Sinh viên thực hiện: Huỳnh Văn Sĩ - Nguyễn Xuân
Đề tài: Ứng dụng công nghệ Edge AI thiết kế hệ thống dập lửa theo định hướng nguồn nhiệt
Năm: 2022

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
KHOA ĐIỆN-ĐIỆN TỬ

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

NGHÀNH: ĐIỆN – ĐIỆN TỬ
CHUYÊN NGHÀNH: KỸ THUẬT ĐIỆN TỬ

ĐỀ TÀI

ỨNG DỤNG CÔNG NGHỆ EDGE AI THIẾT KẾ HỆ THỐNG DẬP LỬA THEO ĐỊNH HƯỚNG NGUỒN NHIỆT

Giáo viên hướng dẫn: Ths. Phan Ngọc Kỳ
Sinh viên thực hiện: Nguyễn Xuân
Lớp: 18D4
Mã sinh viên: 1811505120261

Sinh viên thực hiện: Huỳnh Văn Sĩ
Lớp: 18D3
Mã sinh viên: 1811505120143

Đà Nẵng, Ngày 3 Tháng 6 Năm 2022

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
KHOA ĐIỆN-ĐIỆN TỬ

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

NGHÀNH: ĐIỆN – ĐIỆN TỬ
CHUYÊN NGHÀNH: KỸ THUẬT ĐIỆN TỬ

ĐỀ TÀI

ỨNG DỤNG CÔNG NGHỆ EDGE AI THIẾT KẾ HỆ THỐNG DẬP LỬA THEO ĐỊNH HƯỚNG NGUỒN NHIỆT

Giáo viên hướng dẫn: Ths. Phan Ngọc Kỳ
Sinh viên thực hiện: Nguyễn Xuân
Lớp: 18D4
Mã sinh viên: 1811505120261

Sinh viên thực hiện: Huỳnh Văn Sĩ
Lớp: 18D3
Mã sinh viên: 1811505120143

Đà Nẵng, Ngày 3 Tháng 6 Năm 2022

HD 1

Phụ lục 06-*Khoa/Bộ môn có thể xem xét điều chỉnh, bổ sung các tiêu chí đánh giá phù hợp với đề cương chi tiết học phần ĐATN*

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
KHOA ĐIỆN TỬ- VIỄN THÔNG

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

NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP
(*Dành cho người hướng dẫn*)

I. Thông tin chung:

1. Sinh viên thực hiện: **Huỳnh Văn Sĩ**
Nguyễn Xuân

2. Lớp: 18D3 Mã SV: 1811505410117
1811505410119

3. Tên đề tài: **Ứng dụng công nghệ EDGE AI thiết kế hệ thống dập lửa theo định hướng nguồn nhiệt.**

4. Người hướng dẫn: Phan Ngọc Kỳ. Học hàm/ học vị: Thạc sỹ.

II. Nhận xét, đánh giá đồ án tốt nghiệp:

1. Về tính cấp thiết, tính mới, mục tiêu của đề tài: (điểm tối đa là 1đ)

Đề tài khá phức tạp cho mục tiêu phát hiện lửa và điều khiển hệ thống chữa cháy (vòi phun) với kiến thức hạn chế trong những ngày đầu thực hiện, nhóm sinh viên đã nỗ lực tìm kiếm các giải pháp công nghệ cũng như yêu cầu thực thi trên các dòng chíp cao cấp STM32 Cortex-M mà sinh viên chưa từng được trang bị để hoàn thiện mô hình.

Mô hình thực hiện trên nền tảng cong nghệ AI, nhận diện lửa trên Google Colab bằng ngôn ngữ Python.

Kết quả mô hình chạy hoàn toàn nhúng trên trên chip STM32 Cortex-M Mô hình được triển khai trên vi điều khiển STM32 có thể mở rộng phạm vi ứng dụng của đề tài.

2. Về kết quả giải quyết các nội dung nhiệm vụ yêu cầu của đồ án: (điểm tối đa là 4đ)

Với các nhiệm cụ được giao, nhóm sinh viên (tác giả) đã tích cực vận dụng kiến thức đã học, tìm kiếm thông tin và thực nghiệm để đưa ra mô hình đáp ứng yêu cầu đặt ra.

3. Về hình thức, cấu trúc, bố cục của đồ án tốt nghiệp: (điểm tối đa là 2đ)

Bố cục báo cáo của đồ án được chia làm 4 chương. Nội dung, định dạng các chương theo đúng yêu cầu, đúng quy định.

4. Kết quả đạt được, giá trị khoa học, khả năng ứng dụng của đề tài: (điểm tối đa là 1đ)

Nhóm tiến hành tìm các giải pháp để thực hiện đồ án, tiến hành so sánh để tìm ra giải pháp được đưa ra. Và nhóm đã tìm được giải pháp tối ưu nhất cho đồ án là sử dụng công nghệ Edge AI để xử lý hình ảnh và nhúng trên vi điều khiển STM32.

Nhóm tiến hành thu thập dữ liệu là các hình ảnh về lửa để tiến hành tạo mô hình AI nhận diện lửa.

Xây dựng một hình AI nhận diện lửa trên Google Colab bằng ngôn ngữ Python.

Sử dụng công cụ STM32Cube AI để chuyển đổi thành mã C và nhúng mô hình xuống vi điều khiển STM32 và chạy mô hình và kiểm thử trên vi điều khiển STM32 trong thực tế.

5. Các tồn tại, thiếu sót cần bổ sung, chỉnh sửa:

Lỗi chính tả vfa định dạng văn bản.

III. Tinh thần, thái độ làm việc của sinh viên: (điểm tối đa 2đ)

Nhóm sinh viên đã thực hiện với tinh thần rất tích cực, khắc phục mọi khó khăn do tình hình dịch bệnh, tự trang bị các dụng cụ và thường xuyên liên hệ với giáo viên hướng dẫn trong suốt thời gian thực hiện đề tài.

IV. Đánh giá:

1. Điểm đánh giá: **Huỳnh Văn Sĩ** 9.75/10
   Nguyễn Xuân 9.75/10
2. Đề nghị: ☒ Được bảo vệ đồ án ☐ Bổ sung để bảo vệ ☐ Không được bảo vệ

Đà Nẵng, ngày 04 tháng 06 năm 2022.

Người hướng dẫn

Phan Ngọc Kỳ

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
KHOA ĐIỆN – ĐIỆN TỬ

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

NHẬN XÉT PHẢN BIỆN ĐỒ ÁN TỐT NGHIỆP
(*Dành cho người phản biện*)

I. Thông tin chung:

1. Họ và tên sinh viên: Nguyễn Xuân [(1)]; Huỳnh Văn Sĩ[(2)]
2. Lớp: [(1)]18D4; [(2)]18D3 Mã SV: [(1)]1811505120261; [(2)]1811505120143
3. Tên đề tài: ỨNG DỤNG CÔNG NGHỆ EDGE AI THIẾT KẾ HỆ THỐNG DẬP LỬA THEO ĐỊNH HƯỚNG NGUỒN NHIỆT

4. Người phản biện: GVC. Nguyễn Văn Thịnh Học hàm/ học vị: Thạc sĩ

II. Nhận xét, đánh giá đồ án tốt nghiệp:

1. Về tính cấp thiết, tính mới, mục tiêu của đề tài:

   Đạt tính cấp thiết, có tính mới và thực hiện đúng mục tiêu đề ra.

2. Về kết quả giải quyết các nội dung nhiệm vụ yêu cầu của đồ án:

   Đã hoàn thành nhiệm vụ đồ án đăng ký

3. Về hình thức, cấu trúc, bố cục của đồ án tốt nghiệp:

   Bố cục rõ ràng, hợp lý. Tuy nhiên có những lỗi về câu, từ, mục chưa hợp lý cần sửa chửa lại.

4. Kết quả đạt được, giá trị khoa học, khả năng ứng dụng của đề tài:

   Đã thi công thành công mô hình dập lửa theo định hướng nguồn nhiệt bằng công nghệ EDGE AI.

5. Các tồn tại, thiếu sót cần bổ sung, chỉnh sửa:

   Cần rà soát lại toàn bộ báo cáo về chính tả, từ ngữ, câu, đề mục; quy định chú thích hình, bảng. Ví dụ hình 2.17 là bảng chứ không phải hình.

   Cần giải thích chi tiết các đại lương trong công thức tại trang 23, ý nghĩa của công thức.

   Cần trích dẫn tài liệu tham khảo vào cuối các đoạn, mục nội dung trong báo cáo.

TT	Các tiêu chí đánh giá	Điểm tối đa	Điểm đánh giá
1	**Sinh viên có phương pháp nghiên cứu phù hợp, giải quyết các nhiệm vụ đồ án được giao**	**8,0**	**7,0**

1a	- Tính cấp thiết, tính mới (nội dung chính của ĐATN có những phần mới so với các ĐATN trước đây); - Đề tài có giá trị khoa học, công nghệ; giá trị ứng dụng thực tiễn;	1,0	1,0
1b	- Kỹ năng giải quyết vấn đề; hiểu, vận dụng được kiến thức cơ bản, cơ sở, chuyên ngành trong vấn đề nghiên cứu; - Khả năng thực hiện/phân tích/tổng hợp/đánh giá; - Khả năng thiết kế, chế tạo một hệ thống, thành phần, hoặc quy trình đáp ứng yêu cầu đặt ra;	3,0	2,5
1c	- Chất lượng sản phẩm ĐATN về nội dung báo cáo, bản vẽ, chương trình, mô hình, hệ thống,…;	3,0	2,5
1d	- Có kỹ năng sử dụng phần mềm ứng dụng trong vấn đề nghiên cứu (thể hiện qua kết quả tính toán bằng phần mềm); - Có kỹ năng sử dụng tài liệu liên quan vấn đề nghiên cứu (thể hiện qua các tài liệu tham khảo).	1,0	1,0
2	**Kỹ năng trình bày báo cáo đồ án tốt nghiệp**	**2,0**	**1,0**
2a	- Bố cục hợp lý, lập luận rõ ràng, chặt chẽ, lời văn súc tích;	1,0	0,5
2b	- Hình thức trình bày.	1,0	0,5
3	**Tổng điểm theo thang 10** (lấy đến 1 số lẻ thập phân)		**8,0**

- Câu hỏi đề nghị sinh viên trả lời trong buổi bảo vệ: ……
  ……...
  ……...
  ……...
  ……...
  ……...
- Đề nghị: ☒ Được bảo vệ đồ án ☐ Bổ sung để bảo vệ ☐ Không được bảo vệ

Đà Nẵng, ngày 15 tháng 6 năm 2022

Người phản biện

Nguyễn Văn Thịnh

TÓM TẮT

Họ tên sinh viên: Huỳnh Văn Sĩ Lớp: 18D3 Mã SV: 1811505120143
Nguyễn Xuân 18D4 1811505120261

Khoa: Điện-Điện tử
Ngành: Kỹ thuật Điện tử

Tên đề tài: **ỨNG DỤNG CÔNG NGHỆ EDGE AI THIẾT KẾ HỆ THỐNG DẬP LỬA THEO ĐỊNH HƯỚNG NGUỒN NHIỆT.**

Trong những năm gần đây công nghệ AI được sử dụng trong rất nhiều lĩnh vực khác nhau, với sự ra đời của một chức năng mới là học sâu (Deep Learnig) thì độ chính xác và tốc độ xử lý của một mô hình càng cao. Tuy nhiên từ trước tới nay những mô hình AI thường được chạy trên những máy tính có tốc độ xử lý và chi phí đắt đỏ làm cho người dùng khó tiếp cận. Nhưng trong những năm gần đây một công nghệ mới mang tên Edge AI xuất hiện, với công nghệ này giúp cho người dùng có thể chạy những mô hình AI trên những thiết bị hạn chế về phần cứng như vi điều khiển. Sau quá trình tìm hiểu thì nhóm quyết định sử dụng công nghệ này cho đồ án của mình.

Một số công cụ được nhóm sử dụng để làm đồ án:

- Google colab: dùng để xây dựng mô hình AI nhận diện lửa
- STM32 Cube IDE: xây dựng thuật toán chính trong đề tài
- STM32Cube AI: chuyển đổi mô hình AI trên python về mã C.
- OpenMV IDE: xây dựng thuật toán để chụp ảnh và gửi ảnh về vi điều khiển

Đồ án của nhóm chia thành năm giai đoạn chính:

Giai đoạn 1: Nhóm tiến hành tìm các giải pháp để thực hiện đồ án, tiến hành so sánh để tìm ra giải pháp được đưa ra. Và nhóm đã tìm được giải pháp tối ưu nhất cho đồ án là sử dụng công nghệ Edge AI để xử lý hình ảnh và nhúng trên vi điều khiển STM32.

Giai đoạn 2: Nhóm tiến hành thu thập dữ liệu là các hình ảnh về lửa để tiến hành tạo mô hình AI nhận diện lửa.

Giai đoạn 3: Nhóm tiến hành xây dựng một hình AI nhận diện lửa trên Google Colab bằng ngôn ngữ Python.

Giai đoạn 4: Sử dụng công cụ STM32Cube AI để chuyển đổi thành mã C và nhúng mô hình xuống vi điều khiển STM32.

Giai đoạn 5: Chạy mô hình và kiểm thử trên vi điều khiển STM32 trong thực tế.

LỜI NÓI ĐẦU

Là sinh viên khoa Điện – Điện Tử của trường Đại học Sư Phạm Kỹ Thuật – Đại học Đà Nẵng, sau 4 năm học tập và rèn luyện dưới sự chỉ bảo của thầy cô giúp chúng em tích lũy được rất nhiều kiến thức. Và đồ án tốt nghiệp này là thành quả của quá trình học tập và rèn luyện tại trường.

Nhóm xin gửi lời cảm ơn đến thầy Phan Ngọc Kỳ-Giáo viên hướng dẫn đồ án, đã giúp đỡ nhóm rất nhiều trong quá trình thực hiện đồ. Với sự chỉ dẫn nhiệt tình và định hướng đúng đắn của thầy đã giúp nhóm tìm ra được những giải pháp cho đề tài của mình.

Nhóm cũng gửi lời cảm ơn đến các thầy cô ở Trường Đại học Sư Phạm Kỹ Thuật – Đại học Đà Nẵng nói chung và các thầy cô ở Khoa Điện – Điện tử nói riêng đã dạy và truyền đạt những kiến thức từ những môn đại cương đến những môn chuyên nghành trong suốt bốn năm qua, từ những kiến thức nền này đã giúp nhóm phát triển lên để làm đồ án này.

Nhóm cũng gửi lời cảm ơn đến bạn bè ở trường và các anh chị ở các cộng đồng về điện tử đã góp ý, chia sẽ kiến thức giúp nhóm hoàn thành được đồ án.

Xin chân thành cảm ơn!

Người thực hiện đề tài
Huỳnh Văn Sĩ
Nguyễn Xuân

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Giảng viên hướng dẫn: Ths. Phan Ngọc Kỳ

Sinh viên thực hiện: Huỳnh Văn Sĩ Mã SV: 1811505120143

Nguyễn Xuân Mã SV: 1811505120261

1. Tên đề tài: ỨNG DỤNG CÔNG NGHỆ EDGE AI THIẾT KẾ HỆ THỐNG DẬP LỬA THEO ĐỊNH HƯỚNG NGUỒN NHIỆT.

2. Các số liệu ban đầu:

Mô hình cần kiểm tra là một hệ thống báo cháy và chữa cháy ứng dụng công nghệ xử lý ảnh có đặc tính sau:

- Kích thước: 18cm x 18cm.
- Chất liệu: Board đồng, bìa Carton, giá đỡ nhựa.

3. Nội dung chính của đồ án:

Nội dung 1: Tìm hiểu, lựa chọn các giải pháp, các linh kiện dùng trong đồ án
Nội dung 2: Tính toán, thiết kế các khối chức năng của hệ thống.
Nội dung 3: Thiết kế, thi công mô hình hệ thống
Nội dung 4: Vận hành thử nghiệm và hiệu chỉnh lỗi.
Nội dung 5: Đánh giá kết quả thực hiện.

4. Các sản phẩm dự kiến

5. Ngày giao đồ án: 21/02/2022

6. Ngày nộp đồ án: 06/05/2022

Đà Nẵng, ngày tháng năm 2022.

Người hướng dẫn

LỊCH TRÌNH THỰC HIỆN ĐỒ ÁN

Giảng viên hướng dẫn: Ths. Phan Ngọc Kỳ

Sinh viên thực hiện: Huỳnh Văn Sĩ Mã SV: 1811505120143

Nguyễn Xuân Mã SV: 1811505120261

Đề tài:

ỨNG DỤNG CÔNG NGHỆ EDGE AI THIẾT KẾ HỆ THỐNG DẬP LỬA THEO ĐỊNH HƯỚNG NGUỒN NHIỆT.

