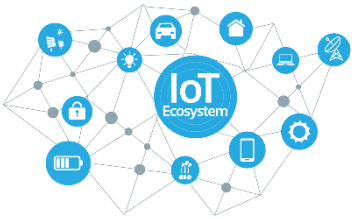
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN
PBL5 - KỸ THUẬT MÁY TÍNH



HỆ THỐNG CẢNH BÁO CHÁY

CÁN BỘ DOANH NGHIỆP HƯỚNG DẪN: Nguyễn Vương Quang
GIẢNG VIÊN ĐỒNG HƯỚNG DẪN: ThS. Trần Thế Vũ

STT NHÓM: 01 HỌ VÀ TÊN SINH VIÊN	LỚP HỌC PHẦN ĐỒ ÁN
Nguyễn Đắc Thái	20Nh10
Bùi Văn Thông	20Nh10
Hoàng Lê Thành Phương	20Nh10
Trần Nguyễn Tới	20Nh10

ĐÀ NẴNG, 06/2023

TÓM TẮT ĐỒ ÁN

Trong quá trình học tập và nghiên cứu, chúng em nhận thấy rằng việc cảnh báo cháy trong các tòa nhà và khu công nghiệp vẫn còn chưa được đảm bảo đầy đủ. Vì vậy, chúng em quyết định xây dựng một hệ thống cảnh báo cháy tự động cho người dùng.

Để thực hiện điều này, chúng em đã áp dụng các công nghệ AI như computer vision và machine learning để phát hiện khói và hình ảnh cháy, kết hợp với cảm biến nhiệt để xác định mức độ nóng của ngọn lửa.

Hệ thống cũng được tích hợp với API để giao tiếp với các thiết bị khác, chẳng hạn như hệ thống phun nước tự động. Để thực hiện hệ thống cảnh báo cháy tự động, chúng em đã sử dụng Raspberry Pi và các cảm biến, cùng với điện thoại thông minh để hiển thị thông báo cho người dùng.

Hệ thống cũng được thiết kế để kết nối với điện toán đám mây để lưu trữ và quản lý dữ liệu. Sau khi nghiên cứu và thử nghiệm, chúng em tự hào thông báo rằng hệ thống cảnh báo cháy tự động của chúng em đã hoạt động tốt. Tuy nhiên, chúng em vẫn sẽ tiếp tục phát triển và cải tiến hệ thống trong tương lai để đảm bảo rằng nó hoạt động hiệu quả và đáng tin cậy.

Cuối cùng nhóm chúng em xin cảm ơn giảng viên Trần Thế Vũ, doanh nghiệp Enclave và anh hướng dẫn Nguyễn Vương Quang đã có những chỉ dạy và đóng góp quý giá để giúp nhóm chúng em ngày càng hoàn thiện hơn.

BẢNG PHÂN CÔNG NHIỆM VỤ

Sinh viên	Nhiệm vụ	Hoàn thành
Nguyễn Đắc Thái	Phân công công việc, đảm bảo tiến độ đồ án	✓
	Tìm hiểu, triểu khai phần phát hiện lửa	✓
	Thu thập dữ liệu	✓
	Xây dựng model và training	✓
	Kết nối và nhận diện trên Picam	✓
	Viết báo cáo	✓
Bùi Văn Thông	Tìm hiểu, triển khai phần server	✓
	Cài đặt hệ điều hành cho Raspberry	✓
	Kết nối dữ liệu với phần cứng và app	✓
	Ghép nối và thử nghiệm sản phẩm	✓
	Viết báo cáo	✓
Hoàng Lê Thành Phương	Xây dựng ứng dụng mobile app	✓
	Xây dựng cơ sở dữ liệu	✓
	Ghép nối và thử nghiệm sản phẩm	✓
	Viết báo cáo	✓
Trần Nguyễn Tới		✓
	Tìm kiếm và chuẩn bị phần cứng	✓
	Lắp ráp cảm biến	✓
	Ghép nối và thử nghiệm sản phẩm	✓
	Viết báo cáo	✓

MỤC LỤC

DANH MỤC HÌNH ẢNH

1. Giới thiệu

1.1. Thực trạng sản phẩm

Hiện nay, sản phẩm hệ thống cảnh báo cháy tự động đang là một trong những sản phẩm được sử dụng rộng rãi trong các tòa nhà, khách sạn, nhà xưởng, trường học, bệnh viện, ... Với tính năng cảnh báo ngay lập tức khi phát hiện cháy, sản phẩm này giúp cho việc phòng cháy chữa cháy và giảm thiểu thiệt hại về tài sản cũng như tính mạng con người. Yếu tố cần thiết nhất để đem lại một hệ thống cảnh báo cháy tự động tốt đó là sự chính xác. Do đó, chúng em tiến hành thử nghiệm đề tài này với mục đích cải thiện độ chính xác của hệ thống nhắm đem lại sự an tâm cho người dùng.

1.2. Các vấn đề cần giải quyết

- Cần có các thiết bị phần cứng để thu nhận dữ liệu.
- Phát hiện và nhận diện nhiều lửa trong đám cháy, tiến hành gởi thông tin cho người dùng.
- Cần ứng dụng tương tác để người dùng có thể xem cảnh báo, ra quyết định.
- Hệ thống chạy theo thời gian thực.

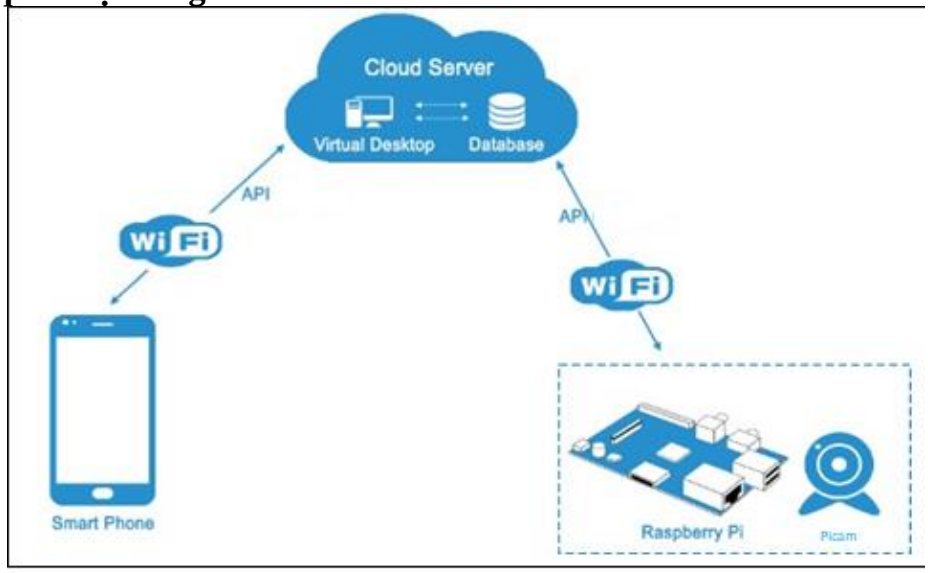
1.3. Đề xuất giải pháp tổng quan

Bảng 1: Đề xuất giải pháp tổng quan

Vấn đề	Giải pháp đề xuất
Phần cứng	Raspberry Pi 3. Cảm biến khói MQ2. Cảm biến nhiệt độ LM35 Điện thoại thông minh.
Phát hiện lửa	Xây dựng và huấn luyện model phát hiện lửa. Thử nghiệm với các model: Yolo, Facenet,Tensorflow lite... Huấn luyện trên Google Colab.
Nhận diện lửa	Xây dựng và huấn luyện model nhận diện lửa. Thử nghiệm với các model: Facenet, Tensorflow,ArcFace,… Huấn luyện trên Google Colab.
Ứng dụng	Xây dựng ứng dụng điện thoại. Có chức năng hiển thị nhiệt độ và nồng độ gas. Lưu trữ dữ liệu cháy. Thông báo cháy về điện thoại.
Server	Viết API bằng Flask

2. Giải pháp

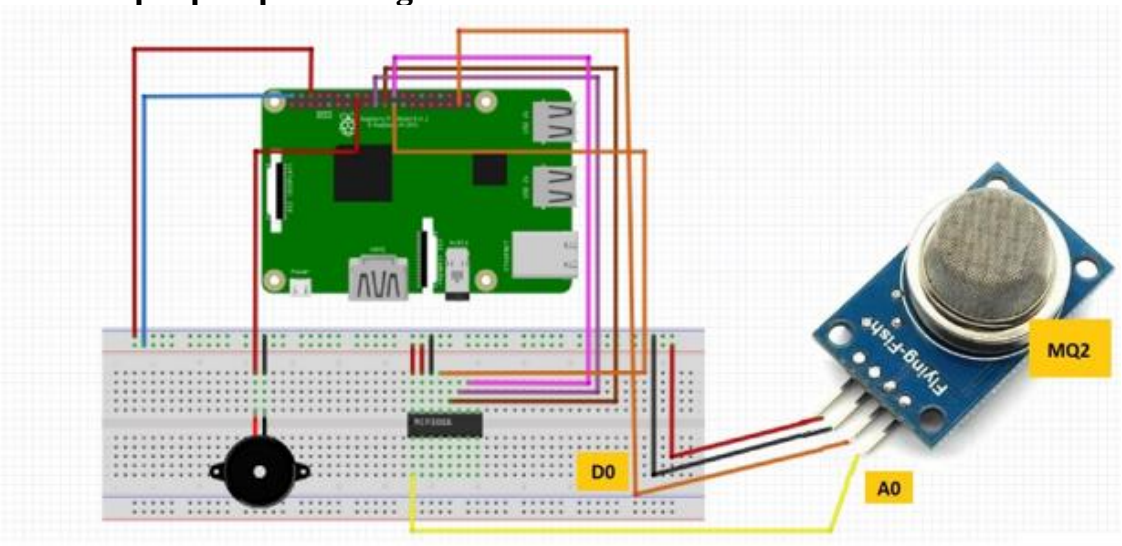
2.1. Tổng quan hệ thống.



