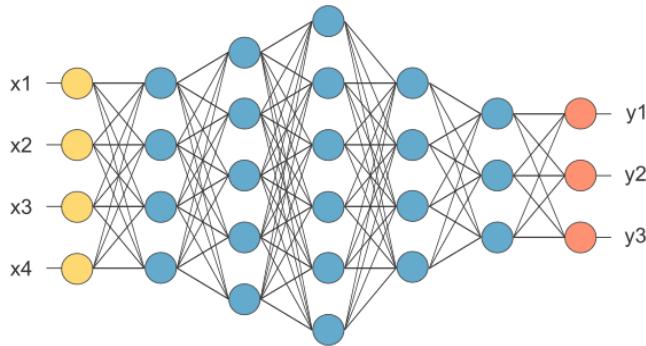
Hình 2-25: Ví dụ về Pooling Layer

- Lớp pooling hay còn gọi là lớp lấy mẫu, sau khi trích xuất đặc trưng qua lớp chập, CNN sử dụng Pooling layer để giảm kích thước của feature map. Từ đó sẽ làm giảm bớt số lượng tham số khi hình ảnh quá lớn, làm tăng hiệu suất tính toán và tránh hiện tượng overfitting. Overfitting là hiện tượng mô hình học quá kĩ dữ liệu huấn luyện nhưng lại hoạt động kém khi gặp dữ liệu mới. Pooling hoạt động bằng cách áp dụng một bộ lọc nhỏ để lấy giá trị đại diện mỗi vùng quét, giúp giữ lại những thông tin quan trọng nhất. Có 2 phương pháp pooling thường gặp là Max Pooling và Average Pooling. Mặc dù pooling làm mất đi một số thông tin, nhưng lại giúp cho mô hình hoạt động hiệu quả hơn, giảm độ phức tạp và cải thiện khả năng tổng quát hóa với dữ liệu mới.



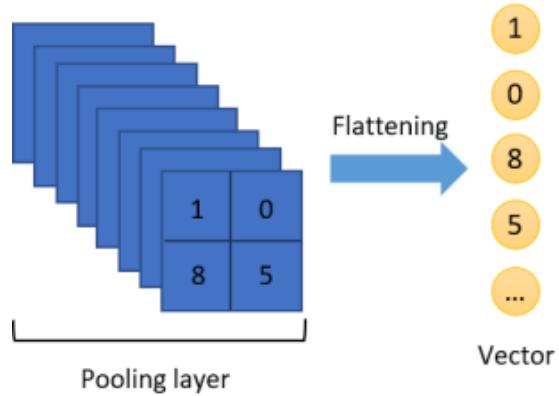
Hình 2-26: Hai phương pháp pooling thông dụng

- Lớp gộp phụ thuộc vào ba tham số là kích thước của bộ lọc  $f$ , bước nhảy và phương pháp pooling. Nhưng ở lớp này, vù không có giá trị đầu vào ngẫu nhiên của một bộ lọc như lớp chập nên không có quá trình tối ưu tham số.
- Lớp cuối cùng là lớp kết nối đầy đủ (Fully connected layer): lớp này là lớp cuối cùng của mô hình mạng CNN, đóng vai trò tổng hợp tất cả các đặc trưng đã trích xuất ở các lớp trước và thực hiện phân loại. Ở lớp này, mỗi nơ – ron được kết nối với toàn bộ nơ – ron ở lớp trước, tạo nên một mạng lưới liên kết chặt chẽ.



Hình 2-27: Lớp kết nối đầy đủ

Các giá trị từ feature map trước đó sẽ được làm phẳng thành một vectơ một chiều và sau đó được đưa làm đầu vào cho lớp fully connected để xử lý. Quá trình làm phẳng này gọi là Flattening. Tiếp đó, CNN sử dụng các hàm kích hoạt phi tuyến như Softmax hoặc Sigmoid để tính toán xác suất cho từng lớp đầu ra. Điều này giúp cho mô hình đưa ra quyết định cuối cùng, chẳng hạn như phân loại hình ảnh thành các nhóm khác nhau (ví dụ: chó, mèo, ô tô, ...).



Hình 2-28: Quá trình làm phẳng Flattening

Tổng quát lại hoạt động của CNN như sau: ảnh đầu vào sẽ đi qua một loạt các lớp để trích xuất đặc trưng và đưa ra kết quả. Đầu tiên là các lớp tích chập (convolutional layers) để quét qua bằng các kernel nhằm phát hiện các đặc trưng như cạnh, góc, họa tiết. Kết quả tạo thành các bản đồ đặc trưng (feature maps), sau đó được xử lý qua các hàm kích hoạt phi tuyến như ReLU để tăng khả năng biểu diễn. Tiếp theo, các lớp gộp (pooling layers), thường là max pooling, sẽ giảm kích thước không gian và giúp mô hình kháng nhiễu. Quá trình này được lặp lại nhiều lần để trích xuất đặc trưng từ thấp đến cao. Sau đó, các bản đồ đặc trưng được làm phẳng thành vector 1 chiều và đưa vào các lớp fully connected để tổng hợp thông tin. Cuối cùng, lớp đầu ra (output layer) sử dụng hàm softmax hoặc sigmoid để đưa ra dự đoán tương ứng với nhiệm vụ như phân loại ảnh hoặc nhận diện đối tượng.

#### 2.6.4 Đánh giá mô hình YOLO

Để đánh giá hiệu quả của mô hình, người ta sử dụng các thông số kỹ thuật quan trọng để phản ánh cả độ chính xác lẫn tốc độ xử lý. Các thông số quan trọng được quan tâm.

- **Precision (Độ chính xác):** thể hiện mức độ chắc chắn của mô hình rằng một đối tượng được phát hiện thuộc về một lớp cụ thể. Độ chính xác giúp lọc các dự đoán, chỉ những phát hiện có độ chính xác trên ngưỡng được chỉ định mới được coi là hợp lệ. Khi precision cao thì sẽ ít cảnh báo giả, hạn chế cảnh báo nhầm.
- **Recall (Độ nhạy):** là chỉ số đo lường khả năng của mô hình trong việc phát hiện đầy đủ tất cả các trường hợp thực sự thuộc về một lớp nào đó. Độ nhạy càng cao thì hệ thống càng phát hiện được nhiều trường hợp xảy ra.
- **Điểm F1:** là giá trị trung bình hài hòa giữa độ chính xác và độ nhạy, cung cấp đánh giá cân bằng về hiệu suất của mô hình trong khi xem xét kết quả dương tính giả và âm tính giả. F1-score đạt giá trị 1 khi mô hình hoàn hảo

và bằng 0 khi mô hình hoàn toàn sai. Trong các bài toán phát hiện đối tượng như YOLO, F1-score giúp đánh giá mô hình không chỉ theo số lần phát hiện đúng mà còn xét đến việc có bỏ sót đối tượng hay không.

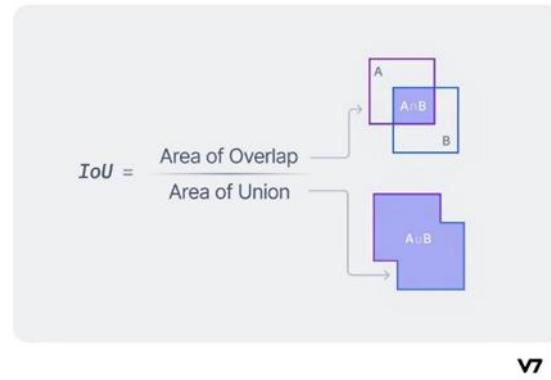
- **Mean Average Precision (Độ chính xác trung bình – mAP):** Là chỉ số đo lường hiệu suất của mô hình, kết hợp giữa 2 chỉ số Precision và Recall. Nó xem xét độ chính xác của việc phát hiện từng lớp đối tượng, tính trung bình các điểm này và đưa ra một con số tổng thể cho thấy mô hình có thể xác định và phân loại đối tượng chính xác như thế nào. Với chỉ số này, khi train mô hình, ta sẽ được kết quả với 2 chỉ số:
  - mAP@0.5: đo độ chính xác trung bình tại ngưỡng IoU duy nhất là 0.5. Chỉ số này để kiểm tra xem mô hình có thể tìm đúng các đối tượng với yêu cầu độ chính xác lỏng lẻo hơn hay không? Nó tập trung vào việc liệu đối tượng có ở đúng vị trí hay không, không cần vị trí hoàn hảo. Nó giúp xem liệu mô hình có giỏi phát hiện ra các đối tượng hay không.
  - mAP@.5:95: đo độ chính xác trung bình các giá trị mAP được tính toán ở nhiều ngưỡng IoU, từ 0.5 đến 0.95 theo mức tăng 0.05. Số liệu này chi tiết và nghiêm ngặt hơn. Nó cung cấp đầy đủ hơn về mức độ chính xác mà mô hình có thể tìm thấy các đối tượng ở các mức độ nghiêm ngặt khác nhau và đặc biệt hữu ích cho các ứng dụng cần phát hiện đối tượng chính xác.

### 2.6.5 Intersection over union (IoU)

Khi cho frame ảnh qua model để train, mỗi frame ảnh sẽ được chia thành nhiều cell, mỗi cell sẽ có một vector khác nhau. Khi dùng thuật toán này, trong một frame ảnh với nhiều cell, mô hình sẽ detect được nhiều bounding box với độ chắc chắn khác nhau cho mỗi đối tượng. IoU sinh ra để giải quyết vấn đề này. Nhờ có IoU nên mô hình có thể lọc ra được bounding box tốt nhất.

IoU là độ đo đánh giá các mô hình thực thể, có thể đánh giá các mô hình khác nhau như YOLO, RCNN, Fasst-RCNN, ... IoU tính toán số lượng chồng chéo giữa hai bounding box – một bounding box dự đoán và một bounding box thực tế.

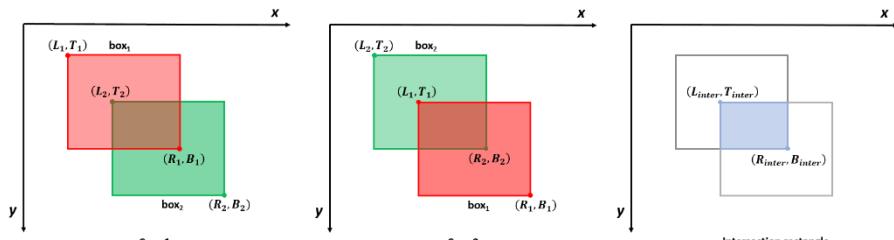
Công thức tính IoU bằng tỉ lệ giữa diện tích phần giao với diện tích phần hợp của hai hộp giới hạn:



Hình 2-29: Cách tính chỉ số IoU

$$IoU = \frac{\text{Diện tích giao (Intersection)}}{\text{Diện tích hợp (Union)}}$$

Điểm IoU sẽ cao nếu có nhiều sự chồng chéo giữa hai hộp. Ngược lại, nếu sự chồng chéo thấp sẽ dẫn đến điểm IoU thấp. Điểm  $IoU = 1$  cho thấy sự trùng khớp giữa hai hộp lớn,  $IoU = 0$  có nghĩa là sự chồng chéo giữa các hộp nhỏ, độ chính xác không cao.



Hình 2-30: Sự chồng chéo giữa các box

Sau đây là cách tính toán IoU thể hiện rõ qua phần toán học: Giả sử một hệ trục tọa độ 2 chiều với trục x và trục y, với hai bounding box là  $box_1$  và  $box_2$ . Mỗi box được biểu diễn bằng các điểm gồm left (L), right (R), top (T) và bottom (B).

- Với  $box_1$  có tọa độ được xác định:  $L_1, T_1$  là tọa độ góc đỉnh trên bên trái box,  $R_1, B_1$  là tọa độ góc đỉnh dưới bên phải.
- Với  $box_2$  có tọa độ được xác định:  $L_2, T_2$  là tọa độ góc đỉnh trên bên trái box,  $R_2, B_2$  là tọa độ góc đỉnh dưới bên phải.
- Với hộp của phần giao nhau, tọa độ được xác định:  $L_{inter}, T_{inter}$  là tọa độ góc đỉnh trên bên trái,  $R_{inter}, B_{inter}$  là tọa độ góc đỉnh dưới bên phải. Các tọa độ này được xác định như sau:

$$L_{inter} = \max(L_1, L_2)$$

$$T_{inter} = \max(T_1, T_2)$$

$$R_{inter} = \min(R_1, R_2)$$

$$B_{inter} = \min(B_1, B_2)$$

- Diện tích vùng giao điểm được tính từ các tọa độ thu được:

$$A_{inter} = (R_{inter} - L_{inter}) * (B_{inter} - T_{inter})$$

Đây là phép nhân để tính được diện tích giao nhau (Intersection), nếu  $A_{inter}$  âm thì có nghĩa là hai vùng không giao nhau và  $\text{IoU} = 0$ .

- Sau khi tính diện tích vùng giao điểm, ta sẽ tính diện tích phần hợp (Union) theo cách tính sau: Đầu tiên ta sẽ tính diện tích của hai hộp:

$$\text{Diện tích của box}_1: A_1 = (R_1 - L_1) * (B_1 - T_1)$$

$$\text{Diện tích của box}_2: A_2 = (R_2 - L_2) * (B_2 - T_2)$$

- Cuối cùng, chỉ số IoU được tính theo công thức:

$$\text{IoU} = \frac{A_{inter}}{A_{union}}$$

Chỉ số IoU rất hữu ích vì nó cung cấp thước đo định lượng cho việc đánh giá độ chính xác của mô hình trong việc phát hiện các đối tượng trong ảnh. Khi train, thường chọn một ngưỡng IoU tối thiểu để xác định bounding box dự đoán là chính xác. Việc lựa chọn ngưỡng IoU phù hợp giúp cân bằng giữa phát hiện sai và phát hiện đúng, và cần được điều chỉnh tùy theo mục tiêu bài toán. Trong mô hình của chúng em sử dụng ngưỡng  $\text{IoU} = 0.7$  để cân bằng các giá trị tính toán.

IoU còn được tính trong phân loại nhị phân như sau:

$$\text{IoU} = \frac{TP}{TP + FP + FN}$$

Trong đó:

- TP (True Positive): mô hình dự đoán đúng các điểm thực sự có khói, lửa.
- FP (False Positive): mô hình dự đoán nhầm – báo có lửa nhưng thực tế không có.
- FN (False Negative): mô hình bỏ sót đối tượng lửa, khói – trên thực tế là có khói, lửa nhưng không phát hiện được.

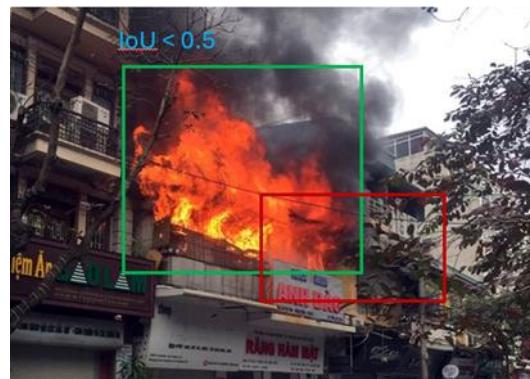
Dựa trên IoU ta có thể tính các chỉ số mAP, sử dụng trong bài toán Non – Max suppression trình bày ở phần dưới đây.

Xem xét một ví dụ về object detection đơn giản để hiểu rõ hơn về IoU:



Hình 2-31: Trường hợp True Positive

Trong trường hợp đầu tiên, bounding box được mô hình dự đoán (màu đỏ) khá gần với bounding box thực tế (màu xanh). Với chỉ số  $\text{IoU} > 0.5$  nên mô hình dự đoán khá chính xác. Hình trên sẽ được cho là TP.

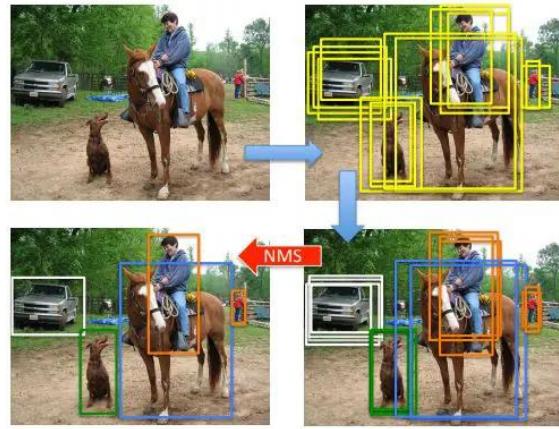


Hình 2-32: Trường hợp False Positive

Trong hình ảnh tiếp theo, bounding box dự đoán chỉ phát hiện được một phần đối tượng nhỏ ở góc, không dự đoán được cả ngọn lửa nên làm cho chỉ số  $\text{IoU} < 0.5$ . Vì thế, trường hợp này được đánh dấu là FP.

## 2.6.6 Bài toán Non – Max suppression

Non – Max suppression (NMS) là một kỹ thuật xử lý hậu kỳ được sử dụng trong các thuật toán phát hiện đối tượng để giảm số lượng hộp giới hạn chồng chéo và cải thiện chất lượng phát hiện tổng thể. Khi một đối tượng được phát hiện nhiều lần với các bounding box có độ tin cậy khác nhau, NMS giữ cho đối tượng tốt nhất và loại bỏ các bounding box dư thừa. Điều này giúp đảm bảo mỗi đối tượng chỉ được tính một lần.



Hình 2-33: Thuật toán Non – max Suppression

Để NMS có thể làm được như vậy thì nhóm em cũng đã nghiên cứu về thuật toán của NMS sử dụng. Và thuật toán đó như sau:

- Đầu vào của thuật toán:
  - Một tập các proposals bounding box:  $B = \{b_1, b_2, \dots, b_n\}$ , với mỗi  $b_i$  là một bounding box được dự đoán.
  - Một tập các confidence score tương ứng:  $S = \{s_1, s_2, \dots, s_n\}$ , với  $s_i$  là độ tin cậy của box  $b_i$  (thường là xác suất của đối tượng).
  - Một ngưỡng chòng chéo (IoU threshold):  $N \in [0,1]$ .
- Đầu ra là một bounding box đã được lọc bỏ các box dư thừa.

Sau đây là ví dụ để tính toán NMS cho hệ thống nhận diện lửa, khói của nhóm em, với đầu vào là một bức ảnh có 4 bounding box được dự đoán như hình vẽ.



Hình 2-34: Ví dụ về thuật toán NMS

Trước tiên, mỗi bounding box được biểu diễn dưới dạng  $[L_1, T_1, R_1, B_1, \text{label}, \text{conf}]$ , trong đó  $(L_1, T_1)$  và  $(R_1, B_1)$  là tọa độ góc trên bên trái và dưới bên phải của hộp, label là nhãn phân loại và conf là độ tin cậy của mô hình đối với hộp đó. Giả

sử hệ thống phát hiện được bốn hộp giới hạn lần lượt là hộp màu đỏ, xanh lá, xanh dương và vàng, tất cả đều được gán cùng một label "fire" nhưng có độ tin cậy khác nhau.

Mỗi bounding box sẽ được biểu diễn:

- Green\_box = [L<sub>1</sub>, T<sub>1</sub>, R<sub>1</sub>, B<sub>1</sub>, fire, 0.92]
- Blue\_box = [L<sub>2</sub>, T<sub>2</sub>, R<sub>2</sub>, B<sub>2</sub>, fire, 0.87]
- Red\_box = [L<sub>3</sub>, T<sub>3</sub>, R<sub>3</sub>, B<sub>3</sub>, fire, 0.80]
- Yellow\_box = [L<sub>4</sub>, T<sub>4</sub>, R<sub>4</sub>, B<sub>4</sub>, fire, 0.75]

Áp dụng thuật toán NMS, bài toán được thực hiện như sau:

- Sắp xếp các box theo độ tin cậy (conf) giảm dần, từ box có độ tin cậy cao nhất đến thấp nhất. box\_list = [green\_box (0.92), blue\_box (0.87), red\_box (0.80), yellow\_box (0.75)]
- Tiếp theo, khởi tạo một danh sách lưu kết quả new\_box\_list, chưa có box nào. Ban đầu, new\_box\_list = {} là danh sách rỗng, chưa có phần tử nào.
- Bước tiếp theo, duyệt và so sánh với IoU. Em chọn ngưỡng IoU threshold = 0.5. Chọn box có conf cao nhất cho vào new\_box\_list, nghĩa là chọn green\_box với conf = 0.92. new\_box\_list = {green\_box}. Sau đó, so sánh green\_box với các box còn lại. So sánh nhờ vào ngưỡng IoU threshold. Nếu box nào có độ giao nhau với green\_box vượt quá ngưỡng 0.5 thì box đó sẽ bị loại do khả năng cao đang bao quanh cùng một đối tượng. Khi đó, sẽ loại bỏ box có conf thấp hơn. Còn nếu IoU thấp hơn ngưỡng, hộp sẽ được giữ lại và tiếp tục so sánh với các hộp còn lại sau.
- Lặp lại quá trình trên với các box tiếp theo, cho đến khi duyệt hết các box. Kết quả cuối cùng là danh sách new\_box\_list gồm các box có conf cao và không trùng lặp đáng kể với nhau.

Thuật toán Non-Maximum Suppression (NMS) đóng vai trò quan trọng trong việc lọc bỏ các hộp giới hạn chòng lặp, đảm bảo rằng mỗi đối tượng chỉ được phát hiện duy nhất một lần với độ chính xác cao. Bằng cách so sánh độ tin cậy và mức độ giao nhau giữa các hộp, NMS giúp hệ thống phát hiện trở nên rõ ràng, hiệu quả và tối ưu hơn. Việc áp dụng NMS góp phần nâng cao chất lượng đầu ra của mô hình, đặc biệt trong các tình huống có nhiều đối tượng xuất hiện gần nhau hoặc bị chòng lấn.

## 2.7 Video streaming

### 2.7.1 Tổng quan về video streaming

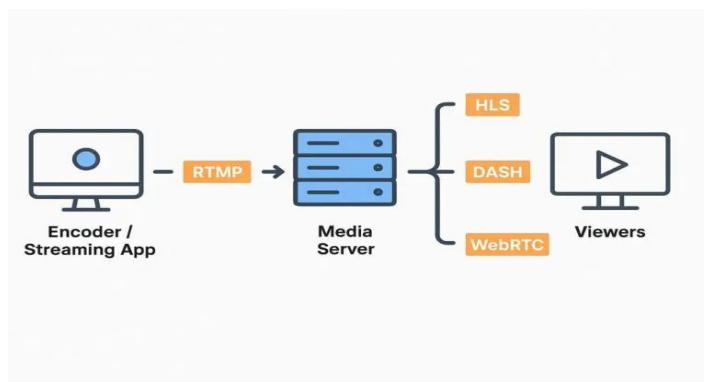
Video streaming là quá trình truyền nội dung video liên tục qua mạng internet để người dùng có thể xem mà không cần tải toàn bộ tệp về trước. Video streaming cho phép truy cập tức thì vào nhiều nội dung: phim, âm thanh, chương trình truyền hình và phát sóng trực tiếp. Trong dự án của chúng em, chúng em stream video để truyền tải hình ảnh từ camera của Raspberry Pi về app cho người dùng xem được.

Mỗi khi người dùng xem video trực tuyến dù là phát trực tiếp hay theo yêu cầu – thì đều có các giao thức truyền tải video (streaming protocol) hoạt động âm thầm phía sau để đưa dữ liệu đến thiết bị qua internet. Những giao thức này thường hoạt động ở các tầng ứng dụng, trình bày và phiên trong mô hình OSI.

Việc truyền video qua Internet sử dụng 2 nhóm giao thức chính: các giao thức streaming truyền thông TCP/ UDP và các giao thức dựa trên HTTP. Trong luồng stream video từ Raspberry Pi lên server và từ server về app, chúng em sử dụng 2 giao thức chính, mỗi giao thức đại diện cho mỗi nhóm giao thức, đó là giao thức RTMP (Real-time Messaging Protocol) và HLS (HTTP live streaming).

### 2.7.2 Giao thức RTMP (Real – Time Messaging Protocol).

Là giao thức truyền tải dữ liệu âm thanh, video và các dữ liệu khác qua Internet, giữa trình phát Flash và Server được phát triển bởi Adobe Systems vào năm 2004. RTMP là một giao thức dựa trên TCP, duy trì các kết nối liên tục và cho phép giao tiếp với độ trễ thấp. Để phân phối luồng một cách trơn tru và truyền càng nhiều thông tin càng tốt, nó chia luồng thành các đoạn (fragments) và kích thước của chúng được thỏa thuận tùy vào thỏa thuận giữa client và server. Kích thước mặc định cho dữ liệu audio là 64 bytes và 128 bytes cho dữ liệu video. Các đoạn này sau đó được sắp xếp lại và ghép thành một kết nối duy nhất. Với các tập tin dài, RTMP chỉ chứa 1 byte header/1 fragment nên không bị quá tải.



Hình 2-35: Mô hình tổng quát sử dụng RTMP

Quá trình hoạt động của RTMP: RTMP hoạt động bằng cách chia nhỏ dữ liệu video chất lượng cao thành các gói nhỏ hơn, giúp dễ dàng quản lý và gửi đi. Quá trình RTMP bao gồm ba giai đoạn chính:

- **Handshake (Bắt tay):** Client sẽ gửi yêu cầu kết nối tới Server. Trong bước này, 3 gói dữ liệu được trao đổi chứa thông tin về phiên bản RTMP và dấu thời gian để đồng bộ kết nối. Quá trình này đảm bảo client và server sẵn sàng trao đổi nội dung video và âm thanh.
- **Connection (Kết nối):** Sau khi hợp tác thành công sẽ thiết lập kết nối. Client gửi yêu cầu tới server sử dụng các thông điệp mã hóa Action Message Format (AMF) gồm có URL kết nối và các loại codec âm thanh, video. Nếu thông tin chính xác, server sẽ phê duyệt, và kết nối được thiết lập, sẵn sàng nhận video từ bộ mã hóa RTMP.
- **Stream (Truyền phát):** Đây là bước cuối cùng của quá trình RTMP. Client sẽ gửi nội dung video và âm thanh tới server, sau đó server sẽ gửi đến người xem.

Hiện nay có nhiều công nghệ mới ra đời như DASH, WEBRTC, nhưng trong hệ thống hiện đại, RTMP vẫn được sử dụng phổ biến nhờ khả năng truyền tải ổn định và dễ triển khai. Các đặc điểm nổi bật của RTMP là:

- **Đảm bảo độ tin cậy cao:** RTMP hoạt động trên giao thức truyền tải TCP, giúp đảm bảo dữ liệu đến đúng thứ tự và không bị mất mát. Điều này đặc biệt quan trọng trong truyền phát video trực tiếp, nơi mỗi khung hình cần được hiển thị tuần tự để đảm bảo trải nghiệm người xem.
- **Duy trì kết nối liên tục (persistent connection):** Khác với các giao thức dựa trên HTTP (như HLS), RTMP thiết lập một kết nối socket liên tục giữa client và server. Nhờ đó, dữ liệu có thể được truyền đi tức thì mà không cần thiết lập lại kết nối, giúp giảm đáng kể độ trễ.
- **Khả năng phân mảnh và ghép luồng (stream multiplexing):** RTMP chia luồng thành các đoạn nhỏ (fragments) với kích thước câu hình được (thường là 64 bytes cho audio và 128 bytes cho video), sau đó sắp xếp lại theo một luồng hợp nhất. Điều này giúp giảm gánh nặng truyền tải, tránh gây quá tải hệ thống khi truyền các file video dài.
- **Hỗ trợ nén và chọn codec linh hoạt:** RTMP có thể truyền dữ liệu được mã hóa bằng nhiều codec, phổ biến nhất là H.264 cho video và AAC cho audio. Điều này cho phép lựa chọn giữa chất lượng và hiệu năng, tùy vào thiết bị phát và mạng.

Trong hệ thống của chúng em, RTMP được lựa chọn làm giao thức truyền tải video từ Raspberry Pi 4 lên server nhờ tính ổn định, dễ triển khai và hỗ trợ độ trễ thấp. Với kết nối TCP liên tục và khả năng truyền tải hiệu quả luồng H.264 đã mã hóa, RTMP đáp ứng tốt yêu cầu truyền hình ảnh thời gian thực từ thiết bị IoT.

Tuy không còn phổ biến ở phía người xem do sự lỗi thời của Flash, RTMP vẫn là lựa chọn lý tưởng ở vai trò giao thức ingest, cho phép server tiếp nhận luồng dễ dàng và chuyển đổi sang định dạng phát lại như HLS. Đây là giải pháp cân bằng tốt giữa hiệu năng, độ trễ và độ phức tạp triển khai trong hệ thống giám sát đa thiết bị.

### 2.7.3 Giao thức HLS (HTTP Live Streaming)

HLS là một HTTP-based adaptive bitrate streaming, một giao thức truyền bitrate dựa trên HTTP. HLS được lập trình bởi Apple Inc. Là một giao thức truyền phát nội dung đa phương tiện khá phổ biến. HLS hoạt động bằng cách băm luồng thành nhiều tập tin nhỏ (HTTP-based transport stream) - .ts files. Các files này được sắp xếp lại thứ tự bởi UTF-8 M3U playlist (.m3u8).

Playlist là tệp văn bản định dạng .m3u8 chứa danh sách các tệp phân đoạn (segments) và thông tin chưa dữ liệu khác. Thường có 2 loại:

- **Master Playlist (Danh sách phát chính):** Đây là tệp danh sách chính chứa các liên kết đến các phiên bản khác nhau của video với độ phân giải, bitrate khác nhau. Khách hàng sẽ chọn phiên bản phù hợp dựa trên băng thông và thiết bị.
- **Media Playlist (Danh sách phát phương tiện):** Tệp này chứa các đường dẫn đến từng phân đoạn video cụ thể.

TS segment là các đoạn video nhỏ được cắt ra từ video gốc, có định dạng .ts (MPEG Transport Stream). Mỗi segment thường có độ dài từ 2 đến 10 giây, tùy thuộc vào cấu hình.

Cách hoạt động của HLS:

- **Server:** Luồng HLS bắt nguồn từ máy chủ và thực hiện hai quá trình chính:
  - Mã hóa (Encoding): Dữ liệu video được mã hóa lại bằng các chuẩn nén phổ biến H.264 (AVC) hoặc H.265 (HEVC) nhằm đảm bảo khả năng tương thích với nhiều thiết bị đầu cuối khác nhau. Hai chuẩn này được chọn vì chúng đạt tỷ lệ nén cao nhưng vẫn giữ chất lượng tốt, và đặc biệt là tương thích rộng rãi với các thiết bị hiện nay như điện thoại, máy tính bảng, TV thông minh. Quá trình mã hóa giúp giảm kích thước video, tiết kiệm băng thông truyền tải mà vẫn giữ được trải nghiệm người xem.
  - Phân mảnh (Segmentation): Video sau khi được mã hóa sẽ được chia nhỏ thành các đoạn video ngắn, thường có độ dài từ 2 đến 6 giây mỗi đoạn. Trước năm 2016, độ dài mặc định là 10 giây. HLS đồng thời tạo một tệp chỉ mục (.m3u8), đây là một danh sách phát (playlist) ghi rõ thứ tự các đoạn video để client biết cần tải đoạn nào tiếp theo. Bên

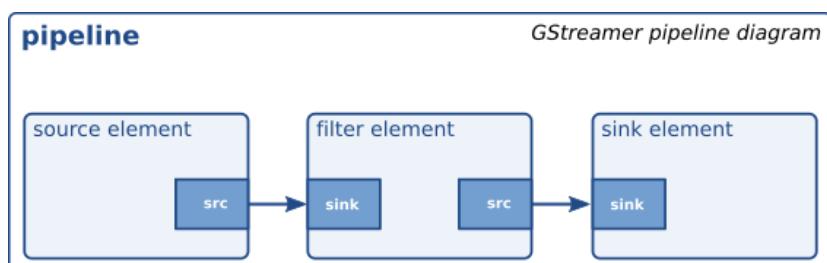
cạnh đó, để hỗ trợ các mạng có tốc độ khác nhau, máy chủ sẽ tạo ra nhiều phiên bản video ở các độ phân giải khác nhau (ví dụ: 480p, 720p, 1080p), mỗi phiên bản đều được phân mảnh riêng và có playlist riêng.

- **Phân phối (Distribution):** Khi client yêu cầu phát video, server sẽ gửi lần lượt các đoạn video thông qua giao thức HTTP, giống như khi truy cập một trang web. Quá trình này không yêu cầu server đặc biệt – chỉ cần máy chủ web thông thường cũng có thể phục vụ luồng HLS. Tuy nhiên, để tăng hiệu suất và giảm độ trễ khi có nhiều người dùng ở các khu vực địa lý khác nhau, hệ thống thường sử dụng CDN (Content Delivery Network). CDN sẽ phân phối bản sao các đoạn video tới các máy chủ biên gần người dùng hơn, giúp giảm thời gian tải và giảm tải cho máy chủ gốc.
- **Thiết bị người dùng (Client Device):** Phía người dùng là nơi tiếp nhận và hiển thị luồng video. Khi người dùng mở video HLS, thiết bị của họ sẽ trước tiên tải về file danh sách phát (.m3u8). Dựa vào nội dung trong file này, thiết bị sẽ lần lượt yêu cầu các đoạn video kế tiếp. Nếu đường truyền mạng ổn định, thiết bị sẽ tải video độ phân giải cao (ví dụ 1080p). Nhưng nếu mạng yếu, thiết bị sẽ tự động chuyển sang phân giải thấp hơn (adaptive bitrate streaming) như 480p để đảm bảo phát mượt. Các đoạn video được phát nối tiếp nhau tạo thành một trải nghiệm xem liền mạch, giống như một video bình thường.

#### 2.7.4 Gstreamer

Trong hệ thống của chúng em, gstreamer được lựa chọn và đóng vai trò rất quan trọng trong việc truyền phát video từ Raspberry Pi lên server để hiển thị trên app Flutter. Nó chính là công cụ xử lý trung gian, thực hiện các tác vụ chuyển đổi, đóng gói và truyền tải video theo thời gian thực.

Gstreamer là một framework mã nguồn mở dùng để xây dựng pipeline (đường ống xử lý) âm thanh và hình ảnh đa phương tiện. Gstreamer được sử dụng rộng rãi trong các hệ thống phát video, xử lý âm thanh, camera, ... Mỗi bước xử lý trong gstreamer là một phần tử được kết nối với nhau tạo thành một chuỗi.



Hình 2-36: Sơ đồ pipeline GStreamer

Gstreamer gồm có 3 thành phần chính tạo nên một luồng xử lý hoàn chỉnh:

- **Nguồn dữ liệu (Source elements):** Đây là nơi bắt đầu của pipeline, nơi Gstreamer nhận tín hiệu đầu vào từ các thiết bị như camera (v4l2src), file (filesrc) hoặc micro (alsasrc) để tạo dữ liệu sử dụng cho pipeline. Trong hệ thống của chúng em, filesrc được dùng để đọc video H.264 từ pipe FIFO do libcamera-vid xuất ra.
- **Bộ xử lý trung gian (Filter/Transform elements):** Đây là phần xử lý chính – dùng để chuyển đổi định dạng, mã hóa, phân tích luồng dữ liệu, hoặc trích xuất thông tin cần thiết. Ví dụ như h264parse để phân tích định dạng video, videoconvert để đổi định dạng ảnh, hoặc queue để tách luồng. Bộ xử lý trung gian gồm 2 pad sink và scr, sink pad để nhận dữ liệu từ element khác và chuyển tiếp dữ liệu đã qua xử lý tới element tiếp theo thông qua scr pad.
- **Đầu ra (Sink elements):** Đây là nơi dữ liệu được ghi lại, hiển thị hoặc truyền đi. Trong hệ thống này, rtmpsink đảm nhiệm việc gửi luồng video đã được đóng gói lên server thông qua giao thức RTMP.

Gstreamer giữ vai trò cốt lõi trong việc truyền phát video từ Raspberry Pi lên server. Cụ thể, nó đảm nhận việc đọc dữ liệu video H.264 được ghi ra pipe bởi libcamera-vid, sau đó phân tích cú pháp định dạng thông qua phần tử h264parse. Tiếp theo, Gstreamer đóng gói luồng video thành định dạng FLV nhờ flvmux và truyền đi bằng rtmpsink đến máy chủ trung tâm thông qua giao thức RTMP. Với thiết kế dạng pipeline linh hoạt, Gstreamer giúp hệ thống thực hiện truyền video thời gian thực một cách hiệu quả, ổn định và tương thích với các server phát như nginx-rtmp. Đây là thành phần không thể thiếu trong quy trình đảm bảo truyền tải luồng video trực tiếp từ Pi lên server.

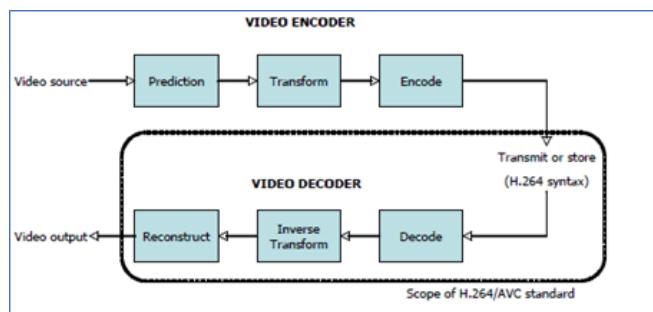
### 2.7.5 Bộ mã hóa và bộ giải mã H.264

Trong hệ thống phát hiện lửa, khói, cảnh báo và stream video lên app, chúng em ưu tiên sử dụng bộ mã hóa và bộ giải mã H.264, còn gọi là AVC – Advanced Video Coding làm chuẩn mã hóa chính cho dữ liệu video. Đây là một trong những chuẩn nén video phổ biến và hiệu quả nhất hiện nay, được sử dụng rộng rãi trong các ứng dụng phát trực tuyến như YouTube, Netflix, cũng như các thiết bị nhúng như Raspberry Pi hoặc camera IP. H.264 cho phép nén video với dung lượng nhỏ nhưng vẫn giữ được chất lượng hình ảnh cao, rất phù hợp cho các hệ thống yêu cầu truyền tải thời gian thực, băng thông hạn chế và độ trễ thấp như trong ứng dụng giám sát, cảnh báo cháy, hoặc phát hiện sự cố mà nhóm đang xây dựng.

Mã hóa video là quá trình chuyển đổi các tệp video thô thành các tệp kỹ thuật số để chúng không được lưu dưới dạng các hình ảnh riêng lẻ mà là các video lưu động. Nói cách khác, đây là quá trình nén và có khả năng thay đổi nguồn tương tự thành nguồn kỹ thuật số. Nói chung, bộ mã hóa/giải mã (codec) được sử dụng để nén tệp video.

Mã hóa video rất quan trọng trong việc truyền nhận video trong các thiết bị nhúng. Ngày xưa, tất cả các tệp video đều ở định dạng thô, tức là chúng là một tập hợp các ảnh tĩnh. Đối với video được ghi ở tốc độ 30 khung hình/giây (fps), bạn phải sử dụng 30 ảnh/giây cảnh quay. Do đó, 1800 hình ảnh/phút được sử dụng cho một video, dẫn đến kích thước tệp video rất lớn. Giải pháp duy nhất để khắc phục vấn đề này là nén các video này, nhưng chất lượng sẽ bị mất trong quá trình này. Sau đó, các kỹ sư đã phát triển các kỹ thuật mã hóa video để nén các tệp này mà không làm giảm chất lượng.

H.264 còn được gọi là Mã hóa video nâng cao (AVC) hoặc MPEG-4 Phần 10. Đây là công nghệ nén video được Liên minh Viễn thông Quốc tế (gọi là H.264) và Tổ chức Tiêu chuẩn hóa Quốc tế/Nhóm chuyên gia hình ảnh động của Ủy ban Kỹ thuật Điện tử Quốc tế (gọi là MPEG-4 Phần 10, Mã hóa video nâng cao hay AVC) cùng phát triển. Là một codec video, H.264 thường được sản xuất theo định dạng chứa MPEG-4, sử dụng phần mở rộng .MP4, cũng như QuickTime (.MOV), Flash (.F4V), 3GP cho điện thoại di động (.3GP) và luồng truyền tải MPEG (.ts). Đôi khi, video H.264 được mã hóa bằng âm thanh được nén bằng codec Mã hóa âm thanh nâng cao (AAC), một tiêu chuẩn ISO/IEC.



Hình 2-37: Bộ mã hóa và giải mã H.264

Cách hoạt động của H.264/AVC: bộ mã hóa video H.264 thực hiện các quy trình dự đoán, chuyển đổi và mã hóa để tạo ra luồng bit H.264 được nén. Nó sử dụng tiêu chuẩn định hướng khối với sự cạnh tranh chuyển động để xử lý các khung nội dung video. Đầu ra sẽ là các khối macro bao gồm các khối có kích thước lớn tới  $16 \times 16$  pixel.

- Dự đoán:** Bộ mã hóa dự đoán khung hình hiện tại bằng cách phân tích các khung hình trước đó. Điều này dựa trên ý tưởng các khung hình liên tiếp thường chứa thông tin tương tự nhau. Thay vì lưu trữ toàn bộ khung hình hiện tại, bộ mã hóa chỉ gửi sự khác biệt giữa khung hình dự đoán và khung hình thực tế.
- Biến đổi:** Sự khác biệt được gửi từ quy trình dự đoán được chia nhỏ thành các thành phần không gian nhỏ hơn bằng kỹ thuật toán học. Điều này giúp lưu trữ và thao tác dữ liệu hiệu quả hơn.

- **Lượng tử hóa:** Để giảm kích thước tệp, dữ liệu ít quan trọng hơn sẽ bị loại bỏ. Đây là hành động cân bằng giữa việc đạt được kích thước tệp nhỏ hơn và duy trì chất lượng video chấp nhận được.
- **Mã hóa Entropy:** Dữ liệu còn lại sau đó được mã hóa bằng một ngôn ngữ cụ thể để giảm thiểu số bit cần thiết để biểu diễn dữ liệu đó. Điều này tương tự như cách mã hóa Huffman hoạt động trong các tệp zip.