TT	Thời gian	Nội dung công việc	Kết quả dự kiến đạt được
1	21-27/02/2022	Găp giáo viên hướng dẫn và nhận đề tài, viết đề cương.	Viết nhiệm vụ và đề cương
2	28/2-6/3/2022	Tìm hiểu các hệ thống báo cháy và chữa cháy đang được các tòa nhà sử dụng	Hiểu được các công nghệ báo cháy và chữa cháy đang được sử dụng nhiều.
3	7-13/3/2022	Tìm hiểu các giải pháp cho đề tài	Chọn công nghệ AI(Deep Learnig) là giải pháp cho đề tài.
4	14-20/3/2022	Tiến hành thiết kế sơ đồ khối	Hình dung được tổng quát mô hình
5	21-27/3/2022	Tìm hiểu về vi điều khiển STM32	Nắm được các kiến thức về vi điều khiển STM32.
6	28/3-3/4-2022	Tìm hiểu các ngoại vi trên STM32 như ADC, Timer,UART,..	Thực hành giao tiếp với các ngoại vi của vi điều khiển STM32.
7	4-10/4/2022	Tìm hiểu về Google Colab và ngôn ngữ Python.	Sử ngôn ngữ Python lập trình trên Google Colab.
8	11-17/4/2022	Tiến hành thu thập dữ liệu về cháy và không cháy để Train Model AI	Có được dữ liệu đầu vào đúng với yêu cầu của mô hình.
9	18-24/4/2022	Tìm hiểu về cấu trúc mạng CNN và thư viện Keras.	Nắm được cấu trúc mạng CNN và cách sử dụng mạng Keras.

10	25/4-1/5/2022	Đánh nhãn và xử lý dữ liệu đầu vào cho mô hình AI.	Chia ảnh thành các phần có lửa và không lửa để đánh nhãn.
11	2-8/5/2022	Huấn luyện mô hình AI nhận diện lửa.	Có được mô hình AI với độ chính xác trên 90%.
12	9-15/5/2022	Đánh giá mô hình AI trên tập dữ liệu mới.	Kiểm tra mô hình có nhận diện đúng không.
13	16-22/5/2022	Sử dụng công cụ STM32Cube AI để chuyển đổi mô hình thành C.	Kiểm tra mức độ phức tạp của mô hình để tìm được vi điều khiển thíc hợp.
14	23-29/5/2022	Kết nối STM32 với các phần cứng liên quan.	Thu thập ảnh đưa vào STM32 xử lý và điều khiển các động cơ.
15	30/5-6/6/2022	Hoàn thiện chương trình và viết báo cáo.	Chạy chương trình trong thực tế và hiệu chỉnh lại mô hình.

Đà Nẵng, ngày .....tháng .....năm 2022

NGƯỜI HƯỚNG DẪN

LỜI CAM ĐOAN

Đề tài này là công sức nghiên cứu của nhóm trong nhiều tháng qua, đề tài có tham khảo một số tài liệu được nhóm liệt kê trong phần phụ lục. Và không sao chép từ các luận văn hay đồ án đã có trước đó.

Người thực hiện đề tài
Huỳnh Văn Sĩ
Nguyễn Xuân

MỤC LỤC

DANH MỤC BẢNG VÀ HÌNH ẢNH

DANH MỤC CÁC CHỮ VIẾT TẮT

AI: Artificial intelligence.

CNN: Convolutional Neural Networks.

ANN: Artificial Neural Networks.

NN: Neural Networks.

CPU: Central Processing Unit.

GPU: Graphics Processing Unit.

RAM: Random Access Memory.

DMA: Direct memory access.

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1 Đặt vấn đề:

Hiện nay hỏa hoạn là một tai nạn để lại hậu quả nặng nề nhất về cả tính mạng lẫn tài sản. Theo thống kê của cục phòng cháy chữa cháy chỉ trong 5/2022 trên toàn quốc xảy ra 154 vụ cháy. Thiệt hại khoảng 33,23 tỷ đồng, 6,36ha rừng và làm hàng chục người bị thương.

Hình 1-1: *Cháy nhà máy ở Quảng Nam.*

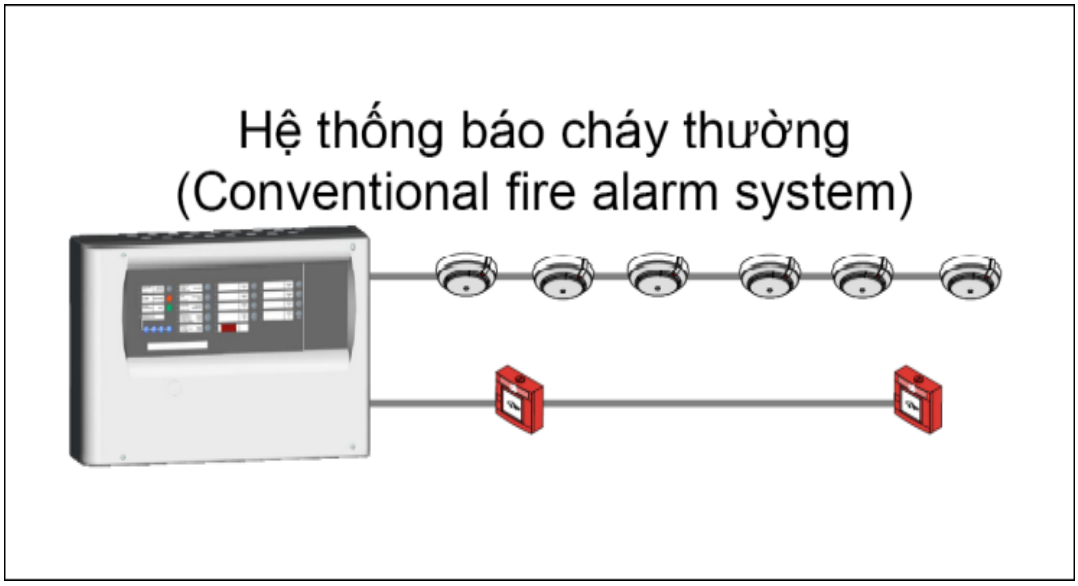
Vì vậy việc phát hiện sớm các nguy cơ dẫn đến cháy như khí gas, khói, lửa rất quan trọng. Khi đã phát hiện cháy ta tiến hành báo động để sơ tán và áp dụng công nghệ xử lý ảnh để phát hiện đám cháy và tiến hành dập lửa sẽ tránh được tối đa những thiệt hại.

1.2 Các giải pháp hiện tại:

1.2.1 Hệ thống báo cháy thông thường:

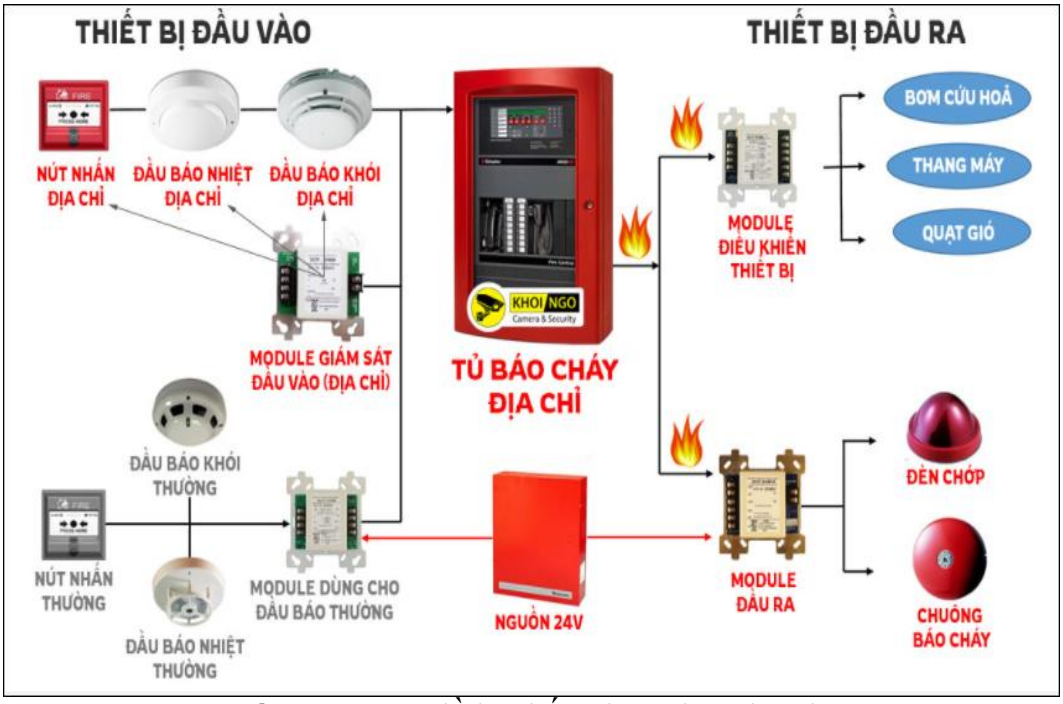
Với tính năng đơn giản, giá thành không cao, hệ thống báo cháy thông thường chỉ thích hợp lắp đặt tại các công ty có diện tích vừa hoặc nhỏ(Khoảng vài ngàn m2, số

lượng các phòng không nhiều(Vài chục phòng); lắp đặt cho những nhà, xưởng nhỏ… Các thiết bị trong hệ thống được mắc nối tiếp với nhau và mắc nối tiếp với trung tâm báo cháy, nên khi xảy ra sự cố trung tâm chỉ có thể nhận biết khái quát và hiển thị toàn bộ khu vực (zone) mà hệ thống giám sát (chứ không cho biết chính xác vị trí từng đầu báo, từng địa điểm có cháy). Điều này làm hạn chế khả năng xử lý của nhân viên giám sát.



Hình 1-2: *Hình ảnh minh họa về hệ thống báo cháy thông thường.*

1.2.2 Hệ thống báo cháy địa chỉ.

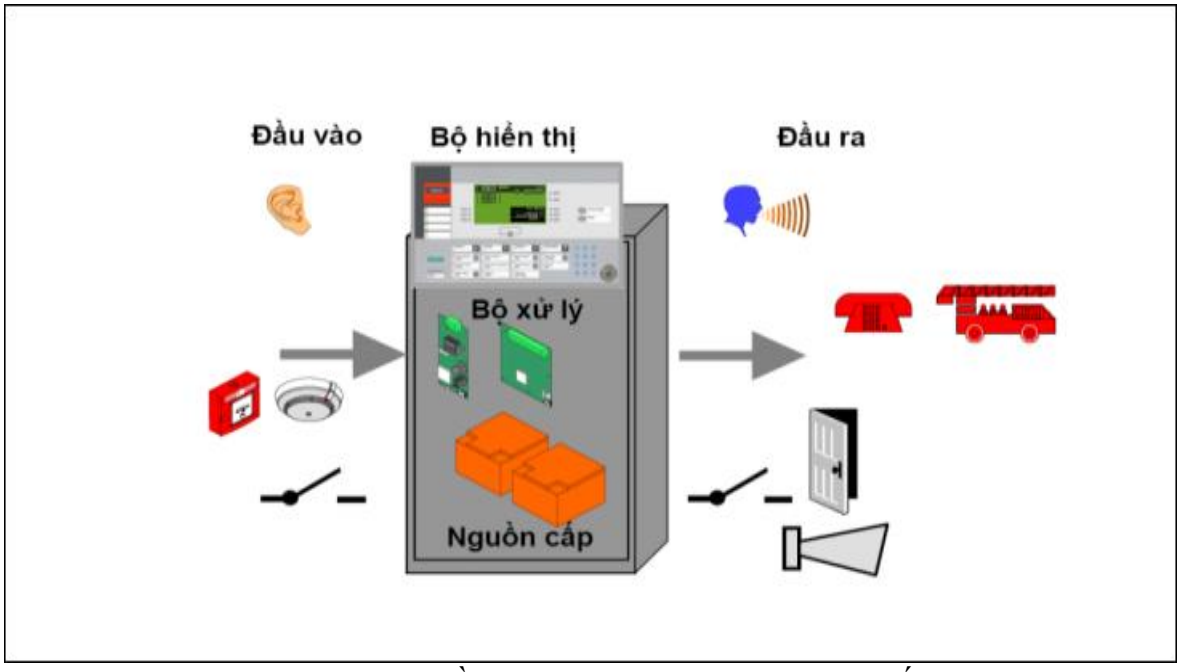


Hình 1-3: *Sơ đồ hệ thống báo cháy địa chỉ*

Với tính năng kỹ thuật cao, hệ thống báo cháy địa chỉ dùng để lắp đặt tại các công trình mà mặt bằng sử dụng rộng lớn (vài chục ngàn m2), được chia ra làm nhiều khu vực độc lập, các phòng ban trong từng khu vực riêng biệt với nhau. Từng thiết bị trong hệ thống được mắc trực tiếp vào trung tâm báo cháy giúp trung tâm nhận tín hiệu xảy ra cháy tại từng khu vực, từng địa điểm một cách rõ ràng, chính xác. Từ đó trung tâm có thể nhận biết thông tin sự cố một cách chi tiết và được hiển thị trên bảng hiển thị phụ giúp nhân viên giám sát có thể xử lý sự cố một cách nhanh chóng.

1.2.3 Nguyên lý hoạt động:

Quy trình hoạt động của hệ thống báo cháy là một quy trình khép kín. Khi có hiện tượng về sự cháy (chẳng hạn như nhiệt độ gia tăng đột ngột, có sự xuất hiện của khói hoặc các tia lửa), các thiết bị đầu vào (đầu báo, công tắc khẩn) nhận tín hiệu và truyền thông tin của sự cố về trung tâm báo cháy. Tại đây trung tâm sẽ xử lý thông tin nhận được, xác định vị trí nơi xảy ra sự cháy thông qua các zone (đối với hệ thống báo cháy thường) hoặc thông qua địa chỉ (đối với hệ thống báo cháy địa chỉ) và truyền thông tin đến các thiết bị đầu ra (bảng hiển thị phụ, chuông, còi, đèn), các thiết bị này sẽ phát tín hiệu âm thanh, ánh sáng để mọi người nhận biết khu vực đang xảy ra sự cháy và xử lý kịp thời.



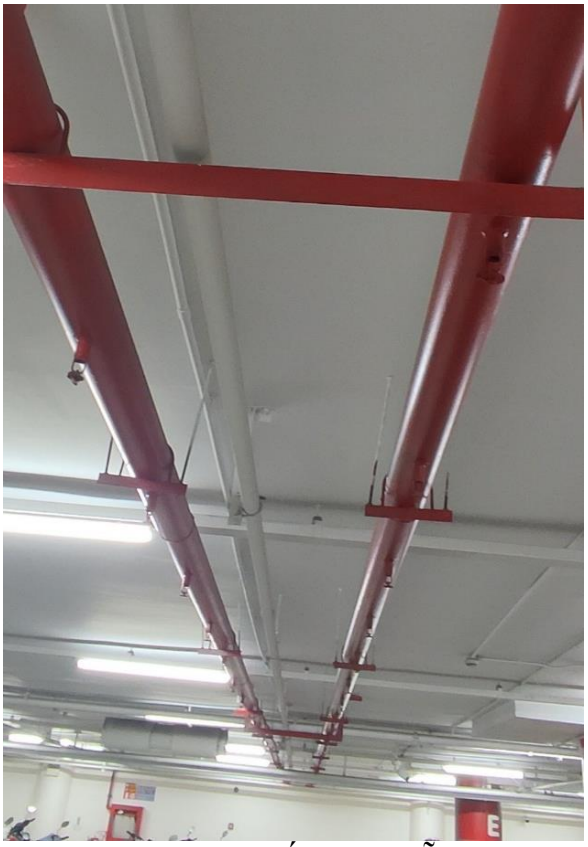
Hình 1-4: *Sơ đồ nguyên lý hoạt động của hệ thống*

1.2.4 Khảo sát các hệ thống báo cháy, chữa cháy được sử dụng ở siêu thị GO:

Hệ thống bao gồm 3 phần: một đầu báo cháy, một loa để phát ra cảnh báo và một ống dẫn nước để chữa cháy.

Hình 1-5: *Đầu báo khói thực tế*

Khi có cháy và cảm biến khói phát hiện có khói thì sẽ kích hoạt các hệ thống vòi phun nước đa hướng phun nước để dập lửa, các hệ thống van nước này được lắp đặt khắp tòa nhà để dập lửa ở bất kì vị trí nào của tòa nhà.



Hình 1-6: *Hệ thống van dẫn nước.*

Ngoài ra còn có các thiết bị để chữa cháy bằng tay như bình chữa cháy và vòi phun nước công suất lớn.