Hình 1: Sơ đồ tổng quan hệ thống

Hệ thống bao gồm Raspberry Pi 3 và picam dùng để chụp ảnh, thiết bị smart phone dùng tương tác và hiển thị kết quả và Cloud Server để thiết lập Server. Thông qua mạng không dây, điện thoại và Raspberry Pi có thể giao tiếp với Server bằng API. API này được lập trình dựa trề Flask ramework.

2.2. Giải pháp về phần cứng

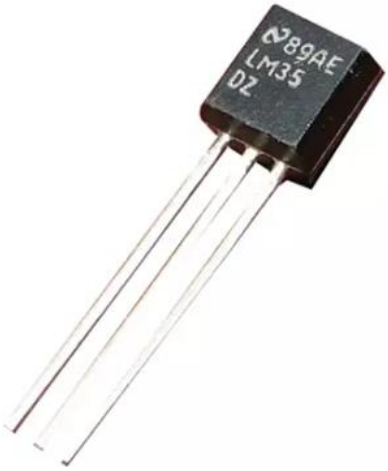


Hình 2: Mô hình phần cứng

2.2.1. Linh kiện sử dụng

Bảng 2: Linh kiện sử dụng

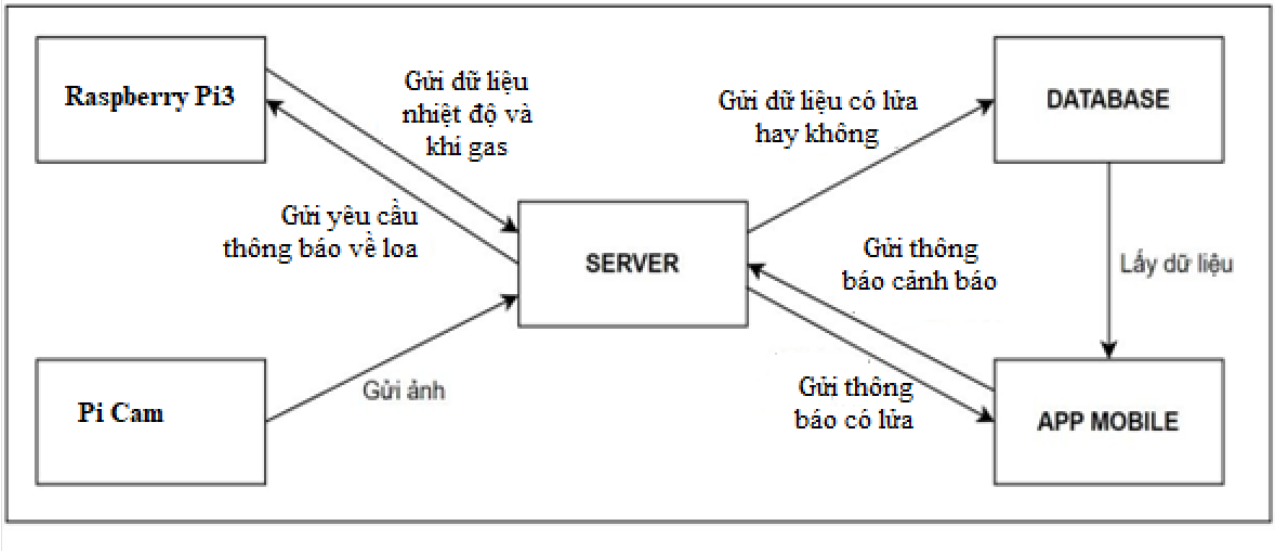
Tên linh kiện	Hình ảnh	Thông số, hoạt động
Raspberry Pi 3		**Thông số kỹ thuật** Bộ vi xử lý 64-bit quad-core ARM Cortex-A53 Mạng không dây 802.11 b/g/n Bluetooth 4.1 Bộ nhớ LPDDR2 1GB Nguồn 2.5A cổng MicroUSB 1 cổng Ethernet 10/100 1 cổng HDMI 1 jack cắm RCA 4 cổng USB 2.0 Khe cắm thẻ nhớ MicroSD
Picam		**Thông số kỹ thuật** Độ phân giải : 2592 x 1944 (5 megapixel) + Ống kính : f = 3,57 mm, f / 2.8 + Góc nhìn : 65 độ + Phạm vi lấy nét : 0,69m đến vô cực （ở mức 1,38m) + Hỗ trợ: 1080p @ 30 khung hình / giây với codec H.264 (AVC) , 720p @ 60fps và 640x480p @ 60/90 khung hình / giây + Giao diện: CSI

Mq2 **Cảm biến khí gas**		Điện áp hoạt động: 3.3V-5V Kích thước PCB: 3cm * 1.6cm Led đỏ báo nguồn vào, Led xanh báo gas VCC: 3.3V-5V GND: 0V DO: Đầu ra tín hiệu số (0 và 1) AO: Đầu ra Analog (Tín hiệu tương tự) Cấu tạo từ chất bản dẫn Sno2
Loa Buzzer		Nguồn : 3.5V - 5.5V. Dòng điện tiêu thụ: <25mA. Tần **số** cộng hưởng: 2300Hz ± 500Hz. Biên độ âm thanh: >80 dB. Nhiệt độ hoạt động:-20 °C đến +70 °C. Kích thước : Đường kính 12mm, cao 9,7mm.
LM35 **Cảm biến nhiệt độ**		Điện áp hoạt động: 4~20VDC. Công suất tiêu thụ: khoảng 60uA. Khoảng đo: -55°C đến 150°C. Điện áp tuyến tính theo nhiệt độ: 10mV/°C. Sai **số**: 0.25°C. Kiểu chân: TO92. Kích thước: 4.3 × 4.3mm.

Bảng 3: Bảng kê chi phí đồ án

Tên linh kiện	Đơn giá	Ghi chú
1x Raspberry Pi 3 model B+	900.000	Mua cũ
1x Thẻ nhớ 16GB	120.000	Mua
1x Pi cam	85.000	Mua
1x MQ2	30.000	Mua
1x LM35 (Cảm biến nhiệt độ)	35.000	Mua
1x Loa Buzzer	10.000	Mua
	Thành tiền: 1.180.000	

2.3. Giải pháp truyền thông



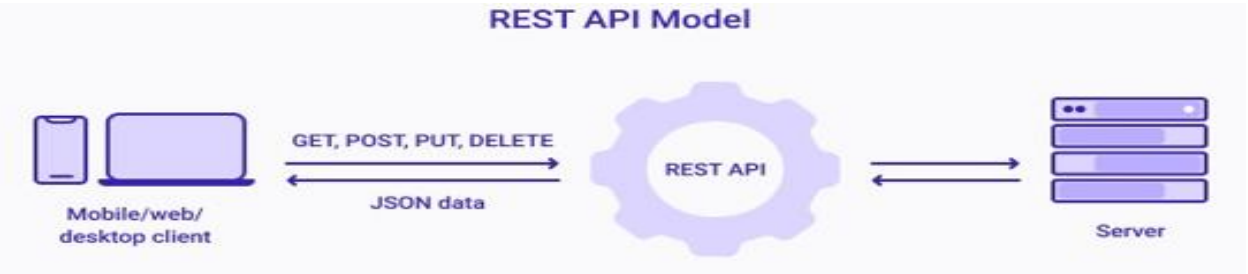
Hình 3: Tiến trình giao tiếp giữa các khối

Raspberry Pi3 giao tiếp với sever thông qua giao thức WebSocket để thực hiện việc gửi dữ liệu và nhận yêu cầu từ mobile. Trong khi đó Mobile sẽ gửi yêu cầu cảnh báo tới Sever thông qua API và nhận dữ liệu từ sever thông qua WebSocket.