Mặt khác, bộ giải mã H.264 hoạt động bằng cách bổ sung các bước của bộ mã hóa: giải mã Entropy, biến đổi ngược và tái tạo.

- Giải mã Entropy: Luồng bit nén được giải mã trở lại thành dữ liệu lượng tử.
- Biến đổi ngược: Dữ liệu được biến đổi trở lại miền không gian.
- Tái tạo: Khung dự đoán được kết hợp với dữ liệu đã giải mã để tạo lại khung gốc.

Nén H.264 làm giảm kích thước của hộp chứa video xuống còn khoảng một nửa so với bản gốc. Khi làm như vậy, các codec dựa trên H.264 không ảnh hưởng đến bất kỳ chất lượng nào.

## 2.8 Tổng quan về cơ sở dữ liệu

### 2.8.1 Giới thiệu về cơ sở dữ liệu

Cơ sở dữ liệu (Database - CSDL) là một tập hợp các dữ liệu có liên quan đến nhau, được lưu trữ trên máy tính, có nhiều người sử dụng và được tổ chức theo một mô hình. Hay nói cách khác, CSDL là một bộ dữ liệu được lưu trữ lại và được các hệ ứng dụng của một đơn vị cụ thể nào đó sử dụng.

CSDL có thể lưu trữ thông tin về con người, sản phẩm, các đơn hàng, hoặc bất kỳ thứ gì khác. Nhiều cơ sở dữ liệu bắt đầu dưới dạng danh sách trong chương trình xử lý văn bản hoặc bảng tính (spreadsheet). Nếu danh sách ấy càng phát triển lớn dần, thì sự dư thừa và không đồng nhất trong dữ liệu bắt đầu xuất hiện. Các dữ liệu sẽ trở nên rất khó để hiểu khi ở dưới dạng danh sách, đồng thời sẽ gặp khó khăn khi tìm kiếm hoặc xuất các phần tử trong dữ liệu để xem xét. Một khi những vấn đề này bắt đầu xuất hiện, thì cách tốt nhất là lưu trữ các dữ liệu trong các “databases” (được phát triển bởi “hệ thống quản lý CSDL – DBMS).

Hệ quản trị cơ sở dữ liệu (DBMS – Database Management System) là phần mềm trung gian giúp người dùng tương tác với cơ sở dữ liệu. Ở trong dự án của chúng em, hệ quản trị cơ sở dữ liệu được sử dụng là MySQL.

### 2.8.2 Hệ quản trị CSDL trong hệ thống

Hệ thống sử dụng cơ sở dữ liệu MariaDB được triển khai trên server để lưu trữ toàn bộ dữ liệu liên quan đến quá trình giám sát và cảnh báo cháy. Cơ sở dữ

liệu này đóng vai trò trung tâm trong việc lưu vết các hoạt động từ cả camera (Pi) lẫn các cảm biến (ESP32), đảm bảo thông tin được lưu trữ đầy đủ, có thể truy xuất và xử lý khi cần thiết.

Để lưu dữ liệu gửi từ Pi và cảm biến lên server, chúng em đã thiết kế các bảng (table) để lưu trữ dữ liệu. Để lưu dữ liệu gửi từ Pi và cảm biến lên server, chúng em đã thiết kế các bảng (table) trong hệ quản trị cơ sở dữ liệu MariaDB với cấu trúc phù hợp cho từng loại nguồn dữ liệu. Cụ thể, dữ liệu từ ESP32 (cảm biến đo nhiệt độ, khói, khí CO) được lưu vào bảng sensor\_data, bao gồm các trường như user\_id, device\_name, temp, smokes, co và timestamp, cho phép phân loại và theo dõi theo từng người dùng và thiết bị cụ thể.

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	user_id	varchar(100)	YES		NULL	
	device_name	varchar(100)	YES		NULL	
	temp	float	YES		NULL	
	smokes	float	YES		NULL	
	co	float	YES		NULL	
	timestamp	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

Hình 2-38: Bảng cơ sở dữ liệu sensor\_data

Đồng thời, dữ liệu phát hiện cháy từ Raspberry Pi (ảnh có khói/lửa) được lưu trong bảng detections, với các trường như image\_path, fire\_score, smoke\_score, level\_id và timestamp, nhằm lưu lại thời gian, vị trí, mức độ nghi ngờ cháy và đường dẫn ảnh phục vụ kiểm tra sau này.

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	image_path	varchar(255)	YES		NULL	
	fire_score	float	YES		NULL	
	smoke_score	float	YES		NULL	
	level_id	int	YES		NULL	
	timestamp	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

Hình 2-39: Bảng cơ sở dữ liệu detections

Ngoài ra, để phục vụ việc gửi thông báo đẩy (push notification) qua Firebase Cloud Messaging, hệ thống còn có bảng fcm\_tokens, nơi lưu các token thiết bị người dùng, được liên kết với user\_id và level\_id, đảm bảo chỉ gửi cảnh báo đến đúng người dùng và tầng tương ứng.

	Field	Type	Null	Key	Default	Extra
▶	user_id	varchar(100)	NO	PRI	NULL	
	token	text	YES		NULL	
	level_id	int	NO	PRI	NULL	

Hình 2-40: Bảng cơ sở dữ liệu fcm\_tokens

Các bảng này đều được thiết kế tối ưu cho truy vấn nhanh, dễ tích hợp với ứng dụng di động và các API truy xuất dữ liệu thời gian thực. Cơ sở dữ liệu MariaDB chạy trên môi trường server ổn định, được bảo mật qua xác thực kết nối và chỉ mở quyền truy cập cho backend xử lý dữ liệu, đảm bảo an toàn và toàn vẹn thông tin trong hệ thống.

## CHƯƠNG 3. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

Tại chương này, nhóm em sẽ tiến hành khảo sát các hệ thống hiện có, để đưa ra các đánh giá tổng quan về các hệ thống này rồi từ đó đưa ra các đề xuất để cải tiến để tài của nhóm. Từ các đề xuất đó nhóm sẽ tiến hành phân tích và thiết kế để hiện thực hóa các đề xuất trên.

### 3.1 Khảo sát các hệ thống hiện có

#### 3.1.1 Các hệ thống hiện có

##### 3.1.1.1 Thiết bị báo cháy độc lập

Thiết bị báo cháy độc lập là một dạng hệ thống cảnh báo cháy hoạt động riêng biệt, không cần kết nối dây dẫn với các trung tâm điều khiển, không cần kết nối dây dẫn với các trung tâm điều khiển. Thiết bị này được phát triển nhằm mang đến giải pháp cảnh báo cháy nhanh chóng và thuận tiện cho người dùng. Khi phát hiện dấu hiệu bất thường như khói hoặc nhiệt độ tăng cao, thiết bị sẽ lập tức phát tín hiệu cảnh báo tại chỗ và có thể gửi thông báo đến số điện thoại đã được thiết lập sẵn để người dùng ứng phó kịp thời.



Hình 3-1: Thiết bị báo cháy độc lập

- Nguyên tắc hoạt động:
  - Khi phát hiện khói trong phạm vi cảm biến thì ngay lập tức phát ra báo động bằng chuông báo cháy rất to và rõ, kết hợp với đèn led nhấp nháy.
  - Chỉ cần phát hiện có khói là cảm biến ngay lập tức báo động ngay để người dùng có thể ứng phó kịp thời.
- Ứng dụng của đầu báo cháy độc lập:
  - Gia đình: Lắp đặt trong phòng ngủ, phòng khách, bếp, hành lang để bảo vệ an toàn cho các thành viên.
  - Căn hộ chung cư: Sử dụng trong các căn hộ nhỏ, nơi không có hệ thống báo cháy trung tâm.

- Văn phòng nhỏ: Bảo vệ tài sản và nhân viên trong các không gian làm việc nhỏ.
- Nhà kho, cửa hàng: Cảnh báo sớm khi có cháy xảy ra trong các khu vực hàng hóa dễ cháy.
- Một số loại đầu báo cháy độc lập:
  - Đầu báo khói quang học: Sử dụng một tia hồng ngoại trong buồng tối, khi có khói bay vào tia sáng bị tán xạ và chạm vào cảm biến, từ đó phát tín hiệu cảnh báo.
  - Đầu báo nhiệt: Sử dụng cảm biến để phát hiện sự gia tăng đột ngột của nhiệt độ, phát ra tín hiệu cảnh báo khi nhiệt độ vượt quá ngưỡng thiết lập. Hoặc cũng có thể dựa trên tốc độ tăng nhiệt độ nhanh bất thường trong một khoảng thời gian ngắn để phát ra cảnh báo.
  - Đầu báo khí gas: Nhận diện nồng độ khí cháy như gas LPG, methane... Khi nồng độ này vượt ngưỡng thiết bị sẽ phát ra cảnh báo.
  - Báo cháy cảm biến CO (Carbon monoxide): Đây là loại báo cháy dựa trên nồng độ khí CO trong không khí. Khí CO là một khí độc hại, không màu, không mùi, sinh ra do quá trình cháy. Khi nồng độ khí CO vượt mức thiết lập thì thiết bị sẽ phát ra cảnh báo. Các thiết bị cảnh báo CO có thể được sử dụng độc lập hoặc kết hợp với báo cháy khói.

*Bảng 3-1: Bảng ưu/nhược điểm của đầu báo cháy độc lập*

Ưu điểm	Nhược điểm
Hoạt động độc lập	Phạm vi bảo vệ hạn chế
Dễ dàng lắp đặt và sử dụng	Thời lượng pin có hạn
Giá thành thấp	Khó mở rộng quy mô hệ thống
Cảnh báo tại chỗ	Dễ bị bỏ qua khi không có người ở gần

### *3.1.1.2. Hệ thống báo cháy thường*

Là hệ thống có tính năng đơn giản, giá thành không cao, hệ thống này thông thường thích hợp lắp đặt tại các công ty, nhà xưởng, nhà kho có diện tích vừa hoặc nhỏ khoảng vài nghìn mét vuông.



Hình 3-2: Hệ thống báo cháy thường

Đặc điểm của hệ thống báo cháy thường:

- Hệ thống báo cháy thường là quản lý một khu vực (zone) nhà xưởng hoặc một tầng nhà. Khu vực (zone) đó sẽ có một vài hoặc tất cả thiết bị báo cháy đầu vào (đầu báo nhiệt, khí gas, đầu báo khói...) được mắc nối tiếp với nhau và mắc nối tiếp với trung tâm báo cháy.
- Zone: Là nhiều thiết bị nằm trên một đường dây tín hiệu khi xảy ra báo cháy thì chúng ta chỉ biết được là khu vực nào báo mà không biết được vị trí chính xác như hệ thống báo cháy địa chỉ. Khi xảy ra cháy thì các thiết bị đầu ra sẽ hoạt động như còi, chuông, đèn báo...
- Trung tâm báo cháy thường có một hoặc nhiều kênh (zone): Một số trung tâm báo cháy cho phép mở rộng được, trong khi một số khác lại không cho mở rộng. Điều này làm giảm khả năng hữu dụng khi cơ sở muốn mở rộng thêm hệ thống thiết bị báo cháy.
- Một số Zone sử dụng 2 hoặc 4 lõi dây nên số lượng dây tín hiệu nối về trung tâm báo cháy là rất lớn.

Bảng 3-2: Ưu/nhược điểm của hệ thống báo cháy thường

Ưu điểm	Nhược điểm
Phát hiện sớm các mối nguy	Không phát hiện được chính xác vị trí cháy
Tăng tính an toàn	Xác suất phát hiện cháy không cao
Tích hợp với các hệ thống khác	Có thể gây nhiễu
Chi phí hợp lí	

### 3.1.1.3. Hệ thống báo cháy địa chỉ

Hệ thống báo cháy địa chỉ là hệ thống các thiết bị tự động phát hiện ra đám cháy và định vị điểm cháy. Chúng có những tính năng vượt trội hơn hệ thống báo cháy thường. Hệ thống này có khả năng giám sát, báo cháy và điều khiển các thiết bị theo từng địa chỉ.



Hình 3-3: Hệ thống báo cháy địa chỉ

Hệ thống này có thể phát hiện ra đám cháy trong khu vực được lắp đặt một cách nhanh chóng, tự động và độ cảnh báo chính xác cao. Khi phát hiện ra mối nguy hệ thống sẽ tự động phát ra các tín hiệu báo động, tự động chỉ thị và điều khiển các thiết bị trong hệ thống nhằm thực hiện một nhiệm vụ nào đó. Đặc biệt, hệ thống còn có khả năng phát hiện ra những sự cháy âm ỉ khi chưa có ngọn lửa.

- Đặc điểm chính của hệ thống báo cháy địa chỉ:
  - Sử dụng mạch loop (vòng tín hiệu) để cấp nguồn và truyền tín hiệu liên lạc.
  - Mỗi mạch loop có thể kết nối hơn 100 thiết bị có địa chỉ, tùy theo nhà sản xuất.
  - Tất cả các thiết bị trong loop được giám sát liên tục bởi tủ trung tâm.
  - Thiết bị không có địa chỉ vẫn có thể kết nối thông qua module địa chỉ.
  - Mỗi địa chỉ được gán duy nhất một vị trí, giúp xác định chính xác điểm cháy.
  - Cho phép lập trình linh hoạt các đầu vào – đầu ra.
  - Hỗ trợ báo cháy theo điểm, giúp người dùng nhanh chóng xác định vị trí cháy.
- Ưu điểm của hệ thống:
  - Chi phí hiệu quả cho các ứng dụng lớn hơn.
  - Vị trí của tình trạng cháy được phát hiện và ghi lại tại mỗi thiết bị riêng lẻ, xác định chính xác nơi xảy ra hỏa hoạn. Điều này sẽ cải thiện thời gian đáp ứng cho người dùng có biện pháp xử lý.

- Chi phí dịch vụ liên tục thấp hơn, bởi vì khi một thiết bị gặp sự cố thì bảng điều khiển sẽ cho bạn biết vị trí chính xác của thiết bị cần dịch vụ.
- **Khả năng trực tuyến:** Bảng thông minh mới có khả năng cung cấp thông báo trực tuyến chi tiết về các sự kiện báo động / sự cố / giám sát.
- Nhược điểm của hệ thống:
  - Chi phí đầu tư cao: Thiết bị và tủ trung tâm có địa chỉ thường đắt hơn so với hệ thống thường.
  - Cài đặt và cấu hình phức tạp: Cần kỹ thuật viên có chuyên môn để lập trình, gán địa chỉ, cấu hình phần mềm.
  - Khó bảo trì, sửa chữa: Khi có lỗi, việc xác định và thay thế thiết bị hỏng yêu cầu công cụ và trình độ chuyên môn.
  - Yêu cầu vật tư đồng bộ: Thường phải sử dụng thiết bị của cùng một hang để đảm bảo tương thích.
  - Mất nhiều thời gian thi công: Do cần kiểm tra từng địa chỉ thiết bị và cấu hình toàn hệ thống.

#### *3.1.1.4. Hệ thống báo cháy thông minh*

Hệ thống báo cháy thông minh (Intelligent fire alarm system) là hệ thống báo cháy tự động. Ngoài chức năng báo cháy thường và địa chỉ, còn có thể đo được một số thông số về cháy của nơi lắp đặt đầu báo cháy như nhiệt độ, nồng độ khói hoặc tự thay đổi ngưỡng tác động của đầu báo cháy theo yêu cầu của nhà thiết kế và lắp đặt.



*Hình 3-4: Hệ thống báo cháy thông minh*

Trong hệ thống báo cháy thông minh, các cảm biến đầu vào sẽ được tích hợp thêm một vi điều khiển hoặc vi xử lý. Sẽ có một chương trình chạy trên vi điều khiển hoặc vi xử lý này để phân tích và đánh giá môi trường xung quanh, sau đó gửi các tín hiệu báo động hoặc thông tin lỗi đến trung tâm điều khiển.

So với hệ thống báo cháy khác thì hệ thống báo cháy thông minh có độ phức tạp cao hơn nhiều. Thiết kế phức tạp này nhằm tăng độ chính xác trong việc phát hiện mối nguy qua đó giảm báo động giả, cảnh báo chính xác.

#### *3.1.1.5. Hệ thống báo cháy không dây*

Báo cháy không dây là giải pháp công nghệ cao, được nâng cấp lên từ hệ thống báo cháy truyền thống, trong đó sử dụng công nghệ không dây để phát hiện và cảnh báo về nguy cơ cháy. Giải pháp này sử dụng sóng radio để liên kết các cảm biến khói, nhiệt, ... và các thiết bị phụ trợ khác mà không cần dùng dây dẫn nguồn hay dây dẫn tín hiệu.

Hệ thống báo cháy không dây hoạt động bằng cách kết nối từ trung tâm báo cháy với các đầu báo khói và nhiệt, nút nhấn báo cháy khẩn cấp, chuông báo cháy, đèn báo cháy và các thành phần khác thông qua các tín hiệu không dây. Những thành phần này giao tiếp với bảng điều khiển trung tâm sử dụng các giao thức truyền thông để liên tục truyền nhận tín hiệu về trạng thái của các thiết bị trong hệ thống.

- Một số trạng thái thường gặp của hệ thống:
  - Trạng thái hoạt động bình thường.
  - Mất đầu báo hoặc đầu báo không hoạt động.
  - Lỗi tiếp địa.
  - Lỗi nguồn AC hoặc lỗi nguồn DC.
  - Cảnh báo khói vượt ngưỡng an toàn.
  - Cảnh báo nhiệt vượt ngưỡng hoặc đang gia tăng bất thường.
- Ưu điểm của hệ thống:
  - Linh hoạt khi lắp đặt: Không cần đi dây, không đục tường, thích hợp cho công trình đã xây dựng xong.
  - Tiết kiệm thời gian và chi phí thi công: Giảm đến 90% chi phí nhân công so với hệ thống có dây.
  - Dễ mở rộng hoặc di dời thiết bị: Phù hợp khi thay đổi bố trí phòng hoặc nâng cấp hệ thống.
  - Tích hợp cao với hệ thống an ninh và BMS: Hỗ trợ kết nối số trong hệ thống nhà thông minh.
  - Giảm gián đoạn vận hành: Không ảnh hưởng đến các hoạt động trong nhà khi nâng cấp hay bảo trì.

- Hỗ trợ theo dõi từ xa: Kiểm tra tình trạng thiết bị pin, sự cố, ... thông qua app hoặc phần mềm quản lý.
- Nhược điểm của hệ thống:
  - Phụ thuộc vào pin: Cần kiểm tra và thay pin định kỳ để đảm bảo hoạt động ổn định.
  - Khoảng cách truyền tín hiệu giới hạn: Ở môi trường có vật cản dày, tín hiệu dễ bị suy giảm.
  - Chi phí đầu tư thiết bị ban đầu cao hơn so với đầu báo có dây đơn lẻ.
  - Dễ bị nhiễu sóng nếu sử dụng gần các thiết bị không dây khác như Router Wi-fi, thiết bị RF.
  - Chưa phổ biến tại các công trình lớn hoặc yêu cầu giám sát nghiêm ngặt do lo ngại về độ ổn định.

### **3.1.2 Đánh giá tổng quan các hệ thống hiện có và đề xuất giải pháp**

Hiện nay, trên thị trường tồn tại nhiều loại hệ thống báo cháy với các đặc điểm và phạm vi ứng dụng khác nhau, từ các thiết bị báo cháy độc lập đơn giản đến các hệ thống báo cháy thông minh phức tạp. Mỗi hệ thống đều có ưu và nhược điểm riêng, phù hợp với nhu cầu và quy mô của từng công trình cụ thể.

Thiết bị báo cháy độc lập là sự lựa chọn phổ biến cho hộ gia đình, cửa hàng nhỏ nhở vào giá thành rẻ, dễ lắp đặt và không yêu cầu kết nối dây dẫn. Tuy nhiên phạm vi bảo vệ hạn chế, không thể mở rộng hoặc kết nối với các hệ thống khác là điểm yếu lớn, khiến thiết bị này không phù hợp với công trình có quy mô lớn hơn.

Hệ thống báo cháy thường là bước phát triển cao hơn, cho phép kết nối nhiều thiết bị trong một khu vực (zone) và tích hợp với trung tâm điều khiển. Ưu điểm của hệ thống này là chi phí lắp đặt hợp lý và khả năng tích hợp các thiết bị đầu vào – đầu ra. Tuy nhiên, việc không thể xác định chính xác vị trí điểm cháy làm giảm hiệu quả ứng phó và bảo trì, đặc biệt trong môi trường nhiều thiết bị.

Hệ thống báo cháy địa chỉ khắc phục được hạn chế trên nhờ khả năng định danh từng thiết bị theo địa chỉ riêng biệt. Điều này giúp xác định chính xác điểm xảy ra cháy, làm giảm thời gian phản ứng và hỗ trợ công tác xử lý hiệu quả hơn. Mặc dù vậy, hệ thống này đòi hỏi chi phí đầu tư cao, quá trình thi công và lắp đặt cầu hình phức tạp, yêu cầu người có chuyên môn.

Hệ thống báo cháy thông minh tiếp tục nâng cấp bằng cách tích hợp vi điều khiển hoặc vi xử lý tại từng cảm biến, giúp xử lý và đánh giá môi trường tại chỗ, từ đó giảm cảnh báo giả và tăng độ chính xác. Đây là xu hướng phát triển tất yếu trong bối cảnh chuyển đổi số, song vẫn gặp rào cản lớn về chi phí và yêu cầu công nghệ cao.

Cuối cùng, hệ thống báo cháy không dây là giải pháp mang tính linh hoạt cao, đặc biệt phù hợp với các công trình cải tạo hoặc không tiện đi dây. Với khả năng kết nối nhanh, dễ mở rộng và hỗ trợ theo dõi từ xa, hệ thống này ngày càng được ưa chuộng. Tuy nhiên, nó vẫn còn một số hạn chế như giới hạn khoảng cách truyền tín hiệu, dễ bị nhiễu và phụ thuộc vào pin.

Từ những đánh giá tổng quan về các hệ thống báo cháy hiện có, có thể nhận thấy mỗi hệ thống đều có những ưu điểm và hạn chế nhất định. Để nâng cao hiệu quả phát hiện và cảnh báo cháy, đồng thời phù hợp với xu thế công nghệ hiện nay, cần có các cải tiến dựa trên nền tảng trí tuệ nhân tạo (AI), Internet vạn vật (IoT) và kỹ thuật số hóa trong quản lý.

Tích hợp công nghệ IoT vào hệ thống báo cháy nhằm hỗ trợ giám sát và điều khiển từ xa. Việc kết nối các thiết bị cảm biến với nền tảng đám mây cho phép theo dõi trạng thái thiết bị theo thời gian thực, nhận cảnh báo qua điện thoại hoặc máy tính và đồng bộ dữ liệu lên hệ thống trung tâm. Người sử dụng có thể biết được tình trạng pin yếu, thiết bị mất kết nối hay lỗi cảm biến mà không cần kiểm tra thủ công.

Ứng dụng trí tuệ nhân tạo (AI) vào phân tích dữ liệu cảm biến sẽ giúp hệ thống nâng cao khả năng nhận diện cháy thực tế và giảm thiểu báo động giả. Thuật toán học máy (Machine Learning) có thể được huấn luyện để phân biệt giữa khói do cháy và khói do nấu ăn, bụi hoặc hơi nước, từ đó tăng độ tin cậy. Hệ thống còn có thể dự đoán nguy cơ cháy thông qua sự thay đổi bất thường về nhiệt độ, nồng độ khói tích tụ liên tục hoặc các hành vi môi trường không bình thường.

Ngoài ra, cần tăng cường khả năng tích hợp của hệ thống báo cháy với các hệ thống quản lý toàn nhà như hệ thống điều hòa, quạt hút khói, hệ thống chiếu sáng khẩn cấp và cửa ngăn cháy. Việc đồng bộ vận hành giữa các hệ thống sẽ giúp tăng tốc độ phản ứng khi có sự cố xảy ra, đảm bảo an toàn cho người và tài sản.

Bên cạnh đó nên phát triển phần mềm giám sát tập trung với giao diện trực quan, hiển thị bản đồ tòa nhà và trạng thái thiết bị theo thời gian thực. Giao diện này có thể hỗ trợ trên nhiều nền tảng như máy tính, điện thoại hoặc thiết bị đeo thông minh, giúp người quản lý theo dõi mọi lúc, mọi nơi. Đồng thời, việc sử dụng các nguồn năng lượng thay thế như pin mặt trời cho thiết bị không dây cũng nên được xem xét để tăng tính ổn định và giảm chi phí.

## 3.2 Phân tích và thiết kế hệ thống

### 3.2.1 Phân tích tổng thể

Trong hệ thống báo cháy, cảm biến và camera đóng vai trò cực kỳ quan trọng. Bên cạnh đó sẽ có những thiết bị cảnh báo như đèn, chuông sẽ được lắp đặt ở những nơi dễ nhận biết, không bị hạn chế âm thanh, nên đặt gần các thiết bị phòng cháy chữa cháy. Qua đó, nhóm em đưa ra cấu trúc hệ thống như sau:

### Bộ cảm biến:

- Cảm biến khói quang
- Cảm biến nồng độ khí CO
- Cảm biến nhiệt độ, độ ẩm
- Vị trí lắp đặt đề xuất:
  - Trần nhà, cách tường ít nhất 50 cm (để phát hiện khói bay lên).
  - Khu vực trung tâm căn hộ (phát hiện khói từ nhiều hướng).
  - Không được quá gần bếp và nhà vệ sinh (tránh báo động giả do khói nấu ăn, hơi nước).
  - Cách xa quạt hút, cửa sổ, điều hòa (hạn chế luồng khí làm loãng CO).
  - Gần phòng ngủ (để phòng ngộ độc CO khi ngủ).

### Các thiết bị báo động:

- Đèn báo
- Chuông báo
- Thông báo qua app

Ngoài ra, để tăng độ chính xác, hạn chế cảnh báo giả, nhóm em quyết định tích hợp thêm trí tuệ nhân tạo (AI) để phát hiện khói, lửa. Nhóm em sẽ sử dụng một camera gắn với một máy tính nhúng làm trung tâm điều khiển. Trung tâm điều khiển này sẽ được gắn ở ngoài căn hộ, tại hành lang hay các khu sinh hoạt chung (như nhà xe, sân thượng, ...). Bình thường trung tâm điều khiển sẽ sử dụng nguồn điện chính từ ổ cắm, nhưng đòi hỏi cần nguồn điện dự phòng để có thể hoạt động khi có sự cố điện xảy ra, đặc biệt là sự cố điện do hỏa hoạn.

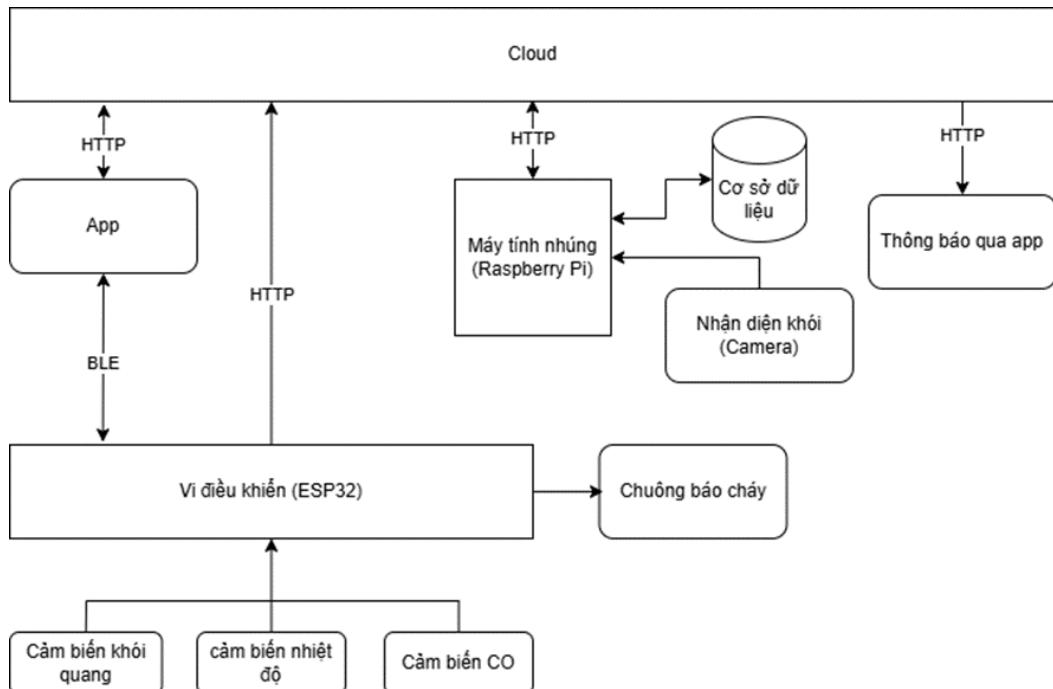
### Một hệ thống trung tâm điều khiển sẽ có chức năng:

- **Nhận dữ liệu:** Hệ thống này sẽ liên tục nhận dữ liệu từ các cảm biến và camera và tiến hành xử lý tại biên. Việc xử lý tại biên này giúp hệ thống có thể nhanh chóng phản ứng trước các tình huống khẩn cấp.
- **Lưu trữ dữ liệu:** Dữ liệu thu thập được sau khi xử lý sẽ được lưu trữ để phân tích, đánh giá hiệu suất hệ thống. Dữ liệu này sẽ được dùng để huấn luyện hệ thống, giúp nâng cao độ chính xác, phát hiện từ sớm các vấn đề.
- **Gửi cảnh báo đến người dùng:** Ngay khi phát hiện các dấu hiệu bất thường nhận được từ cảm biến, hệ thống sẽ ngay lập tức bật đèn cảnh báo. Nếu sau một khoảng thời gian không có phản hồi từ người dùng và chỉ số dữ liệu các cảm biến thu được tiếp tục tăng thì sẽ thông báo qua app.

- **Giao diện app** để người dùng điều khiển, cấu hình, hiển thị.

### 3.2.2 Sơ đồ khái quát của hệ thống

Từ những phân tích ở mục 3.1.1, nhóm em đưa ra sơ đồ khái quát cho toàn hệ thống như sau:



Hình 3-5: Sơ đồ khái quát của hệ thống

Hệ thống được thiết kế để phát hiện và cảnh báo sớm nguy cơ cháy thông qua phân tích hình ảnh khói và dữ liệu từ các cảm biến. Hệ thống xử lý tại biên (ngay trên máy tính nhúng) và gửi dữ liệu lên cloud để lưu trữ, làm cơ sở huấn luyện sau này.

**Vi điều khiển (ESP32)** sẽ đóng vai trò trung gian trong việc điều khiển, xử lý và gửi các dữ liệu từ các cảm biến đến trung tâm điều khiển.

- *Cảm biến khói quang*: Cảm biến này sẽ hoạt động dựa trên nguyên lý tán xạ ánh sáng. Khi có khói đi qua luồng cảm biến, các hạt khói làm tán xạ ánh sáng từ một nguồn sáng (thường là LED hồng ngoại). Cảm biến quang điện sẽ phát hiện mức độ ánh sáng bị tán xạ, từ đó xác định mật độ khói trong không khí. Yêu cầu cảm biến phải phân biệt được các loại khói khác, ví dụ khói do cháy sẽ có nồng độ hạt mịn cao, màu đen hoặc xám, dễ tán xạ ánh sáng mạnh; khói do nấu ăn thường chứa dầu mỡ và hơi nước nên có thể phản xạ ánh sáng khác với khói cháy; hơi nước thì không có hạt rắn, ít làm tán xạ ánh sáng; khói do thắp hương thì hạt khói nhỏ, màu trắng hoặc xám nhẹ, có thể gây nhầm lẫn với khói do cháy nhưng nồng độ sẽ thấp hơn.

- *Cảm biến nhiệt độ*: Cảm biến nhiệt độ có nhiệm vụ đo lường nhiệt độ môi trường xung quanh để phát hiện dấu hiệu hỏa hoạn tiềm tàng. Khi nhiệt độ tăng cao đột ngột vượt ngưỡng an toàn, hệ thống sẽ cảnh báo ngay lập tức bằng cách kích hoạt đèn báo.
- *Cảm biến khí CO (Carbon Monoxide)*: Khi hỏa hoạn xảy ra, nhiệt độ cao sẽ làm phân hủy các chất hữu cơ, nếu không đủ Oxy để phản ứng tạo thành CO<sub>2</sub> thì CO sẽ sinh ra, khi hít phải khí này nó sẽ kết hợp với hemoglobin trong máu, ngăn cản Oxy lưu thông trong cơ thể. Cảm biến khí CO giúp phát hiện sự gia tăng nồng độ CO trong không khí qua đó đưa ra cảnh báo từ sớm.
- *Chuông báo cháy*: Phát ra tín hiệu âm thanh to và rõ, giúp cảnh báo cho mọi người trong chung cư, đặc biệt hữu ích vào buổi đêm.\

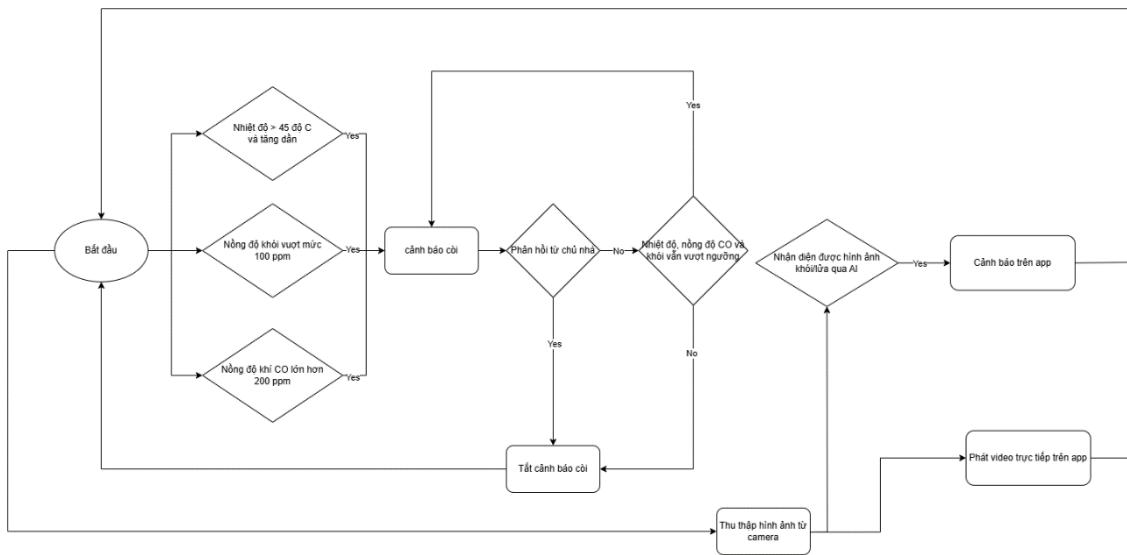
Vi điều khiển sẽ giao tiếp với máy tính nhúng thông qua giao thức HTTP. Đây là giao thức nhẹ dành cho thiết bị IoT để giao tiếp với máy tính nhúng. Điều này giúp truyền dữ liệu nhanh chóng và đáng tin cậy giữa các thiết bị.

**Máy tính nhúng:** Nhận dữ liệu từ vi điều khiển, lưu trữ các dữ liệu nhận được vào cơ sở dữ liệu. Do xử lý tại biên nên máy tính nhúng sẽ lưu trữ dữ liệu cục bộ và chỉ gửi lên cloud để phục vụ phân tích sau này. Ngoài ra máy tính nhúng còn đảm nhận chức năng nhận diện khói bằng trí tuệ nhân tạo (AI) (có thể là sẽ sử dụng YOLOv8). Cuối cùng, máy tính nhúng sẽ giao tiếp với người dùng thông qua app giúp dễ dàng hiển thị các dữ liệu và điều khiển cho người dùng (máy tính nhúng gửi dữ liệu lên cloud, sau đó app sẽ lấy dữ liệu từ cloud để hiển thị).

- *Nhận diện khói thông qua công nghệ AI*: Phân tích hình ảnh thu được từ camera để nhận diện khói. AI sẽ được huấn luyện để nhận biết các dấu hiệu của khói từ hình ảnh/video, giúp phát hiện sớm các nguy cơ cháy.
- *Cơ sở dữ liệu*: Cơ sở dữ liệu giữ tất cả dữ liệu từ các cảm biến và hoạt động của hệ thống để phục vụ cho việc phân tích và huấn luyện.
- *App*: App cung cấp giao diện cho người dùng để họ có thể xem trực tiếp dữ liệu từ hệ thống, nhận thông báo khẩn cấp.

**Cloud:** Đóng vai trò là trung tâm dữ liệu và giao tiếp từ xa, giúp kết nối giữa máy tính nhúng, app người dùng và hệ thống lưu trữ.

### 3.2.3 Sơ đồ thuật toán tổng quát



Hình 3-6: Sơ đồ thuật toán tổng quát

**Kiểm tra điều kiện nguy hiểm vật lý:** Hệ thống sẽ liên tục giám sát các thông số môi trường và kiểm tra các điều kiện nguy hiểm sau:

- *Kiểm tra nhiệt độ vượt quá 45°C:* Đây là mức nhiệt có nguy hiểm tiềm ẩn, môi trường lúc này đang quá nóng (so với nhiệt độ phòng bình thường), có khả năng có cháy. Dự đoán sẽ được cung cấp nếu nhiệt độ ngày càng tăng, không có xu hướng giảm.
- *Kiểm tra nồng độ CO vượt mức 200 ppm (part per million – một phần triệu):* Nồng độ khí CO trong không khí là bình thường khi dưới 0.001% (tức 100 ppm). Nếu nồng độ này từ 0.01% trở lên sẽ gây độc cho người hít phải. Vậy nên lựa chọn ngưỡng 200ppm để giảm cảnh báo giả (do xe cộ, ...), hơn nữa đây chưa phải mức quá nguy hiểm gây tử vong, chỉ gây đau đầu nếu tiếp xúc quá lâu, vẫn đủ thời gian để cảnh báo và phản ứng.
- *Kiểm tra nồng độ khói vượt mức 100 ppm:* Cảm biến khói đo nồng độ khói trong không khí dưới dạng ppm – đơn vị biểu thị số phần tử khói trong một triệu phần tử không khí. Ngưỡng 100 ppm được lựa chọn nhằm phát hiện sớm sự xuất hiện của khói bất thường trong môi trường sinh hoạt, nhưng vẫn đủ để tránh các cảnh báo giả do hơi nước hoặc nấu ăn nhẹ. Mặc dù không tương đương với đơn vị obscuration của các đầu báo cháy quang học, giá trị này đã được kiểm nghiệm phù hợp để phát hiện các hiện tượng như khói cháy âm ỉ hoặc vật liệu bị đốt nóng. Nếu nồng độ khói tiếp tục tăng cao hoặc kết hợp với nhiệt độ và khí CO vượt ngưỡng, hệ thống sẽ đánh giá đây là tình huống có nguy cơ cháy cao và kích hoạt báo động.

**Dấu hiệu nguy hiểm trực quan:** Song song với việc kiểm tra điều kiện nguy hiểm vật lý, hệ thống cũng sẽ liên tục phân tích hình ảnh thu được từ camera. Nếu phát hiện ra hình ảnh khói hệ thống sẽ ngay lập tức gửi cảnh báo đến app.

**Cảnh báo nội bộ:** Bất kỳ điều kiện nguy hiểm vật lý nào được phát hiện thì hệ thống sẽ bật còi cảnh báo.

**Cảnh báo từ xa:** Khi AI nhận diện được hình ảnh cháy từ camera thì ngay lập tức sẽ có cảnh báo trên app.

**Đo lại thông số từ đầu:** Khi không còn nhận diện được hình ảnh khói và các thông số cảm biến đã ở mức an toàn thì hệ thống sẽ dừng cảnh báo.

### 3.2.4 Thiết kế phần mềm hệ thống

#### 3.2.4.1. Nghiệp vụ hệ thống

Dựa trên những phân tích và thông tin thu thập được, nhóm em đưa ra nghiệp vụ hệ thống của hệ thống được xác định như sau:

- Đăng ký tài khoản và đăng nhập:** Người dùng có thể đăng ký tài khoản để truy cập vào hệ thống.
- Kích hoạt tài khoản:** Sau khi đăng ký tài khoản, để có thể sử dụng đầy đủ tính năng của hệ thống thì người dùng sẽ phải kích hoạt tài khoản bằng mã OTP được cung cấp.
- Quản lý thông tin tài khoản:** Hệ thống có thể hiển thị dữ liệu, thông tin tài khoản để người dùng có thể kiểm tra và cập nhật.
- Hiển thị dữ liệu cảm biến:** Hệ thống có thể hiển thị dữ liệu của các loại cảm biến và theo dõi được khu vực lắp đặt máy tính nhúng qua camera.
- Điều khiển thiết bị:** Người dùng có thể điều khiển các thiết bị thông qua hệ thống.
- Thông báo:** Khi máy tính nhúng phát hiện ra các nguy cơ cháy thì sẽ gửi thông báo đến cho người dùng.

#### 3.2.4.2. Yêu cầu về dữ liệu

Bảng 3-3: Bảng yêu cầu dữ liệu

Tên dữ liệu	Mô tả
Tài khoản	ID tài khoản, tên người dùng, mật khẩu, email, số điện thoại, số tầng.
Nhiệt độ	Nhiệt độ, thời gian
Độ ẩm	Độ ẩm, thời gian
Khí CO	Nồng độ khí CO, thời gian
Nồng độ khói	Nồng độ khói, thời gian

Tài khoản:

- Có ID tài khoản để nhận diện duy nhất mỗi người dùng.
- Tên người dùng, số điện thoại, số tầng, email để hiển thị và tương tác.
- Email, mật khẩu sử dụng để đăng nhập vào hệ thống.
- Email để xác thực khi người dùng quên tài khoản.