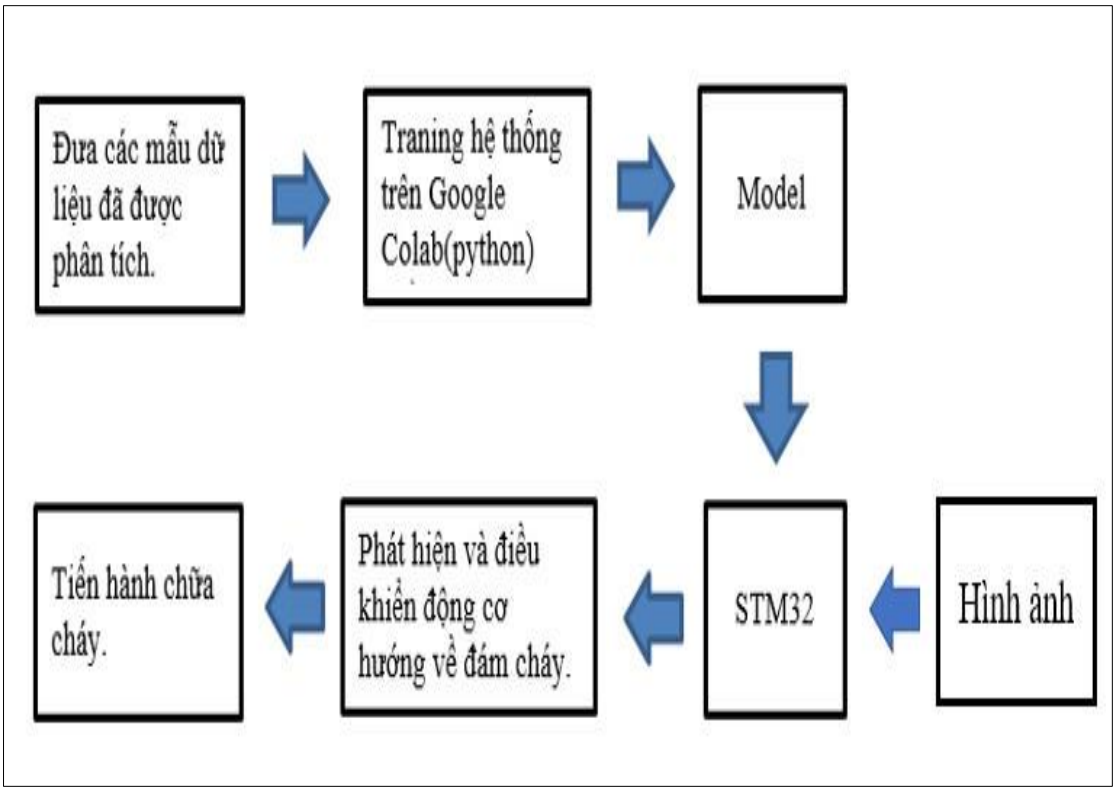
Hình 1-7: *Bình chữa cháy và vòi phun nước công suất lớn.*

Nhược điểm của hệ thống chữa cháy trên: Với những thiết bị phát hiện cháy bằng khói thì khó để phát hiện được những đám cháy nhỏ lúc, vì lúc cháy lớn mới phát sinh ra khói nên lúc đó việc chữa cháy sẽ khó khăn hơn. Lúc chữa cháy hệ thống sẽ tiến hành phun nước cả một khu vực đươc cho là đang có cháy, việc này vô tình gây ảnh hưởng tới vật dụng khác như giấy tờ, …

1.3 Đề xuất giải pháp:

Với vấn đề đã nêu ra ở phần trước, nhóm đã quyết định xây dựng một hệ thống phát hiện đúng vị trí cháy bằng camera và tiến hành dập lửa để tránh đám cháy lan rộng ra gây thiệt hại về tài sản.

Việc xử lý ảnh để phát hiện có cháy hay không thường dùng các máy tính hoặc máy tính nhúng có cấu hình cao để xử lý. Việc này khiến cho việc xây dựng hệ thống này trở nên tốn kém chi phí. Nên nhóm đã sử dụng một công nghệ khác mang tên Edge AI - ứng dụng trí tuệ nhân tạo trên các thiết bị cạnh. Đây là một công nghệ được sử dụng nhiều trong những năm gần đây, công nghệ này cho phép nhúng các mô hình AI xuống các thiết bị vi điều khiển có bộ nhớ và tốc độ xử lý thấp thay vì phải thực hiện trên các máy tính và siêu máy tính như trước.



Hình 1-8: *Sơ đồ khối tổng quan mô hình chữa cháy định hướng nguồn nhiệt*

Trong đề tài này đầu tiên ta sẽ tạo một hình nhận diện lửa AI, tiếp theo sẽ nhúng mô hình này xuống vi điều khiển STM32 để chạy trong thực tế.

Lý do chọn đề tài:

- Dập được nguồn cháy, tránh tình trạng cháy lan.
- Không gây ảnh hưởng tới các tài sản khác.
- Tiết kiệm được nguồn nước.

Mục tiêu của đề tài:

Xây dựng một hệ thống dập lửa theo định hướng nguồn nhiệt ở trong nhà, hệ thống chạy trên vi điều khiển STM32 để tiết kiệm chi phí.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về AI:

2.1.1 AI là gì?

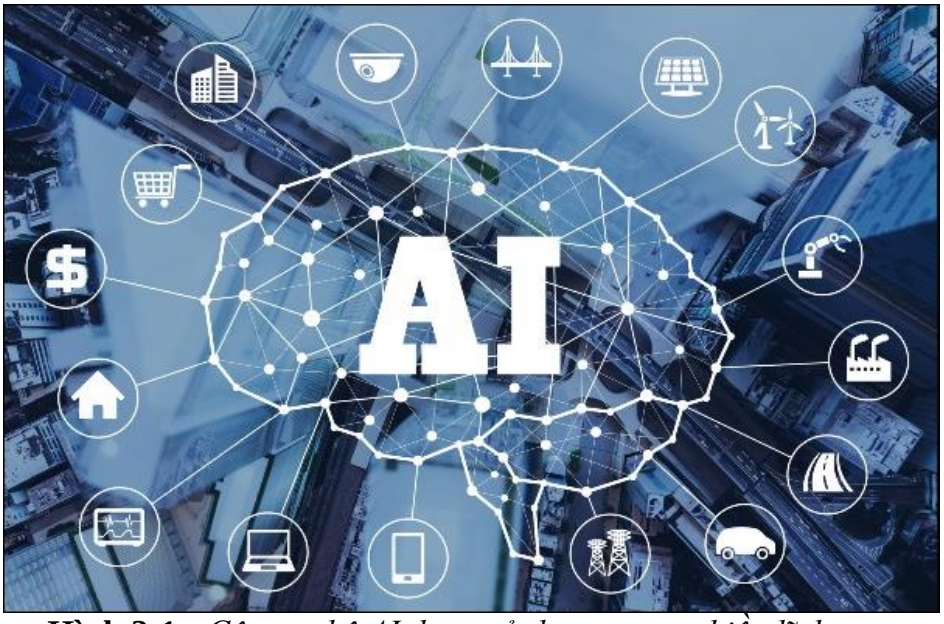
AI (Artificial Intelligence) hay còn gọi là trí tuệ nhân tạo là một thuật ngữ để nói về trí tuệ của các thiết bị như máy tính, robot… được con người lập trình hoặc huấn luyện để có thể tự động làm những công việc như con người.

Các sự kiện AI nổi bật chấn động thế giới khi AI đánh bại được con người

- IBM's Deep Blue Defeats Garry Kasparov in Chess [1996-1997]
- IBM Watson defeats Brad Rutter and Ken Jennings in Jeopardy [2011]
- Deepmind's AlphaGo defeats Lee Sudol in game of Go [2016]
- Deepmind's Alphastar beats StarCraft 2 Champion [2018]
- Facebook's Pluribus defeats professional poker players [2019]

Về thời gian: 1996 -> 2011 -> 2016 -> 2018 -> 2019

Trí tuệ nhân tạo khác với chương trình máy tính thông thường ở chỗ, chương trình máy tính thường sử dụng các câu lệnh dạng if – else để xử lý công việc theo điều kiện đã lập trình từ trước nếu có một sự kiện nằm ngoài việc đã lập trình thì chương trình sẽ dừng. Còn đối với AI nó dựa trên nguyên tắc trí thông minh của con người, nó có thể học để trở nên thông minh hơn và đặc biệt điểm mạnh của AI là khả năng suy luận để giải quyết một vấn đề nào đó.

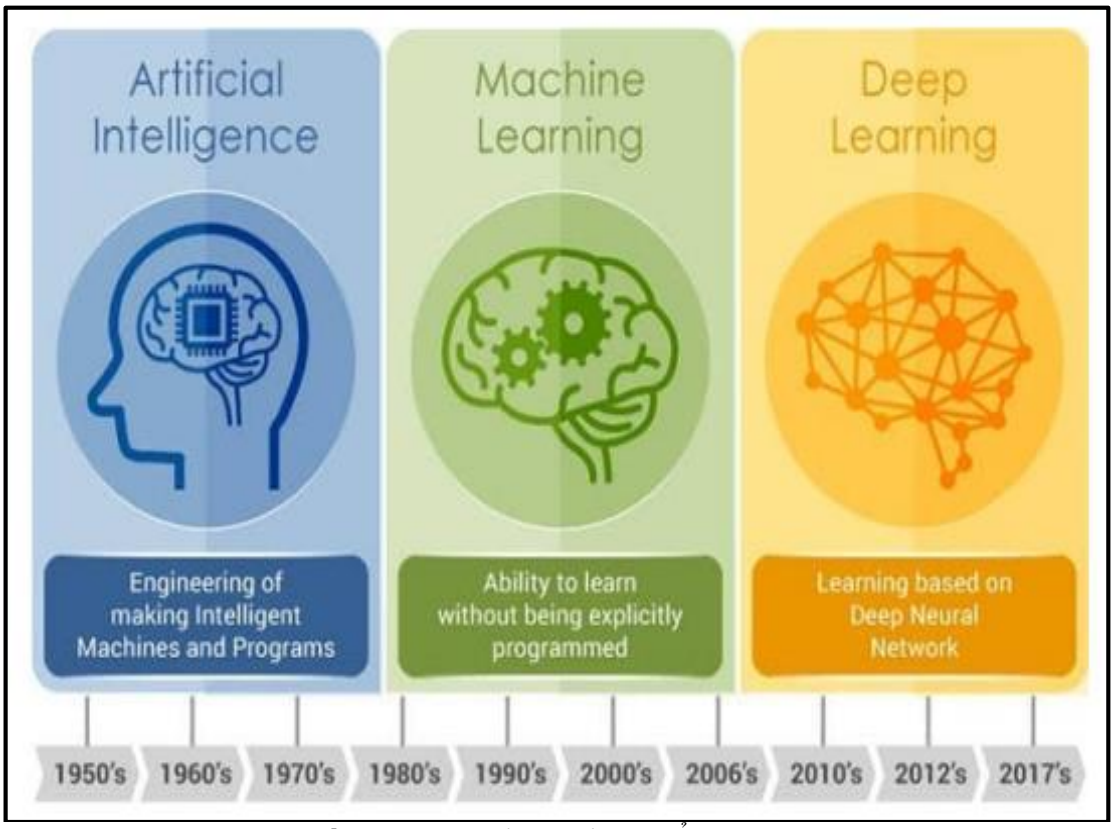


Hình 2-1: *Công nghệ AI được sử dụng trong nhiều lĩnh vực.*

Các ứng dụng của trí tuệ nhân tạo đang được áp dụng cho nhiều lĩnh vực và các ngành công nghiệp khác nhau. Trí tuệ nhân tạo đang được thử nghiệm trong lĩnh vực chăm sóc sức khỏe, nó có thể dự đoán và đề xuất các phương pháp điều trị cho bệnh nhân. Và các ví dụ về AI khác như Siri, ô tô tự lái, đề xuất trên công cụ tìm kiếm của Google.

2.1.2 Phân loại AI:

Vào những năm 50 của thế kỷ trước thì những cỗ máy, những chương trình AI đầu tiên được sử dụng. Những chương trình AI ở giai đoạn này còn khá thô sơ vì hạn chế về tài nguyên máy tính, một chương trình nổi bật là máy tính có khả năng tự học để chơi cờ. Nhưng AI thực sự được sử dụng nhiều là vào năm 1980 trở đi khi một khái niệm mới là học máy (Machine Learning) ra đời và thực sự bùng nổ khi xuất hiện học sâu (Deep Learning).



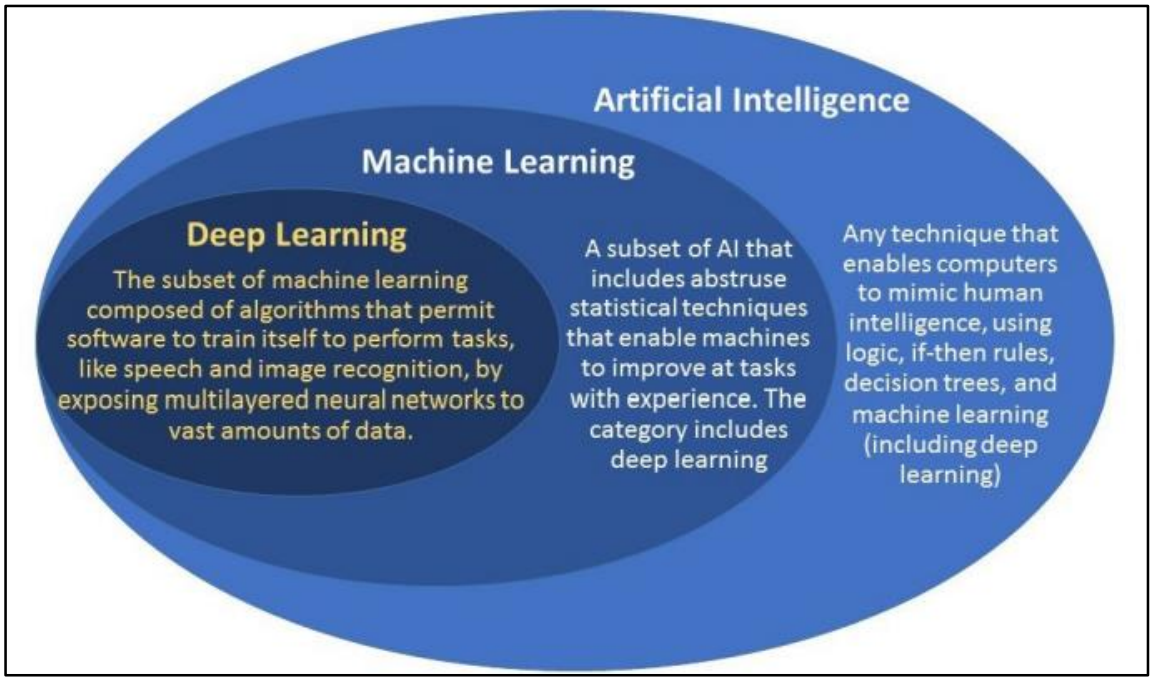
Hình 2-2: *Lịch sử phát triển của AI.*

Có hai cách để xây dựng một mô hình học:

- Cách 1: Viết ra các quy luật và chương trình sẽ thực hiện theo luận đó, ví dụ như if-else.
- Cách 2: Xây dựng chương trình giúp máy có khả năng tự học chứ không phải theo các luật quy định sẵn.

2.1.3 Machine learning là gì?

Học máy (Machine Learning) là một tập hợp con của AI, nó ứng dụng các thuật toán để phân tích các dữ liệu đầu tìm những đặc trưng của dữ liệu đó, học hỏi từ những dữ liệu đó và sau đó đưa ra các dự đoán hoặc thực hiện một quyết định về các vấn đề liên quan.