2.4. Giải pháp phần mềm

2.4.1. Restful API

RESTful API là một tiêu chuẩn được sử dụng trong việc thiết kế API cho các phần mềm, ứng dụng và dịch vụ web để tạo sự thuận tiện cho việc quản lý các resource. Các tài nguyên hệ thống như tệp văn bản, ảnh, video, âm thanh hay dữ liệu di động là mục tiêu mà



Hình 4: Mô hình REST API

API (Application Programming Interface): Là tập hợp các quy tắc và cơ chế mà một ứng dụng hay một thành phần nào đó có khả năng tương tác với một ứng dụng với thành phần khác. API sẽ trả về những kiểu dữ liệu phổ biến như JSON hoặc XML mà ứng dụng của bạn cần sử dụng đến.

REST (Representational State Transfer): Là một dạng chuyển đổi cấu trúc hay kiểu kiến trúc để viết API. Nó có khả năng tạo ra sự tương tác giữa các máy với nhau thông qua phương thức HTTP đơn giản. Chức năng của REST là quy định sử dụng các phương thức HTTP và định dạng URL cho ứng dụng web.

❖ **Cách thức hoạt động của RESTful API**

RESTful API là phương thức tạo ra API và hoạt động dựa trên phương thức HTTP:

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

2.4.2Flask

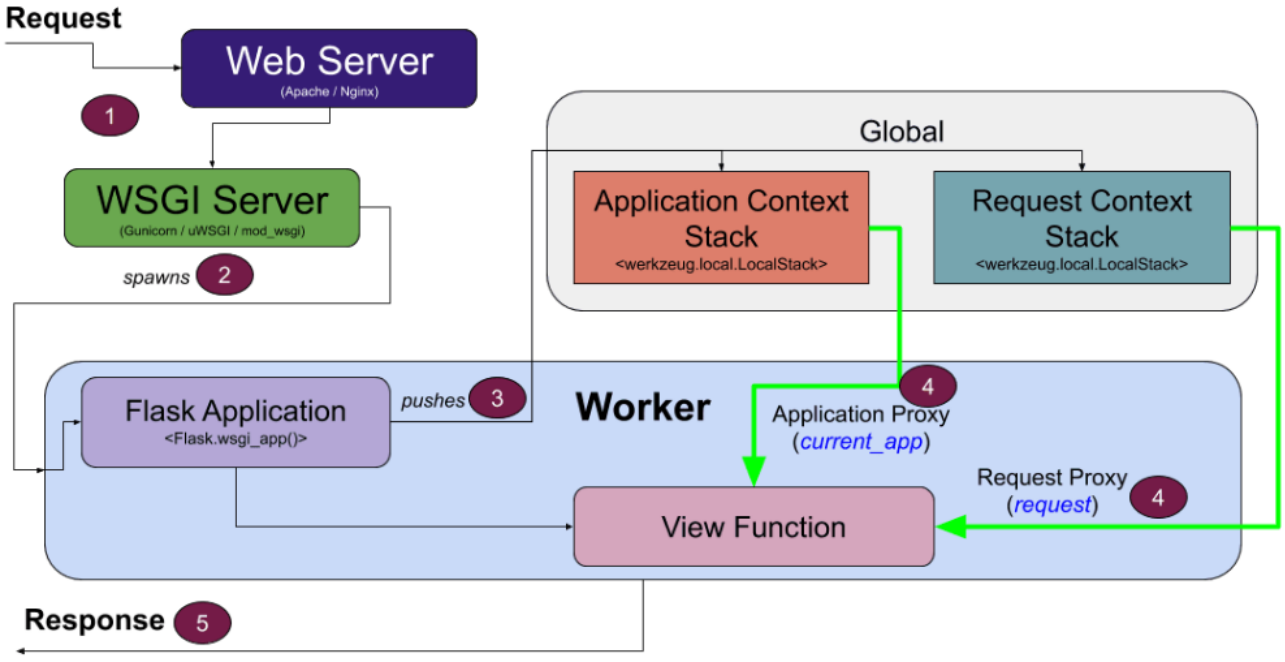
Flask là một framework web cho phép bạn xây dựng các ứng dụng web và RESTful APIs bằng Python. Flask là một trong những framework web phổ biến nhất cho Python, nhẹ nhàng, có tính linh hoạt cao và dễ sử dụng.

Flask cung cấp các công cụ và thư viện để giúp bạn xử lý các yêu cầu HTTP, xử lý biểu mẫu, xử lý tệp tin, xử lý cookie và nhiều hơn nữa. Flask không yêu cầu bạn tuân thủ một cấu trúc dự án cụ thể nào, do đó bạn có thể tùy chỉnh dự án của mình theo cách của riêng bạn.

❖ Kiến trúc của Flask

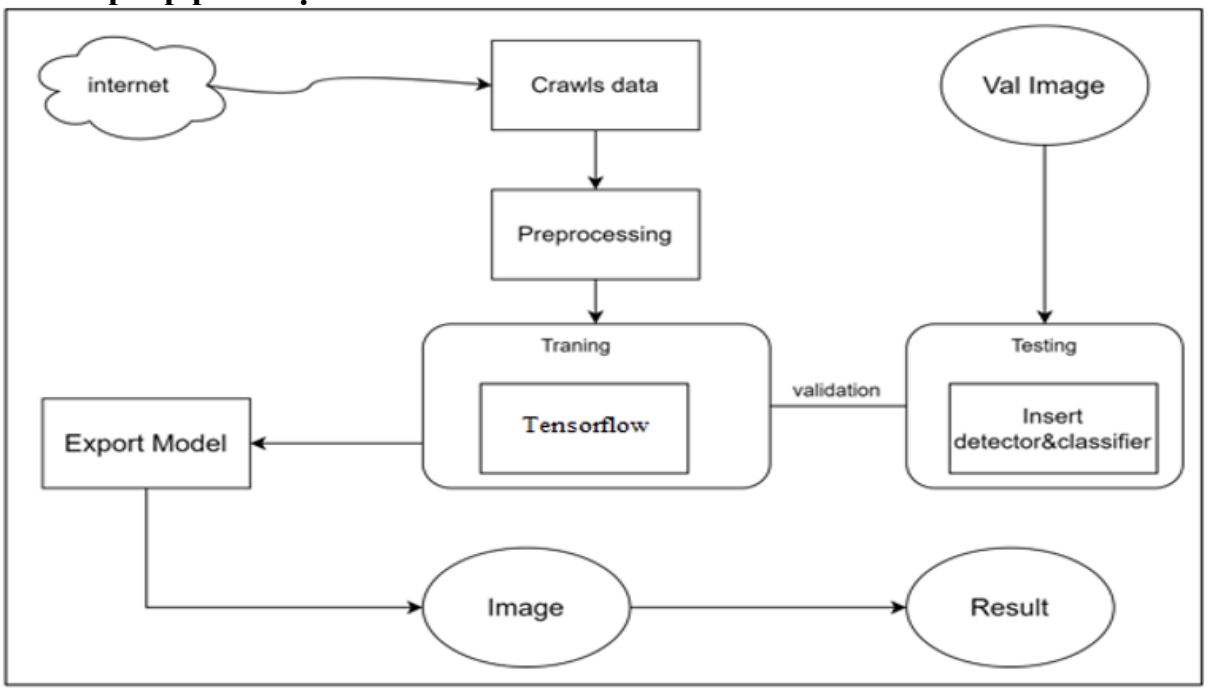
Kiến trúc của Flask tập trung vào việc xây dựng các ứng dụng web RESTful đơn giản và nhẹ nhàng. Dưới đây là một số thành phần chính trong kiến trúc của Flask:

- WSGI (Web Server Gateway Interface): Flask tuân thủ chuẩn WSGI, cho phép nó tương tác với các máy chủ web khác nhau như Gunicorn hoặc uWSGI. WSGI là một giao diện giữa ứng dụng web và máy chủ web để xử lý các yêu cầu HTTP và trả về các phản hồi tương ứng.
- Routes (Đường dẫn): Flask sử dụng decorators (trang trí) để xác định các routes (đường dẫn) cho các chức năng (functions) trong ứng dụng. Ví dụ: @app.route('/') sẽ xác định đường dẫn gốc của ứng dụng.
- Views (Xem): Flask sử dụng views để xử lý các yêu cầu HTTP từ người dùng. Mỗi view là một chức năng (function) nhận yêu cầu và trả về phản hồi. Phản hồi có thể là một chuỗi văn bản, một trang HTML hoặc một đối tượng JSON.
- Templates (Mẫu): Flask hỗ trợ sử dụng mẫu (templates) để tạo ra các trang web động. Mẫu cho phép bạn tách biệt code HTML và logic ứng dụng, giúp quản lý giao diện và logic dễ dàng hơn.
- Models (Mô hình): Flask không cung cấp một ORM (Object-Relational Mapping) tích hợp sẵn, nhưng bạn có thể sử dụng các thư viện ORM khác như SQLAlchemy để tương tác với cơ sở dữ liệu. Models (mô hình) được sử dụng để định nghĩa cấu trúc dữ liệu và các phương thức liên quan để truy vấn và cập nhật dữ liệu.