Nhiệt độ:

- Giá trị nhiệt độ cảm biến đo được.
- Thời gian đo được ghi lại để theo dõi biến động nhiệt độ.

Độ ẩm:

- Giá trị độ ẩm cảm biến đo được.
- Thời gian đo được ghi lại để theo dõi biến động độ ẩm.

Khí CO:

- Giá trị nồng độ CO cảm biến đo được.
- Thời gian đo được ghi lại để theo dõi biến động nồng độ khí CO.

Nồng độ khói:

- Giá trị nồng độ khói cảm biến đo được.
- Thời gian đo được ghi lại để theo dõi biến động nồng độ khói.

#### 3.2.4.3. Yêu cầu về người dùng

Bảng 3-4: Bảng yêu cầu về người dùng

Tên tác nhân	Mô tả tác nhân
Người dùng (user)	Người dùng có thể truy cập vào hệ thống, sau khi kích hoạt tài khoản có thể sử dụng tất cả các chức năng của hệ thống.

Người dùng (user) chính là tác nhân chính trong hệ thống. Khi tài khoản của họ được kích hoạt họ sẽ có quyền truy cập vào toàn bộ các chức năng của hệ thống, bao gồm việc giám sát và điều khiển các cảm biến, nhận thông báo về tình trạng môi trường trong ngôi nhà, thực hiện các thao tác để xử lý khi có cháy nổ xảy ra. Tác nhân này sẽ là trung tâm của hoạt động hệ thống, đóng vai trò quan trọng trong việc tương tác và khai thác tối đa các tính năng mà hệ thống cung cấp.

#### 3.2.4.4. Yêu cầu chức năng

Các yêu cầu chức năng và tác nhân thực hiện trong hệ thống được mô tả trong bảng dưới đây:

Bảng 3-5: Bảng yêu cầu chức năng

STT	Chức năng	Mô tả chức năng	Tác nhân
1	Tạo tài khoản	Cho phép người dùng đăng ký tài khoản để truy cập vào hệ thống	Người dùng
2	Đăng nhập	Cho phép người dùng đã có tài khoản truy cập vào hệ thống	Người dùng
3	Quên mật khẩu	Cho phép người dùng đã có tài khoản xác thực lại tài khoản khi quên	Người dùng
4	Xác thực tài khoản	Cho phép người dùng có mã xác thực hợp lệ sử dụng các chức năng của hệ thống	Người dùng
5	Quản lý tài khoản	Cho phép người dùng cập nhật tài khoản trong hệ thống	Người dùng
6	Quản lý thiết bị	Cho phép người dùng kết nối Bluetooth với các cảm biến để kết nối wifi cho chúng	Người dùng
7	Xem video trực tiếp	Cho phép người dùng theo dõi khu vực lắp đặt máy tính nhúng	Người dùng
8	Thông báo	Cho phép người dùng xem thông báo khi hệ thống phát hiện ra các nguy cơ hỏa hoạn	Người dùng

Dựa trên phân tích yêu cầu của người dùng và mục tiêu xây dựng hệ thống cảnh báo hỏa hoạn kết hợp AI, nhóm em đã xác định các chức năng chính mà phần mềm hệ thống cần đáp ứng. Các chức năng này được trình bày trong bảng yêu cầu chức năng trên.

Sau đây sẽ là mô tả chi tiết các chức năng, bao gồm mục đích sử dụng, cách thức hoạt động và vai trò của người dùng trong quá trình tương tác với hệ thống. Việc triển khai đầy đủ các chức năng này là cơ sở để đảm bảo hệ thống hoạt động ổn định, hiệu quả và thân thiện với người dùng.

- Tạo tài khoản:** Cho phép người dùng tiến hành đăng ký thông tin cá nhân (email, mật khẩu, tên người dùng) để khởi tạo một tài khoản mới trên hệ thống. Đây là bước khởi đầu giúp mỗi người dùng có thể sử dụng các tính năng riêng biệt của mình.

- Sau khi có tài khoản, người dùng sử dụng chức năng **Đăng nhập** để truy cập vào hệ thống bằng thông tin đã đăng ký. Việc xác thực thông tin đăng nhập đảm bảo an toàn và phân quyền dữ liệu truy cập theo từng người dùng.
- Trong trường hợp người dùng quên mật khẩu, chức năng **Quên mật khẩu** hỗ trợ người dùng thực hiện quá trình khôi phục lại toàn khoản thông qua email xác thực. Chức năng này giúp duy trì tính bảo mật và tránh mất quyền truy cập vào hệ thống.
- Để đảm bảo người dùng là hợp lệ, chức năng **Xác thực tài khoản** sẽ kiểm tra mã xác thực mà người dùng được cung cấp khi tạo tài khoản (chỉ kỹ thuật viên lắp đặt mới có thể xem được mã này). Sau khi xác thực thành công, tài khoản mới được phép sử dụng các chức năng quản lý và nhận cảnh báo của hệ thống.
- Chức năng **Quản lý tài khoản** hỗ trợ người dùng cập nhật thông tin cá nhân như mật khẩu, số điện thoại, địa chỉ email, tên người dùng, số tầng khi cần thiết. Điều này giúp người dùng chủ động trong việc duy trì và bảo mật tài khoản.
- Chức năng **Quản lý thiết bị** cho phép người dùng kết nối Bluetooth từ phần mềm hệ thống đến các cảm biến. Thông qua kết nối này, người dùng có thể cấu hình Wi-fi cho cảm biến, giúp thiết bị kết nối với hệ thống mạng để gửi dữ liệu lên Server.
- Chức năng **Xem video trực tiếp (stream video)** là một phần quan trọng trong hệ thống. Máy tính nhúng Raspberry Pi4 được tích hợp camera và thực hiện truyền video thời gian thực lên server. Ứng dụng di động của người dùng có thể truy cập vào luồng video này để theo dõi hình ảnh trực tiếp từ khu vực lắp đặt thiết bị. Điều này giúp người dùng kiểm chứng ngay lập tức tình hình thực tế khi có cảnh báo, từ đó đưa ra xử lý kịp thời.
- Chức năng **Thông báo** giúp người dùng nhận được cảnh báo từ hệ thống khi phát hiện ra các dấu hiệu bất thường là khói, lửa. Các thông báo này được hiển thị ngay trên ứng dụng để người dùng có thể nhận biết để xử lý kịp thời.

#### *3.2.4.5. Yêu cầu phi chức năng*

Hiệu năng (Performance):

- Hệ thống phải xử lý và gửi cảnh báo đến người dùng trong vòng dưới 5 giây kể từ khi cảm biến phát hiện sự cố.
- Ứng dụng di động phải tải giao diện và hiển thị thông báo ngay lập tức khi có dữ liệu mới từ Server.

- Luồng video từ camera truyền tới phải đảm bảo có độ trễ thấp và chất lượng tối thiểu 360P.

Độ tin cậy (Reliability):

- Hệ thống có thể hoạt động liên tục 24/7.

Tính khả dụng (Availability):

- Server phải sẵn sàng hoạt động trong thời gian triển khai thực hiện.
- Ứng dụng di động phải tương thích tốt trên Android, không crash khi chạy nền.

Tính bảo mật (Security):

- Dữ liệu đăng nhập, xác thực tài khoản và các lệnh điều khiển phải được mã hóa (http/https) trong quá trình truyền tải.
- Mỗi tài khoản người dùng chỉ được phép truy cập và xem thông tin của thiết bị mà mình sở hữu (phân quyền truy cập).

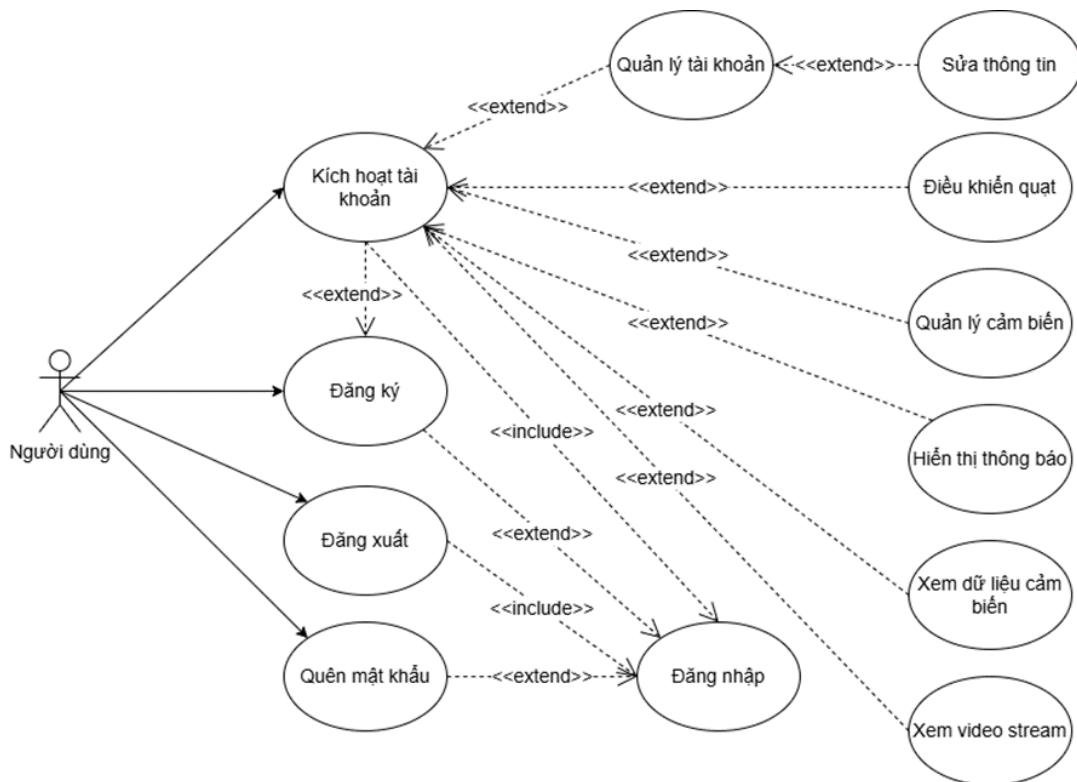
Khả năng mở rộng (Scalability):

- Hệ thống cần hỗ trợ dễ dàng thêm mới các cảm biến hoặc Raspberry Pi khác trong tương lai mà không ảnh hưởng đến hệ thống hiện tại.
- App và Server phải được thiết kế theo mô hình modular, dễ mở rộng để tích hợp các tính năng nâng cao như điều khiển từ xa, AI phân tích hình ảnh, ...

Tính thân thiện với người dùng (Usability):

- Giao diện App phải đơn giản, dễ sử dụng cho cả người không chuyên về công nghệ.
- Các thông báo cảnh báo cần rõ ràng, dễ nhận biết và có hướng dẫn hành động (Ví dụ: Phát hiện khói/lửa ở tầng 2).

### 3.2.4.6. Biểu đồ ca sử dụng tổng quan



Hình 3-7: Biểu đồ ca sử dụng tổng quan

Tác nhân chính (main actor):

- Người dùng: Là tác nhân tương tác chính với hệ thống, thực hiện các thao tác như đăng ký, đăng nhập, sử dụng các chức năng của hệ thống.

Các ca sử dụng chính:

- Quên mật khẩu
- Đăng ký
- Kích hoạt tài khoản
- Đăng xuất

Mối quan hệ:

- <<include>>:
  - Ca sử dụng “Kích hoạt tài khoản” có quan hệ <<include>> với “Đăng nhập”, nghĩa là người dùng phải đăng nhập thì mới có thể thực hiện kích hoạt tài khoản.
  - Ca sử dụng “Đăng xuất” cũng bao gồm <<include>> đến “Đăng nhập”, thể hiện sau khi đăng xuất sẽ quay về trạng thái đăng nhập.
- <<extend>>:

- Ca sử dụng “Kích hoạt tài khoản” có thể được mở rộng từ “Đăng ký”, nghĩa là sau khi đăng ký, người dùng có thể chọn kích hoạt tài khoản nếu muốn sử dụng hệ thống.
- Ca sử dụng “Quên mật khẩu” và “Đăng ký” có quan hệ <<extend>> với “Đăng nhập” nghĩa là nó là một tính năng tùy chọn mở rộng của quá trình “Đăng nhập”.

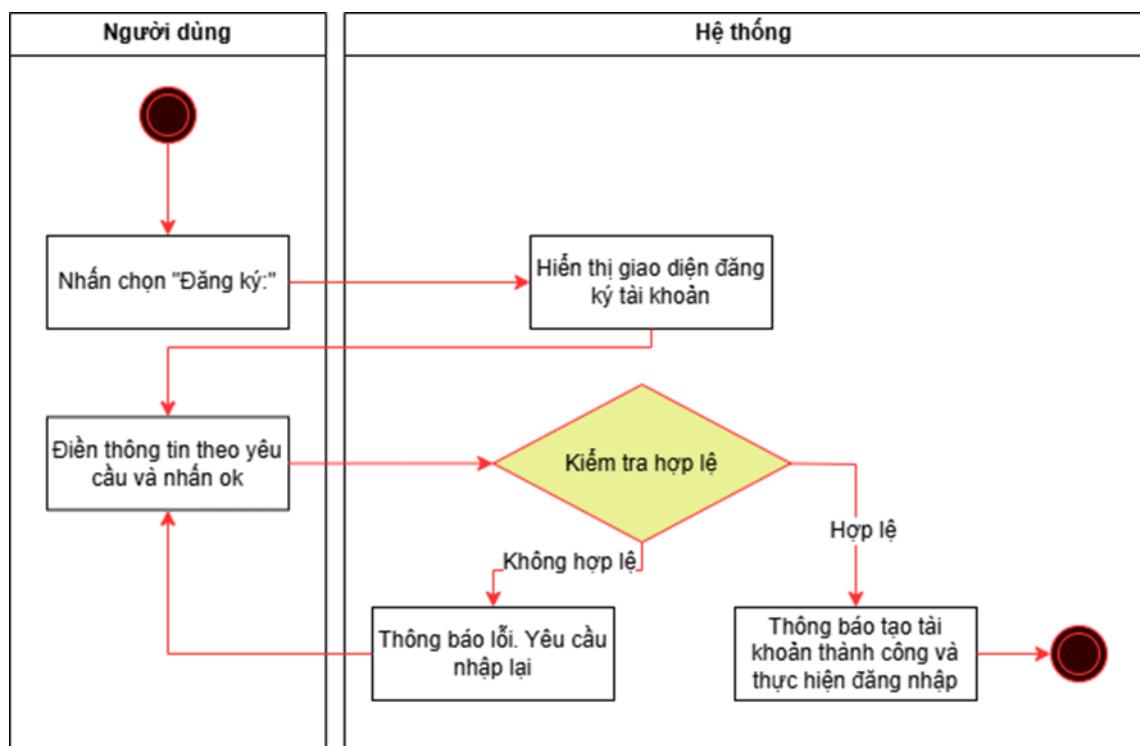
Ca sử dụng phụ thuộc vào trạng thái tài khoản:

- Các chức năng như “Quản lý tài khoản”, “Điều khiển quạt”, “Quản lý cảm biến”, “Hiển thị thông báo”, “Xem dữ liệu cảm biến”, “Xem video stream” chỉ có thể thực hiện được nếu tài khoản đã được kích hoạt, đảm bảo tính bảo mật và phân quyền hợp lý.

Ca sử dụng “Quản lý tài khoản” có chức năng mở rộng thêm là “Quản lý thông tin”.

Sơ đồ ca sử dụng đóng vai trò quan trọng trong việc mô hình hóa yêu cầu hệ thống ở mức chức năng, giúp nắm rõ các tương tác người dùng và hệ thống, từ đó hỗ trợ quá trình thiết kế và triển khai hiệu quả hơn.

#### 3.2.4.7. Chức năng đăng ký tài khoản

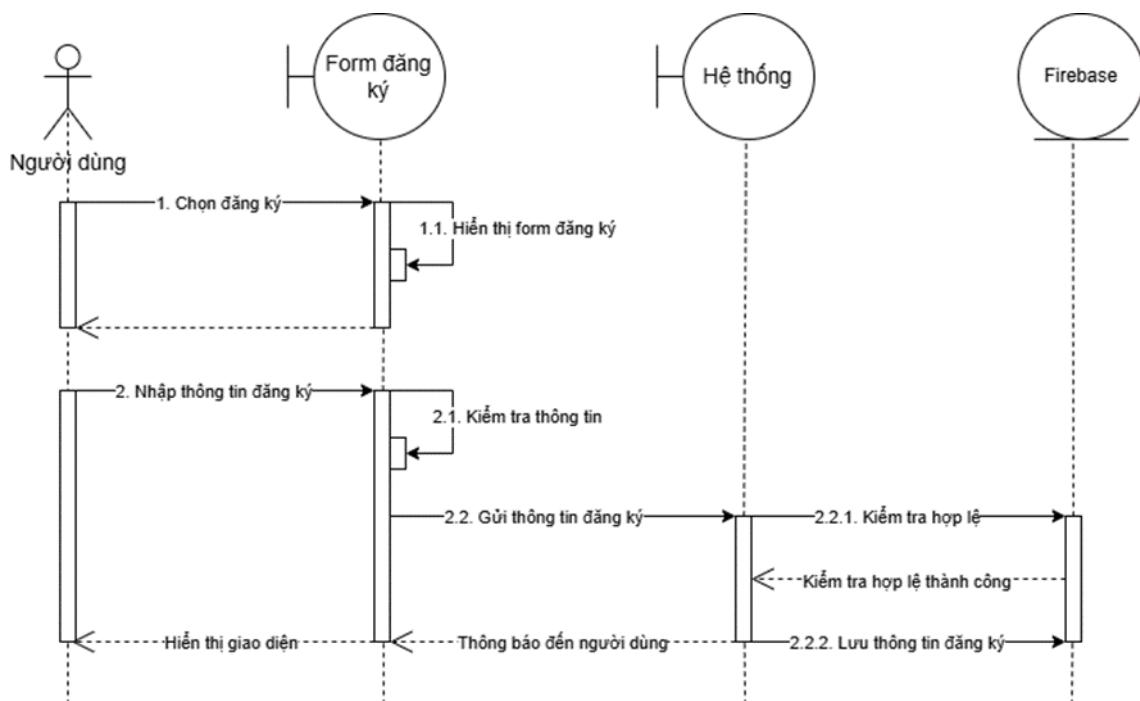


Hình 3-8: Sơ đồ hoạt động chức năng Đăng ký tài khoản

Hình 3-8 thể hiện biểu đồ hoạt động của chức năng “**Đăng ký tài khoản**”. Chức năng này cho phép người dùng khởi tạo một tài khoản mới để sử dụng các dịch vụ trong hệ thống. Quá trình này bắt đầu khi người dùng nhấn chọn “Đăng ký” từ giao diện “Đăng nhập”. Hệ thống sẽ phản hồi bằng cách hiển thị giao diện đăng ký tài khoản, tại đây sẽ có các biểu mẫu yêu cầu nhập thông tin là email, mật khẩu, tên người dùng. Sau khi người dùng điền thông tin và nhấn ok thì hệ thống sẽ thực hiện bước kiểm tra hợp lệ, bao gồm kiểm tra đã điền đầy đủ thông tin trong biểu mẫu, kiểm tra định dạng của email, kiểm tra độ mạnh của mật khẩu và xác minh email chưa tồn tại trong hệ thống.

Nếu thông tin không hợp lệ, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu người dùng nhập lại thông tin. Ngược lại, nếu thông tin hợp lệ thì hệ thống sẽ tiến hành tạo tài khoản, hiển thị thông báo và tự động đăng nhập cho người dùng. Việc kết thúc bằng trạng thái đăng nhập giúp nâng cao trải nghiệm người dùng khi vừa hoàn tất đăng ký là có thể bắt đầu sử dụng hệ thống ngay lập tức.

Sơ đồ hoạt động đã minh họa rõ vai trò của hai tác nhân là “Người dùng” và “Hệ thống”, đồng thời thể hiện đầy đủ các bước xử lý, nhánh điều kiện và phản hồi tương ứng. Đây là cơ sở để nhóm em nắm được luồng xử lý chi tiết, đảm bảo việc thiết kế và thực hiện hóa chức năng đăng ký diễn ra chính xác và đầy đủ.



Hình 3-9: Sơ đồ tuần tự chức năng Đăng ký tài khoản

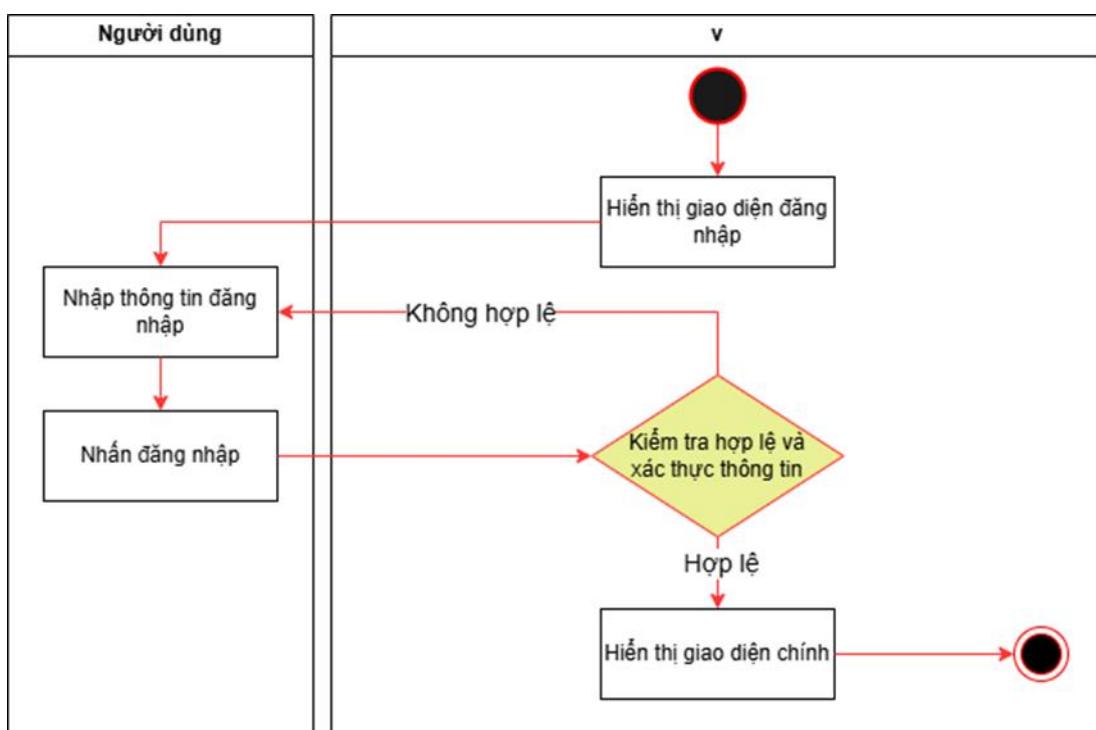
Chức năng “**Đăng ký tài khoản**” được mô tả bằng sơ đồ tuần tự trên đã thể hiện rõ trình tự các bước xử lý khi người dùng khởi tạo tài khoản mới trên hệ thống. Quá trình bắt đầu khi người dùng lựa chọn chức năng đăng ký từ giao diện đăng

nhập. Hệ thống phản hồi bằng cách hiển thị form đăng ký để người dùng nhập thông tin cần thiết bao gồm email, tên người dùng và mật khẩu.

Sau khi điền đầy đủ thông tin, người dùng nhấn ok. Form đăng ký sẽ tiến hành kiểm tra thông tin đầu vào để đảm bảo có đầy đủ dữ liệu và dữ liệu là hợp lệ trước khi gửi đến hệ thống để xử lý. Tiếp theo, hệ thống thực hiện bước kiểm tra hợp lệ là kiểm tra xem email đã tồn tại trên cơ sở dữ liệu ở firebase hay chưa, độ mạnh của mật khẩu có hợp lệ không.

Nếu dữ liệu đăng ký là hợp lệ, hệ thống sẽ tiến hành gửi yêu cầu lưu thông tin người dùng đến Firebase. Khi quá trình lưu thông tin hoàn tất, hệ thống phản hồi về form giao diện, đồng thời thông báo kết quả đăng ký đến người dùng. Kết thúc quy trình.

#### 3.2.4.8. Chức năng đăng nhập

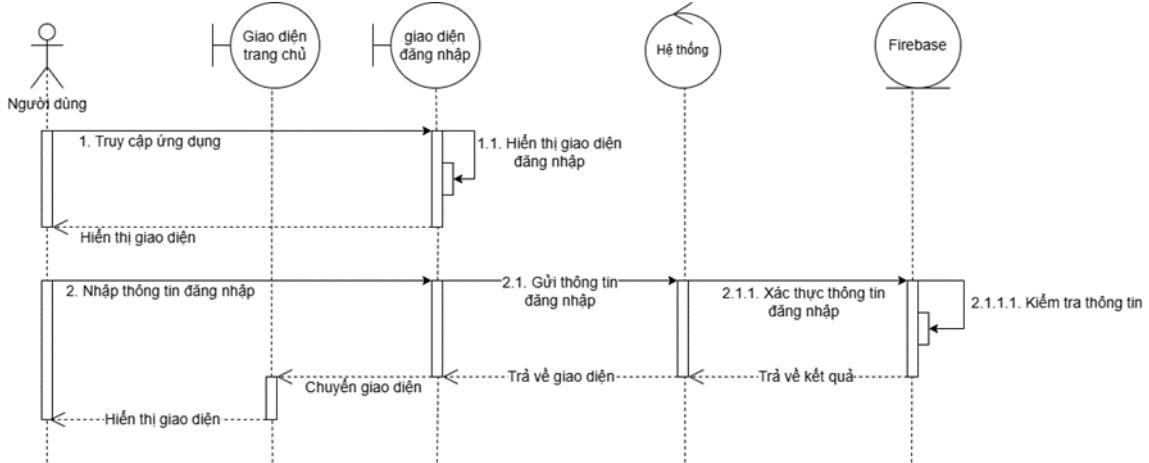


Hình 3-10: Sơ đồ hoạt động chức năng Đăng nhập

Hình 3-10 thể hiện sơ đồ hoạt động chức năng “**Đăng nhập**” mô tả luồng xử lý khi người dùng truy cập vào hệ thống và tiến hành đăng nhập bằng tài khoản đã đăng ký. Quá trình bắt đầu khi hệ thống hiển thị giao diện đăng nhập. Tại đây, người dùng sẽ nhập thông tin đăng nhập là email và mật khẩu, sau đó nhấn đăng nhập để gửi thông tin lên hệ thống xử lý.

Hệ thống sẽ tiếp nhận dữ liệu và tiến hành kiểm tra xác thực thông tin đăng nhập. Có 2 hướng xử lý:

- Nếu thông tin không hợp lệ (sai mật khẩu, sai email, email chưa đăng ký) thì hệ thống sẽ yêu cầu người dùng nhập lại thông tin.
- Nếu thông tin đăng nhập hợp lệ, hệ thống xác nhận quyền truy cập và chuyển hướng đến giao diện chính của ứng dụng. Kết thúc quy trình.



Hình 3-11: Sơ đồ tuần tự chức năng Đăng nhập

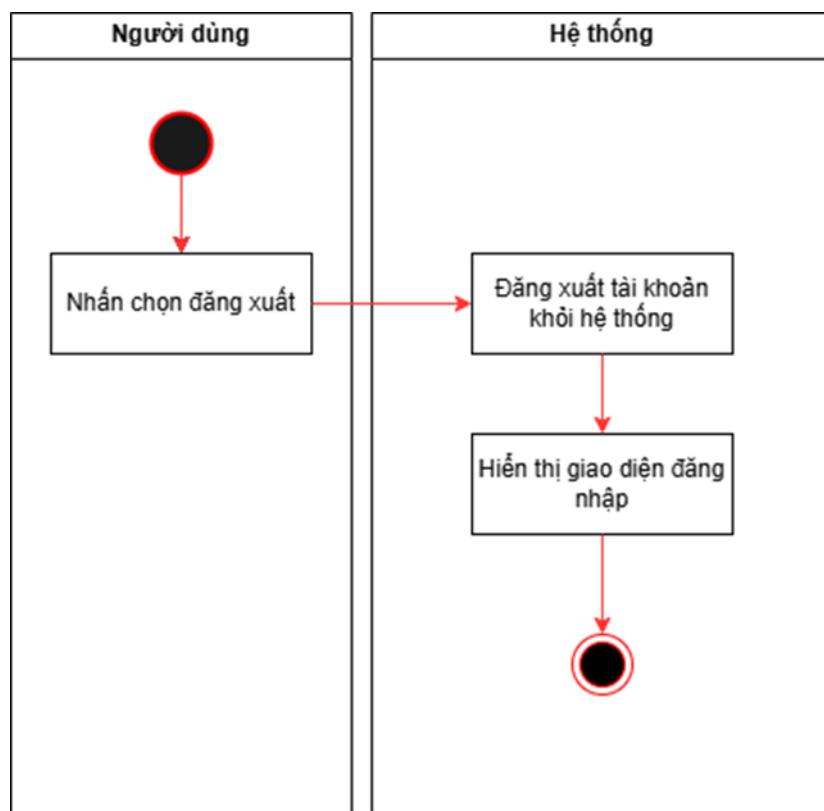
Sơ đồ tuần tự trên mô tả chi tiết trình tự tương tác giữa người dùng, giao diện ứng dụng, hệ thống xử lý và dịch vụ xác thực Firebase. Quá trình bắt đầu khi người dùng truy cập ứng dụng, thao tác này kích hoạt giao diện đăng nhập được hiển thị trên ứng dụng di động. Giao diện này gồm các trường nhập liệu là email và mật khẩu. Sau khi người dùng nhập thông tin đăng nhập, dữ liệu được truyền từ giao diện đến hệ thống xử lý.

Tiếp theo, hệ thống tiếp nhận thông tin và thực hiện bước xác thực thông qua dịch vụ Firebase. Tại đây, hệ thống gửi yêu cầu xác thực (bao gồm email và mật khẩu) đến Firebase để kiểm tra tính hợp lệ. Firebase tiếp nhận yêu cầu, kiểm tra thông tin và trả về kết quả tương ứng cho hệ thống.

Sau khi nhận được phản hồi từ Firebase, hệ thống phân tích kết quả:

- Nếu thông tin đăng nhập không hợp lệ, hệ thống sẽ gửi thông báo lỗi về giao diện yêu cầu người dùng nhập lại thông tin.
- Nếu thông tin đăng nhập hợp lệ, hệ thống cho người dùng đăng nhập và chuyển hướng đến giao diện chính của ứng dụng. Kết thúc quá trình.

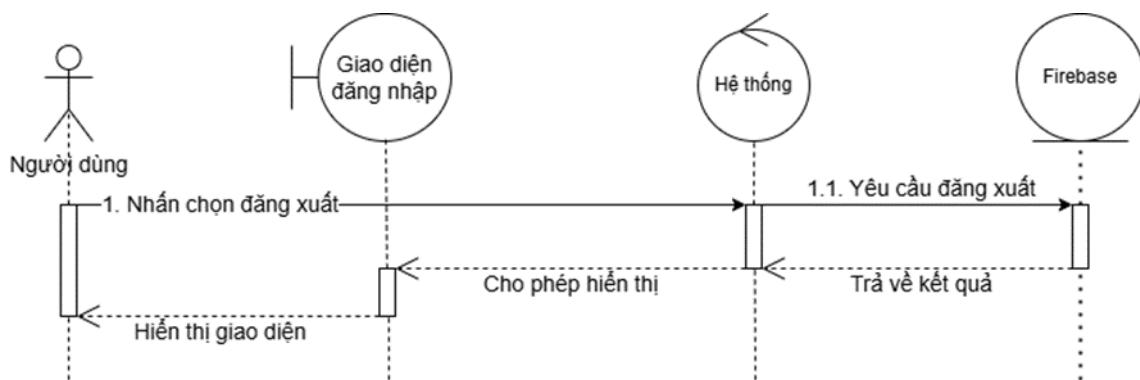
### 3.2.4.9. Chức năng đăng xuất



Hình 3-12: Sơ đồ hoạt động chức năng Đăng xuất

Hình 3-12 mô tả sơ đồ hoạt động chức năng “**Đăng xuất**” cho phép người dùng thoát khỏi phiên làm việc hiện tại, đảm bảo an toàn thông tin và tránh việc truy cập trái phép khi không còn sử dụng ứng dụng.

Quy trình bắt đầu khi người dùng nhấn chọn nút “Đăng xuất” trên giao diện ứng dụng. Hệ thống sau đó sẽ tiến hành đăng xuất tài khoản, đưa người dùng quay lại với giao diện đăng nhập. Kết thúc quy trình.

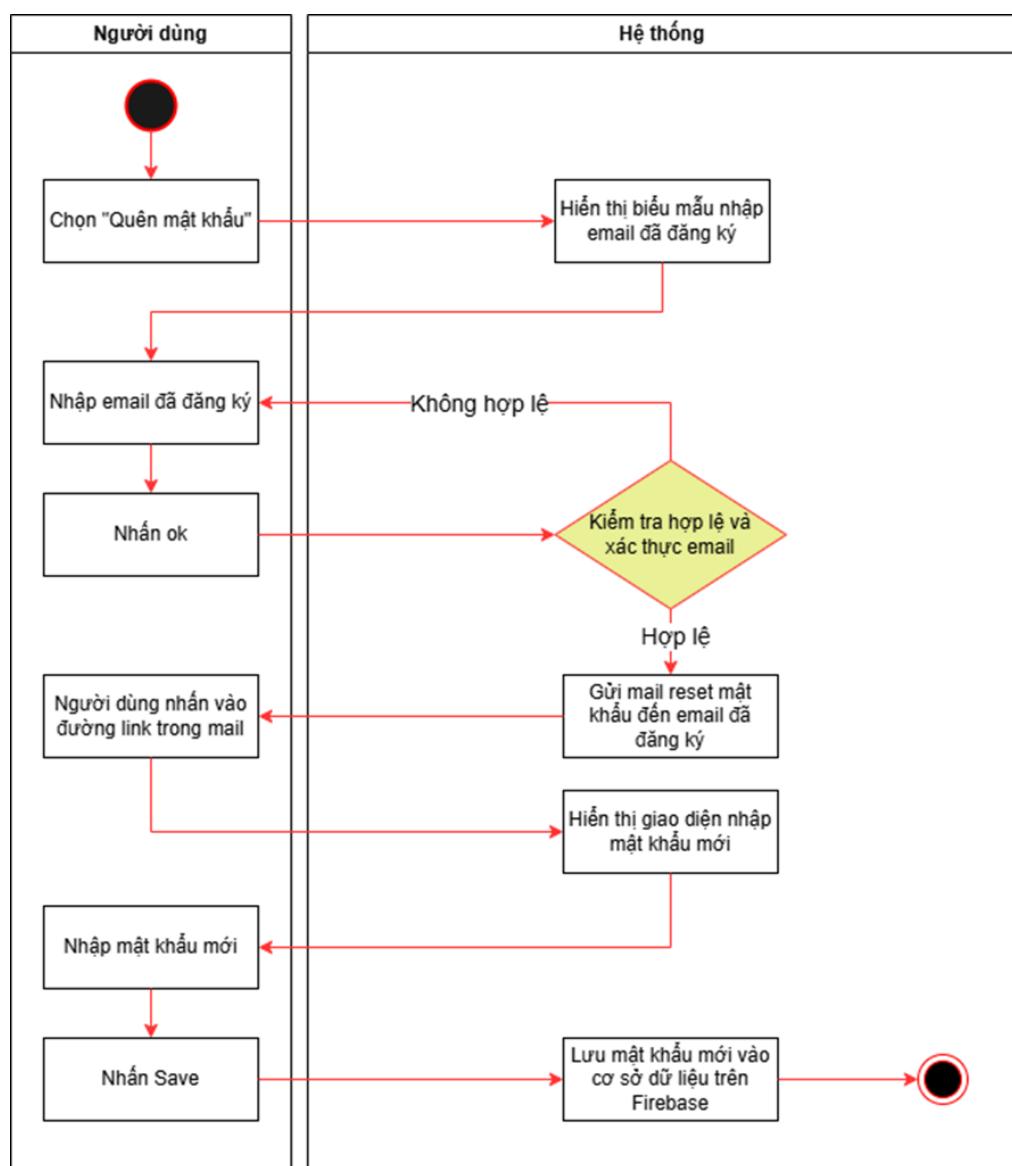


Hình 3-13: Sơ đồ tuần tự chức năng Đăng xuất

Hình 3-13 thể hiện biểu đồ tuần tự cho chức năng “**Đăng xuất**”. Quy trình bắt đầu khi người dùng chọn “Đăng xuất” từ giao diện ứng dụng. Hành động này sẽ gửi 1 yêu cầu đăng xuất đến hệ thống. Từ đó hệ thống tiếp tục gửi yêu cầu đăng xuất tới Firebase (sử dụng dịch vụ Firebase Authentication). Firebase tiếp nhận và xử lý yêu cầu đăng xuất bằng cách hủy phiên xác thực hiện tại, xóa token truy cập.

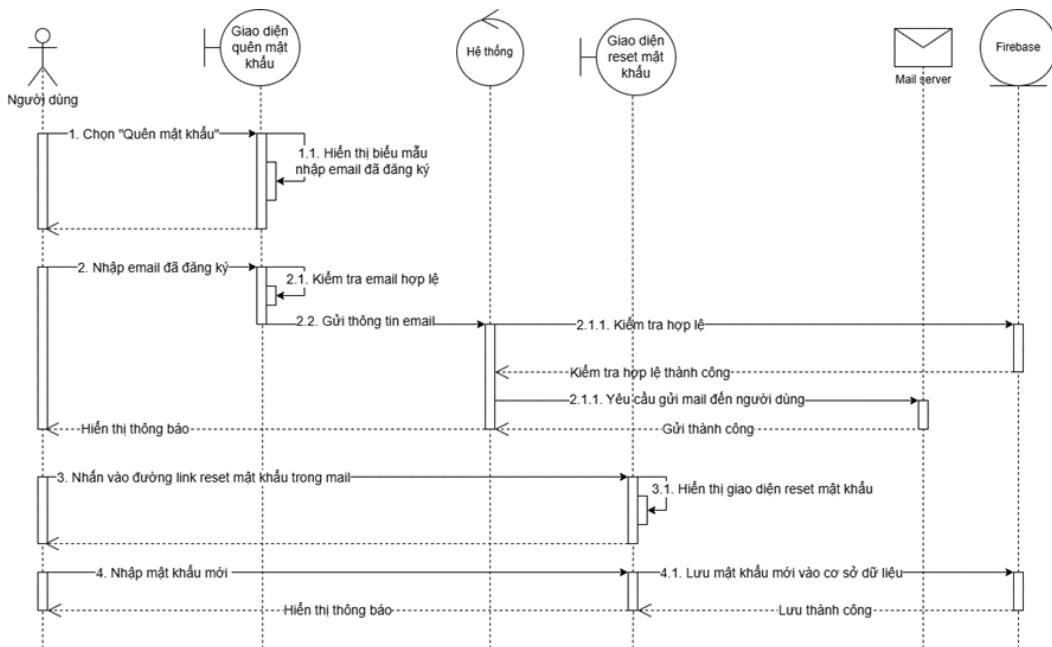
Khi Firebase xử lý thành công, hệ thống nhận được kết quả phản hồi, xác nhận rằng người dùng đã được đăng xuất hoàn toàn khỏi hệ thống. Sau đó chuyển sang chế độ giao diện chuyên về màn hình đăng nhập, sẵn sàng để người dùng đăng nhập lại bằng tài khoản khác. Kết thúc quy trình.

#### 3.2.4.10. Chức năng quên mật khẩu



Hình 3-14: Sơ đồ hoạt động chức năng Quên mật khẩu

Hình trên mô tả sơ đồ hoạt động chức năng “**Quên mật khẩu**”. Quy trình bắt đầu khi người dùng chọn “Quên mật khẩu” trong giao diện đăng nhập. Lúc này, hệ thống sẽ hiển thị biểu mẫu để người dùng nhập email đã đăng ký. Sau đó hệ thống sẽ xác thực email, nếu email không hợp lệ thì yêu cầu người dùng nhập lại, nếu email hợp lệ thì gửi mail đặt lại mật khẩu cho người dùng. Người dùng muốn đặt lại mật khẩu chỉ cần nhấn vào đường link trong mail, giao diện đặt lại mật khẩu mới sẽ hiện ra. Người dùng tiến hành nhập mật khẩu mới rồi lưu để tiến hành lưu mật khẩu mới vào cơ sở dữ liệu trên Firebase. Kết thúc quy trình.