Hình 2-3: *Các tập hợp con của AI.*

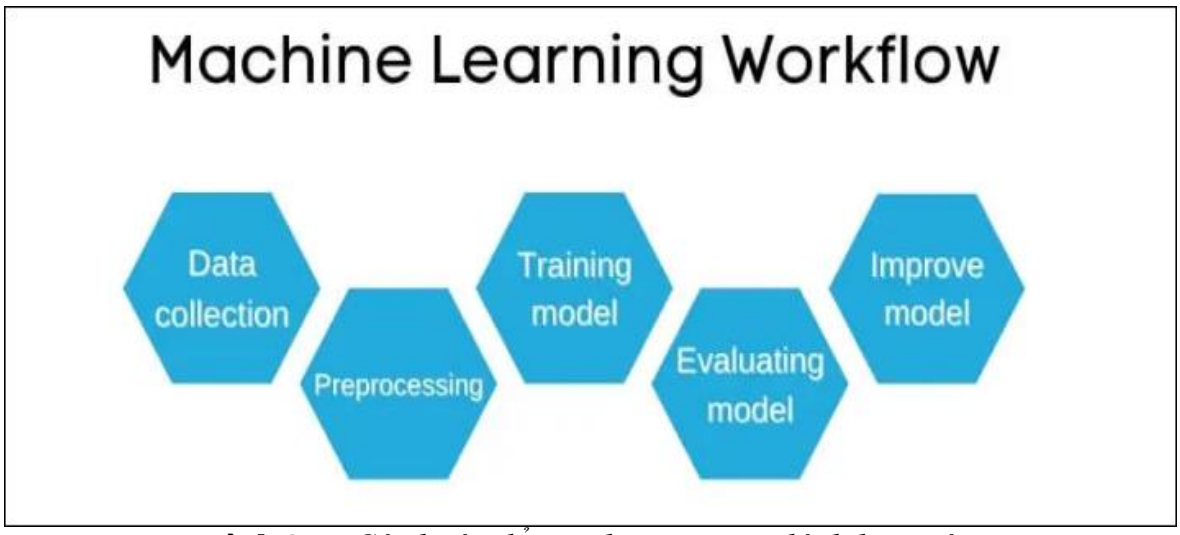
Học máy được sinh ra để chứng minh máy tính có thể học mà không cần được lập trình để thực hiện các công việc cụ thể. Đối với học máy nó có khả năng sử dụng và lặp đi lặp lại những phép toán phức tạp, việc này rất quan trọng vì khi mô hình nhận các dữ liệu mới, chúng có khả năng thích ứng độc lập mà không cần sự can thiệp của con người.

Học máy khác với trí tuệ nhân tạo, đối với trí nhân tạo đây là một nghành khoa học áp dụng các kỹ thuật và công nghệ để bắt chước trí tuệ và hành động của con người. Còn học máy nó là một tập hợp con giúp đào tạo ra những cỗ máy có khả năng tự học và suy luận.

Quy trình làm việc của một hệ thống Machine Learning:

- Thu thập dữ liệu: Để huấn luyện một mô hình máy học trước tiên ta cần xây dựng một bộ dữ liệu. Ta cần nên thu thập dữ liệu ở các nguồn chính thống để đảm bảo dữ liệu được chính xác.
- Quy trình xử lý phân loại dữ liệu đầu vào: Các thuật toán học máy được sử dụng để đưa ra các dự đoán và phân loại. Mô hình học máy dựa trên các dữ liệu đầu vào đã được gắn nhãn và tiến hành phân loại các dữ liệu này.

- Huấn luyện mô hình: Khi đã có dữ liệu ta tiến hành huấn luyện mô hình, mô hình sử dụng các hàm và các lớp lọc để tiến hành trích xuất đặc trưng của dữ liệu vào.
- Quy trình đánh giá: Khi đã huấn luyện mô hình xong tức ta đã cho máy học thành công. Ta sử dụng các hàm để thực hiện so sánh dựa trên kết quả dự đoán và kết quả thực tế để kết luận độ chính xác của mô hình
- Quy trình tối ưu hóa mô hình: Sau khi đã đánh giá mô hình, nếu độ chính xác dưới 80% tức độ chính xác chưa cao thì ta tiến hành huấn luyện lại mô hình.

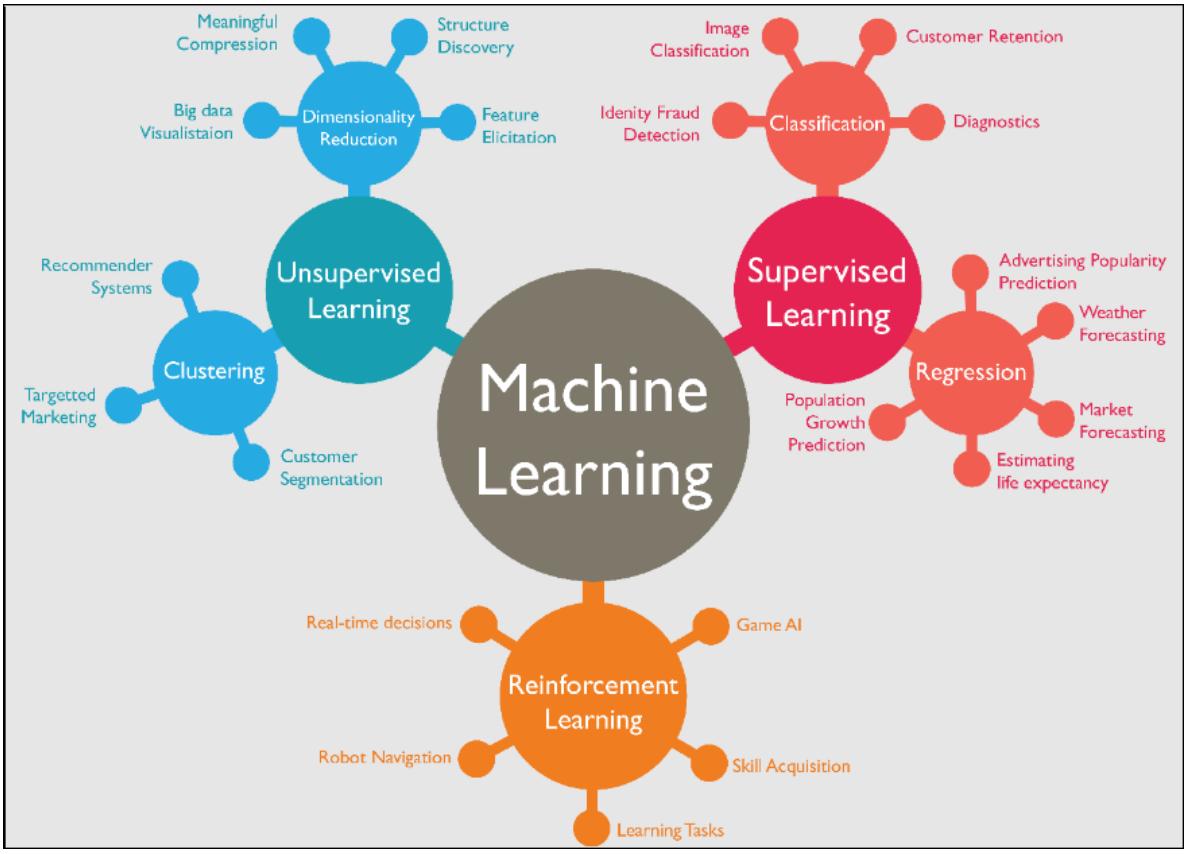


Hình 2-4: *Các bước để xây dựng một mô hình học máy.*

Có 4 kỹ thuật học máy:

- Học có giám sát (Supervised machine learning): Được định nghĩa bằng cách sử dụng các tập dữ liệu đã được gắn nhãn để huấn luyện một mô hình học máy nhằm nâng cao kết quả dự đoán một cách chính xác. Hạn chế của học máy có giám sát cần cung cấp nhiều dữ liệu được gắn nhãn, trong nhiều dự án lớn việc có được các dữ liệu đã được gắn nhãn sẽ tốn rất nhiều chi phí.
- Học không giám sát (Unsupervised machine learning): Được sử dụng nhiều trong việc khám phá cấu trúc và mối quan hệ của dữ liệu. Điểm khác biệt giữa học có giám sát và không giám sát là ở dữ liệu đầu vào, đối với học không giám sát thì dữ liệu đầu vào không cần đánh nhãn. Các thuật toán mà học không giám sát sử dụng để phát hiện những điểm tương đồng và khác biệt trong dữ liệu đầu vào.
- Học bán giám sát (Semi-supervised learning): là sự kết hợp giữa học có giám sát và học không có giám sát. Trong quá trình huấn luyện nó sử dụng một tập dữ liệu nhỏ đã được đánh nhãn để phân loại và trích xuất đặc trưng cho một tập dữ liệu lớn hơn không được đánh nhãn. Học bán giám sát giải quyết đề của một tập dữ liệu lớn không được đánh nhãn đầy đủ.

- Học cũng cố (Reinforcement machine learning): một mô hình học máy hành vi tương tự như học có giám sát, nhưng thuật toán không được đào tạo bằng cách sử dụng dữ liệu mẫu. Mô hình này sẽ học theo cách sử dụng thử và sai. Một chuỗi các kết quả thành công sẽ được củng cố để phát triển khuyến nghị hoặc chính sách tốt nhất cho một vấn đề nhất định.



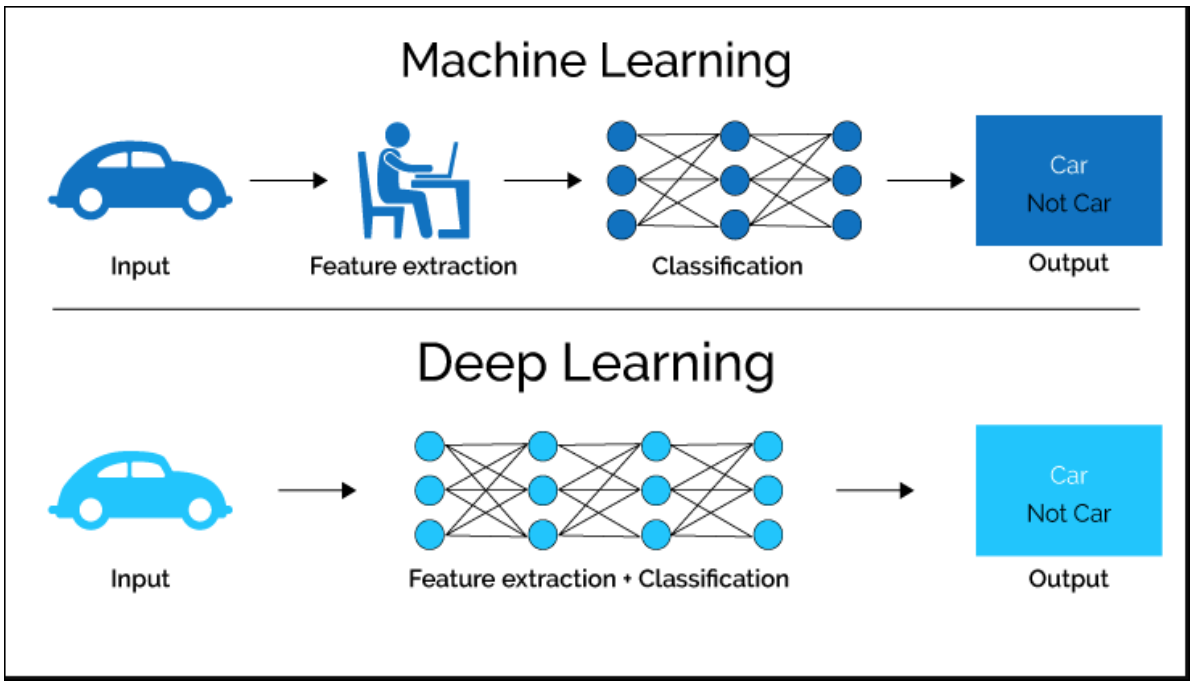
Hình 2-5: *Các kỹ thuật học máy.*

2.1.4 Deep Learning là gì?

Deep learning hay còn gọi là học sâu cũng chỉ là một tập hợp con của máy học. Nhưng nó có khả năng xử lý các bài toán phức tạp hơn nhờ sử dụng mạng nơ-ron (Neural Networks). Mạng nơ – ron gồm nhiều nơ – ron, lớp liên kết với nhau để tạo thành một như một bộ não của con người.

Đối với học máy nó sử dụng một số quá trình xử lý dữ liệu và đánh nhãn mới đưa dữ liệu vào quá trình huấn luyện. Đối với học sâu thì nó bỏ qua một số quá trình tiền xử lý và tăng cường sử dụng mạng nơ-ron nhiều lớp ẩn được nối với nhau, với mỗi lớp được xây dựng dựa trên lớp trước đó để tinh chỉnh và tối ưa hóa dự đoán và phân loại.

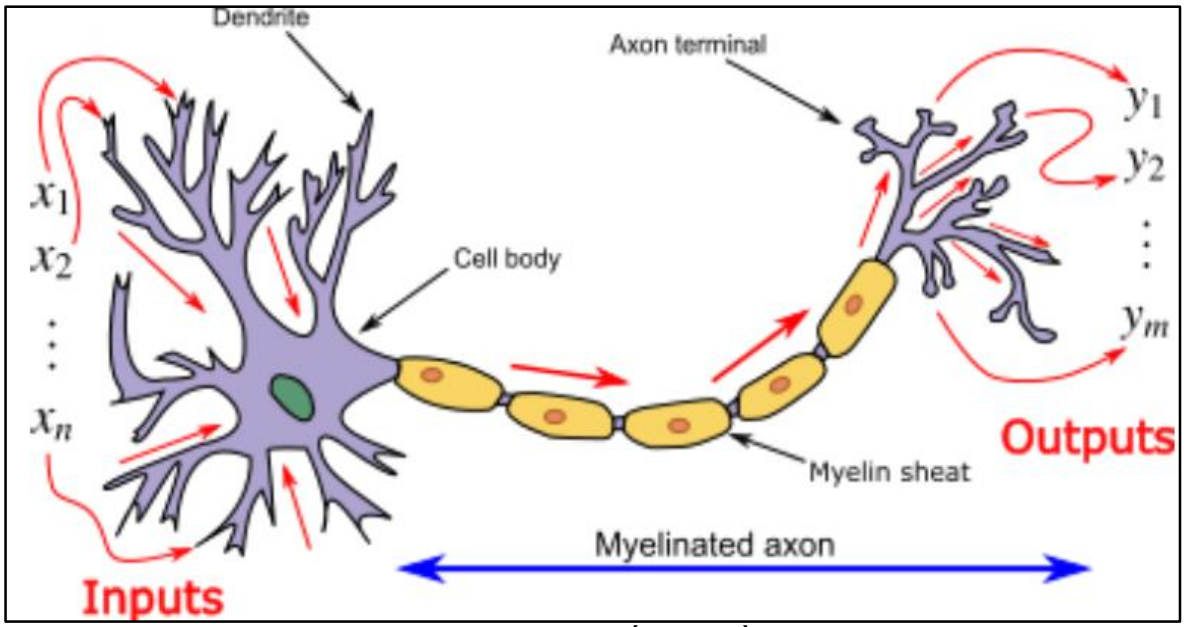
Trong Deep Learning, Mạng nơ-ron hợp pháp hay CNN là một loại mạng nơ-ron nhân tạo, được sử dụng rộng rãi để nhận dạng và phân loại hình ảnh / đối tượng. Do đó, Deep Learning nhận dạng các đối tượng trong hình ảnh bằng cách sử dụng CNN.