Hình 5: Cách thức hoạt động của Flask

2.5. Giải pháp phát hiện lửa



Hình 6: Tổng quan chức năng phát hiện lửa

2.5.1. Mô hình CNN

❖ Giới thiệu về mô hình CNN

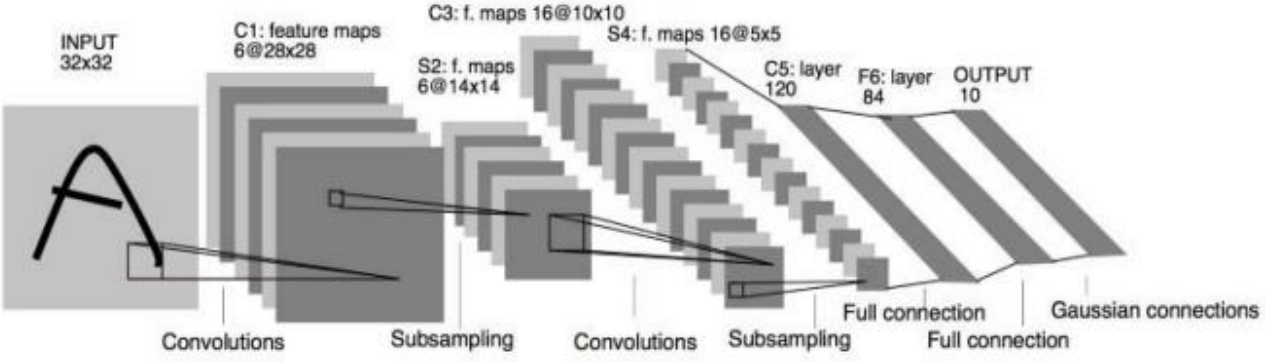
Mô hình CNN (Convolutional Neural Networks) là một trong những mô hình deep learning phổ biến nhất và có ảnh hưởng nhiều nhất trong cộng đồng Computer Vision. CNN được dùng trong nhiều bài toán như nhân dạng ảnh, phân tích video, ảnh MRI, hoặc giải quyết các bài toán về lĩnh vực xử lý ngôn ngữ tự nhiên.

Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác xuất giữa 0 và 1.

❖ Kiến trúc của CNN

a) Mạng nơ-ron tích chập:

Trong các mạng MLP truyền thống, mỗi nơ-ron trong lớp phía trước sẽ được kết nối đến tất cả các nơ-ron ở lớp phía sau, điều này khiến cho khối lượng tính toán trong mạng tăng mạnh khi tăng độ sâu của mô hình (tăng số lượng lớp) cho mô hình.



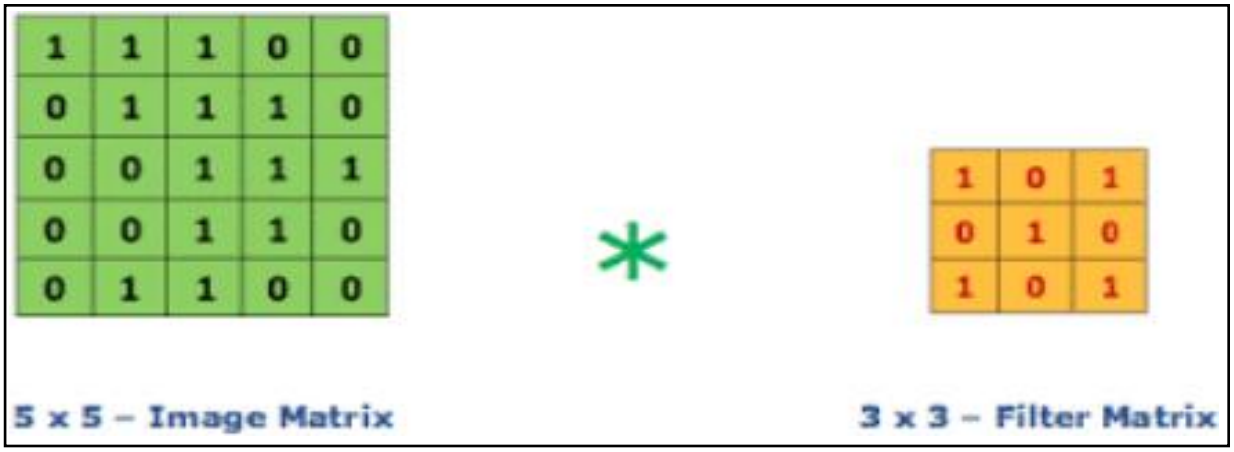
Hình 7: Sơ đồ khối mô tả mô hình mạng lenet-5

Sự ra đời của mạng CNN đã giúp giải quyết vấn đề trên bằng cách sử dụng các vùng tiếp nhận cục bộ, tập trọng số chia sẻ và phương pháp lấy tích chập để trích xuất

b) Lớp tích chập - Convolution Layer:

Lớp tích chập (Convolution Layer) là một phần quan trọng trong kiến trúc của mô hình CNN. Tầng tích chập sử dụng các bộ lọc để thực hiện phép tích chập khi đưa chúng đi qua đầu vào theo các chiều của nó. Các siêu tham số của các bộ lọc này bao gồm kích thước bộ lọc và độ trượt (stride). Kết quả đầu ra được gọi là feature map hay activation map. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.

Ví dụ: Xét 1 ma trận kích thước 5 x 5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3 x 3 như dưới đây:



Hình 8: Ví dụ về Convolution Layer

- Sau đó lớp tích chập của ma trận hình ảnh 5 x5 nhân với ma trận bộ lọc 3 x 3 gọi là "Feature Map" như bên dưới.

4	3	4
2	4	3
2	3	4

Các siêu tham số của bộ lọc trong tầng tích chập (Convolution Layer) bao gồm kích thước bộ lọc (F) và độ trượt (stride - S). Độ trượt ký hiệu số pixel mà cửa sổ sẽ di chuyển sau mỗi lần thực hiện phép tính.

Ngoài ra còn có siêu tham số Zero-padding là tên gọi của quá trình thêm P số không vào các biên của đầu vào. Giá trị này có thể được lựa chọn thủ công hoặc một cách tự động bằng một trong ba những phương pháp: Valid, Same và Full.

c) Lớp gộp - Pooling Layer:

Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Cụ thể, max pooling và average pooling là những dạng pooling đặc biệt, mà tương ứng là trong đó giá trị lớn nhất và giá trị trung bình được lấy ra.

d) Hàm kích hoạt - Activation Function

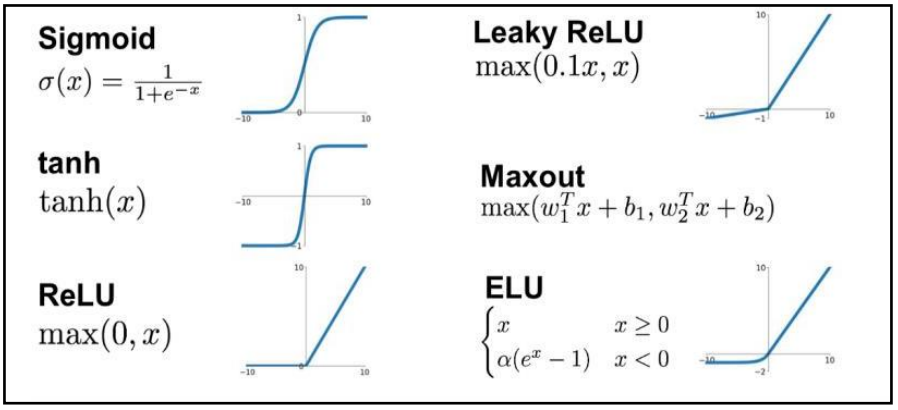
ReLU (Rectified Linear Unit) là một trong những hàm kích hoạt phổ biến được sử dụng trong mô hình CNN.

Mô hình CNN sử dụng các hàm kích hoạt để tạo ra các đặc trưng phi tuyến tính từ các giá trị đầu vào. Các hàm kích hoạt như ReLU giúp mô hình có khả năng học được các đặc trưng phi tuyến tính, cũng như giảm thiểu sự triệt tiêu đặc trưng khi áp dụng các phép tích chập.

Ngoài ReLU, các hàm kích hoạt khác được sử dụng trong CNN bao gồm Sigmoid, Tanh, v. Tùy thuộc vào vấn đề cụ thể và cấu trúc của mô hình, một hàm kích hoạt phù hợp sẽ được chọn để đạt được hiệu quả tốt nhất

e) Dropout Layer:

Lớp dropout (Dropout Layer) là một lớp tắt ngẫu nhiên một số đơn vị đầu vào với một tần suất nhất định trong mỗi bước trong quá trình huấn luyện, điều này giúp ngăn chặn overfitting. Các đầu vào không được đặt thành 0 được thu nhỏ bằng 1 / (1 - rate) để tổng trên tất cả các đầu vào không thay đổi. Lưu ý rằng lớp dropout chỉ áp dụng khi training được đặt thành True để không có giá trị nào bị loại bỏ trong suy luận.