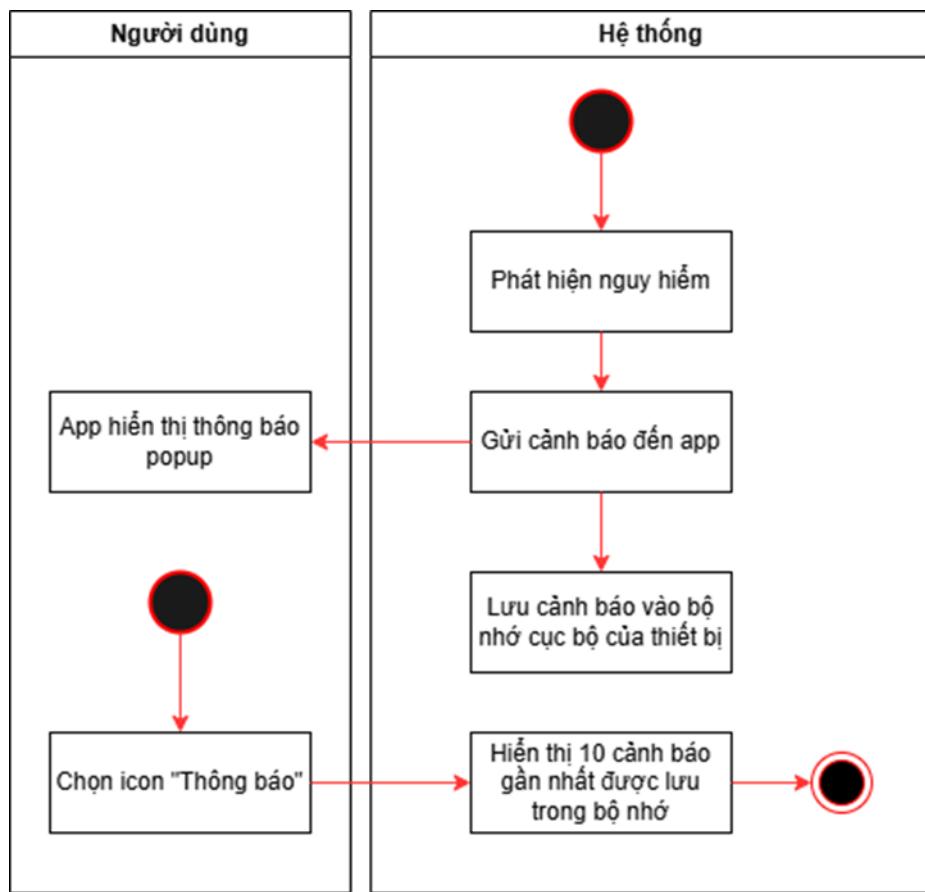
Hình 3-15: Sơ đồ tuần tự chức năng Quên mật khẩu

Hình 3-15 thể hiện sơ đồ tuần tự cho chức năng “**Quên mật khẩu**”. Quy trình bắt đầu khi người dùng chọn chức năng “Quên mật khẩu”. Giao diện ứng dụng phản hồi bằng cách hiển thị biểu mẫu yêu cầu nhập địa chỉ email đã đăng ký. Tại đây người dùng nhập địa chỉ email và gửi yêu cầu đặt lại mật khẩu.

Sau đó hệ thống tiếp nhận email và gửi lên Firebase để kiểm tra hợp lệ. Nếu email hợp lệ, hệ thống tiếp tục gửi yêu cầu đến Mail Server để gửi liên kết khôi phục tới người dùng. Sau đó giao diện hiển thị thông báo người dùng kiểm tra mail.

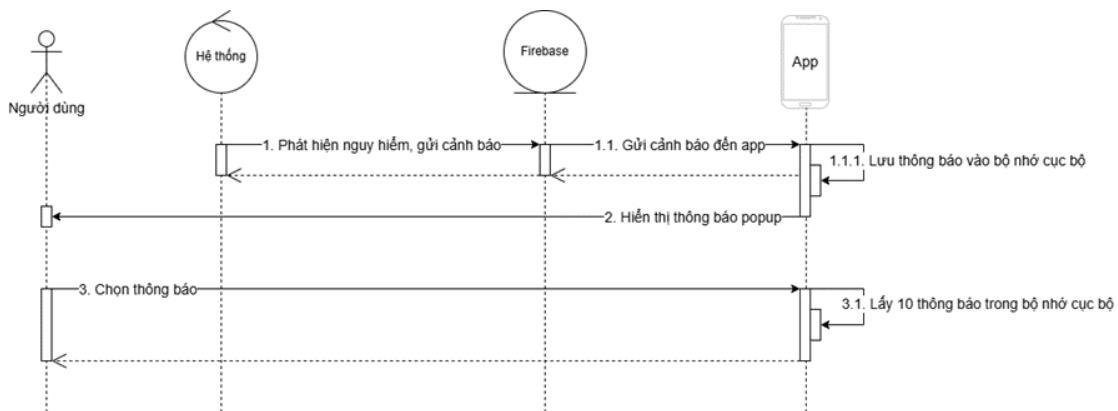
Tiếp theo, người dùng nhấn liên kết khôi phục trong email, giao diện đặt lại mật khẩu sẽ hiện ra. Người dùng nhập mật khẩu mới và nhấn “Save” để lưu vào cơ sở dữ liệu trên Firebase. Sau khi Firebase lưu thành công thì trả về kết quả để hệ thống hiển thị thông báo cho người dùng. Kết thúc quy trình.

### 3.2.4.11. Chức năng thông báo



Hình 3-16: Sơ đồ hoạt động chức năng Thông báo

Hình 3-16 thể hiện biểu đồ hoạt động cho chức năng “Thông báo”. Khi hệ thống phát hiện nguy hiểm thì sẽ gửi thông báo đến ứng dụng cho người dùng. Ứng dụng sẽ hiển thị thông báo popup để người dùng có thể phát hiện được ngay cả khi không sử dụng. Đồng thời những cảnh báo này cũng sẽ được lưu lại vào bộ nhớ cục bộ của thiết bị. Khi người dùng chọn chức năng “Thông báo” trong giao diện ứng dụng thì những thông báo đã lưu sẽ được lấy ra để hiển thị cho người dùng (tối đa 10 thông báo). Kết thúc quy trình.



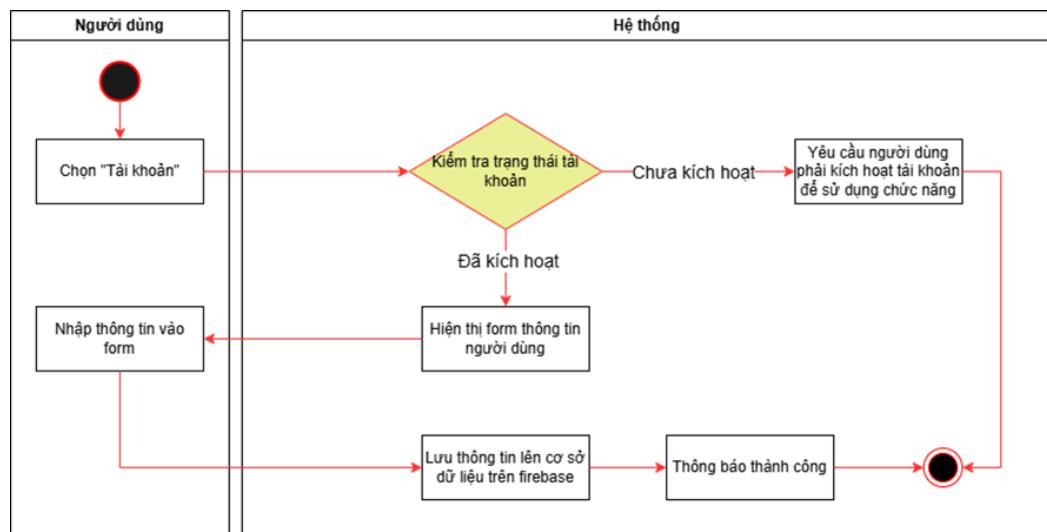
Hình 3-17: Sơ đồ tuần tự chức năng Thông báo

Hình 3-17 mô tả sơ đồ tuần tự cho chức năng “**Thông báo**”. Quy trình bắt đầu khi hệ thống phát hiện nguy hiểm (có khói hoặc lửa bùng lên), ngay lập tức, hệ thống sẽ gửi yêu cầu tạo cảnh báo đến Firebase Cloud Messaging (FCM) – đây là dịch vụ trung gian để phân phối thông báo đến thiết bị người dùng.

Ngay khi ứng dụng nhận được thông báo sẽ hiển thị popup lên màn hình, giúp cảnh báo kịp thời và không cần người dùng chủ động thao tác.

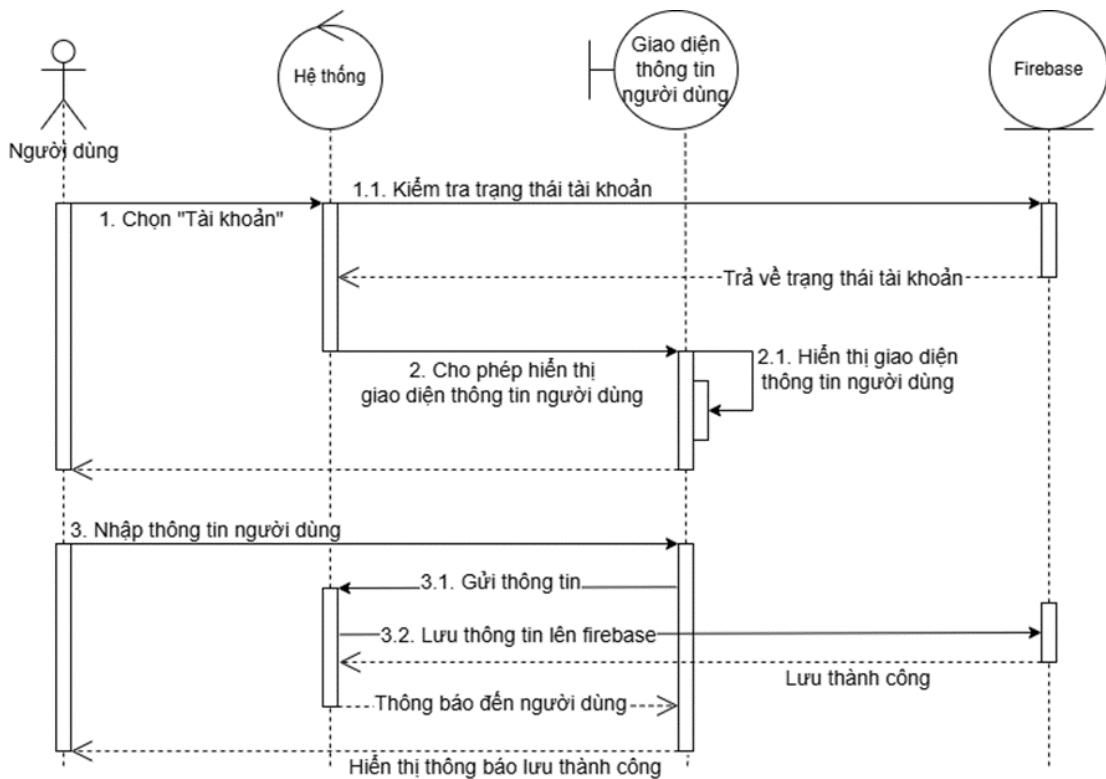
Sau này, khi người dùng muốn xem lại các thông báo đã lưu, họ có thể nhấp vào biểu tượng thông báo trong giao diện ứng dụng. Hành động này sẽ gửi yêu cầu đến chính ứng dụng để truy vấn các thông báo gần nhất đã lưu trong bộ nhớ cục bộ (tối đa 10 thông báo), rồi sau đó hiển thị cho người dùng. Kết thúc quy trình.

#### 3.2.4.12. Chức năng quản lý thông tin tài khoản



Hình 3-18: Sơ đồ hoạt động chức năng Quản lý thông tin tài khoản

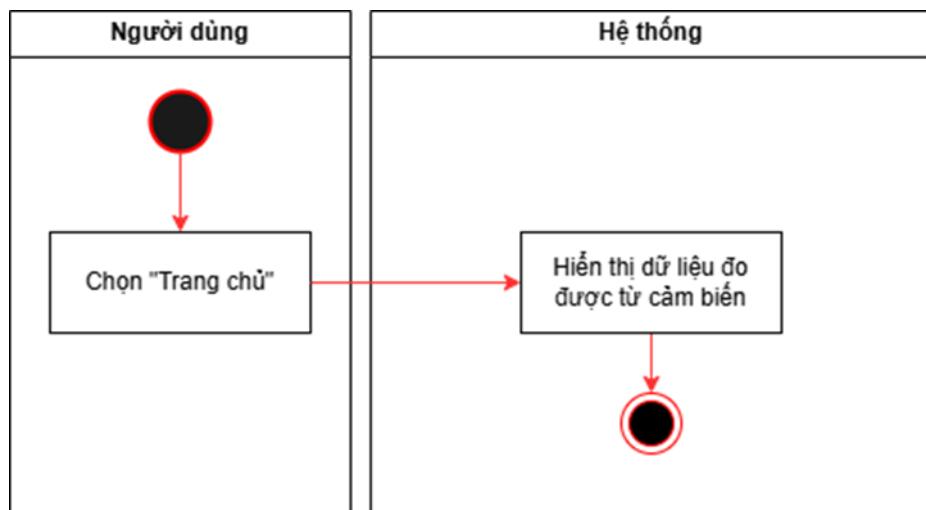
Hình 3-18 thể hiện biểu đồ hoạt động cho chức năng “**Quản lý thông tin tài khoản**”. Quy trình bắt đầu khi người dùng chọn “Tài khoản” trong giao diện ứng dụng. Lúc này hệ thống sẽ kiểm tra trạng thái tài khoản, nếu tài khoản chưa kích hoạt thì hiển thị thông báo yêu cầu người dùng kích hoạt tài khoản để sử dụng chức năng, còn nếu tài khoản đã kích hoạt thì hiển thị giao diện thông tin người dùng. Sau đó người dùng có thể nhập thông tin vào form thông tin rồi nhấn cập nhật. Dữ liệu sẽ được gửi Firebase để lưu vào cơ sở dữ liệu. Sau khi Firebase lưu thành công thì trả về kết quả để thông báo đến cho người dùng. Kết thúc quy trình.



Hình 3-19: Sơ đồ tuần tự chức năng Quản lý thông tin tài khoản

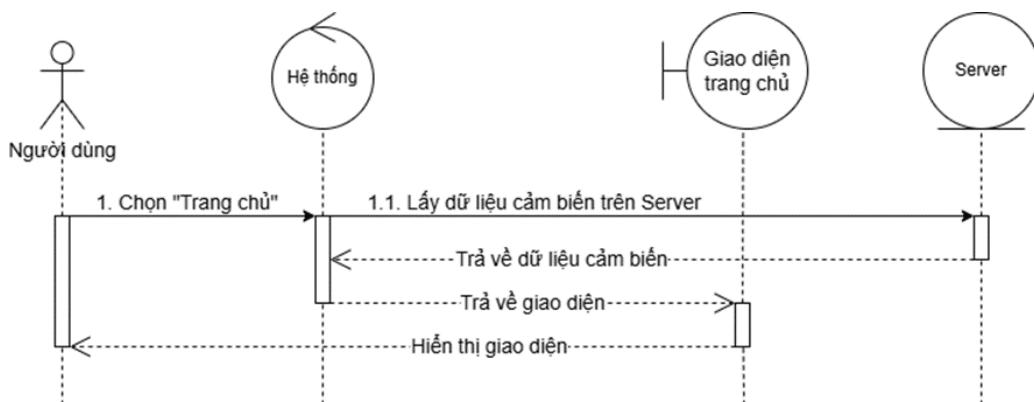
Hình 3-19 thể hiện sơ đồ tuần tự cho chức năng “**Quản lý thông tin tài khoản**”. Quy trình bắt đầu khi người dùng chọn “Tài khoản” trên giao diện ứng dụng. Yêu cầu này sẽ được chuyển đến hệ thống, tại đây sẽ kiểm tra trạng thái tài khoản người dùng (đã kích hoạt chưa) bằng cách gửi yêu cầu truy xuất lên Firebase. Firebase sau khi kiểm tra xong sẽ trả về kết quả, nếu tài khoản đã được kích hoạt thì giao diện ứng dụng sẽ được phép hiển thị giao diện quản lý thông tin cá nhân. Người dùng lúc này có thể nhập hoặc chỉnh sửa thông tin. Khi hoàn tất và ấn nút “Cập nhật”, thông tin sẽ được gửi cho hệ thống, hệ thống tiếp tục gửi yêu cầu cập nhật thông tin lên Firebase để lưu vào cơ sở dữ liệu. Firebase phản hồi kết quả lưu thành công, hệ thống thông báo đến người dùng qua giao diện. Kết thúc quy trình.

### 3.2.4.13. Chức năng xem dữ liệu đo được từ cảm biến



Hình 3-20: Sơ đồ hoạt động chức năng xem dữ liệu đo được từ cảm biến

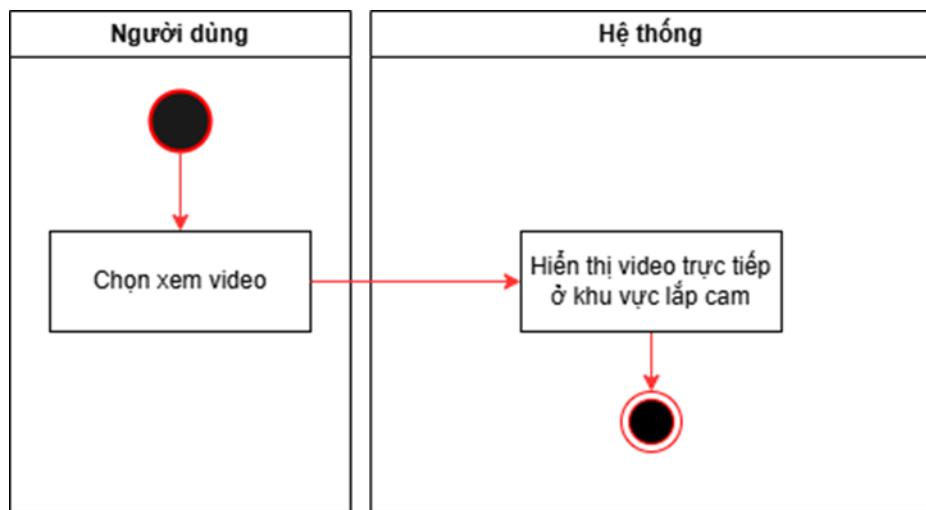
Hình 3-20 mô tả sơ đồ hoạt động của chức năng “**Xem dữ liệu đo được từ cảm biến**”. Dữ liệu đo được này sẽ mặc định hiển thị ở “Trang chủ”, người dùng chỉ cần ở giao diện “Trang chủ” là sẽ xem được dữ liệu đo được từ ba cảm biến nhiệt độ, nồng độ khí CO và nồng độ khói.



Hình 3-21: Sơ đồ tuần tự chức năng Xem dữ liệu đo được từ cảm biến

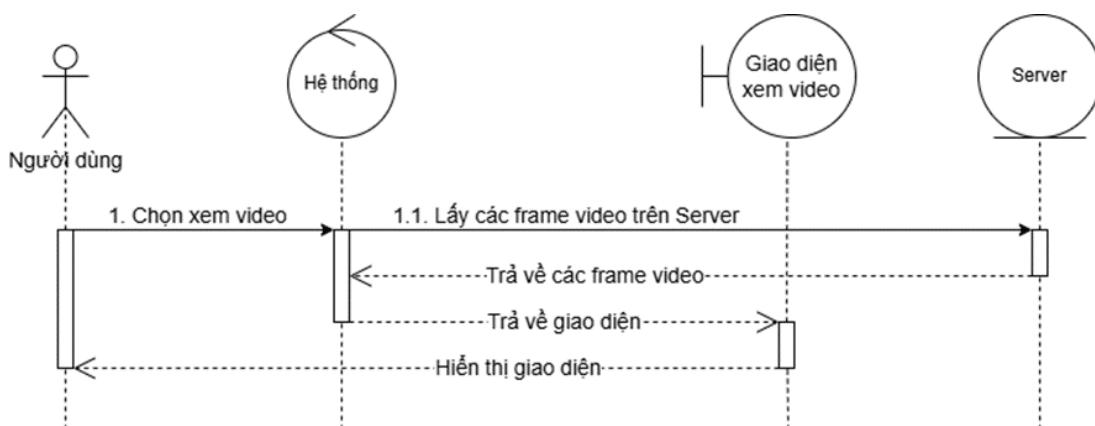
Hình 3-21 mô tả sơ đồ tuần tự của chức năng “**Xem dữ liệu đo được từ cảm biến**”. Quy trình bắt đầu khi người dùng chọn mục “Trang chủ” trong giao diện ứng dụng hoặc khi vừa mới đăng nhập. Hành động này gửi yêu cầu đến hệ thống để thực hiện truy xuất dữ liệu đo được từ cảm biến trên Server. Nhóm em sử dụng 1 Server trung gian để lưu trữ dữ liệu từ cảm biến theo thời gian thực. Hệ thống sẽ gửi yêu cầu đến Server này, sau khi tiếp nhận yêu cầu Server sẽ xử lý và trả về dữ liệu cảm biến dưới dạng các chỉ số đo được gần nhất. Hệ thống nhận dữ liệu rồi trả dữ liệu về cho giao diện “Trang chủ”. Giao diện sẽ tiếp nhận rồi hiển thị dữ liệu cho người dùng dưới dạng con số. Kết thúc quy trình.

### 3.2.4.14. Chức năng xem video trực tiếp



Hình 3-22: Sơ đồ hoạt động chức năng Xem video trực tiếp

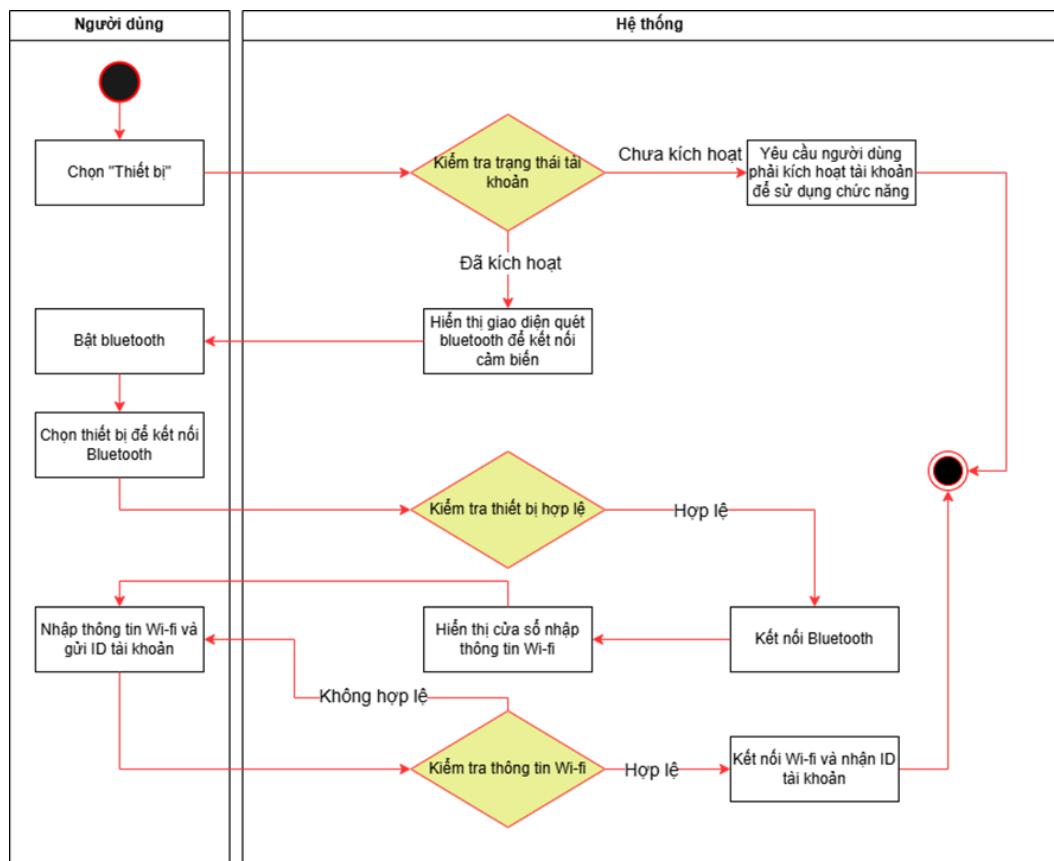
Hình 3-22 mô tả sơ đồ hoạt động của chức năng “Xem video trực tiếp”. Khi người dùng chọn chức năng “Xem video” trên giao diện “Trang chủ” thì giao diện “Xem video” sẽ hiện ra, tại đây sẽ hiển thị video trực tiếp ở khu vực lắp cam. Người dùng chỉ có thể xem, không thể tua video.



Hình 3-23: Sơ đồ tuần tự chức năng Xem video trực tiếp

Sơ đồ tuần tự của chức năng “Xem video trực tiếp” được thể hiện trong hình 3-23. Quy trình bắt đầu khi người dùng chọn mục “Xem video” trên giao diện ứng dụng. Lúc này ứng dụng sẽ gửi yêu cầu đến hệ thống, hệ thống tiếp tục gửi yêu cầu đến Server để yêu cầu lấy các frame video, Server sẽ phản hồi lại bằng cách trả về các chuỗi frame video (theo chuẩn HLS). Khi đã có các frame video rồi hệ thống sẽ trả dữ liệu về giao diện người dùng, giao diện sẽ hiển thị các frame video liên tiếp thông qua player tích hợp.

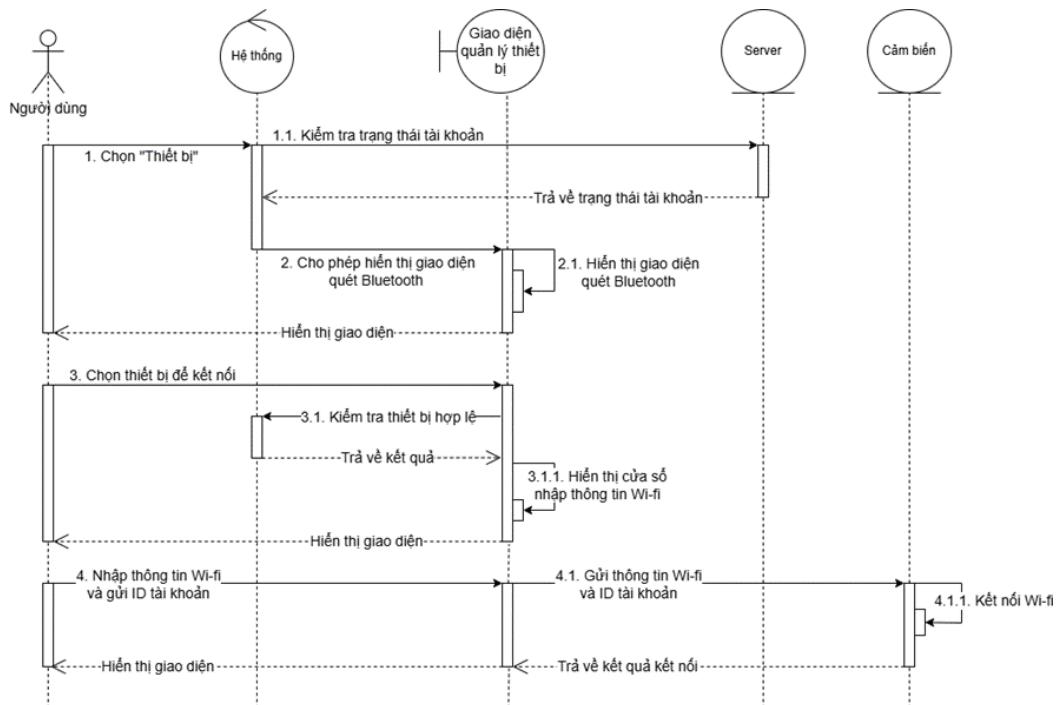
### 3.2.4.15. Chức năng kết nối với cảm biến



Hình 3-24: Sơ đồ hoạt động chức năng Kết nối với cảm biến

Chức năng “Kết nối với cảm biến” này chính là một bước để thêm một cảm biến vào hệ thống để sử dụng. Đầu tiên, khi người dùng chọn mục “Thiết bị” thì hệ thống sẽ thực hiện kiểm tra tài khoản, bắt buộc tài khoản phải kích hoạt mới có thể sử dụng được chức năng này. Nếu tài khoản đã kích hoạt, giao diện quét Bluetooth để kết nối cảm biến sẽ hiện ra. Người dùng bật Bluetooth của thiết bị lên và thực hiện quét, khi thấy tên của cảm biến trong danh sách thiết bị quét được thì người dùng nhấn vào thiết bị đó để thực hiện ghép nối. Nếu đúng là thiết bị cảm biến tương thích với hệ thống thì sẽ thực hiện ghép nối Bluetooth, sau khi ghép nối Bluetooth thành công thì một cửa sổ để nhập thông tin Wi-fi sẽ hiện ra (Yêu cầu người dùng phải nhập Wi-fi thủ công cho từng cảm biến vì cảm biến cần có kết nối internet để gửi dữ liệu trên Server, mỗi vị trí lắp đặt cảm biến có thể sử dụng Wi-fi khác nhau vậy nên không thể fix cứng ssid và password trong phần code mà phải để người dùng thực hiện kết nối thủ công qua ứng dụng di động). Người dùng sẽ nhập thông tin Wi-fi rồi nhấn gửi, cùng với thông tin Wi-fi thì ID của tài khoản cũng sẽ được tự động gửi đi mà không cần người dùng phải thao tác (ID tài khoản này để cảm biến biết mình thuộc sở hữu của người sử dụng nào, ví dụ cảm biến này là của người sử dụng A thì cảm biến đó phải gửi dữ liệu vào cơ sở dữ liệu của A chứ không phải gửi vào của B). Phần thông tin này sẽ được gửi đến cho cảm

biến thông qua Bluetooth, sau khi nhận được, cảm biến sẽ thực hiện kết nối Wi-fi và thông báo kết quả về cho ứng dụng thông qua Bluetooth. Kết thúc quy trình.



Hình 3-25: Sơ đồ tuần tự chức năng Kết nối với cảm biến

Hình 3-25 mô tả sơ đồ tuần tự của chức năng “**Kết nối với cảm biến**”. Bắt đầu quy trình, người dùng chọn mục “Thiết bị” trên giao diện ứng dụng. Hệ thống tiếp nhận yêu cầu và thực hiện kiểm tra trạng thái tài khoản bằng cách truy vấn lên Firebase. Sau khi xác nhận tài khoản đã kích hoạt, hệ thống cho phép người dùng truy cập vào chức năng quản lý thiết bị. Giao diện quét Bluetooth các thiết bị ở gần sẽ hiện ra, khi người dùng bật Bluetooth và bật chức năng quét thì danh sách các thiết bị quét được sẽ hiển thị, người dùng muốn kết nối với thiết bị nào thì nhấn vào thiết bị đó. Nếu thiết bị đó tương thích với hệ thống thì tiến hành ghép đôi, sau khi ghép đôi thành công thì cửa sổ để nhập thông tin Wi-fi sẽ hiện ra. Sau đó thông tin Wi-fi và ID tài khoản sẽ được gửi đến thiết bị cảm biến thông qua Bluetooth. Cảm biến tiếp nhận thông tin và tiến hành kết nối Wi-fi bằng các thông tin đó, đồng thời gửi kết quả lại cho hệ thống cũng qua Bluetooth. Cuối cùng giao diện sẽ hiển thị thông báo kết quả kết nối, “Thành công” hoặc “Thất bại”. Kết thúc quy trình.

### 3.2.5 Thiết kế ứng dụng di động

Để hỗ trợ người dùng giám sát và tương tác với hệ thống cảnh báo hỏa hoạn theo thời gian thực, nhóm em đã thiết kế một ứng dụng di động được phát triển trên nền tảng Flutter. Ứng dụng này đóng vai trò là giao diện chính giữa người dùng và hệ thống, cho phép nhận thông tin cảnh báo, theo dõi dữ liệu cảm biến, xem video tại khu vực lắp đặt và điều khiển thiết bị quạt hút khói khi cần thiết.

Các chức năng chính của ứng dụng:

- **Đăng nhập hệ thống:** Người dùng sử dụng tài khoản đã đăng ký để đăng nhập vào hệ thống, đảm bảo bảo mật và cá nhân hóa trải nghiệm. Hệ thống chỉ cho phép truy cập thông tin liên quan đến tài khoản đã xác thực.
- **Hiển thị dữ liệu cảm biến:** Ứng dụng lấy dữ liệu từ Server và hiển thị theo thời gian thực các giá trị đo được là nồng độ khí CO, nhiệt độ, độ ẩm và nồng độ khói. Các dữ liệu được thể hiện qua đồng hồ đo trực quan, dễ quan sát.
- **Xem video trực tiếp (camera stream):** Ứng dụng cho phép người dùng xem luồng video thời gian thực từ camera gắn trên Raspberry Pi 4. Người dùng có thể theo dõi tình trạng khu vực lắp camera với thông tin của mình để có biện pháp xử lý khi hỏa hoạn xảy ra.
- **Nhận cảnh báo thời gian thực:** Khi hệ thống phát hiện có nguy cơ cháy (khói hoặc lửa), ứng dụng sẽ ngay lập tức nhận được cảnh báo từ Server và hiển thị cho người dùng. Cảnh báo gồm thời gian, thiết bị phát hiện cảnh báo.

Ứng dụng Flutter đóng vai trò như trạm điều khiển cá nhân di động, cho phép người dùng quan sát, nhận cảnh báo và can thiệp từ xa. Nhờ đó, hệ thống cảnh báo cháy không chỉ dừng lại ở khả năng phát hiện mà còn hỗ trợ người dùng ra quyết định kịp thời và chính xác.

Ứng dụng được xây dựng hướng tới giao diện trực quan, thân thiện, đảm bảo sử dụng dễ dàng ngay cả với người không có kiến thức chuyên môn về kỹ thuật. Các chức năng chính được chia thành từng màn hình rõ ràng, tối ưu trải nghiệm người dùng.

## CHƯƠNG 4. TRIỂN KHAI THỰC HIỆN

Chương này trình bày chi tiết quá trình triển khai hệ thống, bao gồm cả phần cứng và phần mềm. Nội dung bao gồm lựa chọn, kết nối các linh kiện, thiết kế mạch, xử lý dữ liệu và huấn luyện mô hình AI. Ngoài ra, chương cũng mô tả cách triển khai mô hình lên Raspberry Pi 4, cấu hình server truyền video và xây dựng ứng dụng di động. Mục tiêu là chuyển từ thiết kế sang triển khai thực tế, đảm bảo hệ thống hoạt động ổn định.

### 4.1 Triển khai thực hiện phần cứng

#### 4.1.1 Các linh kiện được sử dụng

##### 4.1.1.1 Máy tính Nhúng

Trong một hệ thống cảnh báo hỏa hoạn kết hợp AI, việc lựa chọn thiết bị xử lý trung tâm phù hợp đóng vai trò then chốt nhằm đảm bảo khả năng thu thập, xử lý và phản hồi thông tin từ các đầu vào trong thời gian thực. Máy tính nhúng chính là thành phần trung tâm chịu trách nhiệm thu thập dữ liệu từ các cảm biến như CO, nhiệt độ, khói..., thực hiện các thuật toán phát hiện cháy (nhận diện khói/lửa từ hình ảnh camera), điều khiển các thiết bị như quạt hút khói, đồng thời gửi cảnh báo đến người dùng thông qua server đến ứng dụng di động.

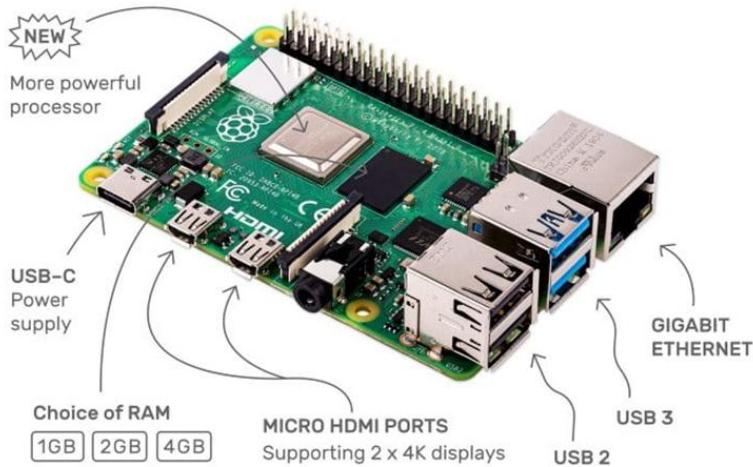
Lựa chọn máy tính nhúng cần đảm bảo sự cân bằng giữa hiệu năng xử lý và mức tiêu thụ điện năng, đồng thời hỗ trợ tốt cho việc kết nối với các loại cảm biến, camera, module mạng và các thiết bị điều khiển ngoại vi. Ngoài ra, khả năng mở rộng phần mềm, tính ổn định và độ tin cậy trong vận hành lâu dài cũng là những yếu tố quan trọng cần xem xét.

Trong mục này, thay vì chỉ đơn thuần liệt kê loại thiết bị mà hệ thống sử dụng, chúng em tiến hành đánh giá tổng quan một số dòng máy tính nhúng tiêu biểu hiện nay. Thông qua việc so sánh các ưu, nhược điểm của từng nền tảng, từ đó xác định được ứng viên phù hợp nhất với yêu cầu của đề tài.

Sau khi khảo sát, nhóm em đưa ra các lựa chọn cho vị trí xử lý trung tâm – máy tính nhúng như sau:

#### Raspberry Pi 4:

Raspberry Pi 4 Model B là một dòng máy tính nhúng phát triển bởi Raspberry Pi Foundation, có kích thước nhỏ gọn nhưng sở hữu khả năng xử lý mạnh mẽ, hỗ trợ nhiều giao tiếp ngoại vi, thích hợp cho các ứng dụng yêu cầu thu thập dữ liệu, xử lý hình ảnh, truyền phát video và điều khiển thiết bị từ xa. Thiết bị đặc biệt phù hợp với các hệ thống nhúng tích hợp trí tuệ nhân tạo (AI) như hệ thống cảnh báo hỏa hoạn thông minh trong đề tài.



Hình 4-1: Raspberry Pi 4

a) Thông số kỹ thuật:

- CPU: Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit, xung nhịp 1.5GHz.
- RAM: Tùy chọn 2GB, 4GB hoặc 8GB LPDDR4.
- GPU: Broadcom VideoCore VI, hỗ trợ giải mã video 4K H.265/H.264.
- Lưu trữ: microSD (hỗ trợ SDHC và SDXC), USB boot (qua ổ cứng ngoài hoặc SSD).
- Kết nối mạng:
  - Gigabit Ethernet.
  - Wi-Fi 802.11 b/g/n/ac (dual-band 2.4GHz/5.0GHz).
  - Bluetooth 5.0, BLE (Bluetooth Low Energy).
- Cổng giao tiếp:
  - 2 cổng micro-HDMI (hỗ trợ 2 màn hình, độ phân giải tối đa 4K).
  - 2 cổng USB 3.0, 2 cổng USB 2.0.
  - 1 cổng USB-C (nguồn 5V/3A).
  - 40 chân GPIO tiêu chuẩn.
  - Cổng CSI (kết nối camera), cổng DSI (kết nối màn hình cảm ứng).
  - 1 cổng âm thanh 4 cực kết hợp với video composite.
- Kích thước: 85.6mm x 56.5mm.
- Nguồn điện:
  - 5V DC qua cổng USB-C (dòng tối thiểu 3A).
  - 5V DC qua chân GPIO (dòng tối thiểu 3A).

- Hỗ trợ cấp nguồn qua Ethernet (PoE – cần thêm module PoE HAT).
  - Nhiệt độ hoạt động:
    - Từ 0°C đến 50°C (nhiệt độ môi trường).
  - Hệ điều hành hỗ trợ: Raspberry Pi OS, Ubuntu, Debian, và các bản phân phối Linux khác.
- b) Ưu điểm của Raspberry Pi 4:
- Hiệu năng xử lý tốt: Với CPU Cortex-A72 4 nhân và RAM lên đến 8GB, Raspberry Pi 4 đủ khả năng xử lý các mô hình AI nhẹ, xử lý video, truyền dữ liệu thời gian thực mà vẫn đảm bảo mượt mà.
  - Hỗ trợ đa dạng kết nối: Raspberry Pi 4 hỗ trợ nhiều cổng kết nối tiêu chuẩn và chân GPIO, dễ dàng tích hợp với các cảm biến, camera, relay điều khiển quạt... giúp phù hợp với hệ thống cảnh báo cháy phức tạp.
  - Khả năng truyền video mạnh: Có thể kết hợp với camera CSI để quay và truyền video trực tiếp lên server bằng giao thức RTMP hoặc HLS, hỗ trợ độ phân giải lên đến 1080p hoặc 4K.
  - Hệ điều hành mở, dễ phát triển: Có thể cài đặt nhiều hệ điều hành mã nguồn mở như Raspberry Pi OS, Ubuntu, hỗ trợ đầy đủ thư viện AI như OpenCV, PyTorch, TensorFlow Lite.
  - Chi phí hợp lý: So với các thiết bị nhúng chuyên dụng (như NVIDIA Jetson), Raspberry Pi 4 có mức giá rẻ hơn nhiều nhưng vẫn đáp ứng tốt phần lớn nhu cầu.
  - Cộng đồng hỗ trợ lớn: Raspberry Pi có cộng đồng lập trình viên toàn cầu, dễ dàng tìm tài liệu, diễn đàn hỗ trợ kỹ thuật.
- c) Nhuược điểm của Raspberry Pi 4:
- Không có bộ xử lý GPU chuyên dụng cho AI: So với các nền tảng như Jetson Nano, khả năng xử lý các mô hình AI lớn hoặc real-time trên Raspberry Pi 4 còn hạn chế.
  - Khả năng tản nhiệt thấp: Khi hoạt động liên tục ở tải cao (ví dụ chạy AI + stream video), Raspberry Pi 4 dễ bị nóng và cần trang bị thêm tản nhiệt hoặc quạt làm mát.
  - Không có bộ nhớ eMMC tích hợp: Raspberry Pi lưu trữ bằng thẻ microSD nên tốc độ truy xuất và độ bền thấp hơn so với bộ nhớ eMMC hoặc SSD tích hợp.

- Không phù hợp cho xử lý song song quy mô lớn: Thiết kế hướng tới các ứng dụng nhúng đơn lẻ hoặc quy mô vừa – không phù hợp với các hệ thống AI lớn, đa camera hoặc tính toán nặng.

### NVIDIA Jetson Nano Developer Kit:

Đây là một nền tảng máy tính nhúng mạnh mẽ, được thiết kế để triển khai các ứng dụng trí tuệ nhân tạo (AI) tại biên (edge AI). Với khả năng xử lý song song và hiệu suất cao, Jetson Nano phù hợp cho các ứng dụng như nhận dạng hình ảnh, phân loại đối tượng, xử lý ngôn ngữ tự nhiên và các ứng dụng AI khác.