Hình 2-6: *Sự khác nhau giữa học máy và học sâu.*

2.1.5 Tế bào thần kinh sinh học:

Tế bào thần kinh sinh học là tế bào thần kinh có các đơn vị cơ bản của não và hệ thần kinh, các tế bào chịu trách nhiệm nhận đầu vào cảm giác từ thế giới bên ngoài thông qua đuôi gai, xử lý nó và đưa ra đầu ra thông qua các sợi trục

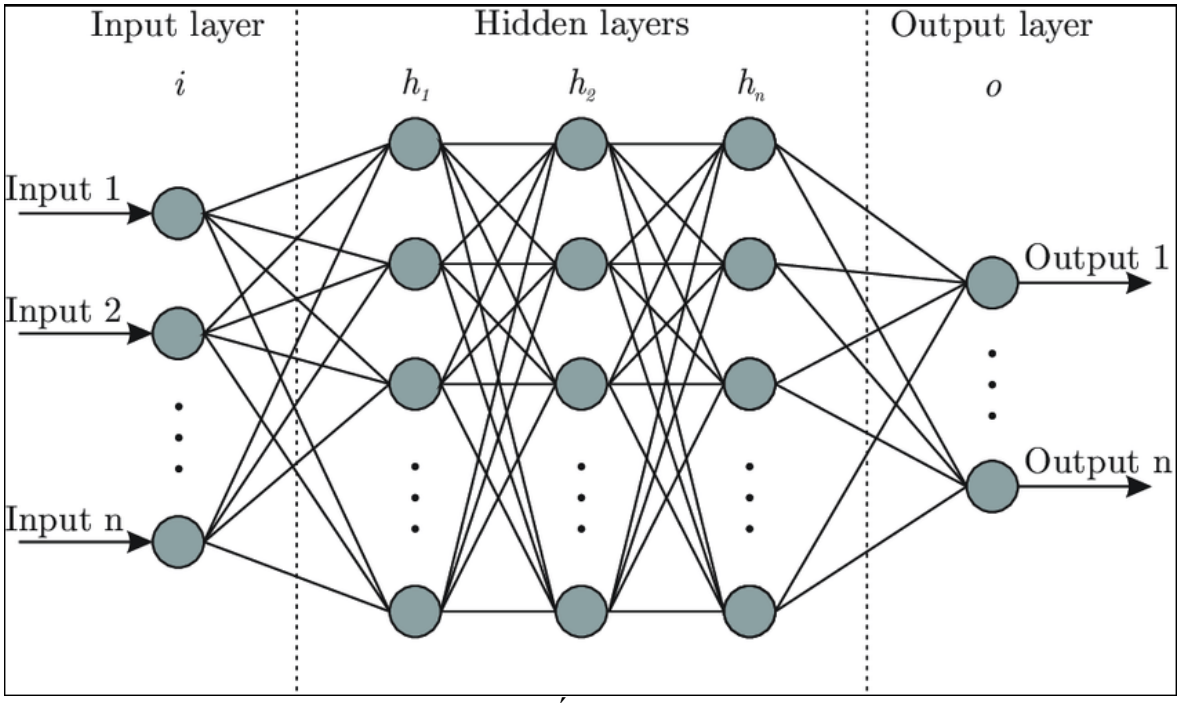


Hình 2-7: *Mô hình của một tế bào thần kinh sinh học.*

2.1.6 Mạng thần kinh nhân tạo (ANN or NN):

Mạng thần kinh nhân tạo hay ANN là một mô hình xử lý thông tin được lấy cảm hứng từ cách hệ thống thần kinh sinh học như não bộ xử lý thông tin. Nó bao gồm một

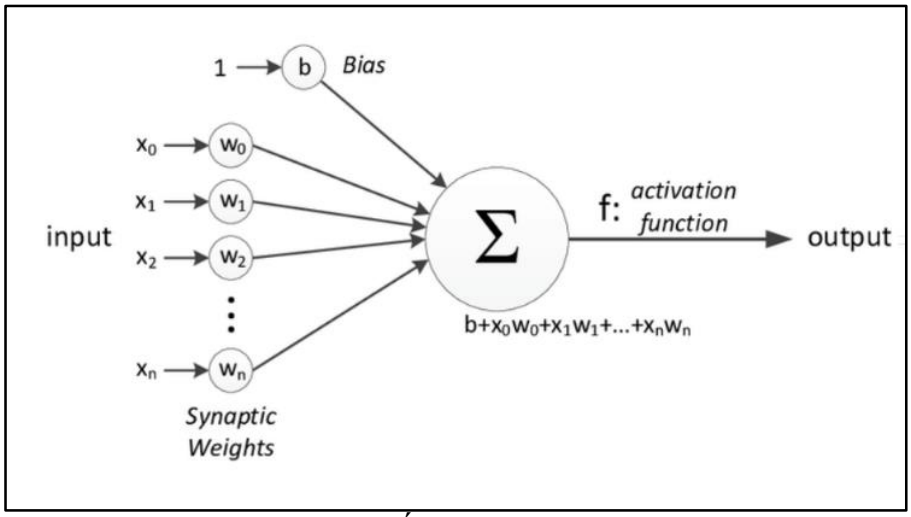
số lượng lớn các phần tử xử lý có tính liên kết cao (tế bào thần kinh) làm việc đồng bộ để giải quyết một vấn đề cụ thể nào đó.



Hình 2-8: *Kiến trúc mạng ANN.*

Kiến trục mạng NN:

Chúng ta sẽ có nhiều đầu vào input từ X0 đến Xn, và đầu ra là một kết quả phụ thuộc vào các đầu vào, mỗi đầu vào sẽ đi theo tương đương với một trọng số w (weights), các trọng số này nó sẽ ảnh hưởng đến kết quả ngỏ ra, đồng thời nó cũng có một trọng số b (bias) để hiệu chỉnh các trọng số ngỏ vào cùng tăng lên hoặc giảm xuống, sau khi xích ma lại thì nó sẽ đi qua một hàm gọi là activation để định hình kết quả ngỏ ra của một kiến trúc mạng

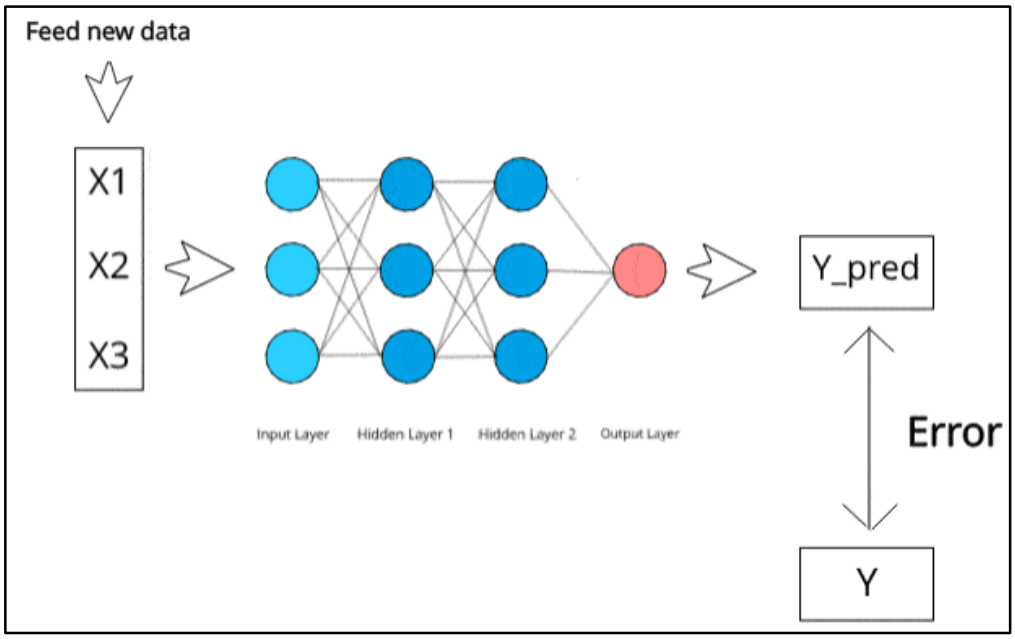


Hình 2-9: *Kiến trúc của mạng NN.*

Mạng Neural Networks gồm ba phần:

- Input layer (Lớp vào): Các đầu vào của mạng.
- Hidden layer (Lớp ẩn): Các lớp tính toán và suy luận logic của mạng.
- Output layer (Lớp ra): các đầu ra của mạng.

Lấy ví dụ chúng ta xây dựng một mô hình AI học có giám sát có 3 dữ liệu đầu vào x1 x2 x3 và các dữ liệu này đã biết trước được kết quả, và mỗi lần chúng ta đưa một bộ dữ liệu đầu vào thì đầu ra nó sẽ có một kết quả, lúc đó kết quả giữa chúng ta biết trước và kết quả của mô hình đưa ra nó sẽ có một khoảng lỗi, chương trình sẽ tự động cập nhật lại các trọng số w, b làm sao cho khoảng lỗi này nó hội tụ lại xấp xỉ bằng 0. Rồi sẽ cho vào vòng lặp để đi huấn luyện và lấy ra mô hình có % chính xác cao nhất.

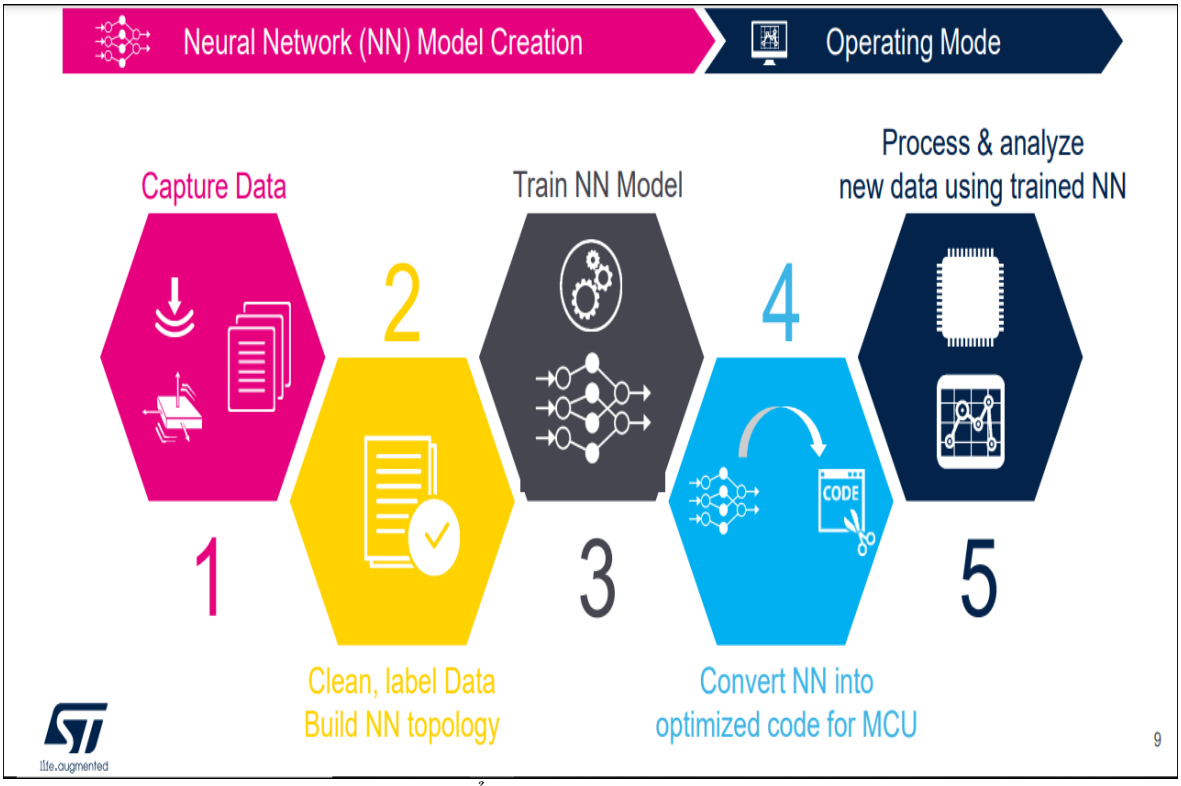


Hình 2-10: *Mô hình AI học có giám sát*

2.2 Edge AI là gì:

Edge AI là việc triển khai và nhúng các ứng dụng AI lên các thiết bị có bộ nhớ và hiệu năng thấp. Việc này giúp các dự án về AI được tính toán và phân tích dữ liệu gần nơi lấy dữ liệu, thay vì gửi dữ liệu lên đám mây để các siêu máy tính xử lý việc này mất thời gian và tiêu tốn nhiều tài nguyên.

Một số ứng dụng nổi bật sử dụng công nghệ EDGE AI như là loa thông minh của Google, Apple HomePod. Ở các thiết bị này chúng đã được nhúng một chương trình Machine Learning để xử lý âm học. Khi người dùng giao tiếp với các trợ lý như Siri hoặc Google Assistans, nó sẽ gửi bản ghi âm yêu cầu của người dùng xuống tới các thiết bị Edge để xử lý và trả về kết quả.



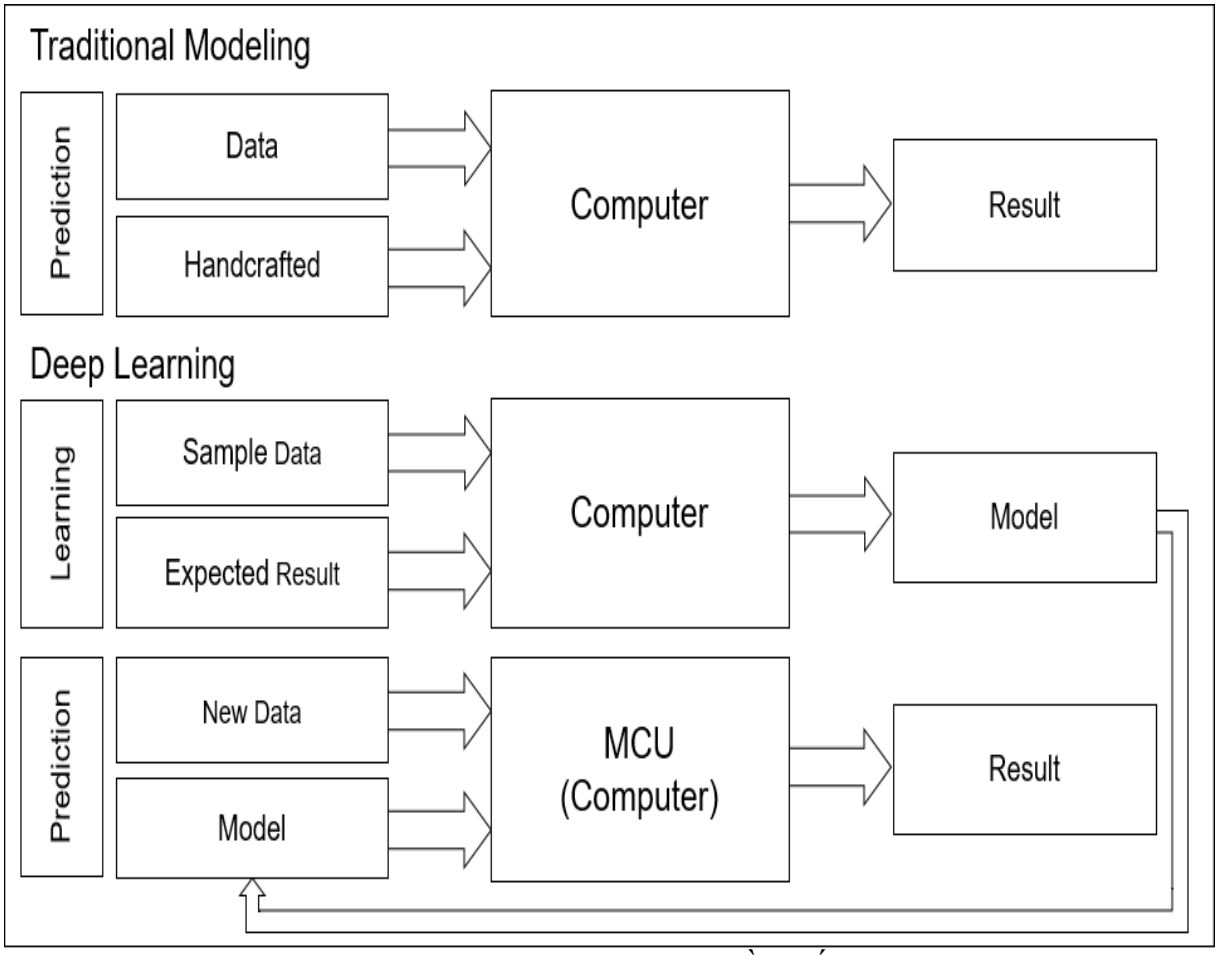
Hình 2-11: *Các bước để xây dựng mô hình Edge AI trên STM32.*

Lợi ích khi sử dụng EDGE AI:

- Phần xử lý AI sát với sensor hơn, nó xử lý tại biên không cần đưa truyền qua môi trường internet
- Tốt hơn về trãi nghiệm người dùng
- Real time, đáp ứng được thời gian thực
- Có độ tin cậy cao, vì nó nằm ở vi điều khiển nên bảo mật và an ninh cao không bị can thiệp bởi bên ngoài.
- Không tốn tài nguyên về cloud
- Có độ riêng tư hình ảnh âm thanh
- Tiêu tốn năng lượng nhỏ
- Mô hình Edge AI được dùng nhiều trên các thiết bị như vi điều khiển thay vì máy tính nên giảm được rất nhiều chi phí.
- Với một mô hình AI nó có khả năng tự học nên nó khiến mô hình được cải tiến liên tục.
- Nhỏ gọn

Nhược điểm:

- Không phát triển ứng dụng, bảo trì server vất vả.
- Không thể phát triển được các ứng dụng có độ phức tạp cao.