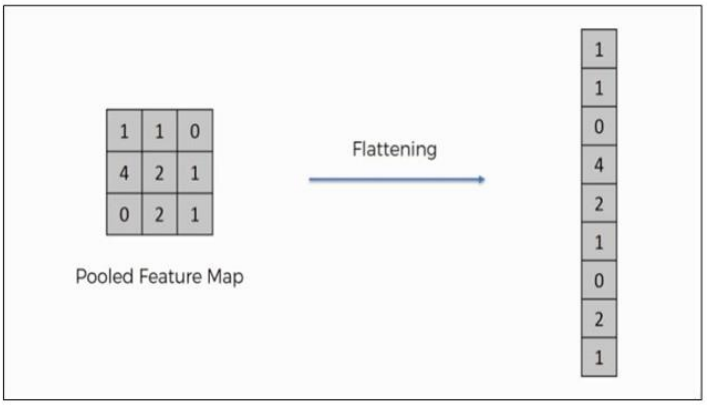
Hình 9: Ví dụ về Activation function

f) Flatten Layer:

Lớp làm phẳng (Flatten Layer) được sử dụng để làm phẳng đầu vào. Ví dụ : nếu lớp làm phẳng được áp dụng cho lớp có đầu vào có kích thước (batch_size, 2,2), thì kích thước đầu ra của lớp sẽ là (batch_size, 4)

g) Lớp kết nối đầy đủ - Fully connected layer:

Tầng kết nối đầy đủ (Fully Connected Layer) nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả neuron. Trong mô hình mạng CNNs, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.



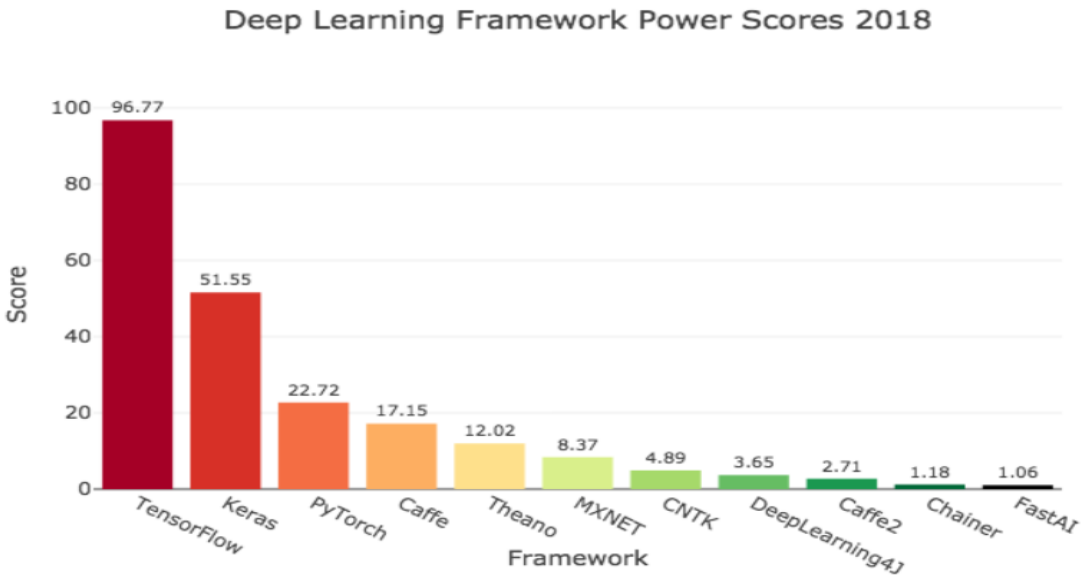
Hình 10: Ví dụ về Fully connected layer

2.5.2. Mô hình nhận diện lửa với Tensorflow

❖ **Lý do chọn Tensorflow**

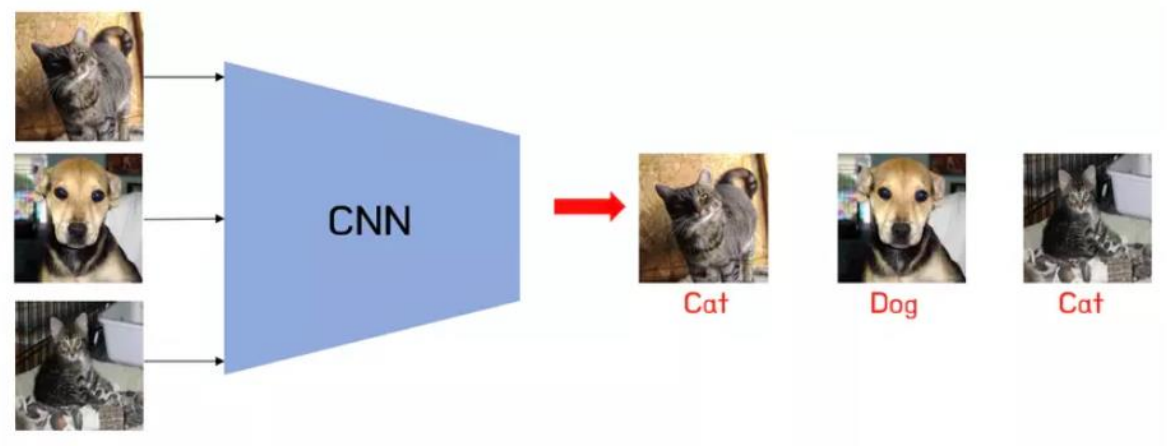
Raspberry Pi3 khá yếu chỉ với 1GB RAM và bị hạn chế tài nguyên.

➢ TensorFlow cung cấp một hiệu suất tốt với việc xử lý và huấn luyện mô hình học sâu. TensorFlow có thể được tối ưu để chạy một mô hình nhận diện lửa một cách hiệu quả.



Hình 11: Các deep learning framework phổ biến trong bài toán nhận diện

Ví dụ: Xây dựng một mô hình neural network để phân loại hình ảnh. Bạn có thể sử dụng TensorFlow để xây dựng một mô hình convolutional neural network (CNN) để phân loại hình ảnh về chó và mèo. Bằng cách sử dụng các tính năng của TensorFlow, bạn có thể tạo ra một mô hình phân loại hình ảnh có độ chính xác cao.



Hình 12: Sơ đồ tổng quát nhận diện với Tensorflow

❖ Giới thiệu về Tensorflow

Tensorflow là một framework mã nguồn mở cho deep learning được viết bằng Python. Nó có thể chạy trên nền của các deep learning framework khác như: Keras, theano, CNTK. Với các API bậc cao, dễ sử dụng, dễ mở rộng, tensorflow giúp người dùng xây dựng các deep learning model một cách đơn giản.

❖ Kiến trúc của Tensorflow

- Tiền xử lý dữ liệu
- Dựng model
- Train và ước tính model

❖ Cách hoạt động

TensorFlow cho phép các lập trình viên tạo ra dataflow graph, cấu trúc mô tả làm thế nào dữ liệu có thể di chuyển qua 1 biểu đồ, hay 1 sê-ri các node đang xử lý. Mỗi node trong đồ thị đại diện 1 operation toán học, và mỗi kết nối hay edge giữa các node là 1 mảng dữ liệu đa chiều, hay còn được gọi là 'tensor'.

TensorFlow cung cấp tất cả những điều này cho lập trình viên theo phương thức của ngôn ngữ Python. Vì Python khá dễ học và làm việc, ngoài ra còn cung cấp nhiều cách tiện lợi để ta hiểu được làm thế nào các high-level abstractions có thể kết hợp cùng nhau. Node và tensor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow bản thân chúng cũng là các ứng dụng Python.

Các operation toán học thực sự thì không được thi hành bằng Python. Các thư viện biến đổi có sẵn thông qua TensorFlow được viết bằng các binary C++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các high-level abstraction lập trình để nối chúng lại với nhau.

Train phân tán dễ chạy hơn nhờ vào API mới và sự hỗ trợ cho TensorFlow Lite cho phép triển khai các mô hình trên khá nhiều nền tảng khác nhau. Tuy nhiên, nếu đã viết code trên các phiên bản trước đó của TensorFlow thì bạn phải viết lại, đôi lúc 1 ít, đôi lúc cũng khá đáng kể, để tận dụng tối đa các tính năng mới của TensorFlow 2.0.

❖ **Các bước training CNN bằng tensorflow**

Hình 13: Các bước training CNN bằng Tensorflow

2.6. Giải pháp ứng dụng di động

2.6.1. Phát triển bài toán

Xây dựng app android để tương tác với hệ thống server và raspberry để quản lí user , hiện thị các thông số cảm biến và hiện thị ảnh. Hệ thống user có thể xem các thông tin , ngoài ra user có thể sửa đổi username và password.