Hình 4-2: NVIDIA Jetson Nano Developer Kit

#### a) Thông số kỹ thuật:

- Bộ xử lý (CPU): Quad-core ARM Cortex-A57 MPCore @ 1.43 GHz.
- Bộ xử lý đồ họa (GPU): 128 nhân NVIDIA Maxwell, hiệu suất 472 GFLOPS.
- Bộ nhớ RAM: 4 GB 64-bit LPDDR4 @ 25.6 GB/s.
- Lưu trữ: Khe cắm thẻ microSD (hỗ trợ SDHC và SDXC).
- Kết nối mạng: Gigabit Ethernet.
- Kết nối không dây: Hỗ trợ qua khe cắm M.2 Key E (cần thêm module Wi-Fi/Bluetooth).
- Cổng giao tiếp:
  - 4 x USB 3.0.
  - 1 x USB 2.0 Micro-B.
  - 1 x HDMI 2.0.

- 1 x DisplayPort 1.4.
- 40 chân GPIO, I<sup>2</sup>C, I<sup>2</sup>S, SPI, UART.
- 2 x MIPI CSI-2 camera connectors.
- Hiển thị: Hỗ trợ độ phân giải lên đến 4K@60fps.
- Mã hóa/giải mã video:
  - Mã hóa: 4K@30fps (H.265), 4x1080p@30fps, 9x720p@30fps
  - Giải mã: 4K@60fps (H.265), 2x4K@30fps, 8x1080p@30fps, 18x720p@30fps.
- Nguồn điện: 5VDC qua cổng Micro-USB hoặc jack DC barrel (công suất từ 5W đến 10W).
- Kích thước: 100 mm x 80 mm x 29 mm.
- Hệ điều hành hỗ trợ: Linux for Tegra (L4T) với JetPack SDK.

b) Ưu điểm:

- Hiệu suất AI cao: Với GPU 128 nhân Maxwell, Jetson Nano có khả năng xử lý các mô hình AI phức tạp với hiệu suất lên đến 472 GFLOPS.
- Hỗ trợ đa dạng giao tiếp: Cung cấp nhiều cổng giao tiếp như USB 3.0, HDMI, DisplayPort, GPIO, giúp dễ dàng kết nối với các thiết bị ngoại vi và cảm biến.
- Hỗ trợ nhiều camera: Với 2 cổng MIPI CSI-2, Jetson Nano có thể kết nối đồng thời với hai camera, phù hợp cho các ứng dụng thị giác máy tính.
- Cộng đồng phát triển mạnh mẽ: Được hỗ trợ bởi cộng đồng lớn và tài liệu phong phú từ NVIDIA, giúp dễ dàng trong việc phát triển và triển khai ứng dụng.
- Tiết kiệm năng lượng: Tiêu thụ điện năng thấp (5W đến 10W), phù hợp cho các ứng dụng nhúng và di động.

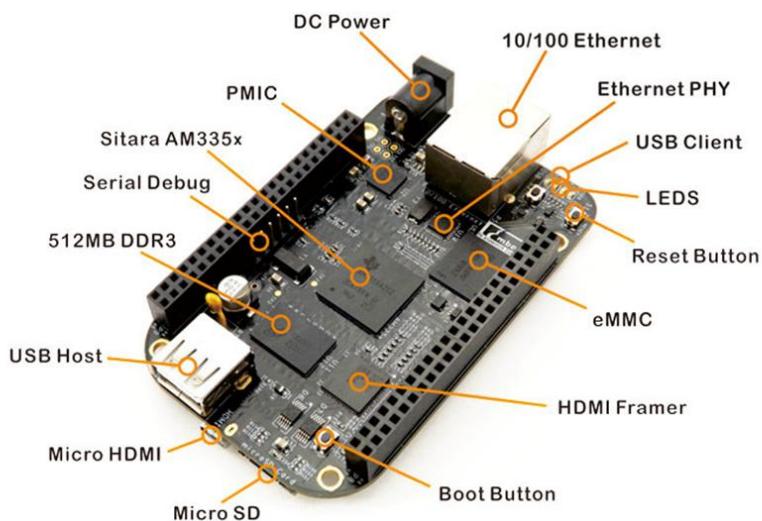
c) Nhược điểm:

- Không có kết nối không dây tích hợp: Cần thêm module Wi-Fi/Bluetooth qua khe cắm M.2 Key E để hỗ trợ kết nối không dây.
- Lưu trữ phụ thuộc vào thẻ microSD: Việc sử dụng thẻ microSD có thể ảnh hưởng đến tốc độ và độ bền của hệ thống so với các giải pháp lưu trữ eMMC hoặc SSD.
- Hạn chế về bộ nhớ RAM: Với 4 GB RAM, Jetson Nano có thể gặp khó khăn khi xử lý các mô hình AI lớn hoặc đa tác vụ.

- Không có bộ xử lý tensor chuyên dụng: Khả năng tăng tốc AI không cao bằng các dòng sản phẩm cao cấp hơn như Jetson Xavier NX hoặc Jetson AGX Xavier.
- Yêu cầu nguồn điện ổn định: Cần nguồn điện ổn định và đủ công suất để đảm bảo hoạt động liên tục, đặc biệt khi sử dụng nhiều thiết bị ngoại vi.

### **BeagleBone Black - Rev C:**

BeagleBone Black là một máy tính nhúng nhỏ gọn, được thiết kế bởi BeagleBoard.org Foundation. Thiết bị nổi bật với khả năng điều khiển thời gian thực mạnh mẽ, hỗ trợ nhiều giao tiếp phần cứng và là lựa chọn phổ biến trong lĩnh vực tự động hóa công nghiệp, điều khiển nhúng và các ứng dụng nhạy với thời gian.



Hình 4-3: BeagleBone Black – Rev C

#### a) Thông số kỹ thuật:

- Bộ xử lý (Processor):
  - TI Sitara AM335x ARM Cortex-A8, xung nhịp 1GHz (~2000 MIPS).
  - GPU tích hợp: PowerVR SGX530 (hỗ trợ OpenGL ES 2.0).
  - 2 đơn vị xử lý thời gian thực (PRU – Programmable Real-Time Unit Subsystem).
- Bộ nhớ:
  - RAM: 512MB DDR3L SDRAM (800MHz).
  - Bộ nhớ trong: 4GB eMMC Flash (Rev C).
  - Hỗ trợ lưu trữ mở rộng qua khe cắm thẻ microS.
- Nguồn và quản lý năng lượng:
  - Nguồn cấp: 5V qua cổng miniUSB hoặc DC Jack.

- Hỗ trợ cấp nguồn ngoài qua header mở rộng.
- IC quản lý điện năng: TPS65217C PMIC + LDO.
- Giao tiếp:
  - 1 x cổng USB 2.0 Host (Type-A).
  - 1 x cổng miniUSB Client.
  - 1 x cổng Ethernet 10/100Mbps (RJ45).
  - 1 x UART0 (6 chân 3.3V TTL).
  - 2 x Header 46 chân (GPIO, I2C, SPI, UART, PWM, ADC...).
- Hỗ trợ video và âm thanh:
  - 1 x cổng HDMI (loại D) hỗ trợ hình ảnh và âm thanh stereo.
  - Hỗ trợ giao tiếp LCD độc lập.
- Tính năng người dùng:
  - Nút nguồn, nút reset, nút boot.
  - 4 đèn LED người dùng lập trình được.
  - Đèn LED hiển thị nguồn.
- Kích thước: 3.4" x 2.1" (tương đương ~86.4mm x 53.3mm).
- Công suất tiêu thụ: ~5V – 0.35A (~1.75W).
- Hệ điều hành hỗ trợ:
  - Debian (pre-installed).
  - Ubuntu.
  - Android.
  - Cloud9 IDE chạy trên Node.js với thư viện BoneScript.

b) Ưu điểm:

- Có bộ nhớ trong eMMC 4GB, không cần thẻ nhớ để khởi động hệ điều hành như Raspberry Pi.
- Hỗ trợ thời gian thực tốt: Nhờ 2 PRU tích hợp trong SoC, BeagleBone Black rất phù hợp cho các ứng dụng cần phản hồi nhanh (robot, điều khiển động cơ...).
- Đa dạng giao tiếp phần cứng: Có hàng chục GPIO, SPI, I2C, UART, ADC giúp dễ dàng tích hợp với các cảm biến và thiết bị ngoại vi.
- Tương thích tốt với Linux: Được cài sẵn Debian, hỗ trợ lập trình Python, C/C++, Node.js...
- Phù hợp với ứng dụng công nghiệp: Bên bì, hoạt động ổn định trong môi trường khắc nghiệt.

c) Nhược điểm:

- Hiệu năng xử lý thấp hơn so với Raspberry Pi 4 hoặc Jetson Nano (chỉ 1 nhân Cortex-A8, RAM 512MB).

- Không có kết nối Wi-Fi hoặc Bluetooth tích hợp, phải gắn thêm module qua USB.
- Không tối ưu cho AI hoặc xử lý hình ảnh do thiếu GPU mạnh và không hỗ trợ nhiều thư viện AI hiện đại.

Cộng đồng nhỏ hơn Raspberry Pi, nên việc tìm tài liệu hỗ trợ hoặc ví dụ mã nguồn có phần hạn chế.

### NVIDIA Jetson Xavier NX

Jetson Xavier NX là một trong những nền tảng máy tính nhúng tiên tiến nhất của NVIDIA, được thiết kế để phục vụ cho các ứng dụng trí tuệ nhân tạo (AI) biên (edge AI) đòi hỏi hiệu năng cao nhưng vẫn bị giới hạn bởi kích thước, công suất tiêu thụ và trọng lượng. Với hiệu suất lên đến 21 TOPS (INT8), thiết bị này có thể xử lý đồng thời nhiều mô hình học sâu hiện đại và dữ liệu đầu vào từ nhiều cảm biến độ phân giải cao trong thời gian thực.

Jetson Xavier NX có kích thước nhỏ hơn thẻ tín dụng nhưng tích hợp một loạt phần cứng chuyên dụng mạnh mẽ như GPU Volta với Tensor Cores, bộ xử lý Carmel đa nhân, bộ tăng tốc deep learning (NVDLA), bộ nhớ tốc độ cao và hỗ trợ nhiều camera qua giao tiếp CSI. Điều này khiến Xavier NX trở thành lựa chọn lý tưởng cho các hệ thống yêu cầu xử lý AI phức tạp như robot tự hành, camera thông minh, thị giác máy tính nâng cao và tất nhiên là cả hệ thống cảnh báo hỏa hoạn kết hợp AI.



Hình 4-4: NVIDIA Jetson Xavier NX

a) Thông số kỹ thuật:

- AI Performance:
  - 14–21 TOPS (INT8), tùy theo chế độ công suất (10W hoặc 15W).

- GPU:
  - 384 nhân NVIDIA Volta™ với 48 Tensor Cores.
  - Tần số tối đa: 800 MHz (10W) / 1100 MHz (15W).
- CPU:
  - 6 nhân NVIDIA Carmel ARM® v8.2 64-bit.
  - Bộ nhớ đệm: 6MB L2 + 4MB L3.
  - Tần số tối đa:
    - 2 nhân @ 1.5GHz và 4 nhân @ 1.2GHz (10W)
    - 2 nhân @ 1.9GHz và 4/6 nhân @ 1.4GHz (15W)
- Bộ nhớ:
  - 8 GB LPDDR4x (128-bit) @ 1600 MH.
  - Băng thông bộ nhớ: 51.2 GB/s.
- Lưu trữ:
  - 16 GB eMMC 5.
- Mạng:
  - Ethernet 10/100/1000 Mbps (RJ45).
- Video:
  - Encode (HEVC):
    - 2 luồng 4K @ 30fp
    - 6 luồng 1080p @ 60fps
    - 14 luồng 1080p @ 30fp
  - Decode (HEVC):
    - 2 luồng 4K @ 60fps
    - 12 luồng 1080p @ 60fps
    - 32 luồng 1080p @ 30fps
    - 16 luồng 1080p @ 30fps (H.264)
- Camera (CSI):
  - Tối đa 6 camera vật lý (36 qua kênh ảo).
  - 12 lane MIPI CSI-2, D-PHY 1.2 (tốc độ lên đến 30 Gbps).
- Giao tiếp mở rộng:
  - PCIe Gen 3: 1 x1 + 1 x4 (Root Port & Endpoint).
  - 2 bô tăng tốc deep learning NVDLA.
  - 2 cổng hiển thị đa chế độ: DisplayPort 1.4, eDP 1.4, HDMI 2.0.
- Công suất tiêu thụ:
  - Có thẻ cầu hình 10W hoặc 15W tùy ứng dụng.
- Kích thước:
  - 45 mm x 69.6 mm.
  - Kết nối qua 260-pin SO-DIMM.

b) Ưu điểm:

- Hiệu năng AI vượt trội: Lên đến 21 TOPS, xử lý tốt nhiều mô hình deep learning cùng lúc.
- GPU chuyên biệt với Tensor Core: Thực thi tối ưu các tác vụ AI như nhận diện hình ảnh, phát hiện đối tượng, phân tích hành vi.
- Tích hợp bộ tăng tốc học sâu (NVDLA): Tăng tốc inference trực tiếp trên thiết bị mà không cần đến GPU.
- Kết nối đa camera tốc độ cao: Hỗ trợ 6 camera vật lý và 36 kênh ảo thông qua giao tiếp MIPI CSI-2.
- Tối ưu hóa cho thiết bị biên: Kích thước nhỏ, cấu hình tiết kiệm năng lượng (10W–15W), phù hợp thiết bị nhúng AI cao cấp.
- Hỗ trợ SDK mạnh mẽ: JetPack SDK, DeepStream, TensorRT giúp triển khai AI nhanh chóng và ổn định.

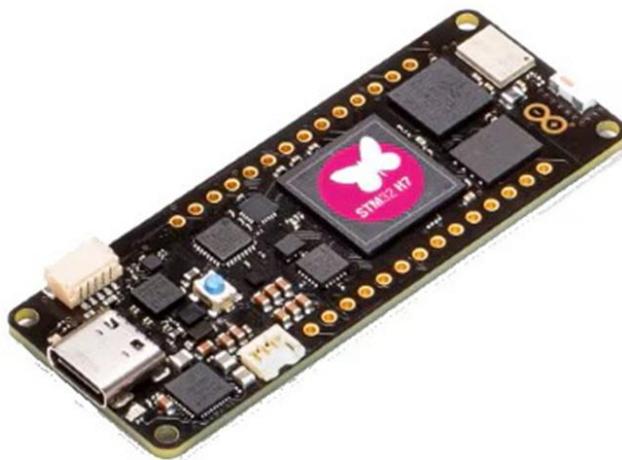
c) Nhược điểm:

- Chi phí cao: Giá thành của Jetson Xavier NX cao hơn nhiều so với các dòng như Raspberry Pi hay Jetson Nano.
- Yêu cầu nguồn ổn định: Cần nguồn công suất cao và ổn định ( $\geq 15W$ ) khi chạy full tải.
- Khó triển khai cho hệ thống đơn giản: Với các ứng dụng không cần AI chuyên sâu, hiệu năng của Xavier NX bị dư thừa.
- Không tích hợp Wi-Fi/Bluetooth: Cần bổ sung module không dây qua PCIe hoặc USB.

### **Arduino Portenta H7:**

Arduino Portenta H7 là một trong những dòng bo mạch máy tính nhúng cao cấp nhất của Arduino, hướng tới các ứng dụng công nghiệp, điều khiển thời gian thực, trí tuệ nhân tạo (AI) và các hệ thống cần hiệu năng cao. Bo mạch này sử dụng vi điều khiển STM32H747 với cấu trúc 2 lõi xử lý song song Cortex-M7 (480 MHz) và Cortex-M4 (240 MHz), cho phép thực hiện đồng thời các tác vụ thời gian thực và xử lý ứng dụng cấp cao.

Portenta H7 có khả năng chạy nhiều môi trường phần mềm như: Arduino, Mbed OS, MicroPython, JavaScript và TensorFlow Lite. Ngoài ra, nó hỗ trợ kết nối không dây (Wi-Fi/Bluetooth), truyền video qua DisplayPort, và đặc biệt là tích hợp GPU Chrom-ART Accelerator để tăng tốc đồ họa.



Hình 4-5: Arduino Portenta H7

a) Thông số kỹ thuật:

- Bộ xử lý:
  - STM32H747 Dual Core:
    - Cortex-M7 @ 480 MHz
    - Cortex-M4 @ 240 MHz
  - 6MB L2 cache tổng cộng.
  - Hỗ trợ Remote Procedure Call (RPC) giữa hai lõi.
- Bộ nhớ:
  - 8 MB SDRAM (tùy chọn mở rộng đến 64 MB).
  - 16 MB NOR Flash (tùy chọn mở rộng đến 128 MB QSPI NAND).
  - Hỗ trợ microSD và bộ nhớ ngoài qua QSPI.
- Đồ họa:
  - GPU tích hợp: Chrom-ART Accelerator™.
  - Hỗ trợ mã hóa và giải mã JPEG.
  - Xuất video: DisplayPort qua cổng USB-C.
- Kết nối không dây:
  - Wi-Fi 802.11 b/g/n (tốc độ lên tới 65 Mbps).
  - Bluetooth Classic + BLE.
  - Anten: gồm onboard hoặc UFL (tùy chọn).
- Kết nối có dây:
  - Ethernet PHY 10/100 Mbps.
  - USB 2.0 HS (480 Mbps) qua cổng USB-C.
  - UART, SPI, I2C, ADC, PWM thông qua chân mở rộng.
- Giao diện mở rộng:
  - 2x header 80 chân công nghiệp (high-density connectors).
  - Tương thích với form-factor MK.

- Tính năng bảo mật:
  - Hỗ trợ chip mã hóa: Microchip ATECC608A hoặc NXP SE050C2 (tùy phiên bản).
- Nguồn cấp:
  - USB-C, hoặc qua chân header mở rộng.
  - Hỗ trợ cấp nguồn cho thiết bị OTG qua USB.
- Kích thước:
  - Kích thước tương đương MKR (~ 25 x 75 mm).
- Hệ điều hành và môi trường phát triển:
  - Arduino IDE.
  - Mbed OS.
  - Cloud9 IDE (qua web).
  - Hỗ trợ Arduino IoT Cloud.

b) Ưu điểm:

- Chạy song song hai lõi (M7 + M4): Có thể xử lý đồng thời tác vụ AI và điều khiển thời gian thực.
- Hỗ trợ nhiều nền tảng phần mềm: Arduino, Mbed, MicroPython, JavaScript, TensorFlow Lite.
- Tích hợp Wi-Fi/Bluetooth và xuất video: Kết nối không dây mạnh mẽ, hỗ trợ DisplayPort cho giao diện người dùng trực quan.
- GPU tích hợp: Hỗ trợ đồ họa cho UI công nghiệp, tăng tốc xử lý hình ảnh.
- Cấu hình linh hoạt: Có thể chọn dung lượng RAM/Flash, chip bảo mật, kiểu anten, theo nhu cầu thực tế.
- Form-factor công nghiệp: Hỗ trợ mở rộng mạnh với 160 chân I/O tiêu chuẩn công nghiệp.

c) Nhuược điểm:

- Chi phí cao: Giá thành cao hơn nhiều so với các bo mạch Arduino thông thường hoặc Raspberry Pi.
- Không có hệ điều hành đầy đủ như Linux: Không thể chạy các ứng dụng phức tạp như streaming hoặc web server như trên Raspberry Pi.
- Chưa phổ biến rộng rãi: Ít dự án thực tế và tài liệu tham khảo hơn so với Raspberry Pi hoặc Jetson.
- Hạn chế về cộng đồng hỗ trợ: Ít thư viện hoặc ví dụ mẫu cho các ứng dụng AI nặng hoặc thị giác máy tính.

Bảng 4-1: Bảng so sánh các loại máy tính Nhúng

Thuộc tính	Raspberry Pi 4	Jetson Nano	Jetson Xavier NX	BeagleBone Black	Arduino Portenta H7
CPU	Quad-core Cortex-A72 @ 1.5GHz	Quad-core Cortex-A57 @ 1.43GHz	6-core Carmel ARM v8.2 @ 1.4–1.9GHz	Cortex-A8 @ 1GHz	Dual-core: Cortex-M7 @ 480MHz + M4 @ 240MHz
RAM	2/4/8GB LPDDR4	4GB LPDDR4	8GB LPDDR4x	512MB DDR3L	8MB SDRAM (tùy chọn đến 64MB)
GPU/AI	VideoCore VI (basic)	Maxwell 128-core GPU (472 GFLOPS)	Volta 384-core + 48 Tensor Cores (21 TOPS)	PowerVR SGX530	Chrom-ART GPU + TensorFlow Lite hỗ trợ
Hệ điều hành	Linux full (Raspberry Pi OS, Ubuntu...)	Linux for Tegra	JetPack SDK (Linux for Tegra)	Debian, Ubuntu	Mbed OS, Arduino, MicroPython, JS
Kết nối không dây tích hợp	Wi-Fi + Bluetooth 5.0	Cần thêm module ngoài	Cần thêm module ngoài	Không có	Wi-Fi + BLE
Kết nối camera	CSI camera x1	MIPI CSI x2	MIPI CSI x6 (tối đa 36 kênh ảo)	Chỉ HDMI	Không chuyên cho xử lý camera trực tiếp
Khả năng chạy AI	Trung bình (chạy AI nhẹ qua CPU)	Tốt (chạy YOLO-tiny, MobileNet...)	Rất mạnh (đa mô hình real-time)	Kém (không phù hợp AI)	Hỗ trợ TensorFlow Lite, hiệu năng thấp hơn
Khả năng truyền video	Stream RTMP/HLS, 4K HDMI	Stream được	Đa luồng 4K/1080p mạnh	Chỉ xuất HDMI cơ bản	DisplayPort qua USB-C

<b>Thuộc tính</b>	<b>Raspberry Pi 4</b>	<b>Jetson Nano</b>	<b>Jetson Xavier NX</b>	<b>BeagleBone Black</b>	<b>Arduino Portenta H7</b>
Giao tiếp ngoại vi	Rất đầy đủ (40 chân GPIO, SPI, UART, I2C...)	Đầy đủ	Đầy đủ, hỗ trợ PCIe	Rất mạnh (46 chân, ADC, PRU...)	160 chân mở rộng qua connector công nghiệp
Hỗ trợ AI trong thực tế	Phù hợp cho mô hình AI nhẹ, học thuật	Tốt cho bài toán AI trung bình	Mạnh cho thị giác máy + multi-camera	Không phù hợp AI	Hỗ trợ TensorFlow Lite, phù hợp ứng dụng IoT
Chi phí	1.720.000đ - 2.700.000đ	5.800.000đ	18.000.000đ	1.900.000đ	4.700.000đ
Cộng đồng, tài liệu hỗ trợ	Rất lớn, dễ tìm ví dụ, thư viện	Lớn, do NVIDIA hỗ trợ tốt	Lớn, nhưng kỹ thuật cao hơn	Nhỏ, ít ví dụ AI	Nhỏ, chủ yếu hướng công nghiệp cao cấp

Mặc dù các nền tảng máy tính nhúng hiện nay như Jetson Nano, Jetson Xavier NX, BeagleBone Black hay Arduino Portenta H7 đều có những ưu điểm nổi bật riêng, đặc biệt là trong các ứng dụng AI, công nghiệp hoặc xử lý song song, nhưng xét trên tổng thể yêu cầu của đề tài – bao gồm hiệu năng xử lý vừa đủ, khả năng kết nối mạnh mẽ, hỗ trợ camera, truyền video, dễ tích hợp phần mềm và chi phí phù hợp thì Raspberry Pi 4 là lựa chọn tối ưu nhất.

Raspberry Pi 4 sở hữu bộ vi xử lý ARM Cortex-A72 bốn nhân, cùng bộ nhớ RAM có thể lên tới 8GB, đủ sức chạy các mô hình AI nhẹ phục vụ nhận diện khói/lửa, đồng thời xử lý việc thu nhận dữ liệu cảm biến, truyền hình ảnh thời gian thực và điều khiển các thiết bị như quạt, còi cảnh báo. Ngoài ra, Pi 4 được tích hợp sẵn Wi-fi và Bluetooth 5.0 giúp giảm thiểu phần cứng phụ trợ, tiết kiệm chi phí và công lắp đặt. Cổng camera CSI giúp việc kết nối và xử lý video trực tiếp từ camera trở nên dễ dàng hơn.

Đặc biệt, cộng đồng hỗ trợ Raspberry Pi rất lớn, tài liệu phong phú, dễ dàng tra cứu khi gặp lỗi hay cần mở rộng tính năng. Hệ thống cũng tương thích tốt với các công nghệ mà đề tài sử dụng như HTTP, Firebase, Flutter, cho phép phát triển và triển khai hệ thống một cách nhanh chóng và hiệu quả. Chính vì vậy, mặc dù rất ấn tượng với các nền tảng khác, nhóm em đã quyết định lựa chọn Raspberry Pi 4 là trung tâm xử lý của hệ thống cảnh báo hỏa hoạn kết hợp AI.

#### *4.1.1.2. Nguồn cấp cho Raspberry Pi 4*

Nguồn cấp điện ổn định là điều kiện tiên quyết để Raspberry Pi 4 hoạt động liên tục và không gặp sự cố, tiền đề để hệ thống hoạt động ổn định.



*Hình 4-6: Nguồn 5V 3A*

Thông số kỹ thuật:

- Điện áp đầu vào: 90~264VAC / 47~63Hz
- Điện áp đầu ra: 5.1 VDC
- Dòng đầu ra: 3A
- Công suất: 15.3 W

Ôn định và bảo vệ: Nguồn điện đạt các tiêu chuẩn quốc tế dành riêng cho Raspberry Pi 4, có bảo vệ cách ly, chống sét an toàn.

#### *4.1.1.3. Thẻ nhớ microSD cho Raspberry Pi 4*

Thẻ nhớ microSD là thành phần lưu trữ chính của Raspberry Pi 4, được sử dụng để cài đặt hệ điều hành và chứa toàn bộ phần mềm điều khiển hệ thống. Trong đê tài này, nhóm em đã sử dụng thẻ microSD Class 10 UHS-I để cài đặt hệ điều hành và chạy các tác vụ trên Raspberry Pi 4.

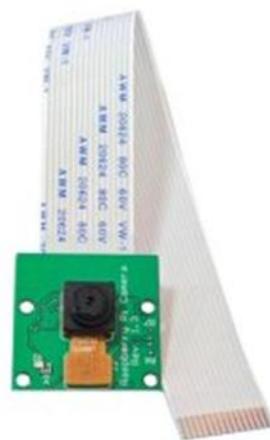


Hình 4-7: Thẻ microSD 64GB

- Dung lượng phù hợp: 64GB, đảm bảo chứa được hệ điều hành Raspberry Pi OS, các thư viện cần thiết cho tác vụ, mô hình AI và các bản cập nhật phần mềm.
- Tốc độ đọc/ghi cao: 100MB/s. Đảm bảo quá trình khởi động và chạy tác vụ nhanh chóng.
- Độ bền và độ tin cậy: Thẻ có độ bền cao, chịu được việc ghi/xóa dữ liệu liên tục, đảm bảo hệ thống hoạt động ổn định lâu dài.
- Khả năng tương thích: Tương thích với Raspberry Pi 4, đảm bảo nhận diện và truy xuất dữ liệu nhanh chóng.

#### 4.1.1.4. Camera cho Raspberry Pi 4

Trong đề tài, camera OV5647 được sử dụng để truyền video trực tiếp lên server và cung cấp hình ảnh cho mô hình AI nhận diện khói và lửa.



Hình 4-8: Camera OV5647

- Độ phân giải phù hợp: 5MP (2592 x 1944), đủ rõ nét để nhận diện khói hoặc lửa ở khoảng cách gần (0.2 – 3m).
- Góc nhìn chấp nhận được: Camera có góc nhìn khoảng 65°, bao quát được không gian dài và hẹp như hành lang chung cư mini.
- Phù hợp giám sát cố định, không cần lấy nét động: Tiêu cự cố định, khẩu độ f/1.8.
- Khả năng tương thích: Kết nối CSI (15 pin), tương thích hoàn toàn với Raspberry Pi 4, độ trễ thấp, không chiếm cổng USB.
- Khả năng hoạt động liên tục: Hoạt động liên tục khoảng 2 giờ 15 phút trong điều kiện thời tiết nóng và không có tản nhiệt, đảm bảo đủ thời lượng để xử lý và giám sát trong các phiên thử nghiệm.

#### 4.1.1.5. Vi điều khiển

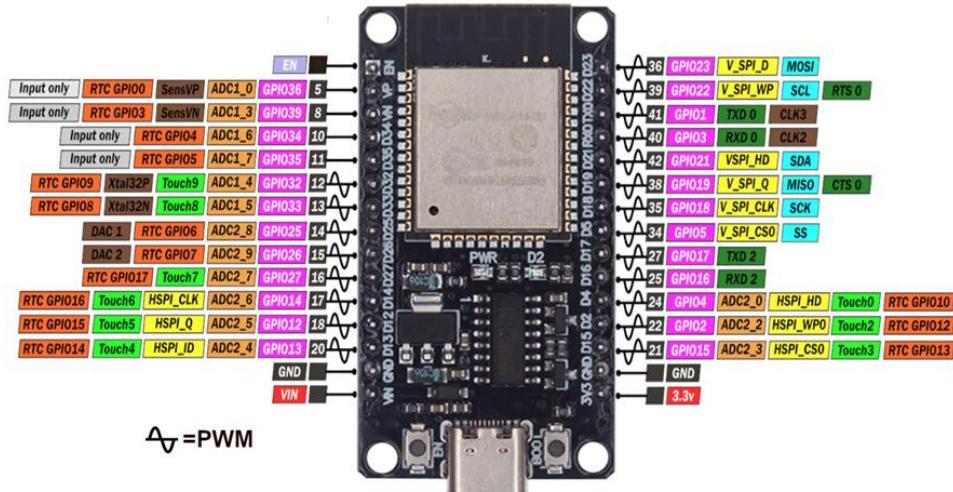
Trong quá trình thiết kế hệ thống, việc lựa chọn vi điều khiển đóng vai trò then chốt vì nó ảnh hưởng trực tiếp đến hiệu năng, khả năng giao tiếp và chi phí. Để đưa ra lựa chọn phù hợp, nhóm em đã tiến hành so sánh một số loại vi điều khiển phổ biến hiện nay như trình bày trong bảng dưới đây:

Bảng 4-2: So sánh các loại vi điều khiển

Thuộc tính	ESP32	ESP8266	Arduino Uno	Arduino Nano 33 IoT	STM32F103 C8T6
Vi xử lý	Dual-core Xtensa LX6 @ 240 MHz	Tensilica LX106 @ 80 MHz	ATmega32 8P @ 16 MHz	ARM Cortex-M0+ @ 48 MHz	ARM Cortex-M3 @ 72 MHz
RAM	~520 KB SRAM	128 KB SRAM	2 KB (ATmega3 28P)	32KB SRAM	20 KB SRAM
Flash	4 MB	4 MB	32 KB (ATmega3 28P), 0.5 KB dùng cho bootloader	256 KB	64/128 KB
Wi-fi	Tích hợp	Tích hợp	Không có	Tích hợp	Không có
Bluetooth	BLE + Classic	Không có	Không có	BLE	Không có

<b>Thuộc tính</b>	<b>ESP32</b>	<b>ESP8266</b>	<b>Arduino Uno</b>	<b>Arduino Nano 33 IoT</b>	<b>STM32F103 C8T6</b>
Giao tiếp ngoại vi	UART, SPI, I2C, ADC, PWM...	UART, SPI, I2C, ADC, PWM	UART, SPI, I2C, ADC, PWM	UART, SPI, I2C, ADC, PWM	UART, SPI, I2C, ADC, PWM
Số chân GPIO	30 chân	17	14	22	47
Nguồn hoạt động	3.3V	5V	5V	3.3V	3.3V
Môi trường lập trình	Arduino IDE, ESP-IDF, PlatformIO	Arduino IDE	Arduino IDE	Arduino IDE	Keil uvision, Arduino IDE
Giá thành	70.000đ – 100.000đ	81.000đ	125.000đ	800.000đ - 900.000đ	60.000đ - 75.000đ
Tài liệu hỗ trợ	Công đồng lớn, tài liệu phong phú	Khả phô biến	Phổ biến, cộng đồng lớn	Khả phô biến	Không phô biến

Sau khi so sánh các vi điều khiển, ESP32 nổi bật lên là lựa chọn tối ưu cho hệ thống cảm biến cảnh báo hỏa hoạn nhờ sự cân bằng giữa hiệu năng, kết nối và giá thành.



Hình 4-9: ESP32 DEV KIT 30 chân

ESP32 tích hợp sẵn cả Wi-fi và Bluetooth, trong khi các vi điều khiển khác như Arduino UNO hay STM32 không có. Nhờ đó, hệ thống không cần thêm module ngoại vi, giúp tiết kiệm chi phí, không gian mạch và giảm độ phức tạp khi lắp đặt. So với ESP8266 chỉ có Wi-fi, ESP32 vẫn chiếm ưu thế với khả năng giao tiếp đa dạng hơn.

ESP32 được trang bị vi xử lý hai nhân tốc độ cao, cùng với dung lượng bộ nhớ RAM và Flash lớn, đảm bảo khả năng đọc và xử lý dữ liệu từ nhiều cảm biến khác nhau một cách ổn định và nhanh chóng. Các vi điều khiển như Arduino Uno hay Nano 33 IoT có bộ nhớ hạn chế, dễ bị quá tải khi cần xử lý đồng thời nhiều tín hiệu đầu vào.

Giá thành của ESP32 cũng khá rẻ, chỉ dao động khoảng 70.000đ – 100.000đ, tương đương hoặc thấp hơn nhiều so với các loại vi điều khiển khác, trong khi vẫn đảm bảo tính năng. Đây là yếu tố then chốt trong dự án triển khai thực tế với chi phí hạn chế.

Ngoài ra, ESP32 có cộng đồng phát triển mạnh mẽ, tài liệu hỗ trợ phong phú, dễ lập trình với Arduino IDE hoặc PlatformIO, dễ dàng trong việc học, triển khai và mở rộng hệ thống khi cần thiết.

#### 4.1.1.6. Cảm biến nhiệt độ, độ ẩm

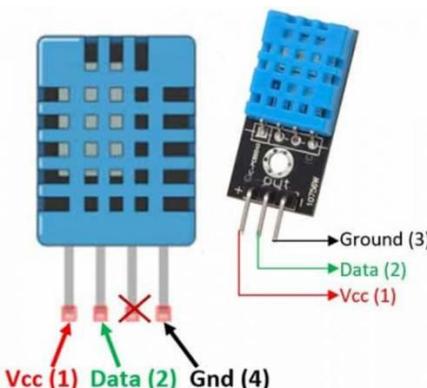
Trong hệ thống cảnh báo hỏa hoạn, việc giám sát nhiệt độ và độ ẩm môi trường là rất quan trọng để phát hiện sớm các dấu hiệu bất thường có thể dẫn đến cháy nổ. Do đó, việc lựa chọn các loại cảm biến nhiệt độ - độ ẩm phù hợp cần được cân nhắc kỹ lưỡng, dựa trên các tiêu chí như dài đo, độ chính xác, giao tiếp, giá thành và sự tương thích với vi điều khiển (trong đề tài này là ESP32).

Hiện nay trên thị trường có nhiều loại cảm biến phổ biến như DHT11, DHT22, BME280 và SHT31. Mỗi loại đều có những ưu nhược điểm riêng, phù hợp với các yêu cầu sử dụng khác nhau. Bảng dưới đây trình bày sự so sánh giữa các cảm biến kể trên để tìm ra được cảm biến phù hợp với hệ thống.

Bảng 4-3: Bảng so sánh các cảm biến nhiệt độ, độ ẩm

Tiêu chí	DHT11	DHT22	BME280	SHT31
Dải đo nhiệt độ	0 – 50 °C	-40 – 80 °C	-40 – 85 °C	-40 – 125 °C
Sai số nhiệt độ	±2 °C	±0.5 °C	±1 °C	±0.2 °C
Điện áp hoạt động	3.3V – 5V	3.3V – 5V	1.2V – 3.6V	2.5V – 5.5V
Giao tiếp	1-wire (digital)	1-wire (digital)	I2C / SPI	I2C
Tốc độ đo	1 Hz	0.5 Hz	1 Hz	1 Hz
Giá thành	22.000đ	135.000đ	130.000đ	84.000đ
Kích thước	28mm x 12mm x 10m	28mm x 12mm x 10mm	2,5mm x 2,5mm x 0,93mm	11 x 13mm

Dựa trên bảng so sánh trên có thể thấy DHT11 không phải loại cảm biến có độ chính xác cao nhất trên thị trường, nhưng với các yêu cầu thực tế của đề tài thì nó vẫn là một lựa chọn hợp lý và tối ưu. DHT11 có chi phí rất thấp, phù hợp để triển khai trong hệ thống có nhiều nút cảm biến ở nhiều vị trí khác nhau, giúp tiết kiệm ngân sách tổng thể.



Hình 4-10: Cảm biến DHT11

Ngoài ra, lập trình và tích hợp DHT11 cũng rất đơn giản nhờ sử dụng giao tiếp một dây (single-wire). Điều này giúp giảm đáng kể thời gian phát triển, lập

trình phần cứng. Không chỉ vậy, DHT11 có kích thước nhỏ gọn, dễ dàng gắn lên mạch in PCB.

Dù không mang lại độ chính xác cao như BME280 nhưng DHT11 vẫn đáp ứng tốt yêu cầu về việc phát hiện sự biến động nhiệt độ bất thường. Đặc biệt, trong hệ thống này, bộ cảm biến đóng vai trò như một lớp bảo vệ bổ sung, giúp phát hiện nguy cơ tại những góc khuất mà camera không được giám sát trực tiếp. Phần lớn khả năng phát hiện vẫn dựa vào camera và mô hình AI, cảm biến DHT11 chỉ góp phần nâng cao độ tin cậy trong hệ thống cảnh báo tổng thể.

#### *4.1.1.7. Cảm biến khí CO*

Khí CO (Carbon Monoxide) là một loại khí không màu, không mùi nhưng cực độc, sinh ra trong quá trình cháy không hoàn toàn. Việc phát hiện sớm sự hiện diện của khí CO giúp cảnh báo nguy cơ hỏa hoạn ngay cả khi chưa có lửa hay khói rõ rệt, đặc biệt tại các khu vực bị khuất tầm nhìn hoặc không thể lắp đặt camera.

Hiện nay có nhiều loại cảm biến CO phổ biến, mỗi loại có ưu nhược điểm riêng về độ nhạy, lượng điện tiêu thụ, giá thành và độ phức tạp khi tích hợp. Bảng sau đây là so sánh một số loại cảm biến tiêu biểu trên thị trường của nhóm em:

Bảng 4-4: Bảng so sánh các cảm biến khí CO

Tiêu chí	MQ-7	MQ-9	ZE07-CO	MiCS-5524
Điện áp hoạt động	5V	2.5V – 5V	5 – 12V	5V
Dải đo CO (ppm)	10 – 1000ppm	10 – 1000ppm (CO + LPG)	0 – 500ppm	1 – 1000ppm
Nhiệt độ hoạt động	-10°C – 50°C	-10°C – 50°C	-10°C – 50°C	-40°C – 85°C
Giao tiếp	Analog	Analog	UART (TTL)	Analog
Ưu điểm	Giá rẻ, phổ biến, nhạy với CO, dễ dùng với ESP32	Cảm biến đa khí, độ nhạy cao	Độ chính xác cao, không cần hiệu chuẩn thủ công	Cảm biến đa khí, kích thước nhỏ, tiêu thụ điện thấp
Nhược điểm	Cần thời gian làm nóng, dễ bị ảnh hưởng bởi môi trường	Ít chuyên biệt cho CO, dễ sai lệch khi có khí khác	Giá cao, giao tiếp phức tạp	Cần mạch đọc ADC chuẩn, tín hiệu yếu
Kích thước	33 x 20 x 16mm	40 x 21mm	25.4x22.4x18.3mm	20.0mm x 12.7mm x 3.1mm
Giá thành	35.000đ	35.000đ	550.000đ	490.000đ

Qua bảng so sánh ta có thể thấy, hiện nay có nhiều cảm biến CO với độ chính xác cao và giao tiếp hiện đại nhưng MQ-7 vẫn là lựa chọn phù hợp cho hệ thống cảnh báo hỏa hoạn được triển khai trong đề tài này.



Hình 4-11: Cảm biến MQ-7

MQ-7 có giá thành rẻ, dễ mua và dễ thay thế – điều rất quan trọng đối với hệ thống có nhiều nút cảm biến được triển khai rộng. Ngoài ra, MQ-7 tương thích tốt với vi điều khiển ESP32, chỉ cần đọc tín hiệu Analog là có thể hoạt động, không cần xử lý các giao tiếp phức tạp như UART hay I2C.

MQ-7 cũng có công đồng hỗ trợ lớn, tài liệu phong phú, thư viện có sẵn để dàng tích hợp vào hệ thống và hiệu chuẩn trong quá trình thử nghiệm. Dù vẫn tồn tại các hạn chế là độ chính xác bị ảnh hưởng bởi điều kiện môi trường nhưng trong mô hình giám sát liên tục và có thể kiểm chứng chéo với cảm biến nhiệt độ, khói hay camera AI thì có thể chấp nhận nhược điểm này.

#### 4.1.1.8. Cảm biến nồng độ khói

Khói là một trong những dấu hiệu rõ ràng và sớm nhất cho thấy khả năng xảy ra hỏa hoạn. Việc phát hiện khói kịp thời, đặc biệt trong các không gian kín như phòng trọ, nhà dân, hành lang chung cư mini, ... giúp cảnh báo sớm và giảm thiểu thiệt hại. Do đó, việc trang bị cảm biến khói cho hệ thống cảnh báo là cần thiết và có ý nghĩa thực tiễn cao.