Hình 2-12: *Chương trình máy tính truyền thống và Edge AI.*

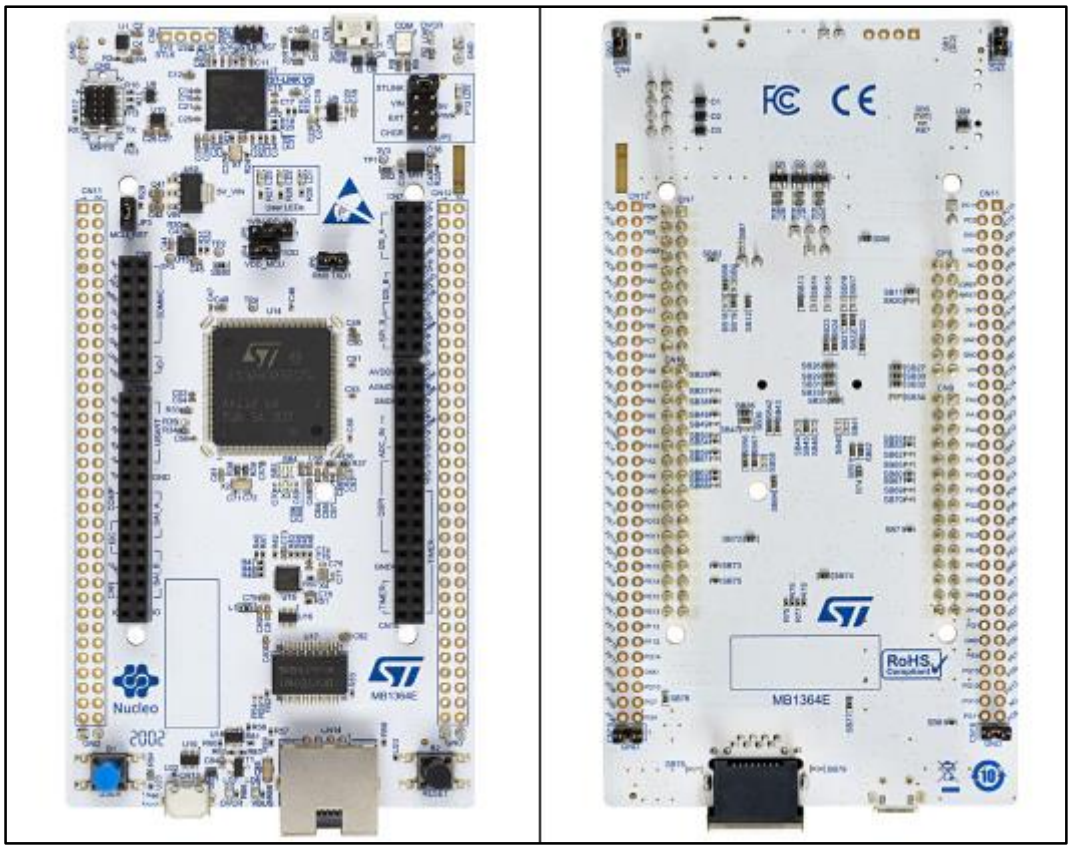
Đối với các chương trình truyền thống khi đưa một dữ liệu đầu vào thì lập trình viên cần viết các chương trình để tìm ra kết quả mong muốn. Và đối với một chương trình AI cụ thể ở đây là một chương trình Deep Learning thì việc đầu tiên ta cần làm là huấn luyện một mô hình để giải quyết một vấn đề cụ thể nào đó và khi đã có được mô hình thì ta tiến hành chạy mô hình đó với các dữ liệu mới lúc này kết quả của chương trình sẽ là các nhãn tương ứng với dữ liệu đầu vào.

2.3 Giới thiệu phần cứng:

2.3.1 Board STM32H743ZI nucleo: [1]

Với việc sử dụng đến AI và có sử dụng camera nên khá tốn tài nguyên cũng như hiệu năng cao để xử lý thì sau khi phân tích nhóm đã lựa chọn và quyết định sử dụng STM32H743ZI nucleo để thực hiện đề tài này.

Board STM32-H743ZI là board điều khiển có cấu hình và hiệu năng cao mà giá cả lại phải chăng, tích hợp trình gỡ lỗi và lập trình ST-LINK V3 và đi kèm với phần mềm toàn diện STM32 và thư viện HAL.

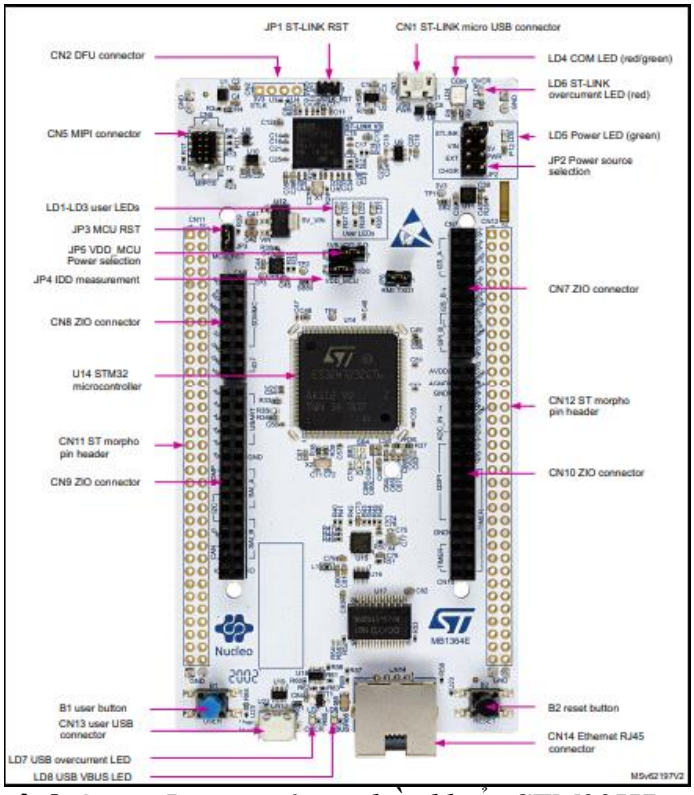


Hình 2-13: *Hình ảnh STM32H743ZI Nucleo.*

Thông số kỹ thuật:

- 32-bit ARM® Cortex®-M7 core
- Điện áp hoạt động: 1.62 V to 3.6 V
- Up to 2 MB of Flash memory
- 32 kHz tần số dao động nội
- Bộ nhớ RAM: 1MB
- Tần số dao động: 64 MHz HSI oscillator, 48 MHz RC oscillator, 4 MHz CSI oscillator, 40 kHz LSI oscillator
- Số lượng chân I/O: 168 chân
- Hỗ trợ giao tiếp: 4xI2C, 4xUSAT/4xUART, LPUART, 6xSPIs, 4xSAIs…
- 22 timer và watchdog.
- Hỗ trợ chuẩn DCMI giao tiếp camera 8 đến 14 bit.
- 3 đèn LED người dùng.
- 2 nút nhấn người dùng và Reset.
- Đầu nối ST Zio, mở rộng kết nối ARDUINO® Uno V3.
- Ethernet tương thích với IEEE-802.3-2002

- USB OTG tốc độ đầy đủ hoặc chỉ thiết bị
- Đầu nối bảng: USB với Micro-AB hoặc USB Type-C ®Ethernet RJ45



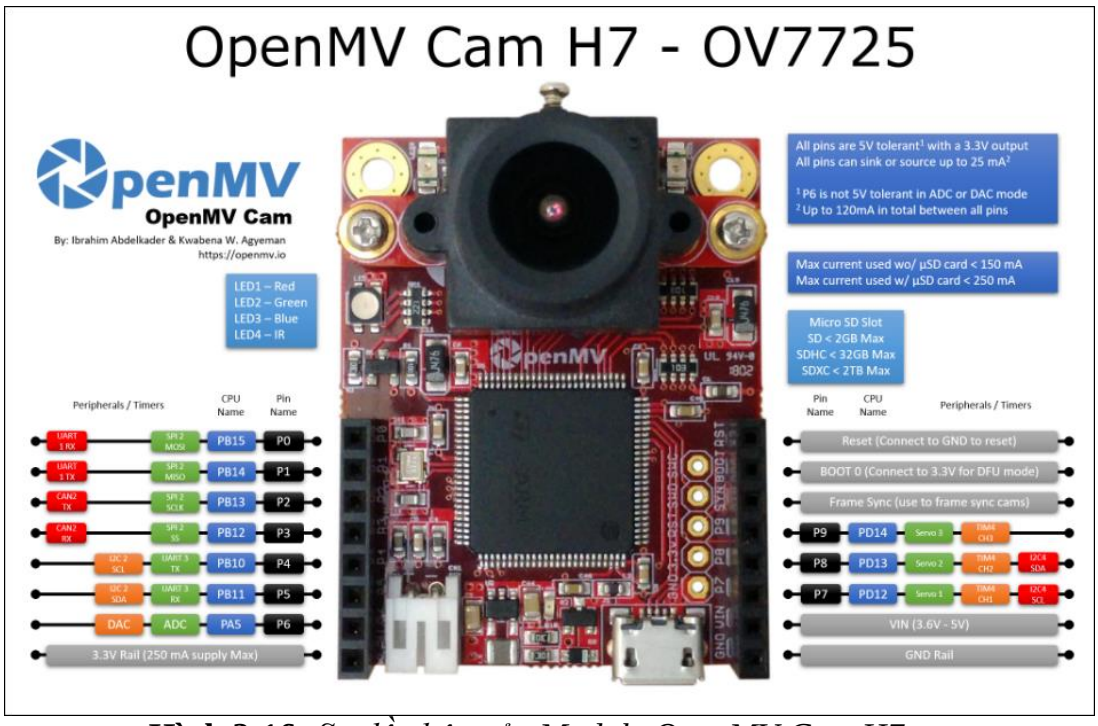
Hình 2-14: *Layout của vi điều khiển STM32H7.*

2.3.2 Module OpenMV Cam H7:

STM32 là một trong những dòng chip 32 bit phổ biến của hãng ST với nhiều họ thông dụng như F1, F4…Module OpenMV Cam H7 tích hợp bộ vi xử lý STM32H743VI ARM Cortex M7 chạy với tốc độ 480MHz.

Hình 2-15: *Hình ảnh Module OpenMV Cam H7.*

- Bộ xử lý STM32H743VI ARM Cortex M7 chạy ở tốc độ 480 MHz với 1MB SRAM và 2MB flash. Tất cả các chân I / O đều xuất ra 3.3V và chịu được 5V. Bộ xử lý có các giao diện I / O sau:
  - Giao diện USB tốc độ đầy đủ (12Mbs) để kết nối với máy tính .
  - Một ổ cắm thẻ SD có khả năng đọc / ghi 100Mbs cho phép OpenMV Cam của bạn chụp ảnh và lưu ảnh.
  - Một bus SPI có thể chạy tới 80Mbs giúp truyền dữ liệu hình ảnh tới LCD hoặc một vi điều khiển khác.
  - Bus I2C (tối đa 1Mb / s), Bus CAN (lên đến 1Mb / s) và Bus nối tiếp không đồng bộ (TX / RX, lên đến 7,5Mb / s) để giao tiếp với các bộ vi điều khiển và cảm biến khác.
  - Một bộ ADC 12-bit và một bộ DAC 12 bit.
  - Ba chân I / O để điều khiển các thiết bị ngoại vi.
  - Ngắt và PWM trên tất cả các chân I / O (có 10 chân I / O trên bo mạch).
  - Và, một đèn LED RGB và hai đèn LED IR 850nm công suất cao.
- Hệ thống mô-đun máy ảnh có thể tháo rời cho phép OpenMV Cam H7 giao tiếp với các cảm biến khác nhau.
- OpenMV Cam H7 đi kèm với cảm biến hình ảnh OV7725 có khả năng chụp ảnh 640x480 8-bit Grayscale hoặc 640x480 16-bit RGB565 ở 75 FPS khi độ phân giải trên 320x240 và 150 FPS khi ở dưới.



Hình 2-16: *Sơ đồ chân của Module OpenMV Cam H7.*

Pin			Description
Header	No	Name	
J1 Pin Configuration			
J1	1	P0	UART1 RX – TM1 CH3N – SPI 2 MOSI
	2	P1	UART1 TX – TM1 CH2N – SPI 2 MISO
	3	P2	CAN2 TX – TM1 CH1N – SPI 2 SCLK
	4	P3	CAN2 RX – SPI 2 SS
	5	P4	TIM2 CH3 – I2C 2 SCL – UART 3 TX
	6	P5	TIM2 CH4 – I2C 2 SDA – UART3 RX
	7	P6	TIM2 CH1 – DAC – ADC
	8	3.3	3.3V Rail (250 mA Supply MAX).
J2 Pin Configuration			
J2	1	RST	Reset (Connect to GND to reset).
	2	BOOT	Boot 0 (Connect to 3.3V for DFU mode).
	3	SYN	Frame synchronization pin (Use to frame sync cams).
	4	P9	Servo3 – TIM4 CH3
	5	P8	Servo2 – TIM4 CH2 – I2C4 SDA
	6	P7	Servo1 – TIM4 CH1 – I2C4 SCL
	7	VIN	VIN (3.6V – 5V).
	8	GND	GND Rail
J3 Pin Configuration			
J3	1	SWC	Serial wire debug clock.
	2	SWD	Serial wire debug data.
	3	RST	Reset (active low).
	4	3.3V	3.3V rail (500 mA Supply MAX)
	5	GND	GND rail

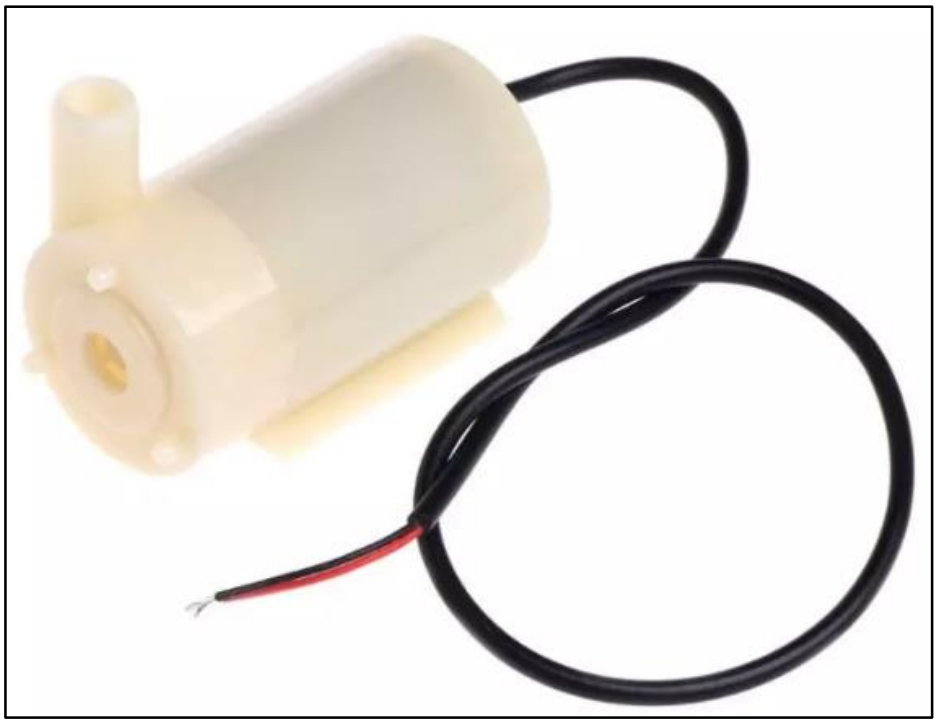
Bảng 2-1: *Chức năng các chân Module OpenMV Cam H7.*

2.3.3 Động cơ bơm nước DC 5v:

Động cơ bơm chìm Mini Water Pump 5VDC có kích thước rất nhỏ gọn, sử dụng điện áp 3~5VDC, vì thuộc dạng bơm chìm nên động cơ có khả năng chống nước và hoạt động khi ngâm chìm trong nước, ứng dụng để bơm nước, dung dịch trong các thiết kế nhỏ, mô hình tưới cây, hồ cá…

Thông số kĩ thuật:

- Điện áp sử dụng: 3~5VDC.
- Dòng điện sử dụng: 100~200mA.
- Lưu lượng bơm: 1.2~1.6L / 1 phút.
- Đường kính ngoài ống dẫn: 7.5mm
- Kích thước: 34 x 43 mm
- Trọng lượng: 28g



Hình 2-17: *Động Cơ Bơm Chìm Mini Water Pump 5VDC.*

2.3.4 Module 1 Relay 5VDC:

Module 1 Relay có một relay hoạt động ở ở điện áp 5VDC. Module có điện thế đóng ngắt tối đa: 250V ~ 10A đối với điện xoay chiều, 30V ~ 10A đối với dòng một chiều. Trên module tích hợp sẵn IC cách ly quang giúp cách ly mạch điều khiển với relay để đảm bảo an toàn cho mạch điều khiển.

Hình 2-18: *Module 1 Relay 5VDC.*

Thông số kỹ thuật:

- Điện áp nuôi DC5V
- Relay tiêu thụ dòng khoảng 80mA
- Có đèn đóng báo ngắt Relay
- Chọn được mức tín hiệu kích 0 hoặc 1 qua jumper.
- Điện thế đóng ngắt tối đa: AC250V ~ 10A hoặc DC30 ~ 10A.