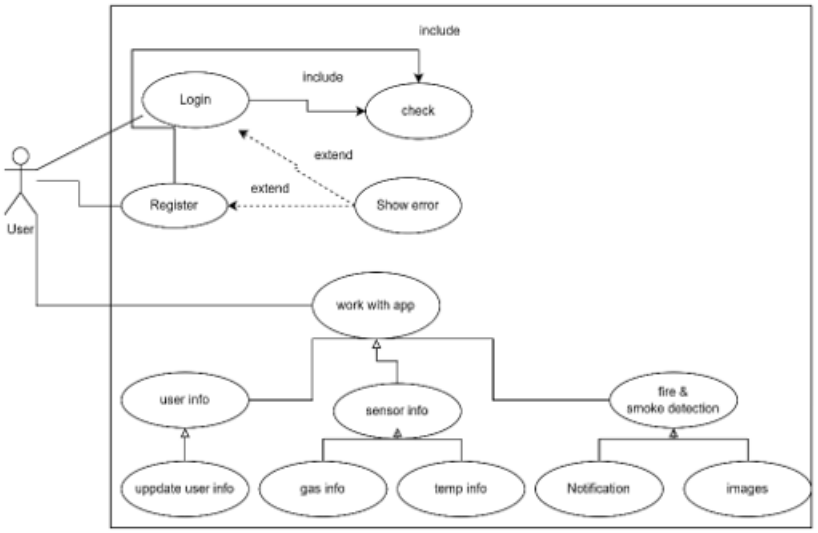
2.6.2. Công nghệ sử dụng

Kotlin : một ngôn ngữ lập trình hiện đại được phát triển bởi JetBrains

Retrofit2 :một thư viện HTTP client cho ngôn ngữ Kotlin, giúp đơn giản hóa việc giao tiếp mạng trong ứng dụng Kotlin và cung cấp một cách linh hoạt để tương tác với API.

Notification: tạo và quản lý thông báo trong ứng dụng

2.6.3. Sơ đồ use-case



Hình 14: Sơ đồ use-case App mobile

3. Kết quả

3.1. Mô hình nhận diện

3.1.1. Tệp dữ liệu

- ❖ Chuẩn bị dữ liệu

Thu nhập các hình ảnh về lửa với lớp phát hiện có lửa hay không. Thông tin về tập dữ liệu:

Nguồn thu thập: https://universe.roboflow.com/

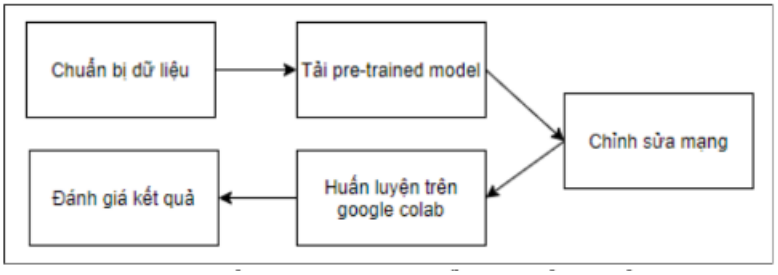
Mô tả: Tập dữ liệu Lửa được thu thập từ nhiều nguồn chuẩn với 4707 ảnh .jpg và các tệp gán nhãn .xml lửa. Tệp được chia sẵn thành: train (3295 hình ảnh), validation (700 hình ảnh) và test (712 hình ảnh).

Phân chia dữ liệu:

Train (70%)	Validation (14.8%)	Test (15.2%)

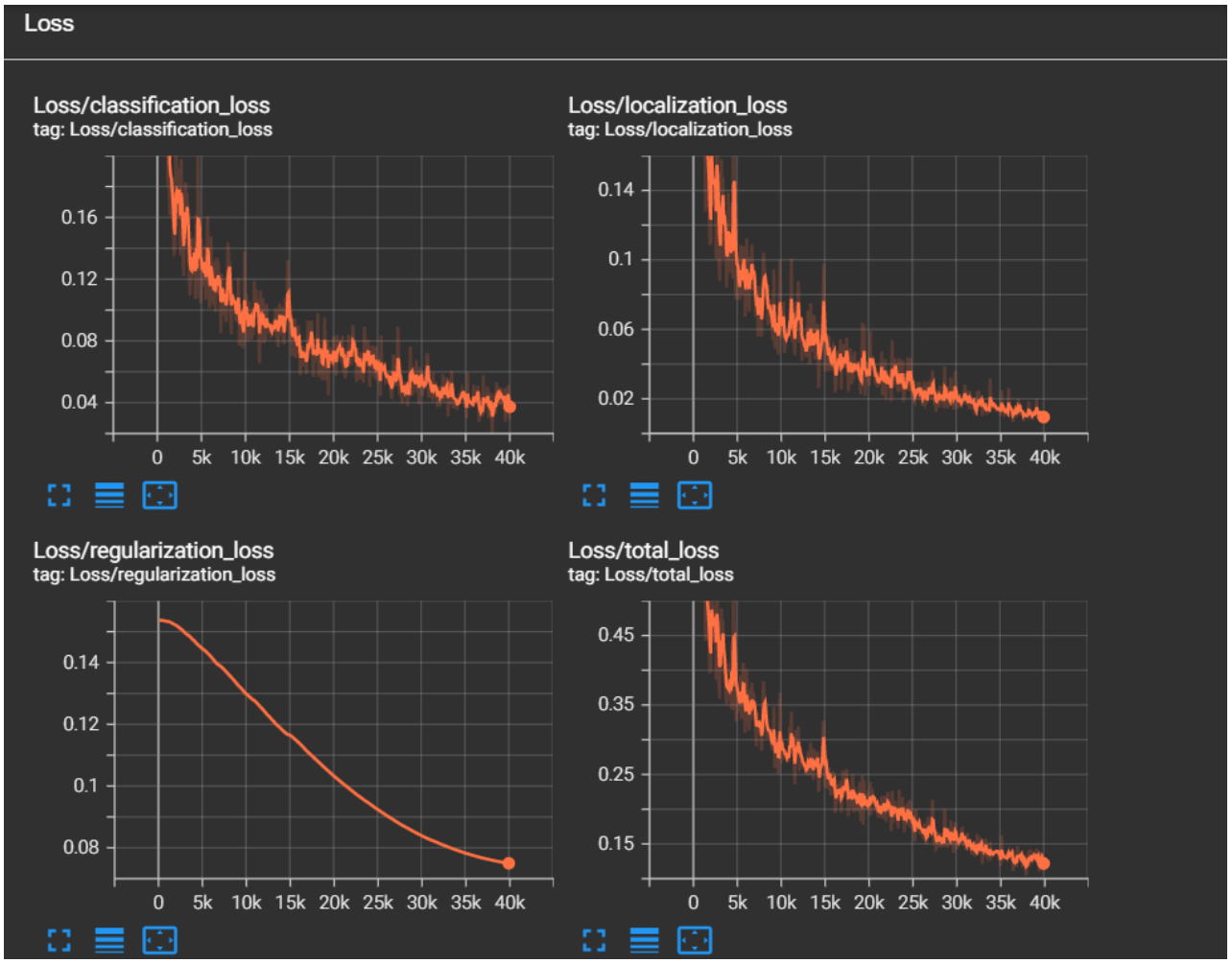
3.1.2. Huấn luyện

Sau khi có được dữ liệu và phân chia dữ liệu theo tỉ lệ phù hợp. Nhóm đã tiến hành tải xuống tệp pre-trained model và tệp cấu hình huấn luyện từ Tensorflow. Các folder này được sử dụng để cấu hình và khởi tạo mô hình nhận diện lửa trong quá trình huấn luyện.

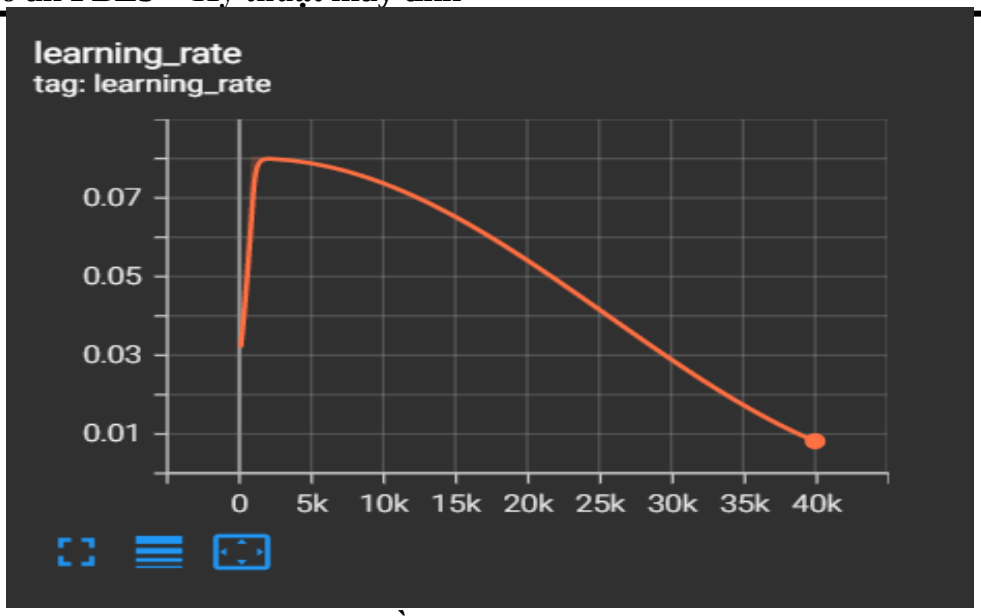


Hình 15: Sở dồ quá trình chuẩn bị đến huấn luyện.