Trong số nhiều loại cảm biến khói hiện nay, nhóm quyết định lựa chọn cảm biến MP2 – một loại cảm biến khói quang điện (photoelectric smoke sensor). MP2 hoạt động dựa trên nguyên lý tán xạ ánh sáng: bên trong cảm biến là một đèn LED hồng ngoại và một photodiode bố trí vuông góc nhau. Khi có khói lọt vào buồng cảm biến, các hạt khói làm tán xạ ánh sáng từ LED đến photodiode, tạo ra tín hiệu điện tỷ lệ với mật độ khói. Điều này giúp MP2 nhận biết chính xác sự xuất hiện của khói trong môi trường.



Hình 4-12: Cảm biến MP-2

Về mặt kết nối, cảm biến MP2 có 3 chân: VCC (nguồn 3.3V hoặc 5V), GND (mass) và AOUT (analog out). Khi kết nối với vi điều khiển như ESP32, chỉ cần đọc giá trị điện áp ở chân AOUT qua analogRead() là có thể biết được mức độ khói. Người dùng có thể thiết lập ngưỡng cảnh báo (ví dụ: nếu giá trị > 300 thì bật còi/báo cháy), từ đó dễ dàng tích hợp vào hệ thống.

Cảm biến MP2 không yêu cầu làm nóng như các cảm biến khí MQ, có độ ổn định cao, dễ sử dụng, không bị ảnh hưởng bởi các loại khí gas khác. Với ưu điểm nhỏ gọn, đơn giản, đáng tin cậy và chi phí thấp, MP2 là lựa chọn phù hợp cho hệ thống cảnh báo hỏa hoạn trong đè tài, đảm bảo khả năng phát hiện sớm khói ngay cả trong các khu vực góc khuất hoặc nơi camera không thể quan sát được.

#### 4.1.1.9. Còi báo động

Để hỗ trợ cảnh báo tức thời tại chỗ khi phát hiện cháy, hệ thống cần một thiết bị âm thanh có thể phát ra tín hiệu rõ ràng, dễ nhận biết. Trong đè tài này, nhóm sử dụng còi báo động 5V vỏ nhựa – là một loại còi chủ động có vỏ bảo vệ, âm thanh lớn (~85 dB), được sử dụng phổ biến trong các hệ thống cảnh báo an ninh, báo cháy mini.



Hình 4-13: Còi báo 5V

Còi có thiết kế gọn, dễ gắn vào mạch cảm biến. Việc kết nối và điều khiển cũng rất đơn giản: còi có 2 dây (đỏ – VCC, đen – GND), chỉ cần cấp điện áp 5V để còi hoạt động. Khi kết hợp với vi điều khiển ESP32, còi sẽ được điều khiển thông qua transistor NPN (ví dụ 2N2222 hoặc S8050) để đóng/mở mạch cung cấp nguồn. Điều này giúp bảo vệ GPIO của vi điều khiển khỏi tải dòng cao.

Nguyên lý kết nối:

- Dây đỏ nối với 5V nguồn ngoài (qua cực collector của transistor).
- Dây đen nối với chân GND.
- GPIO của ESP32 điều khiển cực base của transistor (qua điện trở  $1k\Omega$ ).
- Khi GPIO xuất mức HIGH → transistor dẫn → còi kêu.

#### 4.1.1.10. Nguồn cho mạch cảm biến

Bộ nguồn AC-DC 5V-2A được sử dụng để cấp điện cho mạch cảm biến (gồm vi điều khiển, các cảm biến môi trường và còi).



Hình 4-14: Nguồn 5V 2A

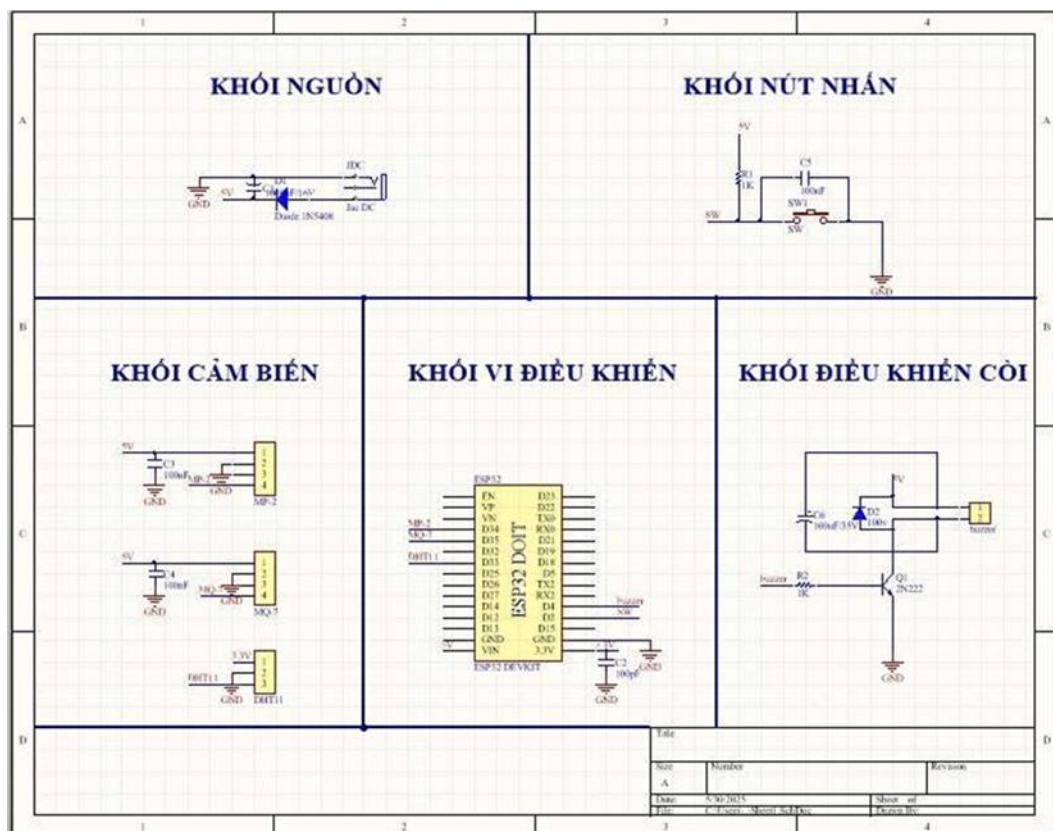
Thông số kỹ thuật:

- Điện áp đầu vào: 100 – 250 VAC (phù hợp lưới điện Việt Nam).
- Đầu ra: Cung cấp điện áp 5V, dòng điện 2A, đảm bảo cung cấp nguồn điện ổn định cho toàn mạch hoạt động, không bị sụt áp hay reset bất ngờ.
- Giao tiếp: Jack DC 5.5mm.

#### 4.1.2 Thiết kế mạch

Sau khi chọn linh kiện, nhóm em tiến hành kết nối các linh kiện và chạy thử trên bảng trắng. Sau khi mạch đã chạy ổn định, nhóm em tiến hành làm mạch in PCB.

##### 4.1.2.1. Sơ đồ nguyên lý của mạch



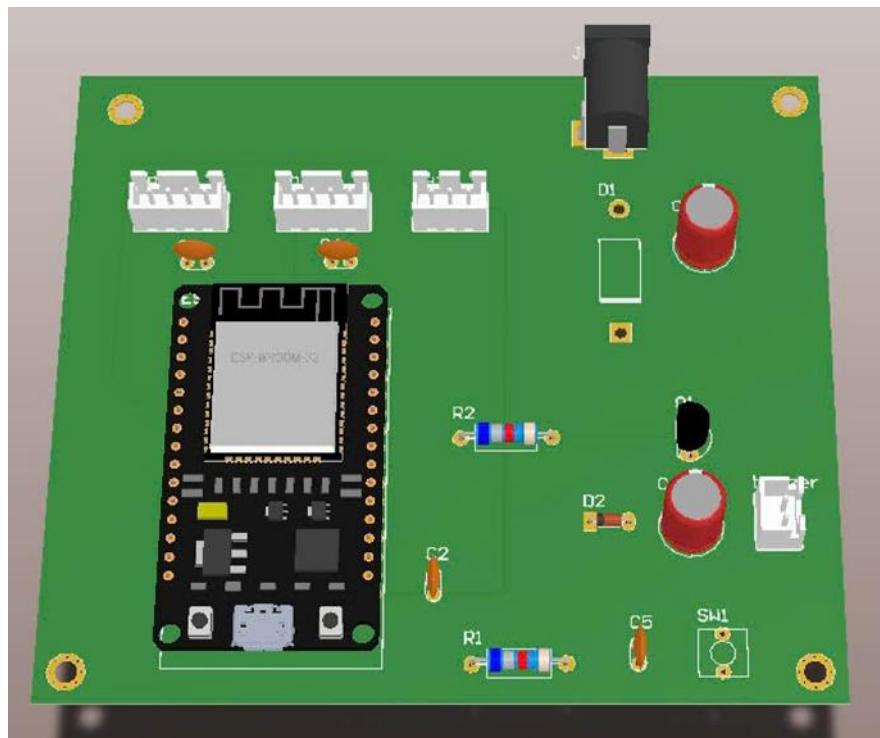
Hình 4-15: Sơ đồ nguyên lý mạch cảm biến

Kết nối chi tiết:

- MP-2: Kết nối với chân D34 của ESP32
- MQ-7: Kết nối với chân D35 của ESP32
- DHT11: Kết nối với chân D33 của ESP32
- Buzzer: Kết nối với chân D4 của ESP32
- Switch: Kết nối với chân D2 của ESP32

Dựa trên sơ đồ nguyên lý, chúng em thực hiện kết nối, đi dây các linh kiện lại với nhau và kết quả cuối cùng được thể hiện ở hình 4-16.

#### 4.1.2.2. Mạch PCB 3D



Hình 4-16: Hình ảnh 3D mạch PCB

## 4.2 Tính toán nồng độ khí CO và khói từ cảm biến MQ-7 và MP-2

### 4.2.1 Giới thiệu

Để giám sát chất lượng không khí, giúp phát hiện sớm nguy cơ cháy thông qua việc đo nồng độ khí CO và khói, hệ thống sử dụng hai loại cảm biến khí bán dẫn phổ biến:

- **MQ-7:** để phát hiện khí CO (carbon monoxide)
- **MP-2:** để phát hiện khói (tổng hợp các khí dễ cháy như CH<sub>4</sub>, LPG, khói thuốc...)

Cả hai cảm biến đều hoạt động theo nguyên lý thay đổi điện trở cảm biến R<sub>s</sub> khi tiếp xúc với khí mục tiêu. Điện trở này được tính toán dựa trên điện áp đầu ra V<sub>RL</sub> đo được qua điện trở tải R<sub>L</sub>

### 4.2.2 Công thức tính toán nồng độ (ppm)

- Tính điện áp đầu ra V<sub>RL</sub> từ ADC:

$$V_{RL} = \frac{ADC}{4095} \times V_C$$

- Tính R<sub>s</sub> (điện trở cảm biến):

$$R_s = R_L \times \frac{V_C - V_{RL}}{V_{RL}}$$

- Tính tỉ số  $R_s/R_0$

$$\frac{R_s}{R_0}$$

Trong đó:

- $R_0$  và  $R_s$  đo được trong môi trường không khí sạch (hiệu chuẩn trước).
- $R_L$  là điện trở tải (MQ-7: 10kΩ, MQ-2: 20kΩ).
- $V_C$  là điện áp cấp (5V).
- Tính ppm từ tỉ số  $R_s/R_0$  bằng phương pháp nội suy logarit:

- **MQ-7 (CO):**

$$\log_{10}(ppm) = \frac{\log_{10}(R_s/R_0) - 1.57}{-0.79} \Rightarrow ppm = 10^{\left(\frac{\log_{10}(R_s/R_0) - 1.57}{-0.79}\right)}$$

- **MQ-2 (Khói):**

$$\log_{10}(ppm) = \frac{\log_{10}(R_s/R_0) - 1.37}{-0.395} \Rightarrow ppm = 10^{\left(\frac{\log_{10}(R_s/R_0) - 1.37}{-0.395}\right)}$$

(Các hệ số 1.57, -0.79 (MQ-7) và 1.37, -0.395 (MQ-2) được trích xuất từ biểu đồ đặc tuyến (sensitivity curve) trong datasheet chính thức của nhà sản xuất).

#### 4.2.3 Hiệu chỉnh và điều chỉnh thực nghiệm

Trong điều kiện phòng bình thường:

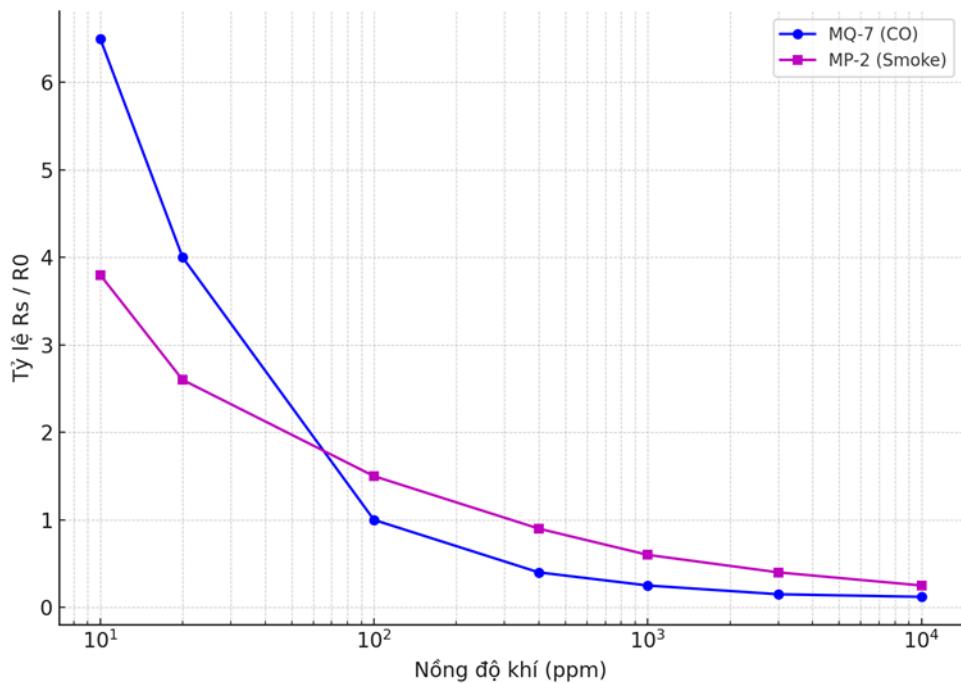
- Nồng độ CO đo được từ cảm biến MQ-7: **0 – 5 ppm**.
- Nồng độ khói đo được từ cảm biến MP-2: **< 30 ppm**.

Để đảm bảo các giá trị đo phù hợp với thực tế, các hệ số  $R_0$  được điều chỉnh thủ công dựa trên giá trị  $R_s$  trong môi trường sạch:

Bảng 4-5: Bảng hiệu chỉnh giá trị  $R_0$  cho cảm biến

Cảm biến	Giá trị $R_L$ ( $\Omega$ )	$R_0$ hiệu chỉnh ( $\Omega$ )	Giá trị ADC sạch
MQ-7	10,000	370,000	90
MP-2	20,000	300,000	110

Việc hiệu chỉnh này giúp đầu ra ppm khớp với ngưỡng khí trong môi trường thật và loại bỏ sai số do cảm biến trôi, sai lệch linh kiện hoặc độ ẩm môi trường.

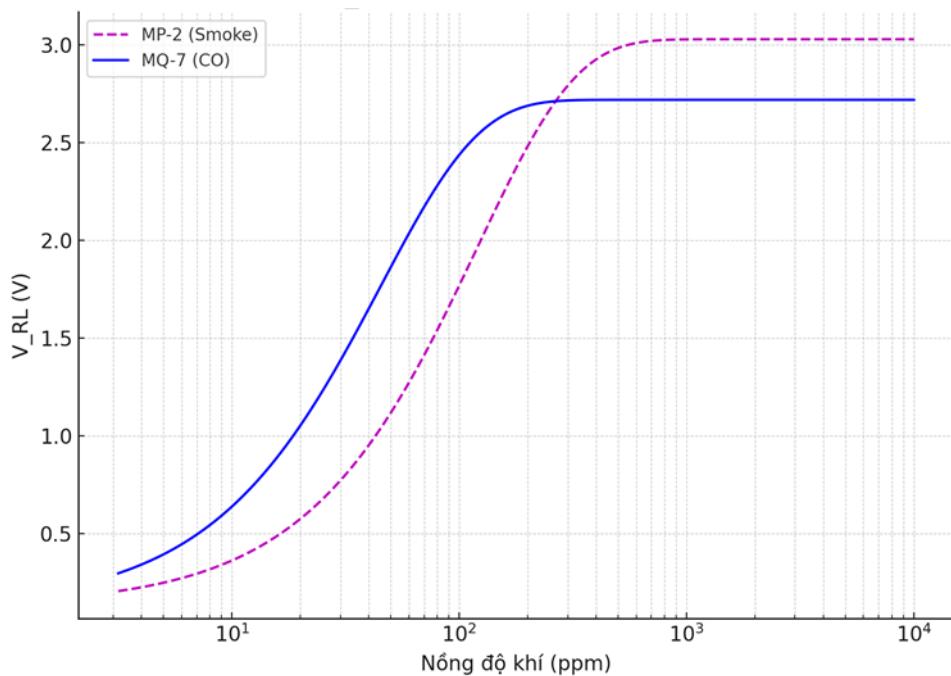


Hình 4-17: Đồ thị mối quan hệ giữa  $\frac{R_s}{R_0}$  và nồng độ khí (ppm)

Cả hai cảm biến đều cho thấy tỷ lệ  $\frac{R_s}{R_0}$  giảm khi nồng độ khí tăng, thể hiện sự giảm điện trở cảm biến trong môi trường có nồng độ CO hoặc khói cao.

Ở mức nồng độ thấp ( $10 - 100$  ppm), MQ-7 có độ nhạy cao hơn MP-2, thể hiện qua việc  $\frac{R_s}{R_0}$  giảm nhanh hơn. Điều này phù hợp với mục tiêu phát hiện khí CO từ sớm.

MP-2 phản ứng ổn định với các nồng độ khói từ trung bình đến cao, giúp hệ thống có thể đánh giá mức độ nghiêm trọng của tình huống cháy.



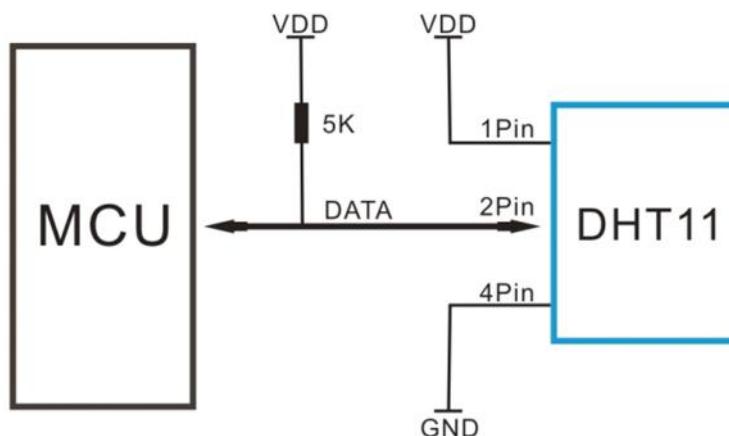
Hình 4-18: Đồ thị mối quan hệ giữa  $V_{RL}$  và nồng độ khí (ppm)

Điện áp ngõ ra  $V_{RL}$  tăng theo nồng độ khí – đây là đặc tính lý tưởng để sử dụng ADC (Analog – to – Digital Converter) của ESP32.

MQ-7 có biên độ điện áp từ khoảng 0.3V đến 2.6V, trong khi MP-2 có thể đạt đến khoảng 3.0V, cho phép phân biệt rõ ràng hơn giữa các mức khói khác nhau.

### 4.3 Đọc dữ liệu nhiệt độ từ cảm biến DHT11

Phần này trình bày quy trình thu thập dữ liệu nhiệt độ từ cảm biến DHT11 sử dụng vi điều khiển ESP32. Đây là một trong những yếu tố quan trọng để đánh giá điều kiện môi trường trong hệ thống phát hiện cháy sớm.



Hình 4-19: Sơ đồ kết nối DHT11 với vi điều khiển

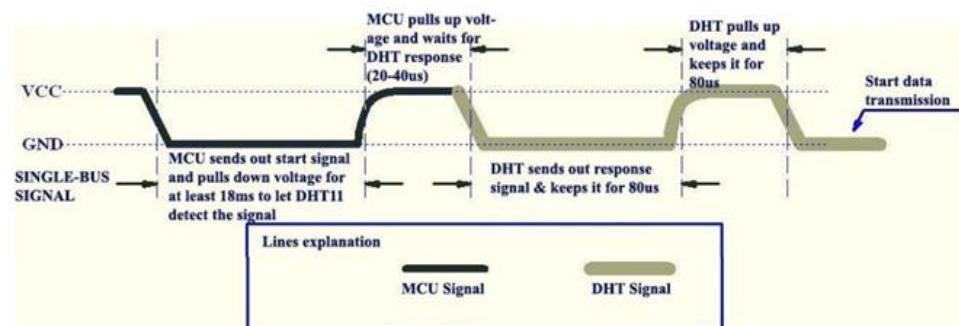
#### 4.3.1 Nguyên lý hoạt động của cảm biến DHT11

Cảm biến sử dụng tín hiệu kỹ thuật số (digital) theo giao thức một dây (single-wire), truyền dữ liệu bằng cách mã hóa thời gian xung.

Quy trình truyền dữ liệu giữa DHT11 và ESP32 bao gồm các bước:

- ESP32 khởi động cảm biến bằng cách kéo chân tín hiệu xuống mức thấp trong 18 – 20ms.
- DHT11 phản hồi bằng một chuỗi xung gồm 40 bit dữ liệu, bao gồm:
  - 8 bit độ ẩm nguyên (humidity integer).
  - 8 bit độ ẩm thập phân (humidity decimal).
  - 8 bit nhiệt độ nguyên (temperature integer).
  - 8 bit nhiệt độ thập phân (temperature decimal).
  - 8 bit kiểm tra (checksum).
- Mỗi bit được truyền dưới dạng xung:
  - Mức thấp kéo xuống khoảng 50 $\mu$ s.
  - Sau đó là mức cao: 26 – 28 $\mu$ s nếu là bit 0, 80 $\mu$ s nếu là bit 1.

ESP32 cần đo độ rộng xung chính xác để giải mã các bit này thành dữ liệu số.



Hình 4-20: Sơ đồ timming giao tiếp của DHT11

#### 4.3.2 ESP32 nhận và xử lý dữ liệu từ DHT11

Việc đọc dữ liệu DHT11 không dùng ADC (Analog-to-Digital Converter) mà là dùng đếm thời gian độ rộng xung (pulse width).

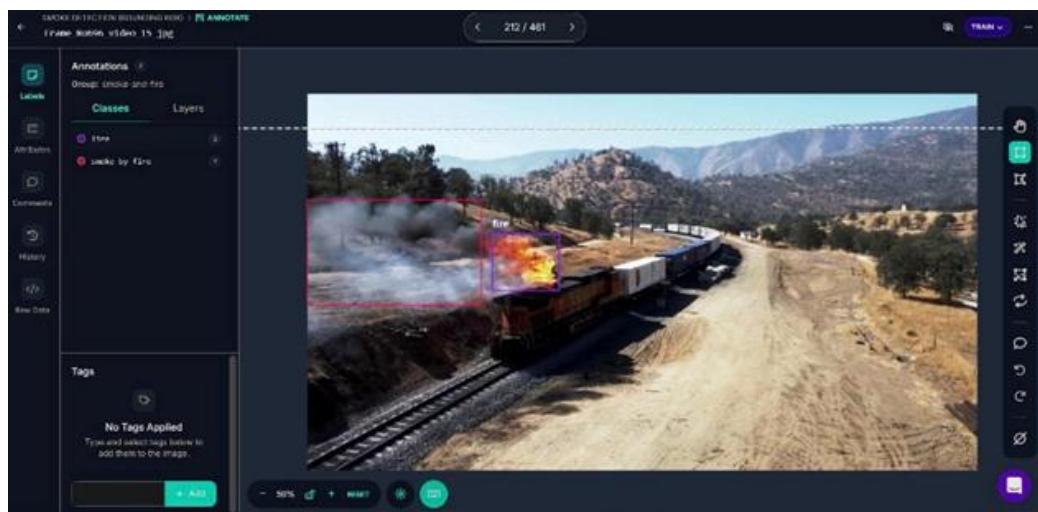
Quy trình xử lý:

- Khởi tạo giao tiếp bằng cách kéo chân GPIO xuống mức thấp  $\geq 18ms$ .
- Đọc 40 bit dữ liệu từ cảm biến bằng cách đo độ dài từng xung digital.
- Ghép dữ liệu.

## 4.4 Thực hiện huấn luyện mô hình

### 4.4.1 Huấn luyện mô hình

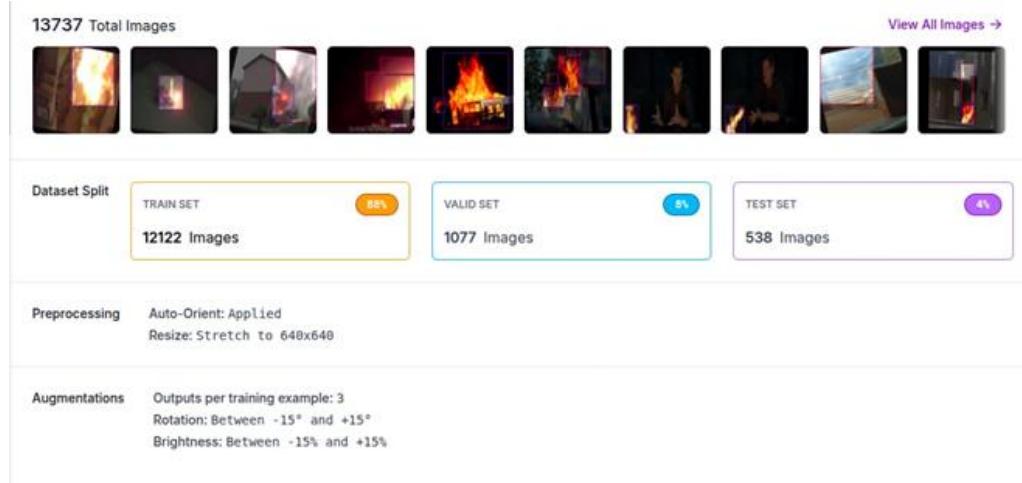
Để huấn luyện được mô hình nhận diện lửa và khói, bước đầu tiên là chúng em chuẩn bị dataset. Bằng cách tìm kiếm, thu thập ảnh trên các trang mạng và lấy ảnh từ các video đám cháy chúng em có được một tập ảnh gồm 5.3k ảnh. Sau đó, chúng em thực hiện gán label thủ công trên trang web roboflow. Khi gán label, sẽ có 2 loại label là fire và smoke by fire.



Hình 4-21: Cách gán nhãn cho ảnh

Sau khi hoàn thành việc gán nhãn cho 13737 ảnh, chúng em chia số ảnh đó thành 3 tập là train, val và test với tỷ lệ 70:20:10 để tạo được tập dataset cuối cùng. Tập train gồm 12122 ảnh là tập dữ liệu chính để huấn luyện mô hình. Mô hình sẽ học các quy luật, đặc điểm trong dữ liệu từ tập này nhờ các thuật toán và các phương pháp tối ưu. Tập val gồm 1077 ảnh là tập dữ liệu được sử dụng để kiểm thử độ chính xác của mô hình trong quá trình huấn luyện mô hình, đánh giá mô hình và giúp tinh chỉnh các siêu tham số. Và tập cuối cùng là tập test gồm 538 ảnh là tập dữ liệu để test sau khi mô hình đã được huấn luyện xong. Nếu đáp ứng được yêu cầu thì đạt, không thì phải xem xét lại mô hình.

Sau khi chia thành 3 tập train, val và test, chúng em sẽ tinh chỉnh một chút và được tập dataset với các điều kiện như sau:



Hình 4-22: Dataset sử dụng

Sau khi có dataset, chúng em huấn luyện với máy tính cá nhân. Quá trình huấn luyện, chúng em thiết lập huấn luyện qua 80 epochs, 8 batchs và 8 workers với epoch là số lần duyệt qua hết các dữ liệu trong tập train, batch là số lượng dữ liệu sẽ sử dụng trong 1 lần để cập nhật tham số và worker là số luồng để tải dữ liệu. Với tập dữ liệu gồm 13737 ảnh, và 80 epochs, trong quá trình train, mỗi epoch cần 5.25 phút, tổng thời lượng cần để huấn luyện mô hình là 7 giờ.

Sau khi hoàn thành việc huấn luyện AI, em sẽ được một tập các kết quả gồm các file lưu trọng số mô hình tốt nhất best.pt, mô hình trạng thái cuối cùng last.pt, file results.png thể hiện biểu đồ biến thiên của các chỉ số như box\_loss, cls\_loss và mAP theo từng epoch giúp đánh giá quá trình hội tụ của mô hình.

#### 4.4.2 Đánh giá kết quả huấn luyện

Sau khi thực hiện huấn luyện mô hình hệ thống, chúng em trích xuất được các biểu đồ đánh giá mô hình.

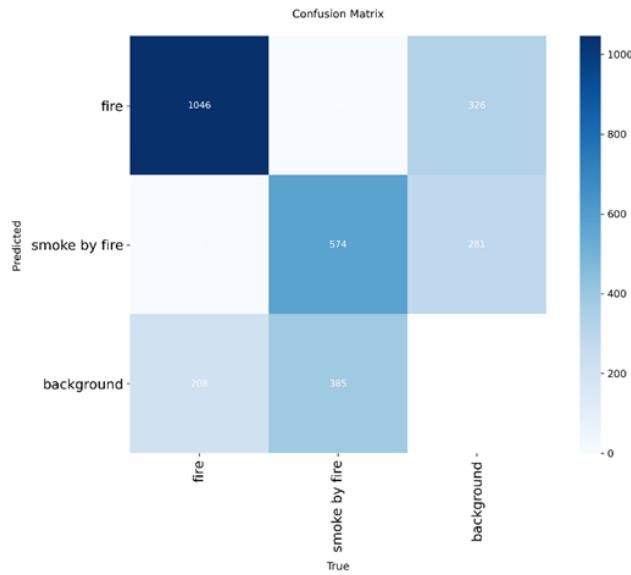
Để đánh giá hiệu quả phân loại của mô hình, cần xét đến ma trận nhầm lẫn (Confusion Matrix) - một công cụ trực quan thể hiện rõ mức độ chính xác và nhầm lẫn giữa các lớp. Với dự án này, mô hình được huấn luyện để phân biệt ba lớp: fire (lửa), smoke by fire (khói do cháy), và background (cảnh nền không có cháy).

```
Validating runs/detect/train2/weights/best.pt...
Ultralytics 8.3.143 Python-3.12.3 torch-2.7.0+cu126 CUDA:0 (NVIDIA GeForce GTX 1650, 3897MB)
Model summary (fused): 72 layers, 3,006,038 parameters, 0 gradients, 8.1 GFLOPs
    Class   Images Instances Box(P      R      mAP50      mAP50-95): 100% |██████████| 68/68 [00:09<00:00,  6.85it/s]
        all     1077    2220   0.749   0.654   0.715   0.407
        fire     975    1257   0.8     0.792   0.831   0.476
        smoke by fire  700    963   0.698   0.516   0.6     0.339
Speed: 0.4ms preprocess, 5.9ms inference, 0.0ms loss, 0.9ms postprocess per image
```

Hình 4-23: Kết quả sau khi train

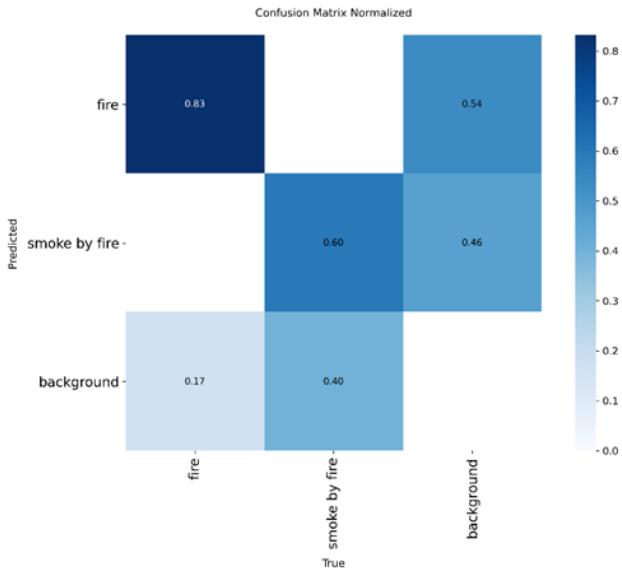
Dưới đây là hai dạng biểu diễn ma trận nhầm lẫn: Ma trận đầu tiên thể hiện số lượng tuyệt đối các dự đoán đúng và sai giữa các lớp và ma trận thứ hai là phiên

bản chuẩn hóa theo hàng, thể hiện tỷ lệ phần trăm đúng/sai theo từng lớp thực tế, giúp đánh giá mô hình một cách trực quan và chính xác hơn.



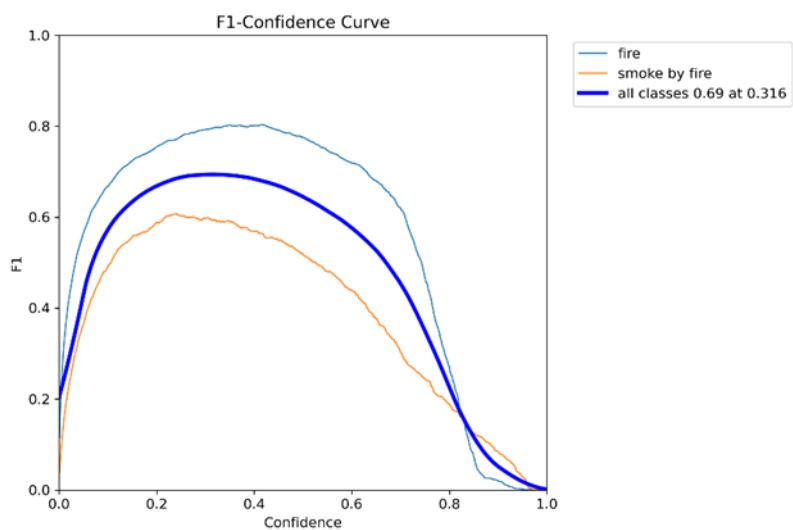
Hình 4-24: Confusion Matrix

Ma trận nhầm lẫn tuyệt đối thể hiện số lượng dự đoán của mô hình cho từng lớp so với nhãn thật. Nhận xét thấy với lớp “fire” được mô hình dự đoán tốt, tuy nhiên trong một số trường hợp nhất định, tín hiệu của lửa có thể bị che khuất hoặc lẫn với môi trường xung quanh khiến mô hình đánh giá sai. Với lớp “smoke by fire” thì mô hình dự đoán không tốt bằng “fire”, mô hình gặp khó khăn trong việc phân biệt rõ ràng khói với lửa hoặc môi trường nền – đặc biệt trong các tình huống hình ảnh bị mờ, thiếu sáng, hoặc khói loãng. Vì khói là thứ rất nhiều hình dạng, màu sắc, khói đặc, khói ít làm cho mô hình rất khó dự đoán. Nhưng nhìn chung mô hình có hiệu quả khá cao với lớp “fire”, nhưng vẫn cần cải thiện khả năng phân biệt giữa “smoke” và “background”.



Hình 4-25: Confusion Matrix Normalized

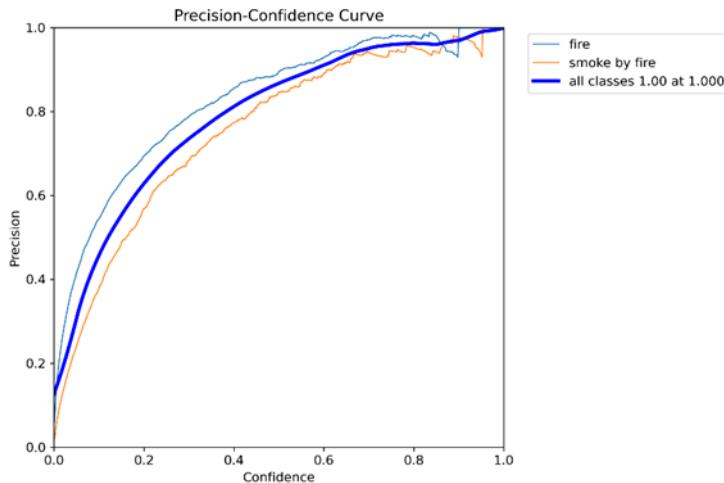
Ma trận nhầm lẫn chuẩn hóa cho thấy với lớp “fire” mô hình đạt tỷ lệ chính xác ấn tượng lên tới 83%, cho thấy đây là lớp dễ nhận biết nhất và được mô hình học tốt. Chỉ còn khoảng 17% bị nhầm sang lớp "background", phần lớn có thể do đặc điểm hình ảnh của lửa bị nhiễu hoặc không rõ ràng. Với lớp smoke by fire, tỷ lệ chính xác đạt 60% – tức cứ 10 ảnh thì có khoảng 6 ảnh được nhận diện đúng là khói, trong khi 40% còn lại vẫn bị nhầm, chủ yếu sang "background". Khói vốn có hình dạng không cố định, màu sắc gần giống nền và thường xuất hiện mờ, loãng, vì vậy đây là lớp khó nhận diện nhất trong bài toán này.



Hình 4-26: F1-Confidence Curve

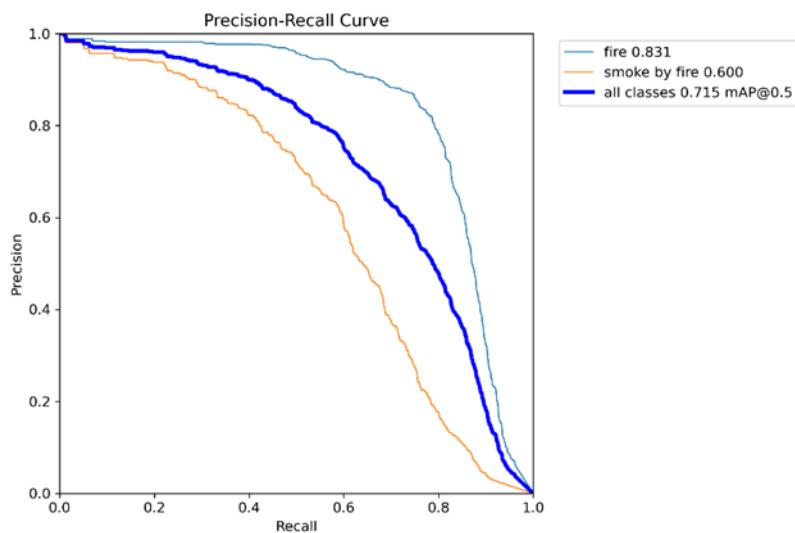
F1-score là trung bình điều hòa giữa Precision và Recall. Đây là chỉ số cân bằng giữa độ chính xác và khả năng bao phủ, rất quan trọng trong bài toán cảnh báo cháy – nơi mà bỏ sót (false negative) và cảnh báo giả (false positive) đều có

thể gây hậu quả nghiêm trọng. Dựa vào kết quả của đường cong F1 theo confidence, nhóm đã lựa chọn ngưỡng confidence = 0.316 làm mức mặc định khi triển khai hệ thống cảnh báo. Đây là ngưỡng mang lại F1-score cao nhất cho toàn bộ mô hình và cho thấy sự cân bằng giữa cảnh báo chính xác và độ tin cậy của hệ thống.



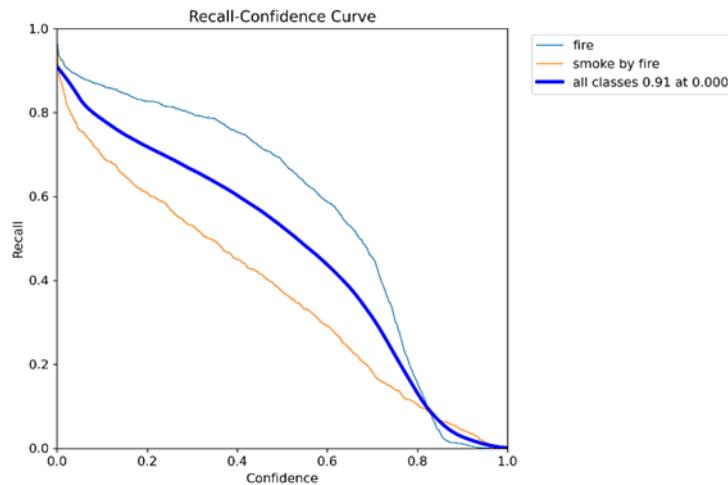
Hình 4-27: Precision-Confidence Curve

Đường cong Precision theo Confidence thể hiện mối quan hệ giữa ngưỡng conf và độ dự đoán của mô hình. Biểu đồ cho thấy, mô hình càng tự tin thì độ chính xác dự đoán càng cao, đặc biệt với lớp “fire” đạt precision > 0.95 khi confidence > 0.9. Điều này chứng tỏ mô hình có khả năng kiểm soát cảnh báo giả tốt, đặc biệt phù hợp với các ứng dụng yêu cầu độ tin cậy cao như hệ thống cảnh báo cháy.