2.3.5 Động cơ Servo SG90:

Động cơ Servo SG90 là một động cơ có kích thước nhỏ gọn có góc quay lên đến 180°, cách điều khiển giống như các động cơ RC Servo phổ biến trên thị trường hiện nay: MG996, MG995,.. Phù hợp cho nhiều ứng dụng khác nhau: Robot cánh tay máy, robot nhện, cơ cấu chuyển hướng, cơ cấu quay góc,...

Hình 2-19: *Hình ảnh Servo SG90 thực tế.*

Thông Số Kỹ Thuật:

- Khối lượng: 9g
- Kích thước: 22.2x11.8.32 mm
- Momen xoắn: 1.8kg/cm
- Tốc độ hoạt động: 60 độ trong 0.1 giây
- Điện áp hoạt động: 4.8V - 5V.
- Nhiệt độ hoạt động: 0 ºC – 55 ºC

Phương Pháp Điều Khiển:

- Động cơ điều khiển bằng xung PWM với tần số f = 50Hz và chu kì T = 20ms.
- Độ rộng xung với 0.5ms = 0° và 2.5ms = 180°.

CHƯƠNG 3: XÂY DỰNG MÔ HÌNH PHÁT HIỆN LỬA

3.1 Giới thiệu mô hình mạng Convolutional neural networks (CNN): [2]

3.1.1 Khái niệm về Convolutional neural networks:

Mạng Convolutation neural networks (CNN) hay còn gọi là mạng nơ – ron tích chập. Tích chập là một phép toán được sử dụng nhiều trong các mô hình Deep Learning, tích chập các dữ liệu đầu vào như ma trận hình ảnh, âm thanh vào các bộ lọc để trích xuất đặc trưng của dữ liệu đó.

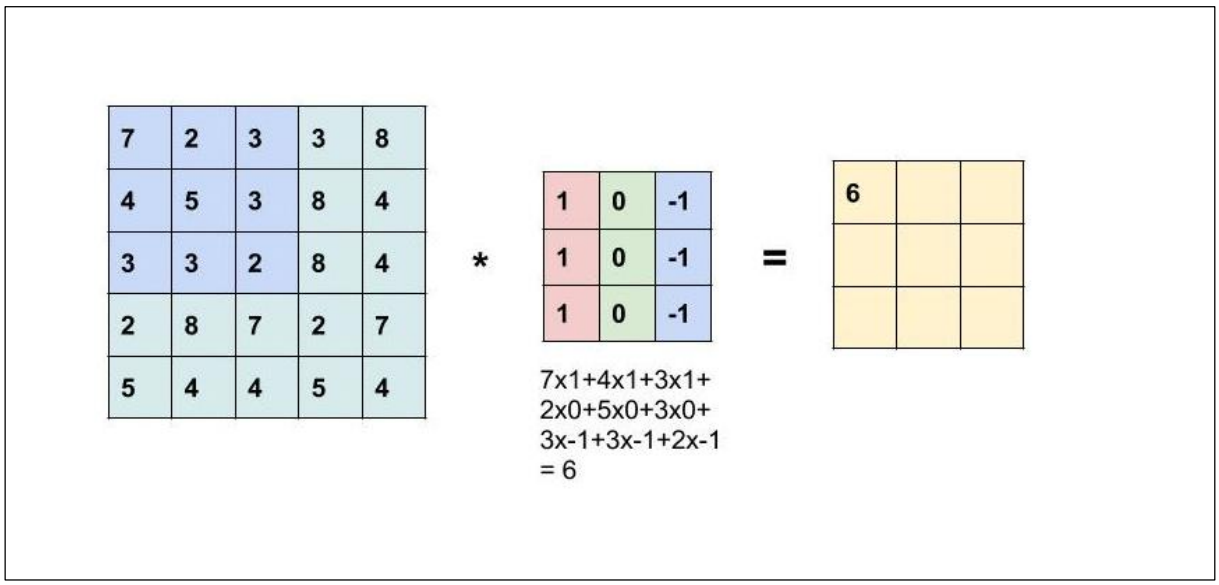
Tích chập được sử dụng nhiều nhất là tích chập hai chiều được áp dụng trên ma trận đầu vào và ma trận bộ lọc hai chiều. Phép tích chập của một ma trận X € $\mathbf{R}^{W1*H1}$ với bộ lọc (receptive field) F € $\mathbf{R}^{F\times F}$ là một ma trận Y € $\mathbf{R}^{W2*H2}$ sẽ trải qua 2 bước:

- Tính tích chập tại một điểm: Tại vị trí đầu tiên trên cùng của ma trận đầu vào ta sẽ lọc ra một ma trận con $X_{sub} \in R^{F\times F}$ có kích thước bằng với kích thước của bộ lọc. Giá trị y_{11} tương ứng trên Y là tích chập của $\boldsymbol{X_{sub}}$ với F được tính như sau:

$$y_{11} = \sum_{i=1}^{F}\sum_{j=1}^{F} x_{ij}y_{ij}$$

Trong đó x_{ij} là ma trận đầu vào, y_{ij} là ma trận lọc và y_{11} là kết quả của phép tính trên.

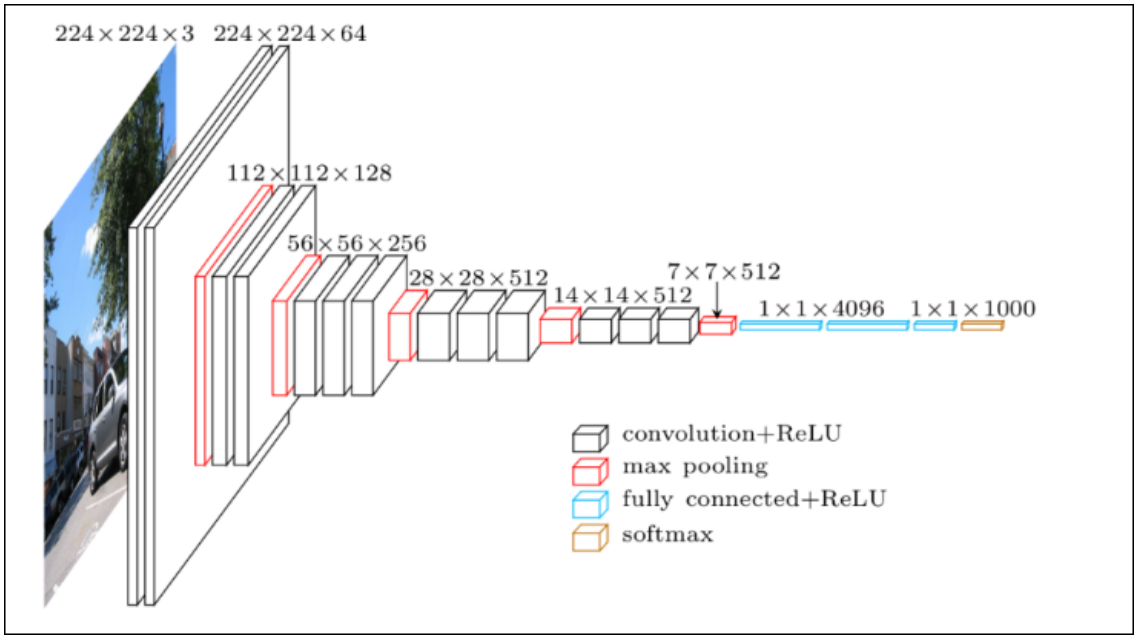
- Tiến hành trượt dọc theo ma trận theo chiều từ trái sang phải, từ trên xuống dưới theo bước nhảy(stride) S ta sẽ tính được các giá trị y_{ij} tiếp theo. Sau khi quá trình này kết thúc ta sẽ thu được trọn vẹn ma trận Y.



Hình 3-1: *Tích chập giữa hai ma trận.*

3.1.2 Cấu trúc cơ bản của một mạng CNN:

Mạng CNN bao gồm nhiều lớp liên kết với nhau để lọc và trích xuất đặc trưng, một mạng CNN hai chiều cơ bản có các lớp sau:



Hình 3-2: *Các lớp cơ bản của một mô hình CNN.*

Trong đó:

- Input: là lớp các dữ liệu đã qua bước tiền xử lý để gắn nhãn và chuyển thành một ma trận, dữ liệu có thể là hình ảnh, âm thanh…
- Convolution Layer: là lớp tích chập với các kernel (có kích thước như 1x1, 2x2, 3x3…) đã được học để hiệu chỉnh lấy ra thông tin chính xác nhất.
- Relu Layer: là hàm một hàm phi tuyến để giữ cho dữ liệu của ta luôn tuyến tính (không âm).
- Pooling Layer: Khi ảnh của ta có kích thước quá lớn hàm này sẽ giúp ta giảm được kích thước ma trận đầu vào mà vẫn giữ được các thông tin quan trọng. Có hai loại Pooling phổ biến là Maxpoling lấy phần tử lớn nhất từ ma trận đối tượng và Average Pooling lấy trung bình ma trận đối tượng.
- Fully-Connected + softmax: Lớp tìm ra xác xuất của từng class đối với bài toán classification

3.2 Công cụ để xây dựng mô hình:

3.2.1 Google Colab:

Google Colab là một sản phẩm của Google Research, Google Colab cho phép các người dùng chạy các mã chương trình Python thông qua trình duyệt. Google Colab cung

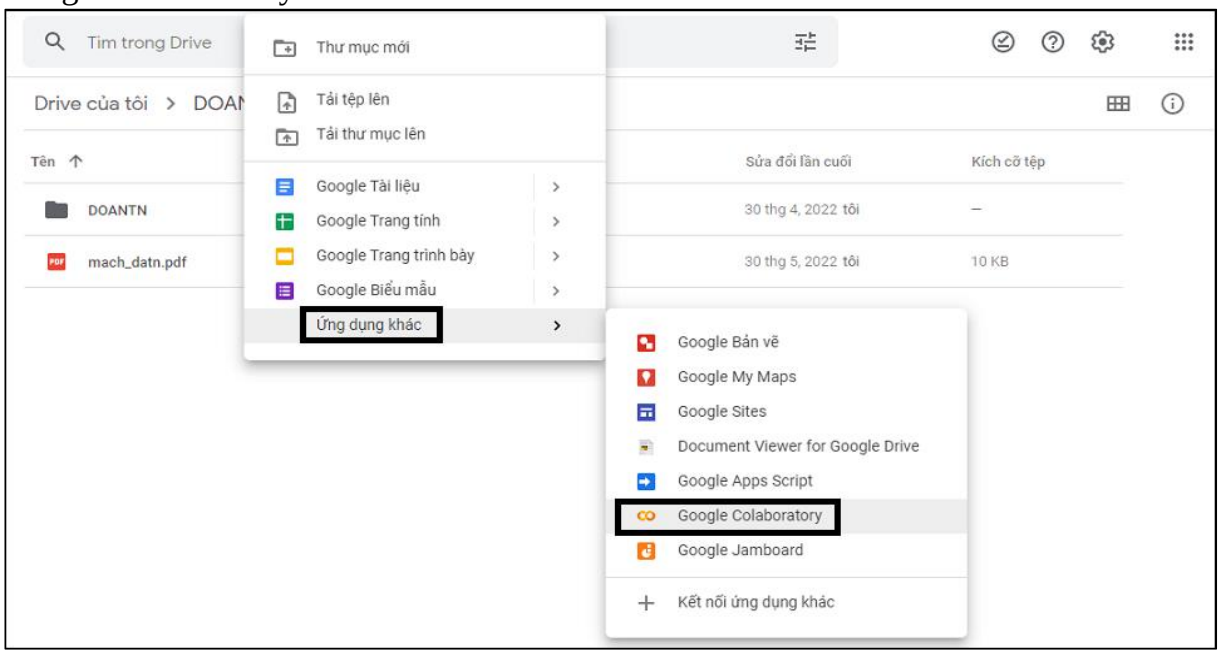
cấp cho người dùng một bộ GPU và một bộ TPU miễn phí, với những bộ phần cứng này giúp cho việc tính toán về toán học của những mô hình như Deep Learning và Machine Learning trở nên nhanh hơn rất nhiều so với việc chỉ sử dụng CPU.

Ngoài ra Google Colab còn cài sẵn cho người dùng những thư viện thường được dùng như OpenCV, Numpy để xử lý dữ liệu và các thư viện hỗ trợ quá trình huấn luyện mô hình học máy như Keras, TensorFlow.

Với việc kết nối với Google Drive giúp người dùng có thể lưu trữ và sử dụng dữ liệu của mình trên hệ thống đám mây. Và có thể sử dụng chương trình trên Google Colab bằng bất cứ khi nào người dùng muốn sử dụng.

Hình 3-3: *Google Colab có thể chạy các chương trình Python.*

Vì Google Colab đang hoạt động trên Google Drive nên quá trình tạo một chương trình Colab rất đơn giản. Đầu tiên ta nhấp chuột phải chọn ứng dụng khác và chọn Google Colaboratory.



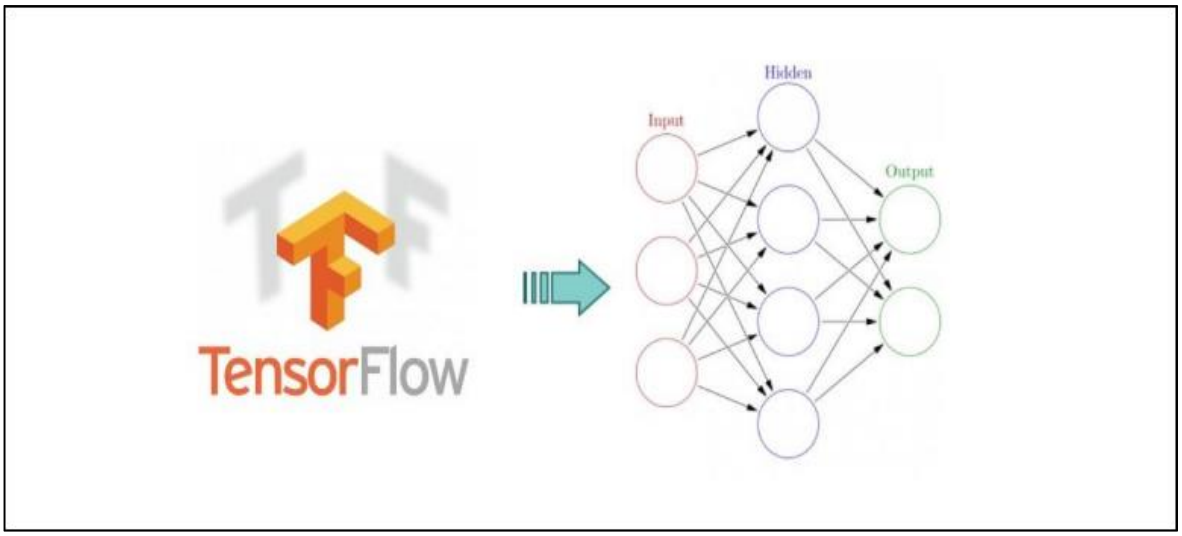
Hình 3-4: *Tạo một chương trình trên Google Colab.*

Lúc này chỉ việc đổi lại tên và thực thi các chương theo từng ô.

Hình 3-5: *Google Colab chạy chương trình theo từng phần.*

3.2.2 TensorFlow:

Được tạo bởi nhóm Google Brain, TensorFlow là một thư viện mã nguồn mở để tính toán số và học máy quy mô lớn. TensorFlow kết hợp một loạt các mô hình và thuật toán học máy và học sâu (hay còn gọi là mạng thần kinh) và làm cho chúng trở nên hữu ích bằng một phép ẩn dụ thông thường. Nó sử dụng Python để cung cấp một API front-end thuận tiện cho việc xây dựng các ứng dụng với framework, đồng thời thực thi các ứng dụng đó bằng C ++ hiệu suất cao.



Hình 3-6: *TensorFlow*

TensorFlow cho phép bạn xây dựng biểu đồ và cấu trúc luồng dữ liệu để xác định cách dữ liệu di chuyển qua biểu đồ bằng cách lấy đầu vào là một mảng đa chiều được gọi là Tensor. Nó cho phép bạn xây dựng một sơ đồ các hoạt động có thể được thực hiện trên các đầu vào này, đi ở một đầu và đến ở đầu kia là đầu ra.