Nhóm huấn luyện với 40.000 steps (Tổng số bước đào tạo mô hình) ta thấy hàm Loss đi từ cao xuống (xấp xỉ 0.43 xuống 0.15) và hàm Learning rate cũng vậy (giảm từ 0.07 xuống 0.01).



Hình 16: Đồ thị hàm Loss quá trình huấn luyện



Hình 17: Đồ thị hàm Learning rate

- Việc giảm hàm loss cho thấy mô hình đang học và cải thiện độ chính xác của dự đoán. Đồng thời, việc giảm learning rate có thể giúp mô hình tiếp tục học tập một cách ổn định và cải thiện hiệu suất.

3.1.3. Kết quả giải pháp nhận diện lửa

❖ **Kết quả định tính**

•Khi camera có độ phân giải thấp, rất khó phát hiện ở khoảng cách xa.

•Vẫn còn phát hiện nhầm lửa khi điều kiện ánh sáng không đủ.

•Vì hạn chế về tốc độ xử lý nên nhiều lúc phát hiển ra lửa chậm

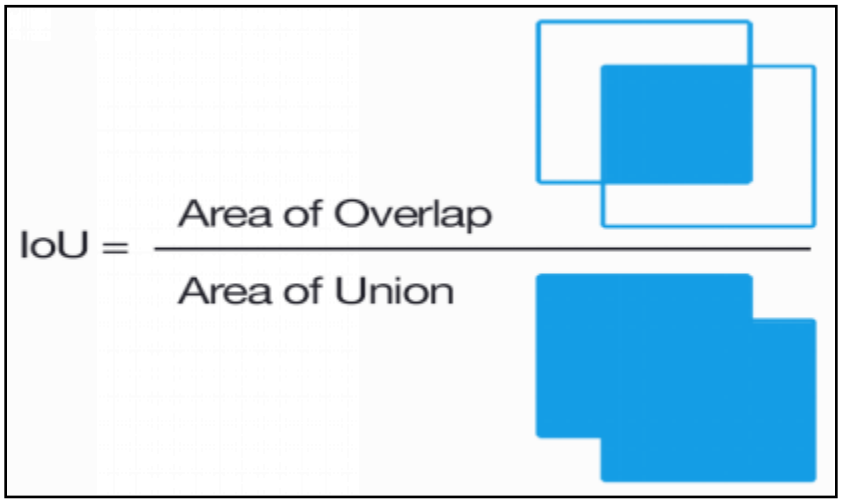
❖ **Kết quả định lượng**

Để tiến hành đánh giá hiệu suất mô hình, nhóm sử dụng mAP (mean Average Precision). mAP là độ đo được sử dụng rất phổ biến cho các bài toán Object Detection.

Tuy nhiên, trong mAP còn có nhiều khái niệm khác về Precision, Recall, IOU.

IOU (Intersection over union)

IOU (INTERSECTION OVER UNION) là hàm đánh giá độ chính xác của object detection trên tập dữ liệu cụ thể. IOU được tính bằng:



Hình 18: Công thức tính IoU

Trong đó Area of Overlap là diện tích phần giao nhau giữa predicted bounding box với grouth-truth bouding box , còn Area of Union là diện tích phần hợp giữa predicted bounding box với grouth-truth bounding box. Những bounding box được đánh nhãn bằng tay trong tập traing set và test set. Nếu IOU > 0.5 thì prediction được đánh giá là tốt.

Độ đo

Precision x Recall curve

Precision x Recall curve là cách tốt nhất đánh giá hiệu suất của một bộ phát hiện khuôn mặt với độ tự tin được thay đổi theo đồ thị đường cong cho mỗi lớp. Một bộ phát hiện khuôn mặt được coi là tốt nếu precision của nó vẫn cao khi recall cao, hoặc là khi thay đổi ngưỡng thì precision và recall vẫn cao.

Precision: Là độ đo đánh giá độ tin cậy của kết luận đưa ra (bao nhiêu % lời kết luận của model là chính xác).

Recall: Là độ đo đánh giá khả năng tìm kiếm toàn bộ các ground truth của mô hình (bao nhiêu % positive sample mà model nhận diện được).

True Positive (TP): Một kết quả phát hiện đúng (IOU >= ngưỡng).

False Positive (FP): Một kết quả phát hiện sai (IOU < ngưỡng).

False Negative (FN): 1 đốm lửa nhưng không phát hiện ra.

True Negative (TN): Một kết quả không phải là lửa nhưng phát hiện là lửa.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

TP = True positive

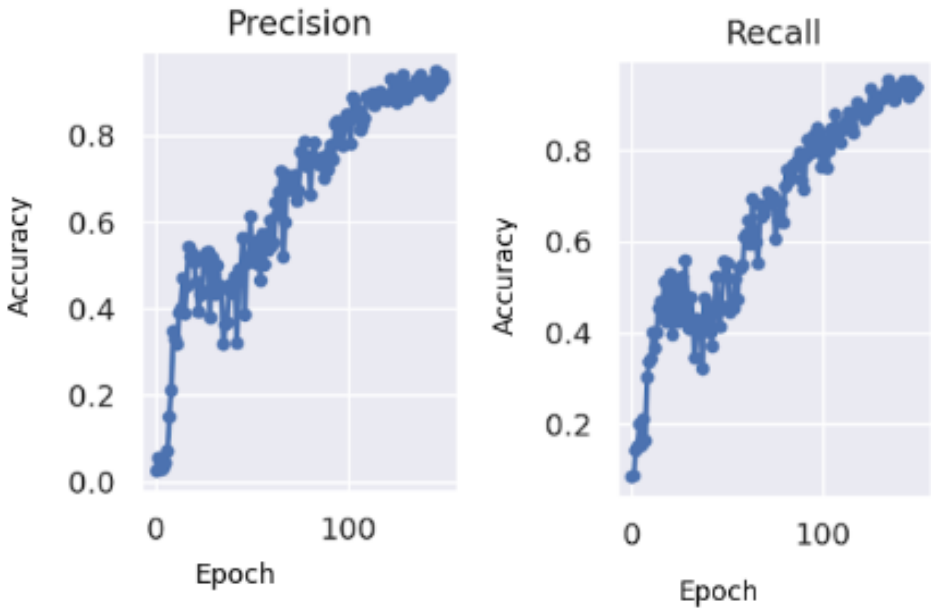
TN = True negative

FP = False positive

FN = False negative

Hình 19: Công thức tính Precision và Recall

Kết quả sau khi train :



Hình 20: Đồ thị biểu thị độ chính xác

Mean Average Precision (mAP): mAP bằng trung bình của chỉ số Average Precision (AP)trên tất cả các lớp trong một mô hình.

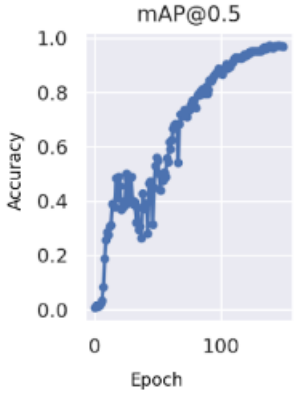
$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = \text{the AP of class } k$$

$$n = \text{the number of classes}$$

Hình 21: Công thức tính mAP

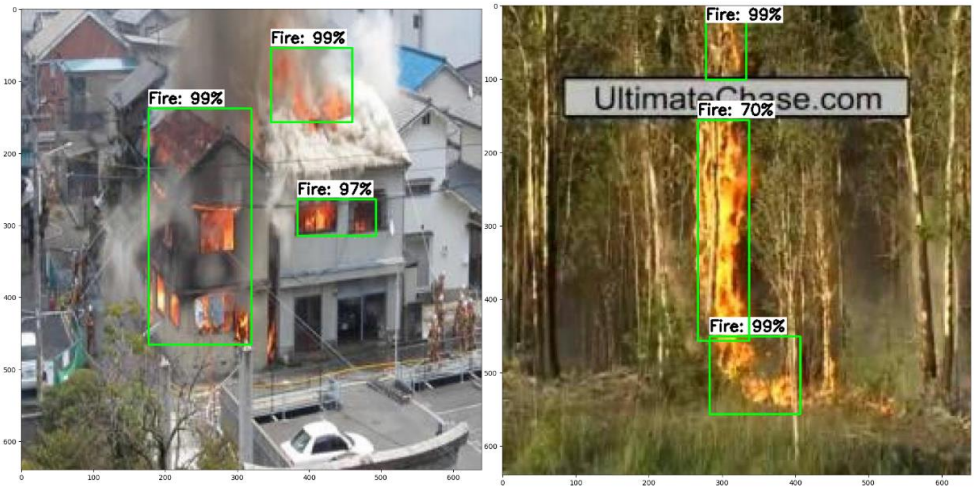
Kết quả mAP sau khi huấn luyện:



Hình 22: Đồ thị mAP

- Các chỉ số learning rate, loss, recall, precision và mAP đều phù hợp với một mô hình nhận diện tương đối tốt.