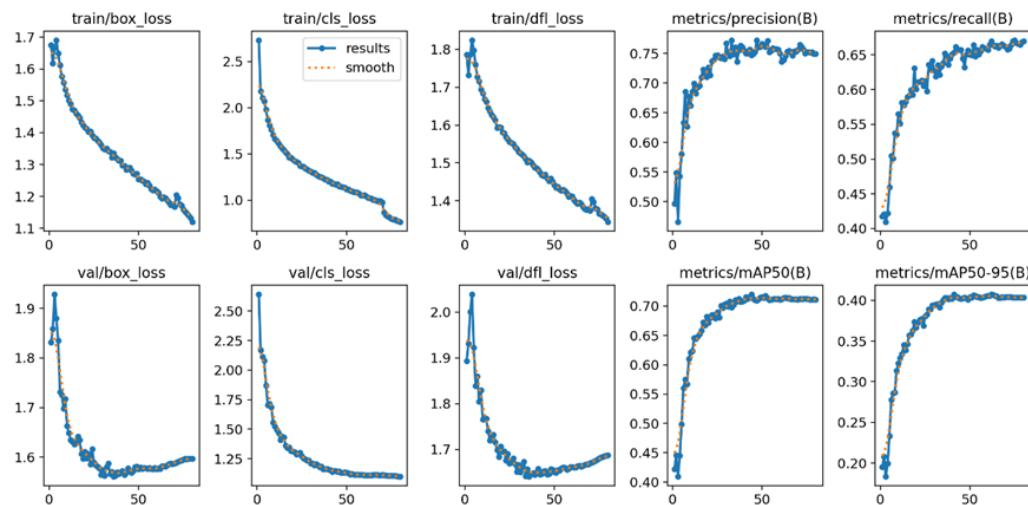
Hình 4-28: Precision-Recall Curve

Đường cong PR (Precision – Recall) cho thấy mô hình đạt hiệu quả rất cao trong việc phát hiện lửa với AP = 0.831, trong khi lớp khói chỉ đạt 0.600 do đặc tính khó nhận diện hơn. Tổng thể, mô hình đạt mAP@0.5 = 0.715, đủ tốt để triển khai trong các hệ thống cảnh báo cháy thực tế.



Hình 4-29: Recall-Confidence Curve

Đường cong Recall theo Confidence ho thấy mô hình đạt độ bao phủ rất cao với recall lên đến 91% ở confidence thấp, đặc biệt hiệu quả với lớp “fire”. Tuy nhiên, lớp “smoke by fire” có recall giảm nhanh khi confidence tăng, cho thấy mô hình dễ bỏ sót khói nếu yêu cầu độ tin cậy quá cao.



Hình 4-30: Kết quả chung

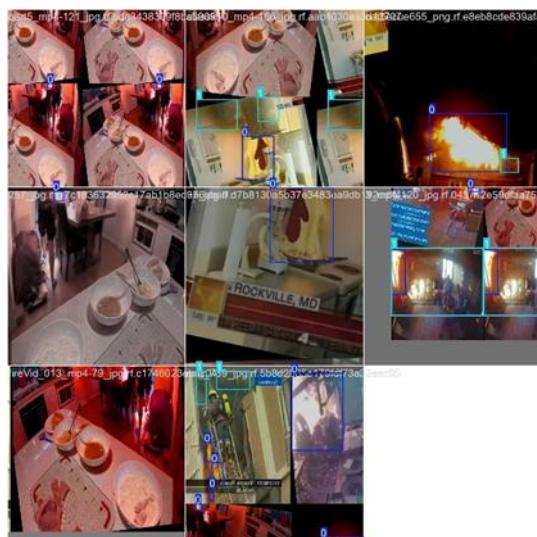
Quá trình huấn luyện và đánh giá mô hình được thể hiện qua các biểu đồ loss và chỉ số đánh giá cho cả tập huấn luyện (train) và kiểm tra (validation). Các hàm loss chính bao gồm box\_loss, cls\_loss và dfl\_loss đều giảm đều theo thời gian, cho thấy mô hình học hiệu quả và hội tụ tốt. Đặc biệt, đường loss của tập validation

gần như song song và đồng pha với loss của tập huấn luyện, điều này phản ánh mô hình không gặp hiện tượng overfitting, đồng thời có khả năng khai quát hóa tốt trên dữ liệu chưa từng thấy.

Về mặt hiệu năng, các chỉ số đánh giá tăng ổn định qua các epoch:

- Precision tăng từ ~0.5 lên gần 0.75, cho thấy mô hình ngày càng đưa ra các dự đoán chính xác hơn.
- Recall cũng được cải thiện rõ rệt, từ ~0.4 lên khoảng 0.67, tức là mô hình giảm thiểu bỏ sót các đối tượng cần phát hiện.
- Đặc biệt, mAP@0.5 đạt 0.715 và mAP@0.5:0.95 đạt 0.41, là các kết quả rất tốt đối với mô hình YOLOv8n, chứng minh rằng mô hình không chỉ dự đoán đúng mà còn định vị chính xác vật thể ở nhiều mức IoU khác nhau.

Với tập dataset của nhóm, khi dùng để huấn luyện mô hình còn nhận được các kết quả mô hình nhận diện trên chính tập train và val của dataset. Và cũng như để đánh giá trực quan hiệu quả mô hình phát hiện khói và lửa, nhóm cũng đưa những ảnh tổng hợp đầu ra trong quá trình huấn luyện và kiểm thử. Các ảnh như train\_batch0.jpg, train\_batch1.jpg thể hiện các mẫu dữ liệu huấn luyện cùng với vùng nhãn (bounding box) đã được gán. Qua đó có thể quan sát được rằng dữ liệu huấn luyện khá đa dạng, bao gồm cả môi trường trong nhà, ngoài trời, ban ngày, ban đêm, và các trường hợp cháy ở quy mô khác nhau.



Hình 4-32: train\_batch0



Hình 4-31: train\_batch1

Trong khi đó, các ảnh như val\_batch0\_labels.jpg và val\_batch0\_pred.jpg cho phép so sánh giữa nhãn thực tế và dự đoán của mô hình trên tập kiểm thử. Để

nhận thấy mô hình có khả năng nhận diện lửa tương đối chính xác với độ tin cậy cao, vùng phát hiện sát với thực tế. Tuy nhiên, ở một số ảnh, mô hình phát hiện khói còn chưa ổn định, đặc biệt là với khói mỏng hoặc lẩn vào nền sáng. Điều này phù hợp với đánh giá định tính trước đó về việc phát hiện khói thường gặp nhiều khó khăn hơn do đặc điểm hình ảnh không rõ ràng.



Hình 4-34: val\_batch0\_labels



Hình 4-33: val\_batch0\_pred

Các kết quả đầu ra này không chỉ phản ánh hiệu quả nhận diện của mô hình mà còn giúp nhóm điều chỉnh chiến lược huấn luyện (như tăng cường dữ liệu khói hoặc tinh chỉnh ngưỡng confidence) để nâng cao độ chính xác của hệ thống trong các lần huấn luyện tiếp theo.

## 4.5 Triển khai mô hình sang Raspberry Pi 4

Sau khi huấn luyện xong mô hình trên máy tính cá nhân, nhóm em tiến hành triển khai mô hình lên Raspberry Pi để thực hiện nhận diện trong môi trường thực tế. Trọng số mô hình tốt nhất best.pt được chuyển sang Pi thông qua `scp`. Do hạn chế về tài nguyên trên Pi, nhóm em đã chuyển đổi mô hình sang định dạng nhẹ hơn là NCNN. Sau đó, chúng em đã viết một đoạn script Python để sử dụng mô hình đã chuyển đổi để xử lý ảnh đầu vào từ camera Pi theo thời gian thực. Kết quả nhận diện sẽ được hiển thị lên terminal và có thể lưu ảnh, phát cảnh báo hoặc gửi dữ liệu về server tùy theo yêu cầu ứng dụng. Việc triển khai giúp mô hình hoạt động độc lập mà không cần kết nối đến máy tính huấn luyện ban đầu.

### 4.5.1 Xử lý tại biên trên Raspberry Pi 4

Để triển khai hệ thống giám sát thông minh trên Raspberry Pi 4, chúng em đã thiết lập một pipeline xử lý song song dựa trên Gstreamer và thư viện libcamera, sử dụng camera CSI OV5647 làm nguồn ghi hình. Khi camera hoạt động bình

thường, hình ảnh được ghi nhận liên tục và luồng dữ liệu video được chia tách thành hai nhánh bằng cơ chế FIFO (First-In First-Out) thông qua pipe tạm thời trong hệ điều hành. Cụ thể, một script shell sử dụng libcamera-vid để xuất dữ liệu video thô (raw YUV) ra luồng đầu ra chuẩn, sau đó được phân chia thành hai pipe khác nhau thông qua phần tử tee trong pipeline GStreamer: một pipe phục vụ cho quá trình phát hiện lửa/khói và một pipe khác phục vụ truyền phát video thời gian thực.

Nhánh thứ nhất được xử lý bởi GStreamer, một framework mạnh mẽ trong xử lý đa phương tiện. GStreamer sẽ đọc dữ liệu từ pipe pipe\_stream.yuv, thực hiện mã hóa video bằng x264enc, sau đó đẩy luồng video này lên server thông qua giao thức RTMP. Server đích sử dụng nginx-rtmp để tiếp nhận luồng và phân phối lại cho ứng dụng Flutter theo giao thức HLS, giúp người dùng theo dõi video thời gian thực với độ trễ thấp.

Trong khi đó, nhánh thứ hai dùng cho việc phát hiện lửa và khói. Một script Python sử dụng mô hình YOLO tích hợp với thư viện OpenCV sẽ liên tục đọc từng khung hình từ pipe pipe\_detect.yuv. Các khung hình này được chuyển đổi định dạng màu, sau đó đưa vào mô hình để nhận diện các dấu hiệu cháy như ngọn lửa hoặc khói trắng. Nếu phát hiện, hệ thống sẽ gửi metadata về server để lưu vào cơ sở dữ liệu và kích hoạt cảnh báo trên ứng dụng Flutter. Nhờ chia hai nhánh xử lý độc lập, hệ thống vừa đảm bảo stream mượt mà cho người dùng, vừa duy trì khả năng phát hiện cháy nổ theo thời gian thực ngay trên thiết bị biên mà không ảnh hưởng đến hiệu suất hoạt động.

Sau đây là phần mô tả chi tiết hoạt động của từng pipe trong hệ thống phát hiện, cảnh báo lửa và khói:

#### *4.5.1.1. Cơ chế chia luồng*

Trong hệ thống, công cụ libcamera-vid đóng vai trò then chốt trong việc ghi hình từ camera OV5647 và xuất trực tiếp luồng video ở định dạng yuv420 (raw video). Dữ liệu video thô này sau đó được đưa vào pipeline GStreamer, tại đây sử dụng phần tử tee để chia thành hai nhánh xử lý song song: một nhánh ghi vào pipe phục vụ mô hình phát hiện YOLO, một nhánh mã hóa thành H.364 và gửi qua RTMP để stream lên server.

Tuy nhiên, với việc dùng FIFO là một tiến trình ghi vào pipe sẽ bị treo nếu tại thời điểm đó, không có tiến trình nào mở đầu đọc pipe. Để tránh tình trạng này, chúng em đã khởi động trước tiến trình đọc ở hai đầu pipe.

#### *4.5.1.2. Phát hiện lửa và khói*

Để thực hiện nhiệm vụ phát hiện lửa và khói trong hệ thống, một pipe riêng là detect\_pipe sẽ thực hiện.

Đầu vào là nhánh detect là luồng video thô được chia ra từ camera thông qua pipr FIFO. Chương trình yolo\_detect\_pipe.py trên Raspberry Pi đảm nhiệm vai trò phát hiện khói và lửa trong thời gian thực bằng mô hình học sâu YOLO

(You Only Look Once). Ngay khi khởi động, chương trình sẽ tải mô hình YOLO đã huấn luyện sẵn (được chỉ định thông qua tham số --model) và thiết lập ngưỡng độ tin cậy cho các dự đoán (mặc định là 0.5, có thể điều chỉnh thông qua tham số --thresh). Sau đó, chương trình mở pipe FIFO đã được tạo từ trước để đọc liên tục các khung hình video ở định dạng YUV420. Với kích thước  $640 \times 360$  pixel, mỗi khung hình có dung lượng 345.600 byte, sau khi đọc đủ sẽ được chuyển đổi sang định dạng BGR sử dụng thư viện OpenCV để chuẩn bị đưa vào mô hình. Nhằm tiết kiệm tài nguyên tính toán, chương trình chỉ xử lý một phần các khung hình nhò cơ chế bỏ qua (--skip-frame), ví dụ mỗi 5 khung hình chỉ xử lý 1, tương ứng với tốc độ phân tích 5 fps. Các khung hình không đến lượt sẽ bị bỏ qua ngay lập tức, không đưa vào hàng đợi xử lý. Điều này rất quan trọng đối với thiết bị biên như Raspberry Pi, vốn có hạn chế về hiệu năng xử lý. Nhờ cơ chế này, mô hình vẫn có thể phát hiện khói và lửa kịp thời mà không làm gián đoạn luồng video hoặc gây nghẽn bộ nhớ.

Khi phát hiện khói hoặc lửa trong một khung hình, hệ thống chưa vội gửi cảnh báo ngay mà sẽ tiếp tục phân tích một chuỗi liên tiếp các khung hình tiếp theo (thường 5–6 frame) để xác nhận xem có đúng là có lửa, khói hay chỉ là nhầm lẫn. Nếu xác suất xuất hiện lửa/khói tiếp tục được xác nhận trong các khung hình kế tiếp, hệ thống sẽ kích hoạt cơ chế gửi cảnh báo lên server. Dữ liệu gửi bao gồm: số lượng bounding box phát hiện, lửa hay khói, Raspberry Pi của tầng nào và id. Thông tin này được đóng gói dưới dạng JSON và gửi đến server thông qua giao thức HTTP, để lưu vào database trên server, dựa vào đó để đưa server đưa ra quyết định cảnh báo về app flutter.

#### 4.5.1.3. Truyền video lên server

Để truyền phát video trực tiếp từ Raspberry Pi lên ứng dụng di động, hệ thống phải thực hiện một chuỗi các bước xử lý nhằm đảm bảo chất lượng video, tiết kiệm băng thông và giảm độ trễ. Các bước chính trong quy trình kỹ thuật bao gồm:

- **Nén dữ liệu (Compression) và mã hóa (Encoding) trực tiếp trên Raspberry PI:** Camera CSI gắn vào Raspberry Pi có khả năng ghi lại video theo thời gian thực với độ phân giải và tốc độ khung hình tùy chỉnh (ví dụ:  $1280 \times 720$  tại 30fps). Tuy nhiên, nếu dữ liệu video được giữ ở dạng thô (raw YUV hoặc RGB), dung lượng sẽ rất lớn (hơn 600 MB/phút đối với 720p30), gây quá tải cho cả bộ xử lý lẫn đường truyền mạng. Vì vậy, ngay từ đầu hệ thống đã sử dụng công cụ libcamera-vid để nén và mã hóa video trực tiếp thành định dạng H.264, tận dụng bộ mã hóa phần cứng tích hợp trên chip Broadcom của Raspberry Pi. Việc mã hóa bằng phần cứng cho phép Raspberry Pi xử lý video với mức tiêu thụ tài nguyên rất thấp, đồng thời tạo ra luồng dữ liệu video có kích thước nhỏ hơn (thường chỉ 3–5 Mbps) mà vẫn giữ chất lượng hình ảnh tốt, phù hợp với truyền phát thời gian thực trong môi trường mạng giới hạn như Wi-Fi hoặc 4G.

- **Chuyển đổi RTMP thành HLS tại server:** Máy chủ trung tâm nhận luồng RTMP từ Raspberry Pi sẽ chịu trách nhiệm chuyển đổi định dạng RTMP thành HLS để phù hợp với trình phát video trên thiết bị di động. Việc chuyển đổi này được thực hiện bởi nginx-rtmp chạy liên tục trên server, với nhiệm vụ: tách luồng video từ RTMP, phân đoạn thành các đoạn .ts nhỏ có độ dài khoảng 1-3 giây và cập nhật file danh sách .m3u8 chứa thứ tự và thông tin các phân đoạn video hiện hành.
- **Phân phối nội dung đến ứng dụng di động Flutter:** Tại phía người dùng, ứng dụng di động được phát triển bằng Flutter có tích hợp thư viện để phát video từ nguồn HLS. Ứng dụng sẽ tự động truy cập file .m3u8 từ server, sau đó lần lượt tải về các đoạn .ts tương ứng. Việc phát video theo từng đoạn nhỏ giúp giảm thiểu độ trễ tải ban đầu, đồng thời tăng độ linh hoạt khi cần tạm dừng hoặc thay đổi chất lượng luồng (nếu có nhiều bitrate).
- **Giải mã (Decoding) và phát lại video trên thiết bị người dùng:** Sau khi các đoạn video .ts được tải xuống, ứng dụng sẽ tiến hành giải mã luồng H.264 và phát lại trên màn hình người dùng. Quá trình giải mã này có thể được xử lý bằng phần cứng (nếu thiết bị hỗ trợ) hoặc phần mềm (trên các thiết bị yếu hơn). Việc sử dụng H.264 giúp tăng khả năng tương thích và giảm thiểu tiêu tốn năng lượng trên thiết bị.

### 4.5.2 Hoạt động của Server

Trong hệ thống cảnh báo cháy và giám sát môi trường, server đóng vai trò trung tâm điều phối và xử lý thông tin từ các thiết bị đầu cuối như ESP32 và Raspberry Pi, đồng thời làm cầu nối giữa thiết bị và ứng dụng di động. Nó lưu trữ metadata phát hiện lửa/khói, điều phối việc gửi cảnh báo cháy đến đúng người dùng qua Firebase. Đồng thời, server cũng chuyển đổi và phân phối luồng video từ camera theo chuẩn HLS để người dùng có thể xem trực tiếp trên app. Đây là thành phần không thể thiếu giúp kết nối và quản lý toàn bộ hệ thống một cách tập trung, chính xác và thời gian thực.

#### 4.5.2.1. Nhận dữ liệu từ Raspberry Pi và gửi cảnh báo đến app

Nhận dữ liệu từ PI: Trong hệ thống, mỗi Raspberry Pi đóng vai trò giám sát hình ảnh tại các khu vực riêng biệt, sử dụng mô hình YOLO để phát hiện khói và lửa. Khi phát hiện một hiện tượng nghi ngờ, Pi sẽ tạo ra các bản ghi metadata bao gồm thời gian, số lượng đối tượng khói (smoke\_boxes), số lượng đối tượng lửa (fire\_boxes), một mã session\_id duy nhất đại diện cho đợt phát hiện đó và một mã level\_ID cố định là ID của PI tầng đó. Các bản ghi này sẽ được gửi đến server thông qua API /upload-metadata sử dụng phương thức HTTP POST.

Lưu dữ liệu và database: Sau khi nhận được dữ liệu từ Raspberry Pi, server sẽ tiến hành lưu trữ các thông tin này vào bảng detections trong cơ sở dữ liệu. Mỗi

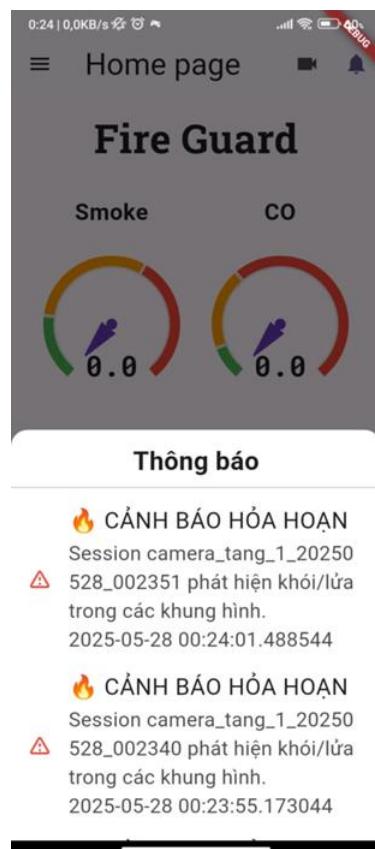
bản ghi trong bảng thể hiện một khung hình phát hiện, kèm theo các thông tin như thời điểm phát hiện, số lượng vật thể, level\_ID mặc định và session\_id của đợt phát hiện tương ứng. Điều này giúp hệ thống có thể theo dõi và kiểm tra lại toàn bộ diễn biến của một sự kiện nghi ngờ cháy thông qua truy vấn các bản ghi có cùng session\_id.

Xử lý điều kiện cảnh báo: Server sẽ theo dõi số lượng bản ghi có cùng session\_id trong cơ sở dữ liệu. Khi đủ 6 bản ghi được gửi từ một Raspberry Pi (tức một đợt phát hiện hoàn chỉnh), server sẽ đánh giá mức độ nghiêm trọng của sự kiện. Nếu ít nhất 4 trong số 6 bản ghi có phát hiện khói hoặc lửa, server sẽ coi đây là một đợt phát hiện hợp lệ và gửi tin cậy để cảnh báo. Ngược lại, nếu không đạt ngưỡng xác thực, server sẽ xóa toàn bộ 6 bản ghi đó để loại bỏ dữ liệu nhiễu, đảm bảo hệ thống không gửi cảnh báo sai lệch đến người dùng.

Gửi cảnh báo đến ứng dụng: Khi phát hiện đủ điều kiện để gửi cảnh báo, server sẽ tra cứu bảng fcm\_tokens để lấy token tương ứng với user\_id đã đăng ký trước đó. Trong hệ thống hiện tại, mỗi user\_id được gán cho một token duy nhất đại diện cho thiết bị của người dùng trong hệ thống Firebase Cloud Messaging (FCM). Sau đó, server sử dụng dịch vụ FCM để gửi thông báo cảnh báo “CẢNH BÁO HỎA HOẠN” đến đúng thiết bị đã đăng ký, kèm theo thông tin session\_id.

Việc gửi đúng cảnh báo đến đúng người dùng giúp đảm bảo tính cá nhân hóa và phản ứng nhanh trong các tình huống nguy hiểm.

#### 4.5.2.2. Nhận dữ liệu từ cảm biến



Hình 4-35: Cảnh báo hỏa hoạn trên app

Mỗi phòng của tầng đều có trang bị hệ thống cảm biến và ESP32 để thu thập dữ liệu từ cảm biến: cảm biến nhiệt độ, cảm biến khói và cảm biến CO. Các thiết bị này sau khi kết nối Wi-Fi sẽ định kỳ gửi dữ liệu đo được về server thông qua HTTP POST.

Server sẽ nhận yêu cầu và trích xuất các thông tin bao gồm: user\_id hoặc mã định danh thiết bị, giá trị nhiệt độ, nồng độ khói, và chỉ số CO. Sau đó, server lưu các bản ghi này vào bảng sensor\_data trong cơ sở dữ liệu, giúp lưu trữ toàn bộ lịch sử đo của từng thiết bị.

```

MariaDB [fire_guard]> SELECT * FROM sensor_data;
+-----+-----+-----+
| user_id | smokes | temp | co |
+-----+-----+-----+
| XoyIGv5Er2X0qyJjtkJEq8mCY2l1 |     130 |    -1 | 1368 |
+-----+-----+-----+
1 row in set (0.001 sec)

MariaDB [fire_guard]> SELECT * FROM sensor_data;
+-----+-----+-----+
| user_id | smokes | temp | co |
+-----+-----+-----+
| XoyIGv5Er2X0qyJjtkJEq8mCY2l1 |     105 |    -1 | 1445 |
+-----+-----+-----+

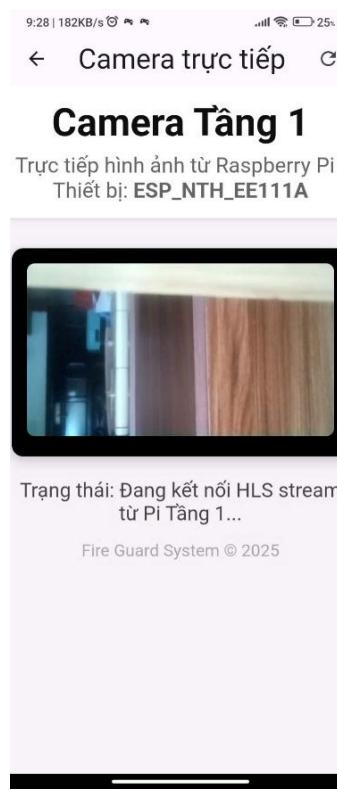
```

Hình 4-36: Dữ liệu từ cảm biến lưu vào database

Việc thu thập và lưu trữ dữ liệu cảm biến không chỉ hỗ trợ việc phát hiện sớm nguy cơ cháy, mà còn đóng vai trò quan trọng trong việc phân tích các yếu tố môi trường bất thường, hỗ trợ xác minh lại thông tin khi có cảnh báo từ phía hệ thống hình ảnh.

#### 4.5.2.3. Truyền video về app

Nhận luồng video từ Raspberry Pi thông qua RTMP: Trên mỗi Raspberry Pi, sau khi ghi hình từ camera và xuất dữ liệu video thô ở định dạng YUV420, pipeline GStreamer sẽ thực hiện mã hóa video thành định dạng H.264 bằng phần tử x264enc. Sau đó, GStreamer sử dụng phần tử rtmpsink để gửi luồng video đã mã hóa đến máy chủ trung tâm qua giao thức RTMP. Tại server, một máy chủ nginx kết hợp module nginx-rtmp được triển khai để lắng nghe tại cổng 1935 và tiếp nhận đồng thời luồng RTMP từ nhiều Raspberry Pi khác nhau. Nhờ đặc tính duy trì kết nối liên tục và độ trễ thấp của RTMP, server có thể nhận luồng video thời gian thực một cách ổn định, phục vụ cho quá trình phân phối video đến người dùng cuối.



Hình 4-37: Giao diện xem video trực tiếp trên app

Chuyển đổi RTMP sang HLS bằng ffmpeg: Do các trình duyệt và thiết bị di động hiện đại không hỗ trợ RTMP trực tiếp, server cần chuyển đổi luồng RTMP thành định dạng HLS (HTTP Live Streaming) để phát qua giao thức HTTP. Hệ thống sử dụng chính nginx-rtmp-module để thực hiện việc này một cách tự động, không cần FFmpeg. Cấu hình trong file nginx.conf đã bật hls on; và chỉ định thư

mục lưu tạm các đoạn .ts và file .m3u8. Mỗi luồng video được phân đoạn thành các tệp .ts có độ dài khoảng 1–2 giây, và danh sách phát .m3u8 sẽ luôn được cập nhật để chứa các đoạn video mới nhất. Việc phân mảnh như vậy giúp trình phát trên thiết bị di động có thể tải và phát luồng video một cách mượt mà, giảm đáng kể độ trễ và hỗ trợ buffer tốt khi kết nối mạng không ổn định.

Lưu trữ tạm thời các phân đoạn video HLS: Các file .ts và .m3u8 do nginx-rtmp tạo ra được lưu tạm trong một thư mục con trên máy chủ (/usr/local/nginx/html/hls/). Web server nginx sẽ được cấu hình để phục vụ các file này thông qua HTTP. Các đoạn .ts cũ sẽ được tự động xóa sau một thời gian nhất định (thường 15–30 giây), đảm bảo tiết kiệm tài nguyên lưu trữ và tránh tràn ô đĩa. Nhờ đó, server có thể phục vụ nhiều thiết bị và người dùng cùng lúc mà không cần tồn tại nhiều tài nguyên hệ thống.

Phân phối video đến dụng di động: Ứng dụng Flutter của người dùng sẽ truy cập luồng video thông qua địa chỉ HLS mà server đã cung cấp. Ứng dụng sử dụng thư viện video\_player để mở file .m3u8, và từ đó tự động tải lần lượt các đoạn .ts tương ứng để phát video. Việc sử dụng HLS giúp đảm bảo tính tương thích cao trên mọi nền tảng di động, đồng thời cho phép phát lại với độ trễ thấp và chất lượng ổn định. Người dùng có thể xem video gần như theo thời gian thực mà không cần tải toàn bộ dữ liệu một lượt.

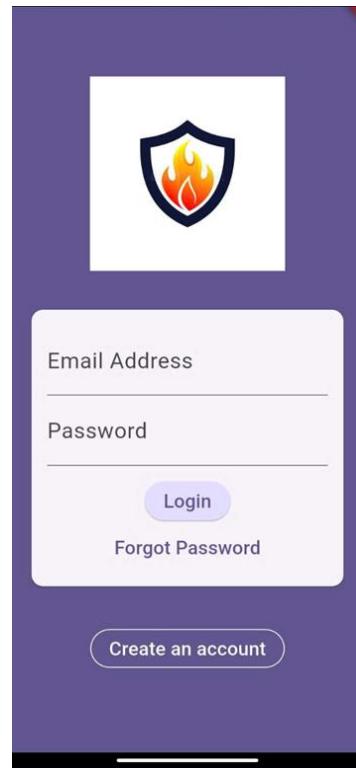
Quản lý stream và điều hướng người dùng: Mỗi Raspberry Pi sẽ tương ứng với một stream riêng biệt trên server, được định danh bằng level\_ID. Server có thể mở rộng khả năng điều hướng bằng cách liên kết level\_ID với từng người dùng trong hệ thống. Ví dụ, khi người dùng đăng nhập ứng dụng, app có thể truy vấn đến stream tương ứng với thiết bị của họ. Điều này giúp hệ thống có khả năng mở rộng và cá nhân hóa, phục vụ đồng thời nhiều người dùng và nhiều camera mà không bị nhầm lẫn luồng video.

## 4.6 Triển khai ứng dụng di động

### 4.6.1 Thiết kế giao diện người dùng

#### 4.6.1.1. Thiết kế giao diện

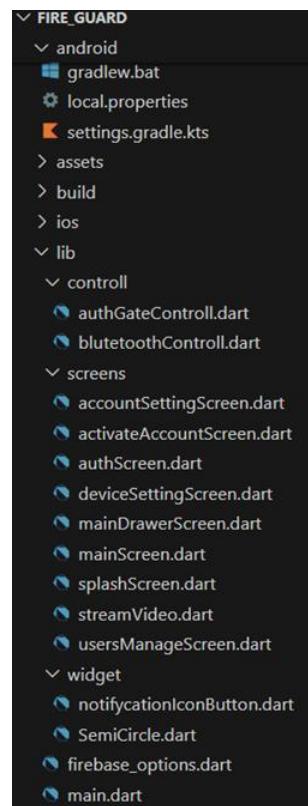
Nhóm em sử dụng Flutter & Dart để thiết kế giao diện người dùng. Các Widget như biểu mẫu, nút nhấn, logo được sắp xếp và tùy chỉnh để hiển thị thông tin thân thiện với người dùng.



Hình 4-38: Giao diện đăng nhập

#### 4.6.1.2. Lập trình logic chức năng

Ứng dụng được lập trình bằng Dart, sử dụng các thư viện và công cụ của Flutter để nhận dữ liệu từ cảm biến, xem video trực tiếp, quản lý thiết bị.



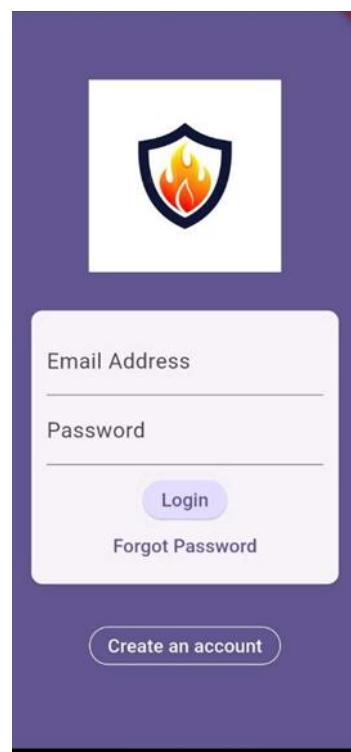
Hình 4-39: Các file .dart điều khiển logic của phần mềm

#### *4.6.1.3. Kiểm thử và gỡ lỗi*

Mục đích của giai đoạn kiểm thử và gỡ lỗi là nhằm đảm bảo ứng dụng di động hoạt động ổn định, đúng chức năng và có thể xử lý chính xác dữ liệu từ hệ thống. Trong quá trình phát triển, ứng dụng được kiểm tra toàn diện từ giao diện, luồng xử lý dữ liệu, đến khả năng nhận thông báo từ Firebase. Qua đó, nhóm em có thể phát hiện và sửa chữa các lỗi tiềm ẩn, nâng cao độ tin cậy và tính sẵn sàng triển khai của hệ thống. Các lỗi phát sinh trong quá trình này được gỡ lỗi bằng công cụ debug của Visual Studio Code.

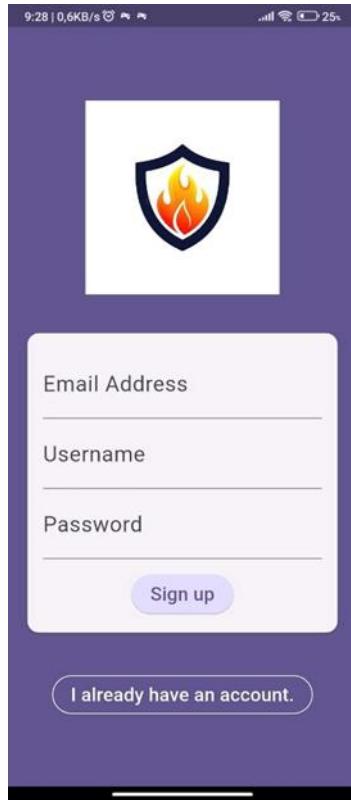
#### *4.6.1.4. Kết quả và đánh giá*

- Giao diện đăng nhập:



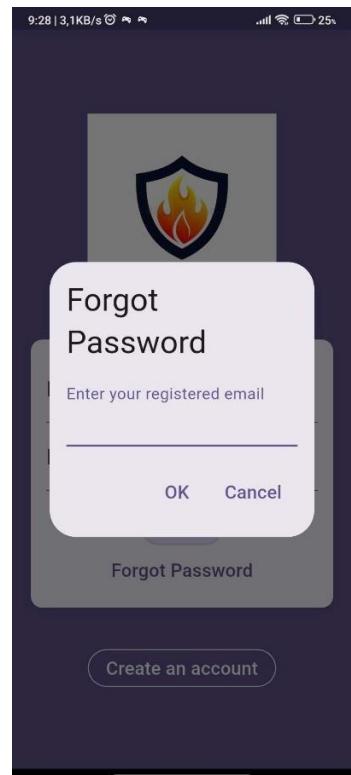
*Hình 4-40: Giao diện đăng nhập*

- Giao diện đăng ký tài khoản:



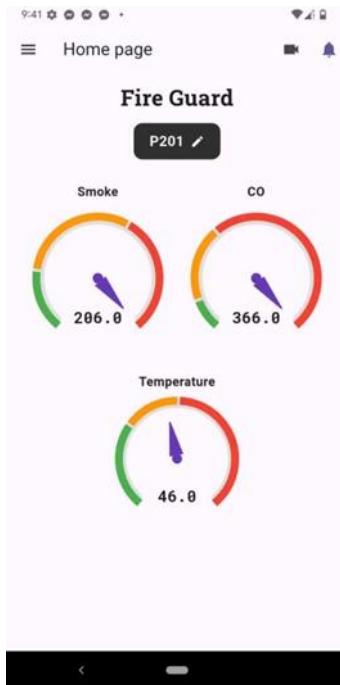
Hình 4-41: Giao diện đăng ký tài khoản

- Giao diện quên mật khẩu:



Hình 4-42: Giao diện quên mật khẩu

- Giao diện chính (xem dữ liệu từ cảm biến):



Hình 4-43: Giao diện màn hình chính

- Giao diện quản lý thông tin tài khoản:

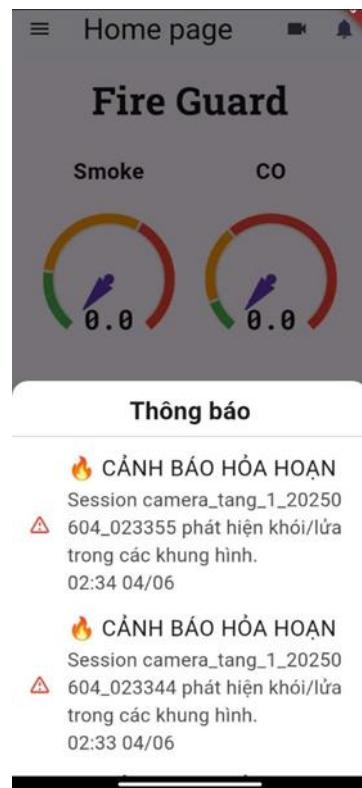
Hình 4-44: Giao diện quản lý thông tin tài khoản

- Giao diện kết nối với cảm biến:



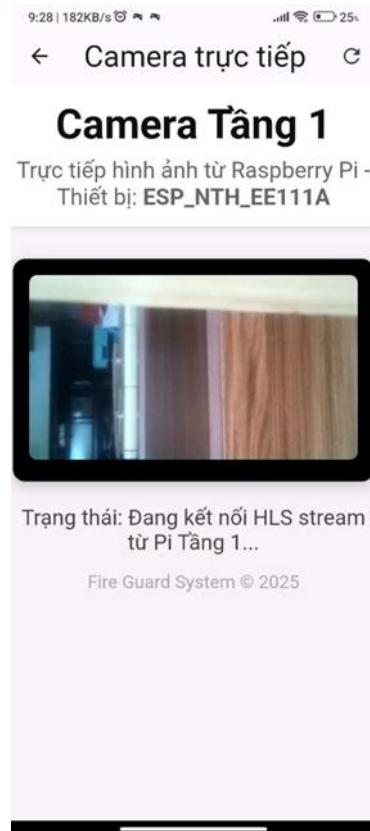
Hình 4-45: Giao diện kết nối với cảm biến

- Giao diện xem cảnh báo từ Server:



Hình 4-46: Giao diện xem cảnh báo

- Giao diện xem video trực tiếp:



Hình 4-47: Giao diện xem video trực tiếp

Hệ thống sử dụng camera gắn với Raspberry Pi 4 để stream video giám sát lên server thông qua giao thức RTMP (để giảm độ trễ), sau đó trên server chuyển đổi thành HLS, phát lại dưới dạng trang HTML tĩnh chứa trình phát video. Ứng dụng di động Flutter chỉ cần mở đường dẫn HTML tương ứng bằng WebView để người dùng có thể xem luồng video theo thời gian thực.

## 4.7 Kết quả thực nghiệm

### 4.7.1 Triển khai thực nghiệm mô hình nhận diện đám cháy

Trong quá trình huấn luyện mô hình, để đánh giá về hiệu quả của mô hình, nhóm em đã tiến hành thử nghiệm trực tiếp trên nhiều dữ liệu đầu vào khác nhau, bao gồm cả ảnh thực tế và video. Vì giới hạn trong báo cáo, nên chúng em xin phép chỉ đưa các ví dụ về hình ảnh qua mô hình và được mô hình nhận diện. Mô hình được huấn luyện để phân biệt giữa hai lớp chính: fire (lửa) và smoke by fire (khói). Khi phát hiện, mô hình sẽ đánh giá xác suất và đánh dấu vùng phát hiện tương ứng với lớp đó, đồng thời hiển thị thông tin trên hình ảnh đầu ra.



Hình 4-48: Nhận diện lửa

Hình 4-43 thể hiện mô hình đã xác định rõ ngọn lửa tại trung tâm ảnh với nhãn "fire" và xác suất 76%, cho thấy khả năng nhận diện lửa trong điều kiện ánh sáng yếu và môi trường phức tạp là khá tốt.



Hình 4-49: Nhận diện khói

Hình 4-44 cho thấy mô hình đã phát hiện một đám khói lớn xuất phát từ phần giữa tòa nhà cao tầng. Khung phát hiện được gán nhãn "smoke by fire" với xác suất 85%, cho thấy mô hình nhận định với độ tin cậy cao rằng đây là khói do hỏa hoạn gây ra.