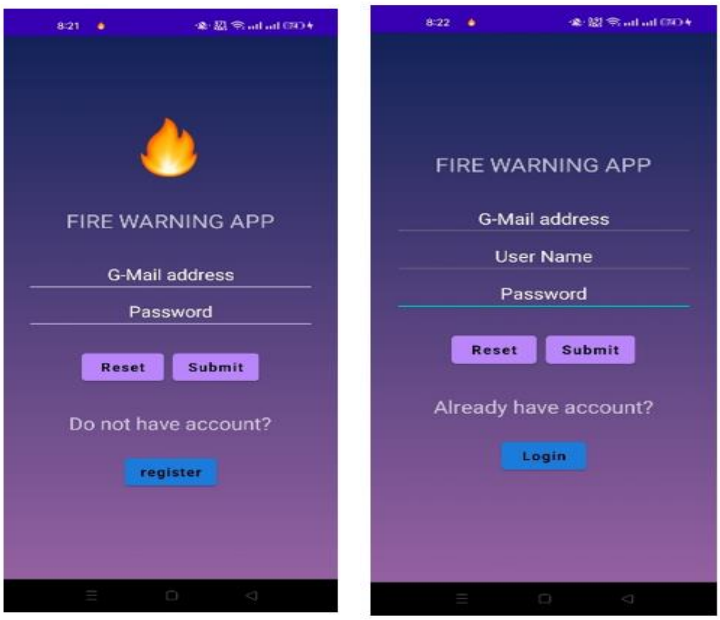
❖ **Kết quả nhận diện bằng hình ảnh (trên tập dữ liệu)**



Hình 23: Kết quả nhận diện

3.2. Phần mềm

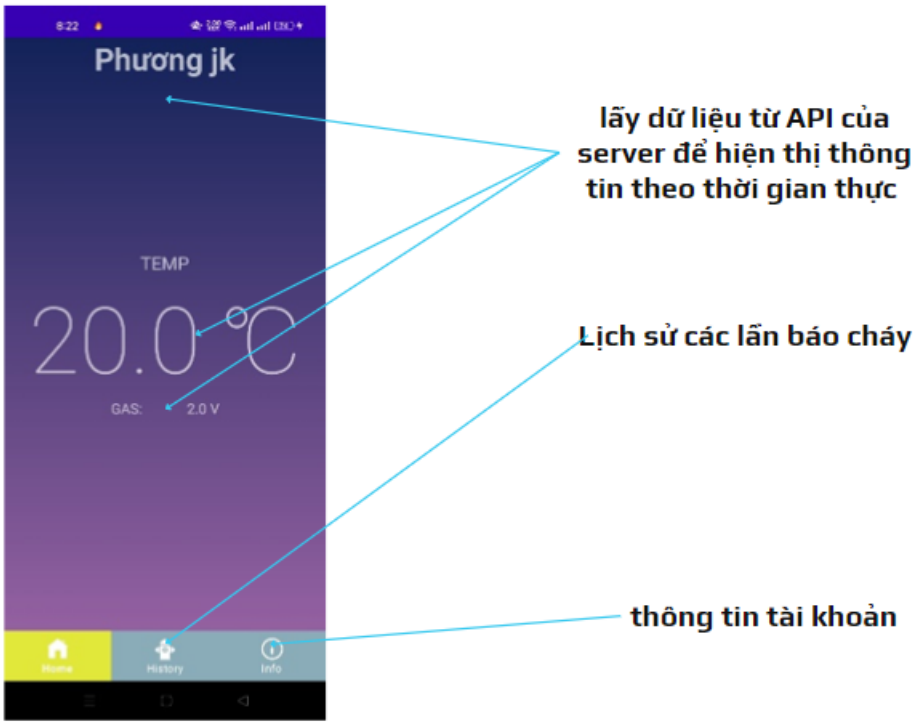
Giao diện phần mềm:



Hình 24: Giao diện đăng nhập và đăng ký

Sau khi đăng nhập thành công, giao diện sẽ có các chức năng hiển thị sau:

- Hiển thị nhiệt độ
- Hiển thị nồng độ gas
- Xem lịch sử báo cháy
- Thông tin tài khoản



Hình 25: Giao diện chính của app mobile

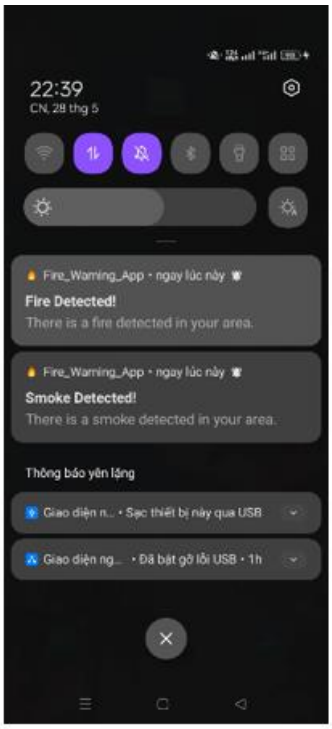
Giao diện trong các chức năng:

Lịch sử cháy Thông tin tài khoản

Hình 26: Giao diện chức năng

Trong chức năng lịch sử báo cháy, app sẽ chỉ lưu những tấm ảnh từ video được quay trực tiếp từ Pi cam khi nhận diện được lửa.

Ngoài việc chuyển ảnh về, điện thoại sẽ nhận được thông báo cháy và còi báo động.



Hình 27: Thông báo cháy

4. Kết luận và hướng phát triển

4.1. Kết quả đạt được

Thiết kế: Sản phẩm thiết kế gọn gàng, trọng lượng nhẹ.

Phát hiện lửa: Mô hình nhận diện lửa hiện tại chưa hoàn thiện và vẫn có thể cải tiến. Tuy nhiên độ chính xác khi thực nghiệm tương đối cao, mặc dù vài thời điểm còn có sai xót.

Server: Server hệ thống xử lý ổn định, gần như không xảy ra lỗi. Đáp ứng yêu cầu đồ án.

Ứng dụng di động: Ứng dụng có giao diện bắt mắt, dễ sử dụng, đầy đủ các chức năng, phù hợp với yêu cầu đồ án.

Phần cứng: Về tổng quan, các cảm biến có thể xuất ra giá trị chính xác, tuy nhiên độ bền có thời gian khá ngắn.

Raspberry Pi 3 còn hạn chế về tốc độ xử lý (1Gb RAM) và dung lượng bộ nhớ nên khó khăn trong việc nhận diện bằng Pi cam khi có độ trễ nhất định.

4.2. Hướng phát triển

Cần cải thiện nhiều về phần cứng bao gồm:

- Sử dụng Raspberry mạnh hơn để xử lý tốt trong việc nhận diện.
- Trang bị một Camera có chất lượng hình ảnh và xử lý mượt hơn.
- Cung cấp cho Raspberry một nguồn điện di động nhằm tăng khả năng linh hoạt và tránh trường hợp mất điện.
- Nâng cấp các cảm biến để có chỉ số đúng nhất.

Tiếp tục tăng khả năng nhận diện

- Tiền xử lý dữ liệu: đa dạng dataset, tăng giảm kích thước, độ sáng...
- Cải tiến training : tăng các tham số mô hình.

Nâng cấp sever: tối ưu hóa tốc độ gửi và nâng cấp dung lượng bộ nhớ.

Thêm một vài chức năng nữa cho phần mềm được tối ưu hơn.

5. Tài liệu tham khảo

[1] https://www.youtube.com/watch?v=XZ7FYAMCc4M&t=433s
[2] https://websitehcm.com/xay-dung-va-training-cnn-trong-tensorflow/
[3] https://niithanoi.edu.vn/flask-la-gi.html
[4] https://viblo.asia/p/building-cnn-model-layer-patterns-and-rules-Do754qXQKM6
[5] https://daotao.vku.udn.vn/uploads/2021/05/1622477559-doanchuyennghanh2.pdf#page=13&zoom=100,129,465
[6] https://nttuan8.com/bai-7-gioi-thieu-keras-va-bai-toan-phan-loai-anh/
[7] https://vi.wikipedia.org/wiki/Kotlin_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)