## 4.7.2 Triển khai thực nghiệm hệ thống

### 4.7.2.1. Kiểm thử chức năng đăng ký tài khoản

Bảng 4-6: Bảng mô tả các bước kiểm thử chức năng Đăng ký tài khoản

STT	Hành động thực hiện	Dữ liệu đầu vào	Kết quả mong đợi
1	Mở ứng dụng và chọn tạo tài khoản	-	Màn hình đăng ký tài khoản hiển thị
2	Nhập email, mật khẩu hợp lệ, tên người dùng hợp lệ, nhấn “Đăng ký”	<u>newuser@gmail.com</u> / 000000 / hieunguyen	Đăng ký thành công, chuyển sang màn hình chính
3	Nhập email đã tồn tại, mật khẩu hợp lệ, nhấn “Đăng ký”	<u>user@gmail.com</u> / 000000	Báo lỗi “email” đã tồn tại
4	Nhập email không hợp lệ	user@ / 000000	Báo lỗi email không hợp lệ
5	Nhập email hợp lệ, mật khẩu không hợp lệ (không đủ độ dài)	<u>newuser1@gmail.com</u> / 00000	Báo lỗi mật khẩu ít nhất phải 6 ký tự
6	Bỏ trống một trong các trường nhập	Email rỗng hoặc Username rỗng hoặc mật khẩu rỗng	Báo lỗi các trường phải đạt độ dài tối thiểu

Bảng 4-7: Bảng kết quả kiểm thử chức năng đăng ký

STT	Dữ liệu kiểm tra	Kết quả thực tế	Kết luận
1	Email mới, mật khẩu hợp lệ, tên người dùng hợp lệ	Tạo tài khoản, chuyển đến màn hình chính	Đạt
2	Email đã tồn tại	Báo lỗi email đã được sử dụng	Đạt
3	Mật khẩu không đạt độ dài tối thiểu	Báo lỗi mật khẩu phải có ít nhất 6 ký tự	Đạt
4	Nhập email không hợp lệ	Báo lỗi email phải đúng định dạng	Đạt
5	Nhập thiếu thông tin	Báo lỗi các trường phải đạt độ dài tối thiểu	Đạt

#### 4.7.2.2. Kiểm thử chức năng đăng nhập người dùng

Bảng 4-8: Bảng mô tả các bước kiểm thử chức năng Đăng nhập

STT	Hành động thực hiện	Dữ liệu đầu vào	Kết quả mong đợi
1	Mở ứng dụng và truy cập màn hình đăng nhập	-	Màn hình đăng nhập hiển thị
2	Nhập <b>email</b> và <b>mật khẩu</b> hợp lệ, nhấn “ <b>Đăng nhập</b> ”	<u>hieunguyen3921@gmail.com</u> / 000000	Đăng nhập thành công, chuyển đến màn hình chính
3	Nhập <b>email</b> sai và <b>mật khẩu</b> đúng, nhấn “ <b>Đăng nhập</b> ”	<u>hiuenguyen3921@gmail.com</u> / 000000	Hiện thông báo sai thông tin xác thực
4	Nhập đúng <b>email</b> và sai <b>mật khẩu</b> , nhấn “ <b>Đăng nhập</b> ”	<u>hieunguyen3921@gmail.com</u> / 000001	Hiện thông báo sai thông tin xác thực
5	Để trống <b>email</b> và <b>mật khẩu</b> , nhấn “ <b>Đăng nhập</b> ”	rõng	Hiện thông báo yêu cầu nhập đầy đủ thông tin

Bảng 4-9: Bảng kết quả kiểm thử chức năng Đăng nhập

STT	Dữ liệu kiểm tra	Kết quả thực tế	Kết luận
1	Email & Mật khẩu đúng	Chuyển đến màn hình chính	Đạt
2	Email sai, mật khẩu đúng	Báo lỗi sai thông tin xác thực	Đạt
3	Email đúng, mật khẩu sai	Báo lỗi sai thông tin xác thực	Đạt
4	Bỏ trống cả hai trường	Báo lỗi, yêu cầu nhập đầy đủ thông tin	Đạt

#### 4.7.2.3. Kiểm thử chức năng quên mật khẩu

Bảng 4-10: Bảng mô tả các bước kiểm thử chức năng Quên mật khẩu

STT	Hành động thực hiện	Dữ liệu đầu vào	Kết quả mong đợi
1	Mở ứng dụng và chọn quên mật khẩu	-	Màn hình quên mật khẩu hiển thị
2	Nhập email hợp lệ, nhấn OK	user@gmail.com	Hiển thị thông báo kiểm tra email để xác nhận yêu cầu đổi mật khẩu
3	Nhập email chưa đăng ký	notregist@gmail.com	Hiển thông báo email chưa đăng ký
4	Nhập email không đúng định dạng	usert@gmail	Hiển thông báo email không hợp lệ
5	Bỏ trống trường email	rõng	Hiển thông báo email không hợp lệ

Bảng 4-11: Bảng kết quả kiểm thử chức năng Quên mật khẩu

STT	Dữ liệu test	Kết quả thực tế	Kết luận
1	Email hợp lệ	Hiển thông báo kiểm tra email để xác nhận yêu cầu đặt lại mật khẩu	Đạt
2	Email chưa đăng ký	Vẫn thông báo kiểm tra email để xác nhận yêu cầu đổi mật khẩu	Chưa đạt
3	Nhập email không đúng định dạng	Báo lỗi, yêu cầu nhập email đúng định dạng	Đạt
4	Bỏ trống trường email	Báo lỗi, yêu cầu nhập email đúng định dạng	Đạt

#### 4.7.2.4. Kiểm thử chức năng theo dõi dữ liệu từ cảm biến

Bảng 4-12: Bảng mô tả các bước kiểm thử chức năng xem dữ liệu từ cảm biến

STT	Tiêu chí	Nội dung
1	Mục tiêu	Kiểm tra khả năng hiển thị chính xác và ổn định các chỉ số cảm biến trên ứng dụng di động
2	Quy trình	Cảm biến gửi dữ liệu định kỳ về server thông qua HTTP. Ứng dụng di động kết nối Server và truy xuất dữ liệu theo device_name và user_id. Thử nghiệm trong các điều kiện thường gặp như mất kết nối, dữ liệu bất thường
3	Kết quả mong đợi	Ứng dụng hiển thị đúng giá trị mới nhất, xử lý tốt tình huống dữ liệu bất thường
4	Thực tế	App hiển thị đúng dữ liệu, hoạt động ổn định. Tuy nhiên khi không có mạng hoặc không có nguồn điện, dữ liệu bất thường thì giao diện vẫn hiển thị mà không báo lỗi, cần cải thiện thêm.

Bảng 4-13: Bảng kết quả kiểm thử chức năng xem dữ liệu từ cảm biến

STT	Tình huống kiểm thử	Kết quả mong đợi	Kết quả thực tế	Đánh giá
1	Mở ứng dụng khi cảm biến hoạt động bình thường	Hiển thị đúng giá trị nhiệt độ, CO, khói theo thời gian thực	App hiển thị đúng, cập nhật liên tục	Đạt
2	Cảm biến bị tắt nguồn đột ngột	App hiện về số đo về 0	App hiển thị số đo về 0	Đạt
3	Thiết bị di động bị mất kết nối Internet	App không cập nhật dữ liệu mới, hiển thị thông báo lỗi	App hiển thị thông báo kết nối lỗi đến cảm biến	Đạt
4	Đổi tên thiết bị để phù hợp với yêu cầu sử dụng	App vẫn hiển thị đúng dữ liệu ứng với cảm biến sau khi đổi tên	App vẫn hiển thị dữ liệu tương ứng với cảm biến	Đạt
5	Cảm biến đo được dữ liệu bất thường	App bỏ qua, không hiển thị giá trị sai lệch hoặc vô lý	App vẫn hiển thị số bất thường	Cần cải thiện

#### 4.7.2.5. Kiểm thử chức năng xem camera giám sát

Bảng 4-14: Bảng mô tả các bước kiểm thử chức năng xem camera giám sát

Tiêu chí	Nội dung
Mục tiêu	Kiểm tra khả năng truy cập, tải và hiển thị luồng video giám sát trực tiếp từ camera Pi trên ứng dụng di động
Quy trình	Truy cập màn hình xem camera trong ứng dụng, đảm bảo kết nối Internet ổn định. Server cần đang phát stream RTMP chuyển đổi HLS. Kiểm tra khả năng tải video, độ trễ và tính liên tục của luồng.
Kết quả mong đợi	Video giám sát hiển thị được sau thời gian chờ hợp lí ( $\leq 1s$ ). Không đứng hình, không mất tín hiệu khi mạng ổn định. Có hiển thị lỗi nếu không truy cập được.
Thực tế	Video stream hoạt động bình thường khi server đang phát. Trên mạng yếu có hiện tượng delay, đơ hình, sau khi mạng kết nối ổn định có thể tua tiến để khắc phục delay. App không thông báo lỗi khi lỗi kết nối.

Bảng 4-15: Bảng kết quả kiểm thử chức năng xem camera giám sát

STT	Tình huống kiểm thử	Kết quả thực tế	Đánh giá
1	Truy cập camera khi server đang phát	Video hiện gần như ngay lập tức ( $\leq 1s$ )	Đạt
2	Server không phát stream	Màn hình hiển thị đang load video, không có thông báo cụ thể	Cần cải thiện UX
3	Kết nối Internet yếu	Video đôi khi đơ, đứng hình	Cần tối ưu stream
4	Mở App khi chưa kết nối Internet	Hiển thị đang load video, App không bị crash	Chấp nhận được
5	Mở stream nhiều lần liên tiếp	App hoạt động ổn định, không bị đơ hay crash	Đạt

#### 4.7.2.6. Kiểm thử còi báo khi các chỉ số vượt ngưỡng

Bảng 4-16: Bảng mô tả các bước kiểm thử chức năng còi báo

Tiêu chí	Nội dung
Mục tiêu	Đảm bảo hệ thống còi chỉ cảnh báo một lần khi vượt ngưỡng và chỉ báo lại nếu người dùng đã xác nhận tắt còi và giá trị cảm biến trở về bình thường.
Quy trình	Giả lập vượt ngưỡng để còi hú, sau đó người dùng nhấn nút để tắt còi. Giữ giá trị cảm biến cao để kiểm tra xem còi có hú lại không. Sau đó đưa giá trị về bình thường để reset trạng thái còi. Cuối cùng lại cho các chỉ số vượt ngưỡng để kiểm tra còi có hoạt động lại không.
Kết quả mong đợi	Còi hoạt động theo đúng logic: Chỉ hú lại sau khi người dùng đã nhấn còi và giá trị cảm biến trở lại bình thường. Tránh cảnh báo lặp lại cho tình huống mà người dùng đã xử lý.
Thực tế	Hệ thống đáp ứng đúng logic thiết kế. Còi không lặp lại khi giá trị vẫn vượt ngưỡng sau khi người dùng đã nhấn nút. Khi cảm biến về ngưỡng an toàn, còi được reset và sẵn sàng hoạt động cho lần vượt ngưỡng mới.

Bảng 4-17: Bảng kết quả kiểm thử chức năng còi báo

STT	Tình huống kiểm thử	Kết quả thực tế	Đánh giá
1	Cho 1 trong các chỉ số vượt ngưỡng	Còi hú liên tục	Đạt
2	Người dùng nhấn nút tắt còi khi còi đang hú	Còi dừng hú	Đạt
3	Giá trị vẫn vượt ngưỡng sau khi nhấn nút	Còi không hú lại, hệ thống ghi nhận tình huống đã được xử lý	Đạt
4	Giá trị trở lại bình thường sau khi nhấn nút	Hệ thống reset trạng thái, chuẩn bị cho lần cảnh báo tiếp theo	Đạt
5	Sau khi reset, cho các chỉ số vượt ngưỡng	Còi hú liên tục	Đạt
6	Không nhấn nút, chỉ số dưới ngưỡng	Còi vẫn kêu liên tục cho đến khi có nhấn nút hoặc hết vượt ngưỡng	Đạt

#### 4.7.2.7. Kiểm thử toàn hệ thống

**Detection Time (DT):** Thời gian từ khi có nguồn cháy (lửa hoặc khói) xuất hiện trong khung hình camera đến thời điểm hệ thống nhận diện và ghi log phát hiện đầu tiên.

- Ý nghĩa, vai trò:
  - DT càng nhỏ thì hệ thống càng nhanh chóng nhận diện cháy, tăng cơ hội sơ tán, xử lý sớm, giảm thiệt hại.
  - Là chỉ số quan trọng, đánh giá khả năng phản ứng thời gian thực của hệ thống.
- Phương pháp đo thực nghiệm:
  - Khởi động hệ thống, sử dụng đồng hồ bấm giờ để đo.
  - Đưa nguồn cháy vào camera tại thời điểm xác định.
  - Ghi lại thời điểm xuất hiện log nhận diện được lửa hoặc khói.
  - DT = thời điểm log nhận diện – thời điểm đưa nguồn cháy vào khung hình.
- Kết quả thực nghiệm:
  - Khi ngọn lửa rõ ràng, đủ lớn và ở gần camera (khoảng 20 – 50cm), hệ thống detect ngay lập tức, log xuất hiện gần như đồng thời với thời điểm ngọn lửa lọt và khung hình.
  - Trong các thử nghiệm thành công, độ trễ giữa thời điểm nguồn cháy xuất hiện và thời điểm xuất hiện log chỉ dao động rất ít, khó nhận biết bằng mắt thường.
  - VỚI video cháy rõ nét trên điện thoại, hệ thống cũng nhận diện ngay lập tức nếu khoảng cách không quá xa (dưới 3m).
- Kết luận: Detection Time < 0.5 giây trong các điều kiện phát hiện thành công.
- Giới hạn của hệ thống:
  - Nếu ngọn lửa nhỏ, quá xa hoặc hình ảnh cháy không đủ rõ, hệ thống không phát hiện được.
  - Khả năng phản ứng nhanh chỉ đạt được trong phạm vi quan sát hiệu quả của camera và model AI.
  - Camera được sử dụng trong hệ thống là OV5647, độ phân giải 5 megapixel, đặt ở chế độ quay video với độ phân giải  $640 \times 360$ . Khi ngọn lửa hoặc vùng cháy ở quá xa, kích thước vùng cháy trong khung hình trở nên quá nhỏ, không đủ chi tiết để AI nhận diện.
- Giải pháp để xuất:
  - Tăng độ phân giải camera.
  - Dùng camera có góc nhìn rộng hơn.
  - Tối ưu vị trí lắp camera để tăng kích thước vùng cháy trong khung hình.
  - Sử dụng thuật toán xử lý làm nét hoặc phóng to vùng nghi ngờ.

**True Positive Rate (TPR):** Là tỷ số giữa số lần hệ thống phát hiện đúng các tình huống có cháy trong tổng số các tình huống cháy được tạo ra trong quá trình thử nghiệm.

$$TPR = \frac{\text{số lần phát hiện đúng}}{\text{tổng số lần tạo ra cháy thật}} \times 100\%$$

- Ý nghĩa và vai trò:
  - Chỉ số thể hiện khả năng bao phủ và độ tin cậy của hệ thống trong việc phát hiện sự cố thật.
  - TPR càng cao thì hệ thống càng đáng tin cậy – đặc biệt quan trọng trong phòng cháy chữa cháy.
  - Là tiêu chí bắt buộc trong mọi hệ thống cảnh báo an toàn.
- Kết quả thực nghiệm:
 

Hệ thống được thử nghiệm với tổng cộng 40 tình huống, gồm:

  - 20 lần bật lửa ở khoảng cách 20 – 70cm.
  - 20 lần phát video cháy ở khoảng cách 0.2 – 3.5m.

Bảng 4-18: Bảng kết quả thực nghiệm TPR

Tình huống	Số lần thử	Phát hiện đúng
Bật lửa ở gần ( $\leq 50$ cm)	16	15
Bật lửa ở xa (~70 cm)	4	1
Video cháy rõ, ở gần ( $\leq 3$ m)	16	14
Video cháy ở xa (3.5m)	4	0
<b>Tổng cộng</b>	<b>40</b>	<b>30</b>

$$\Rightarrow TPR = 30 / 40 = 75\%.$$

- Phân tích kết quả:
  - Hệ thống hoạt động rất tốt trong các tình huống có hình ảnh cháy rõ ràng và ở gần camera.
  - Hệ thống bỏ sót nhiều nguồn cháy thật ở xa.
  - Nguyên nhân chủ yếu là do giới hạn độ phân giải camera và model AI chưa đủ nhạy với các đặc điểm nhỏ, mờ trong ảnh.
- Giải pháp đề xuất:
  - Huấn luyện lại mô hình với nhiều mẫu ảnh ở khoảng cách xa hơn, ngọn lửa nhỏ hơn.
  - Sử dụng camera có độ phân giải cao hơn hoặc điều chỉnh vị trí lắp đặt để tăng kích thước vùng cháy trong khung hình.

**Thời gian hoạt động liên tục (Uptime / Availability):** Uptime là tổng thời gian hệ thống thực sự hoạt động ổn định, không bị treo, lỗi hoặc phải khởi động lại. Availability là tỷ lệ phần trăm thời gian hệ thống sẵn sàng hoạt động trên tổng thời gian thử nghiệm.

$$Availability = \frac{Uptime}{Uptime + Downtime} \times 100\%$$

- Ý nghĩa và vai trò:
  - Là chỉ số cốt lõi để đánh giá độ tin cậy của phần cứng (Raspberry Pi 4), AI và toàn bộ pipeline xử lý.
  - Hệ thống cảnh báo cháy bắt buộc phải hoạt động liên tục 24/7. Chỉ cần mất tín hiệu vài phút cũng có thể bỏ sót đám cháy.
  - Availability thấp đồng nghĩa với rủi ro cảnh báo, gây gián đoạn cảnh báo qua app.
- Kết quả thực nghiệm:
  - Hệ thống được thử nghiệm trong 4 phiên riêng biệt, mỗi phiên kéo dài 2 giờ hoạt động liên tục trước khi Raspberry Pi 4 bị treo và yêu cầu khởi động lại thủ công.
  - Tổng cộng có 8 giờ Uptime trên Raspberry Pi 4.
  - Thời gian vận hành mục tiêu: 24 giờ liên tục.

⇒ Nếu tính theo tổng thời gian cộng dồn:

$$Availability_{cộng\ dồn} = \frac{8}{24} \times 100\% = 33.3\%$$

Tuy nhiên trong thực tế, nếu không có người can thiệp để reset thiết bị, hệ thống chỉ có thể vận hành liên tục khoảng 2 giờ trước khi ngừng hoạt động.

⇒ Availability thực tế để đáp ứng yêu cầu vận hành 24/7 là:

$$Availability_{thực\ tế} = \frac{2}{24} \times 100\% = 8.3\%$$

- Phân tích kết quả:
  - Raspberry Pi 4 phải xử lý đồng thời mô hình AI và truyền AI, dẫn đến quá tải tài nguyên (CPU, RAM).
  - Không có watchdog phần mềm để tự reset nếu bị treo.
  - Chưa có thiết bị tản nhiệt phù hợp.
  - Thiếu cơ chế theo dõi hiệu năng hoặc khôi phục tự động sau lỗi.
  - Dù uptime cộng dồn đạt 8 giờ nhưng  $Availability_{thực\ tế}$  chỉ đạt khoảng 8.3%, cho thấy hệ thống chưa đủ ổn định để triển khai liên tục trong môi trường thật. Việc treo định kỳ và không thể tự động phục hồi là rào cản lớn cần khắc phục trước khi triển khai rộng rãi.
- Giải pháp đề xuất:
  - Tách riêng pipeline AI và stream video: Raspberry Pi 4 chỉ stream, AI detect chạy trên thiết bị khác mạnh hơn (Jetson Nano, ...).
  - Tối ưu pipeline: Giảm độ phân giải đầu vào, giảm tần suất detect.

- Theo dõi hiệu năng hệ thống theo thời gian thực: log CPU/RAM, cảnh báo nhiệt độ, ghi lại thời điểm treo để phân tích thêm.

**False Alarm Rate (FAR) & False Positive Rate (FPR):** Tỷ lệ báo động giả. FAR là số lần cảnh báo sai (không có cháy) trên mỗi phiên hoạt động (tối đa 2 giờ 15 phút / Pi). FPR là tỷ lệ số lần báo nhầm trong tổng số tình huống không có cháy được thử nghiệm.

$$FPR = \frac{\text{số lần báo động giả}}{\text{tổng số tình huống không cháy được tạo ra}} \times 100\%$$

- Ý nghĩa và vai trò:
  - Báo động giả gây mất niềm tin của người dùng, làm phiền người sử dụng, dễ khiến bỏ qua cảnh báo thật (hiệu ứng “còi báo sói”).
  - FAR / FPR thấp là yêu cầu bắt buộc đối với hệ thống cảnh báo thực tế, đặc biệt là ở môi trường dân cư có nhiều nguồn gây nhiễu như nấu ăn, tắm nóng, khói nhang, ánh nắng.
- Kết quả thực nghiệm: Hệ thống được thử nghiệm với 30 tình huống không cháy, phân bổ trong 4 phiên làm việc (mỗi phiên 2 giờ), mô phỏng các tình huống dễ gây nhiễu trong thực tế.

Bảng 4-19: Bảng kết quả thực nghiệm FAR & FPR

Tình huống không cháy	Số lần thử	Số lần báo nhầm
Khói do nấu ăn (rán, đun nước)	8	2
Khói do tháp nhang	6	0
Hơi nước từ phòng tắm (sau khi tắm)	6	1
Vật có ánh sáng mạnh, chuyển động nhanh	10	3
Tổng cộng	30	6

$$\Rightarrow FPR = 6 / 20 = 20\%$$

$$\Rightarrow FAR = 1.5 \text{ cảnh báo giả / phiên (2 giờ hoạt động)}.$$

- Phân tích kết quả:
  - Cảnh báo giả chủ yếu xuất hiện ở tình huống vật thể có ánh sáng mạnh, cho thấy mô hình AI còn nhầm lẫn giữa nguồn cháy với vật thể có màu sáng giống với nguồn cháy.
  - Một số cảnh báo nhầm đến từ khói và hơi nước (những tình huống sinh ra khói nhiều, tập trung tại một điểm), cho thấy khả năng phân biệt khói thật – khói sinh hoạt chưa tốt. Nguyên nhân chủ yếu đến từ

đặc điểm vật lý và thị giác của khói: khói thường mờ, loãng, có hình dạng và màu sắc biến đổi liên tục, dễ bị hòa lẫn với nền trời, mây, sương mù, hoặc các yếu tố môi trường khác.

- Thời lượng hoạt động trên mỗi phiên giới hạn trong 2 giờ do thiết bị sẽ bị treo nếu chạy lâu hơn.
- Với FAR = 1.5 lần / phiên, hệ thống chưa đủ ổn định để triển khai độc lập nếu không cải thiện thêm.
- Giải pháp đề xuất:
  - Huấn luyện lại mô hình với ảnh mẫu chứa các loại khói, ánh sáng nhiều.
  - Cải thiện pipeline và thiết bị để kéo dài phiên hoạt động.

**Khả năng chịu lỗi thành phần (Component Failure Tolerance):** Là khả năng hệ thống tiếp tục vận hành một phần hoặc toàn bộ, ngay cả khi có một số thành phần gặp lỗi – ví dụ như một cảm biến ngừng hoạt động, mất kết nối internet tạm thời hoặc server không phản hồi.

- Ý nghĩa và vai trò:
  - Đảm bảo hệ thống không bị tê liệt toàn bộ chỉ vì một phần bị lỗi.
  - Trong mô hình phân tán như chung cư mini – mỗi thiết bị, mỗi phòng là một đơn vị độc lập, tính cục bộ hóa cao nên khả năng chịu lỗi là cực kỳ quan trọng.
  - Đặc biệt quan trọng với các sự cố như mất mạng, gián đoạn internet, phần cứng hỏng cục bộ.
- Kết quả thực nghiệm và đánh giá thực tế:

Bảng 4-20: Bảng kết quả thực nghiệm khả năng chịu lỗi

Tình huống mô phỏng	Phản ứng hệ thống
Một Raspberry Pi hoặc một cảm biến bị hỏng	Chỉ riêng thiết bị đó ngừng hoạt động. Các node khác vẫn hoạt động bình thường. Hệ thống không bị ảnh hưởng
Mất điện tại thiết bị (không có pin dự phòng)	Thiết bị tắt hoàn toàn. Chỉ tiếp tục hoạt động với thiết bị nào có điện. Không thể duy trì hoạt động nếu không có nguồn điện liên tục
Mất kết nối Internet tại thiết bị	Thiết bị vẫn hoạt động cục bộ nhưng không gửi được dữ liệu lên server, dẫn đến bỏ sót cảnh báo
Server bị mất kết nối	Các thiết bị vẫn hoạt động cục bộ, nhưng toàn bộ dữ liệu bị mất do không có cơ chế lưu tạm và gửi lại sau khi phục hồi.
Mất Internet toàn mạng	Hệ thống chỉ có thể cảnh báo nội bộ bằng còi ở cảm biến chứ không cảnh báo đèn app. Mất toàn bộ khả năng nhận diện bằng AI & thông báo nếu không có kết nối mạng.

- Phân tích kết quả:
  - Mỗi thiết bị cảm biến và Raspberry Pi hoạt động như một đầu báo cháy độc lập, khi một node hỏng thì chỉ node đó bị ảnh hưởng, các thiết bị khác không bị gián đoạn.
  - Tuy nhiên hệ thống phụ thuộc hoàn toàn vào Internet, không có pin dự phòng, không có lưu trữ tạm thời khi mất mạng và không có cơ chế tự phục hồi hoặc gửi lại dữ liệu.
  - Hệ thống hiện tại chịu lỗi tốt ở mức phần cứng, chưa có cơ chế phục hồi lỗi mạng, mất kết nối hoặc gián đoạn Server.
- Giải pháp đề xuất:
  - Tích hợp cơ chế lưu đệm cục bộ: Nếu không gửi được ảnh hoặc dữ liệu lên server cần lưu lại tạm thời và tự động gửi lại sau khi mạng hồi phục.
  - Giám sát trạng thái kết nối: Thiết bị gửi “heartbeat” định kỳ để server phát hiện mất kết nối (qua đó thông báo để người dùng can thiệp kịp thời).
  - Cân nhắc bổ sung pin dự phòng cho Raspberry Pi 4 và cảm biến nếu triển khai ở khu vực dễ mất điện.
  - Hiển thị trạng thái hoạt động và kết nối để người dùng biết node nào đang lỗi.

**Thời gian lắp đặt & Cấu hình trung bình:** Là thời gian cần thiết để lắp đặt vật lý một cảm biến, Raspberry có gắn cam và cấu hình thiết bị để kết nối với Wi-fi và tích hợp vào hệ thống trung tâm (gửi dữ liệu lên server, nhận diện).

- Ý nghĩa và vai trò:
  - Lắp đặt nhanh, gọn và không cần kỹ thuật chuyên sâu là tiêu chí quan trọng trong môi trường nhà ở dân dụng – nơi có hạn chế về không gian, thẩm mỹ và thời gian thi công.
  - Thời gian lắp đặt càng ngắn thì chi phí triển khai càng thấp, khả năng hệ thống càng cao.

- Thực tế triển khai:

Hệ thống gồm 2 loại node:

- Cảm biến môi trường (nhiệt độ, khói, CO):

Lắp đặt vật lý đơn giản: Chỉ cần lấy hộp thiết bị, đặt tại vị trí mong muốn. Không cần khoan đục, đi dây.

Yêu cầu: Có ổ cắm điện gần đó và Wi-fi phủ sóng đủ mạnh.

Cấu hình ban đầu (kết nối Wi-fi): Có thể thực hiện qua app cực kỳ nhanh gọn.

- Raspberry Pi 4 và Camera (Phát hiện khói / lửa bằng AI):

Vị trí linh hoạt, đặt tại vị trí có tầm nhìn camera rộng, miễn sao có ổ cắm điện và Wi-fi phủ tới (tốt nhất dùng mạng LAN để ổn định và kết nối Internet dễ dàng, nếu kết nối Wi-fi cần người có chuyên môn).

Gắn camera vào Raspberry Pi qua CSI, khởi động Raspberry Pi, camera sẽ được khởi động.

Không yêu cầu lắp đặt cố định hay kỹ thuật chuyên sâu.

- Kết luận:

- Việc lắp đặt và cấu hình hệ thống hiện tại là đơn giản, ít xâm lấn, phù hợp cho triển khai nhanh với quy mô lớn trong các không gian dân dụng.
- Các node hoạt động độc lập, không cần dây mạng nếu Wi-fi ổn định, giúp tối ưu thời gian và chi phí nhân công.

**Khả năng tích hợp & Mở rộng (Integration & Scalability):** Là khả năng thêm cảm biến hoặc node mới vào hệ thống mà không cần thay đổi kiến trúc tổng thể. Kết nối hệ thống hiện tại với hệ thống khác (như hệ thống chữa cháy, cửa thoát hiểm tự động, hoặc các nền tảng cảnh báo bên ngoài như SMS, app, ...).

- Ý nghĩa và vai trò:

- Một hệ thống báo cháy hiện tại cần khả năng thích nghi với thay đổi bộ cục phòng cháy, mở rộng quy mô, hoặc kết nối với các giải pháp khác để nâng cao hiệu quả phản ứng.
- Khả năng tích hợp tốt giúp hệ thống không bị lỗi khu vực cần tăng lên.

- Thực tế hệ thống hiện tại:

- Mở rộng hệ thống:

Rất dễ dàng: Mỗi node hoạt động độc lập và kết nối trực tiếp tới Internet và Server.

Không cần thay đổi cấu trúc hay lắp đặt dây dẫn.

Chỉ cần kết nối Wi-fi, cấu hình tên thiết bị là node mới sẽ tự động được hệ thống nhận diện và lưu dữ liệu riêng biệt.

⇒ Không có giới hạn rõ ràng về số lượng node, miễn là server chịu tải được.

- Tích hợp với hệ thống khác:

Hệ thống đã hỗ trợ cảnh báo qua Firebase Cloud Messaging (FCM) tới app di động.

Không có kết nối vật lý với các thiết bị như máy bơm nước, cửa thoát hiểm, ...

Tuy nhiên kiến trúc phần mềm mở, nếu cần hoàn toàn có thể thêm chức năng gửi SMS, gửi tín hiệu ra các thiết bị điều khiển khác.

- Kết luận:
  - Hệ thống có khả năng mở rộng rất linh hoạt về mặt thiết bị đầu cuối: Chỉ cần cắm điện và kết nối mạng.
  - Không cần thay đổi hệ thống hiện có, ít bị ràng buộc về sơ đồ dây hay vị trí lắp đặt.
  - Tuy chưa hỗ trợ điều khiển thiết bị ngoại vi, nhưng có tiềm năng mở rộng dễ dàng nhờ sử dụng phần mềm tùy biến, server riêng, ...
- Giải pháp đề xuất:
  - Thêm chức năng xử lý khi có cháy như điều khiển máy bơm nước, mở cửa tự động, ...
  - Tích hợp gửi SMS cho người dùng không có app.

#### 4.7.2.8. Kiểm thử hiệu năng phần cứng

Mục đích kiểm thử phần thử để đánh giá khả năng xử lý liên tục của Raspberry Pi 4 khi thực hiện các tác vụ nặng như stream video kết hợp nhận diện khói/lửa, từ đó kiểm chứng hiệu quả tính toán, quản lý tài nguyên hệ thống, cũng như mức độ ổn định khi hoạt động dài hạn trong môi trường thực tế.



Hình 4-51: Hình ảnh thực tế phần detect



Hình 4-50: Hình ảnh thực tế bộ cảm biến

#### Sử dụng CPU:

- Khi không chạy tác vụ, các core hoạt động ở mức rất thấp (~1 – 5%).

- Khi chạy nhận diện khói/lửa liên tục kết hợp stream video, từng core dao động trong khoảng 65 – 74%. Tuy nhiên tổng CPU toàn hệ thống thường dưới 70% nhờ hệ điều hành phân phối đều tải trên cả 4 core.
  - ⇒ Việc sử dụng CPU như vậy là hợp lý và cho thấy hệ thống vẫn còn dư tải để xử lý các tác vụ khác nếu cần thiết.
  - ⇒ Hiệu suất xử lý đạt mức ổn định, tận dụng tốt khả năng tính toán mà không gây quá tải cho CPU.

### Sử dụng bộ nhớ:

- Khi không tải, bộ nhớ RAM chiếm khoảng 315 – 396MB.
- Khi chạy tác vụ nhận diện khói lửa và stream video, bộ nhớ tăng lên khoảng 783 – 800MB.
  - ⇒ Với tổng dung lượng RAM 4GB, mức sử dụng này là khá nhỏ, cho thấy mô hình chạy nhẹ, phù hợp với thiết bị nhúng.
  - ⇒ Hệ thống có độ ổn định cao, dư bộ nhớ cho các thành phần khác (lưu log, giao tiếp mạng, ...).

### Nhiệt độ hệ thống:

- Trạng thái không tải: Nhiệt độ CPU ổn định ở mức 40.1 – 41.8°C.
- Khi chạy tác vụ liên tục trong thời gian dài, nhiệt độ tăng dần, dao động ở mức 76.4 – 79.8°C và đạt mức đỉnh 80.3°C.
  - ⇒ Nhiệt độ tuy cao nhưng vẫn nằm dưới ngưỡng throttling (85°C) của Raspberry Pi 4, nên chưa gây sụt hiệu năng quá nhiều.
  - ⇒ Hệ thống có thể vận hành liên tục trong môi trường thực tế, tuy nhiên cần chú trọng đến thiết bị tản nhiệt.

### Hiệu suất xử lý hình ảnh (FPS):

- FPS ban đầu khi mới chạy mô hình dao động từ 0.2 – 0.4 FPS. Sau vài giây ổn định ở mức 1.5 – 1.6 FPS.
  - ⇒ Đây là tốc độ phù hợp với yêu cầu nhận diện khói/lửa theo thời gian thực trong ứng dụng cảnh báo.
  - ⇒ Tốc độ xử lý đủ nhanh để phát hiện sớm sự cố, đồng thời giúp tiết kiệm tài nguyên và tránh quá tải.

### Công suất tiêu thụ:

Để đánh giá mức tiêu thụ năng lượng của các thành phần chính trong hệ thống, nhóm tiến hành đo điện áp, dòng điện và công suất thực tế bằng thiết bị đo USB ở hai điểm:

- Bộ cảm biến:



Hình 4-52: Kết quả đo công suất tiêu thụ của bộ cảm biến

- Thiết bị đo: Nguồn cấp USB qua mạch chuyển đổi.
- Thông số đo được:
  - Điện áp: 4.855 V.
  - Dòng điện: 0.179 A.
  - Công suất tiêu thụ: 0.869 W.

⇒ Mức công suất nhỏ, phù hợp với hoạt động liên tục của cảm biến như MQ-7, MP-2, DHT11. Thiết bị đo cho thấy điện áp và dòng điện được duy trì ổn định trong quá trình hoạt động, không có dao động lớn.

- Raspberry Pi 4 kết nối camera:



Hình 4-53: Kết quả đo công suất tiêu thụ của Raspberry Pi 4 kết nối camera

- Thiết bị đo: Nguồn cấp USB-C chuẩn 5V-3A.
  - Thông số đo được:
    - Điện áp: 5.007 V.
    - Dòng điện: 0.959 A.
    - Công suất tiêu thụ: 4.801 W.
- ⇒ Khi Raspberry Pi 4 vận hành với camera và xử lý video (stream + AI detect), mức tiêu thụ điện tăng rõ rệt. Giá trị gần 5W là phù hợp với công suất trung bình của Pi 4 khi hoạt động nặng. Nguồn cấp cần đảm bảo ổn định  $\geq 5V\ 3A$  để hệ thống có thể hoạt động ổn định.

## CHƯƠNG 5. KẾT LUẬN

### 5.1 Kết luận

Theo đà phát triển của thế giới, ứng dụng IoT và AI đang trở thành thứ không thể thiếu trong cuộc cách mạng công nghiệp 4.0. Qua đề tài này, nhóm em đã đạt được một số kết quả đáng kể.

- **Xây dựng mô hình AI nhận diện được lửa và khói:** Phát triển một mô hình trí tuệ nhân tạo nhận diện lửa và khói từ hình ảnh, giúp sớm phát hiện ra các tình huống nguy hiểm để có biện pháp ứng phó kịp thời.
- **Xây dựng ứng dụng di động:** Phát triển một ứng dụng di động chạy trên hệ điều hành Android, thân thiện với người dùng, giúp người dùng nắm bắt được tình hình từ xa.
- **Thiết kế giao tiếp giữa vi điều khiển và các thiết bị ngoại vi:** Đảm bảo sự tương tác chính xác và hiệu quả giữa các thiết bị trong hệ thống.
- **Tiến hành thử nghiệm và đánh giá kết quả:** Thực hiện các thử nghiệm để kiểm tra và đánh giá hiệu suất của hệ thống.

Các chức năng chính đã được kiểm thử riêng biệt cũng như toàn hệ thống.  
Kết quả cho thấy:

- Hệ thống hoạt động ổn định trong từng phiên hoạt động.
- Giá trị cảm biến hiển thị chính xác trên app, phản ánh đúng trạng thái môi trường.
- Còi hoạt động theo đúng logic mong muốn, tránh cảnh báo lặp hoặc quá nhạy.
- App hiển thị luồng video camera tốt trong điều kiện mạng ổn định.

Tuy nhiên, hệ thống vẫn còn tồn tại một số hạn chế:

- Thời gian hoạt động liên tục của thiết bị chưa cao (khoảng 2 giờ mỗi phiên), do thiết bị xử lý đồng thời tác vụ nặng (AI detect và stream).
- Một số trường hợp kết nối lỗi chưa được xử lý tối ưu (không hiển thị lỗi rõ ràng).
- App chưa có chức năng cảnh báo chủ động khi server không phản hồi hoặc khi cảm biến ngắt kết nối.

### 5.2 Hướng phát triển của đồ án trong tương lai

Mặc dù hệ thống đã cơ bản hoàn thiện các chức năng chính như cảnh báo thời gian thực, xem dữ liệu cảm biến, và giám sát video, tuy nhiên vẫn còn nhiều tiềm năng để mở rộng và nâng cấp nhằm tăng tính ứng dụng và độ tin cậy. Một số hướng phát triển cụ thể trong tương lai:

- **Tích hợp AI và phát hiện bất thường:** Thay vì chỉ cảnh báo khi vượt ngưỡng cảm biến, có thể tích hợp mô hình học máy (Machine Learning) để phát hiện bất thường dựa trên hành vi cảm biến, từ đó giảm báo động giả và tăng độ chính xác.
- **Cải thiện độ trễ và hiệu suất video giám sát:** Tối ưu hóa luồng truyền video để hỗ trợ xem video gần như thời gian thực, ngay cả trong điều kiện mạng yếu.
- **Bổ sung cơ chế phát hiện và xử lý lỗi hệ thống:** Thêm watchdog để tự động reset thiết bị khi bị treo, lưu trạng thái còi cảnh báo vào bộ nhớ không mất điện (NVS) để phục hồi chính xác sau khi mất nguồn.
- **Tự động lưu trữ và thông kê dữ liệu:** Phát triển hệ thống báo cáo theo ngày/tuần/tháng, kết hợp biểu đồ thống kê để người dùng dễ theo dõi lịch sử và xu hướng môi trường.
- **Hỗ trợ nhiều hình thức thông báo nâng cao:** Ngoài FCM, có thể tích hợp thêm thông báo qua email, cuộc gọi tự động hoặc đồng bộ với hệ thống loa cảnh báo cục bộ.

Những hướng phát triển trên không chỉ giúp cho hệ thống hoạt động ổn định và thông minh hơn, mà còn mở rộng khả năng ứng dụng trong các lĩnh vực như: "Nhà thông minh, giám sát cháy nổ, an toàn môi trường,...".

## TÀI LIỆU THAM KHẢO

- [1] PCCC 3S. (n.d.). Nhũng vụ cháy kinh hoàng nhất trong nhũng năm gần đây. PCCC3S.vn.<https://pccc3s.vn/nhung-vu-chay-kinh-hoang-nhat-trong-nhungnam-gan-day/>
- [2] Flutter Team, *Flutter Documentation: Build apps for any screen*, Google, <https://docs.flutter.dev>
- [3] R. P. Foundation, "Raspberry Pi Documentation," [Online]. Available: <https://www.raspberrypi.com/documentation/>.
- [4] Z. C. M. L. A. J. Aston Zhang, DIve into Deep Learning, 2021.
- [5] Microsoft, "Add citations in a Word document," 2017.
- [6] N. T. Tuân, Deep Learning cơ bản, 2020.
- [7] [Online]. Available: <https://docs.ultralytics.com/vi/>.
- [8] [Online]. Available: <https://www.phamduytung.com/blog/2019-12-13-nms/>.
- [9] D. M.-E. Juan R.Terven, *A COMPREHENSIVE REVIEW OF YOLO ARCHITECTURES IN COMPUTER VISION: FROM YOLOV1 TO YOLOV8 AND YOLO-NAS*, 2024.
- [10] X. L. a. Y. Y. Z. Zhang, *Scaling yolov8 for high-precision real-time object detection*, 2024.
- [11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov8: You only look once, but improved. CVPR 2023, 2023
- [12] Datasheet Sensor MP-2 : <https://www.winsen-sensor.com/d/files/MP-2.pdf>
- [13] Datasheet Sensor MQ-7: <https://cdn.sparkfun.com/assets/b/b/b/3/4/MQ-7.pdf>
- [15] ESP32 with DHT11/DHT22 Temperature and Humidity Sensor using Arduino IDE, Random Nerd Tutorials.  
<https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/>

## PHỤ LỤC

### A1. Bảng phân chia công việc

Công việc	Mô tả chi tiết	Người thực hiện
Phân tích yêu cầu	Khảo sát hệ thống hiện có, xác định yêu cầu người dùng và hệ thống	Nguyễn Thạc Hiếu, Nguyễn Thị Tuyết Trinh, Doãn Đăng Nhật
Thiết kế hệ thống	Vẽ sơ đồ khối, phân tích chức năng, đề xuất giải pháp	Nguyễn Thạc Hiếu
Thiết kế phần cứng	Chọn linh kiện, thiết kế mạch trên Altium	Doãn Đăng Nhật
Hiệu chỉnh cảm biến	Hiệu chỉnh các cảm biến. Đo mẫu nhiều lần, so sánh, điều chỉnh hệ số trong code	Doãn Đăng Nhật
Lập trình ESP32	Viết chương trình đọc dữ liệu cảm biến, cảnh báo, gửi dữ liệu lên server	Nguyễn Thạc Hiếu, Doãn Đăng Nhật
Huấn luyện mô hình AI	Thu thập dữ liệu, gán nhãn ảnh, huấn luyện mô hình YOLOv8 phát hiện khói/lửa	Nguyễn Thị Tuyết Trinh
Triển khai chương trình lên Raspberry Pi 4	Thiết lập nhận luồng video từ camera, phát video và nhận diện khói/lửa	Nguyễn Thạc Hiếu, Nguyễn Thị Tuyết Trinh
Xây dựng ứng dụng	Phát triển ứng dụng di động để hiển thị cảnh báo xem dữ liệu cảm biến và camera giám sát từ hệ thống	Nguyễn Thạc Hiếu
Kết nối hệ thống	Giao tiếp ESP32 – Server – Firebase – App	Nguyễn Thạc Hiếu, Nguyễn Thị Tuyết Trinh
Kiểm thử hệ thống	Thực nghiệm mô phỏng cháy, kiểm tra độ chính xác AI, kiểm thử tính năng App	Nguyễn Thạc Hiếu, Nguyễn Thị Tuyết Trinh, Doãn Đăng Nhật
Hoàn thiện báo cáo và slide thuyết trình	Viết báo cáo, chuẩn bị slide bảo vệ, demo sản phẩm	Nguyễn Thạc Hiếu, Nguyễn Thị Tuyết Trinh, Doãn Đăng Nhật