TensorFlow gồm ba phần:

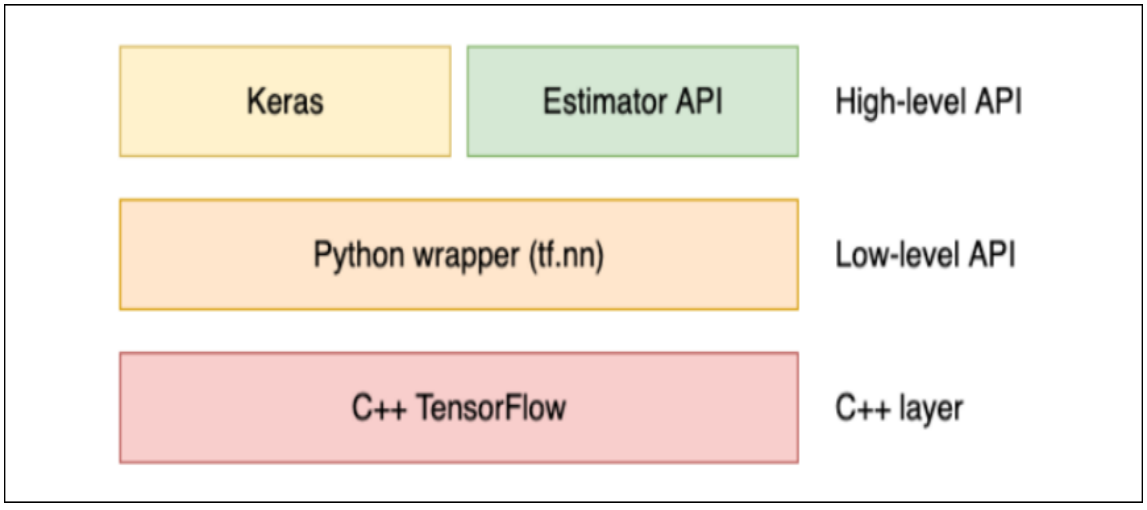
- Tiền xử lý xử liệu
- Xây dựng mô hình
- Đào tạo và đánh giá mô hình

TensorFlow có thể chạy trên nhiều nền tảng Windowns, Linux, IOS, Android… Nhưng ta ưu tiên xây dựng mô hình trên những thiết bị có hỗ trợ GPU để thuận tiện cho việc tính toán song song.

3.2.3 Keras:

Keras được nhúng trong TensorFlow và có thể được sử dụng để thực hiện học sâu nhanh chóng vì nó cung cấp các mô-đun có sẵn cho tất cả các tính toán mạng nơ-ron. Đồng thời, tính toán liên quan đến tensors, đồ thị tính toán, phiên, v.v. có thể được tùy chỉnh bằng cách sử dụng Tensorflow Core API, giúp bạn hoàn toàn linh hoạt và kiểm soát ứng dụng của mình, đồng thời cho phép bạn triển khai ý tưởng của mình trong thời gian tương đối ngắn.

Keras thực sự là một high-level APIs cho việc xây dựng và huấn luyện các mô hình học sâu. Keras giúp đưa ra các mô hình mẫu một cách nhanh chóng, giúp đưa ra các sản phẩm hoặc nghiên cứu với các thuật toán hiện đại.



Hình 3-7: *High-level APIs của TensorFlow.*

Các lợi ích của Keras:

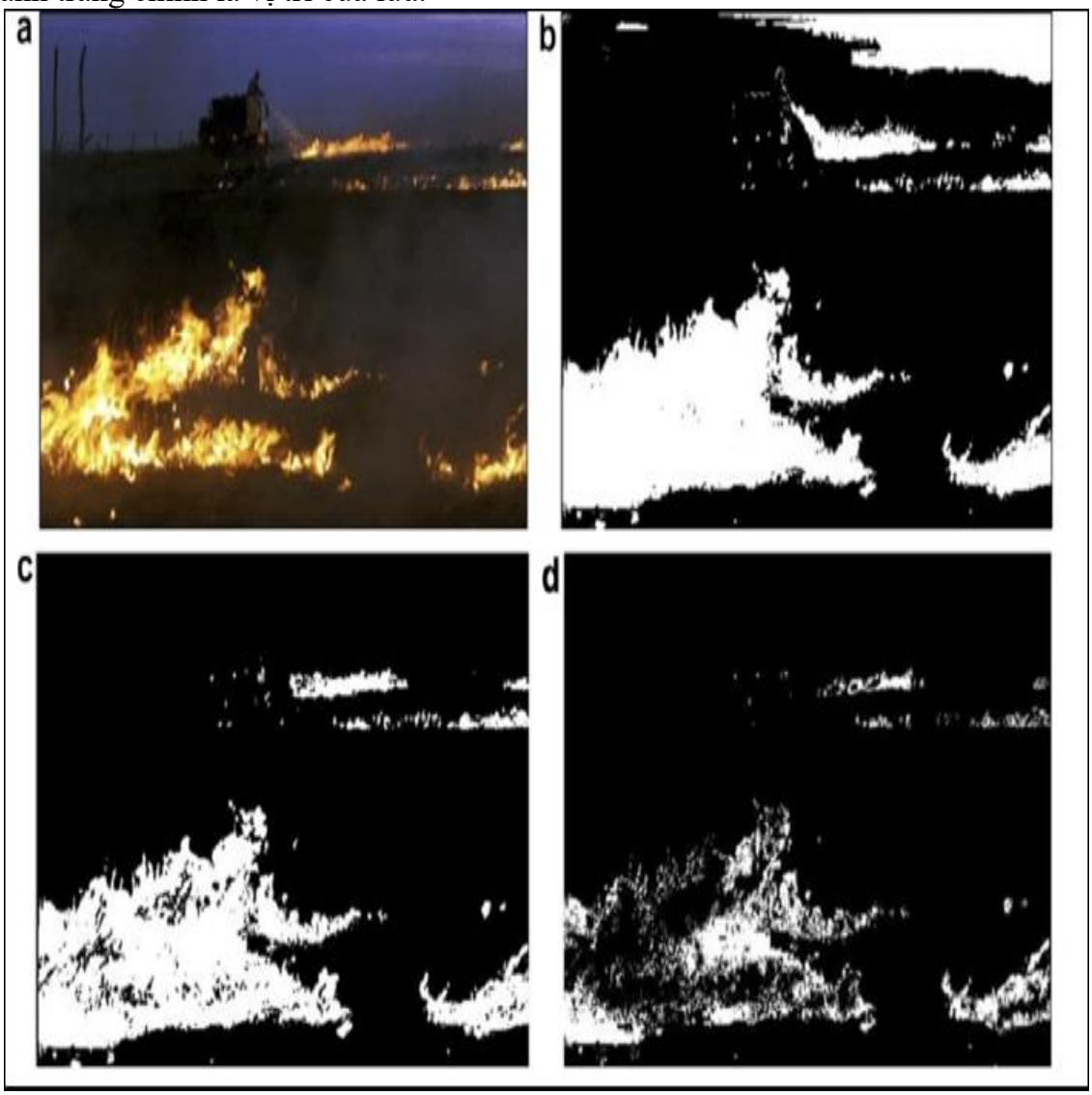
- Thân thiện với người dùng.
- Được chia thành các mô-đun và có thể kết hợp.
- Dễ dàng mở rộng.
- Dành cho cả người mới bắt đầu cũng như các chuyên gia.
- Thực hiện tốt trên cả CPU, GPU.

- Có thể tính toán cùng lúc trên nhiều thiết bị.
- Thực hiện được trên đa nền tàng như trình duyệt, thiết bị di động, thiết bị nhúng,
- Keras là một hệ sinh thái rộng lớn với mối liên hệ với nhiều nền tảng khác nhau.

3.3 Xây dựng mô hình:

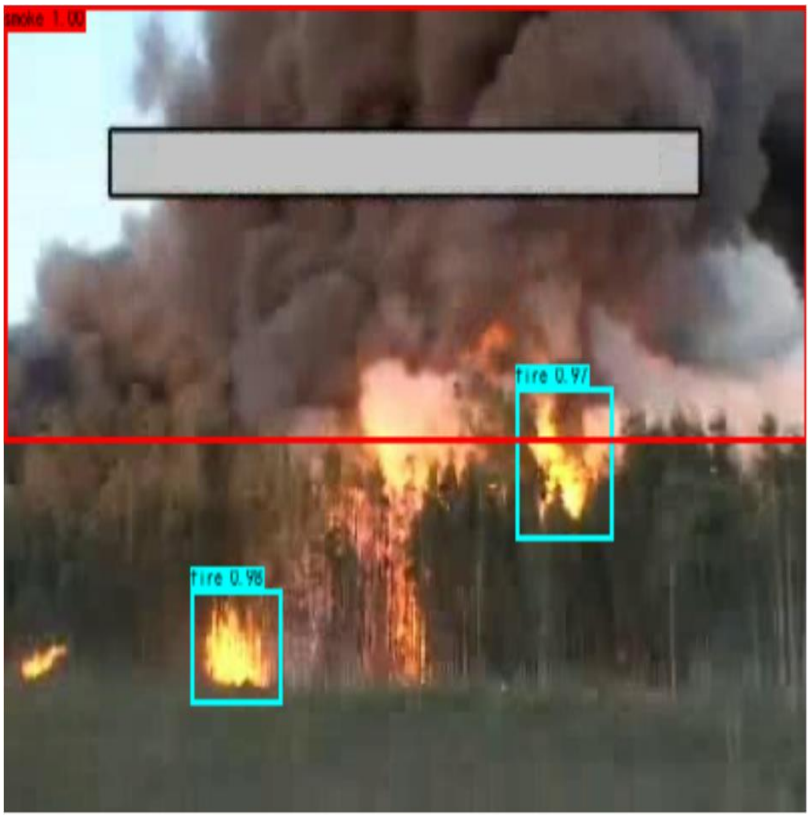
3.3.1 Các phương pháp để nhận diện lửa:

- **Phương pháp Sematic Segmentation:** Đối với phương pháp này ta sẽ tiến hành nhận diện lửa theo từng pixel, kết quả của phương pháp này sẽ là một ảnh nhị phân và phần ảnh trắng chính là vị trí của lửa.



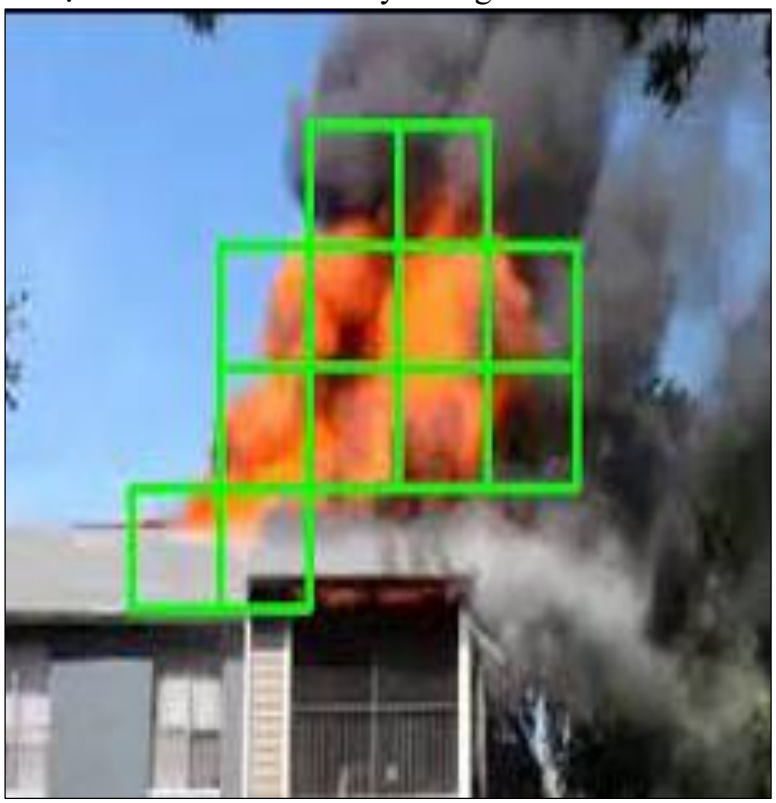
Hình 3-8: *Phương pháp Segmentation.*

- **Phương pháp Object Detection:** Phương pháp này sẽ tiến hành phân loại các đối tượng trong ảnh, trong một đám cháy thì có hai đối tượng cần phân loại đó chính là khói và lửa. Và kết quả của phương pháp này các đối tượng sẽ được khoanh vùng và đánh các nhãn tương ứng như là lửa hoặc khói.



Hình 3-9: *Phương pháp Object Detection.*

- **Phương pháp Classfication:** Đối với phương pháp này ta sẽ chia ảnh ra thành từng ảnh nhỏ, và xác định ảnh đó là có lửa hay không.



Hình 3-10: *Phương pháp Classfication*

Nhóm tiến hành so sánh ba phương pháp kể trên thì kết luận rằng đối với phương pháp **Segmentation** có độ phức tạp cao nhất nhưng cũng mang lại độ chính xác cao nhất và độ phức tạp của phương pháp **Detection** sẽ thấp hơn phương pháp **Segmentation**. Và cuối cùng phương pháp **Classfication** sẽ có độ phức tạp thấp nhất trong ba phương pháp kể trên, phương pháp này phù hợp với các thiết bị có hạn chế về phần cứng như vi điều khiển STM32, nhóm đã quyết định sử dụng phương pháp này cho đề tài của nhóm. Để xây dựng một mô hình nhận diện cháy sử dụng mạng nơron tích chập (CNN), nhóm chia công việc thành 5 bước:

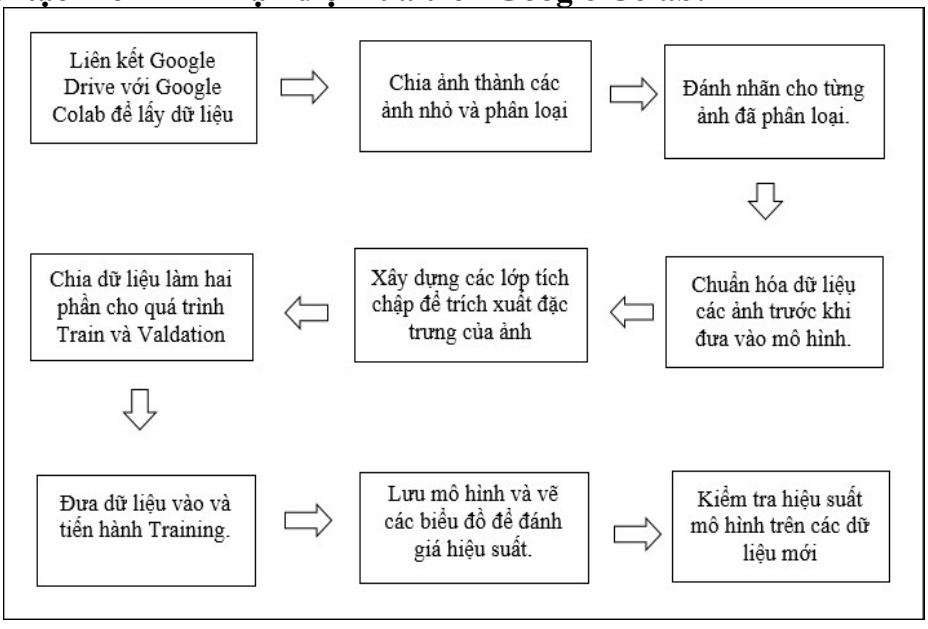
- Thu thập và tổng hợp các dữ liệu về cháy.
- Hiệu chỉnh lại các thuộc tính của ảnh và gắn nhãn cho ảnh.
- Xây dựng và phân tích kiến trúc mạng nơron tích chập (CNN).
- Huấn luyện mô hình và đánh giá mô hình.
- Kiểm tra hiệu suất mô hình trên các ảnh cháy mới.

3.3.2 Xây dựng sơ đồ khối mô hình nhận diện lửa:

Để tạo được một hình AI việc đầu tiên ta cần làm là thu thập dữ liệu. Sau quá trình thu thập các ảnh về cháy và ảnh nhị phân phân vùng cháy(Segmentation), nhóm tiến hành upload các dữ liệu này lên Google Drive. Nhóm tiến hành tạo ba file Google Colab trong đó:

- File đầu tiên dùng để xử lý các dữ liệu đầu vào.
- File thứ hai sử dụng các lớp tích chập và các bộ lọc để tạo ra mô hình nhận diện lửa.
- File cuối cùng để kiểm tra mô hình với các ảnh mới.

Sơ đồ khối tạo mô hình nhận diện lửa trên Google Colab:



Hình 3-11: *Sơ đồ khối mô hình nhận diện lửa.*

3.3.3 Thu thập dữ liệu về cháy:

Các hình ảnh về cháy thì có rất nhiều ở trên Internet, nhưng nhóm ưu tiên tìm các tập dữ liệu về cháy có thêm phần ảnh đã được phân đoạn cháy, đó là các ảnh nhị phân chỉ có cháy. Các ảnh có nhiều nguồn để thu thập như các bài báo nói về cháy và từ các video có cháy.

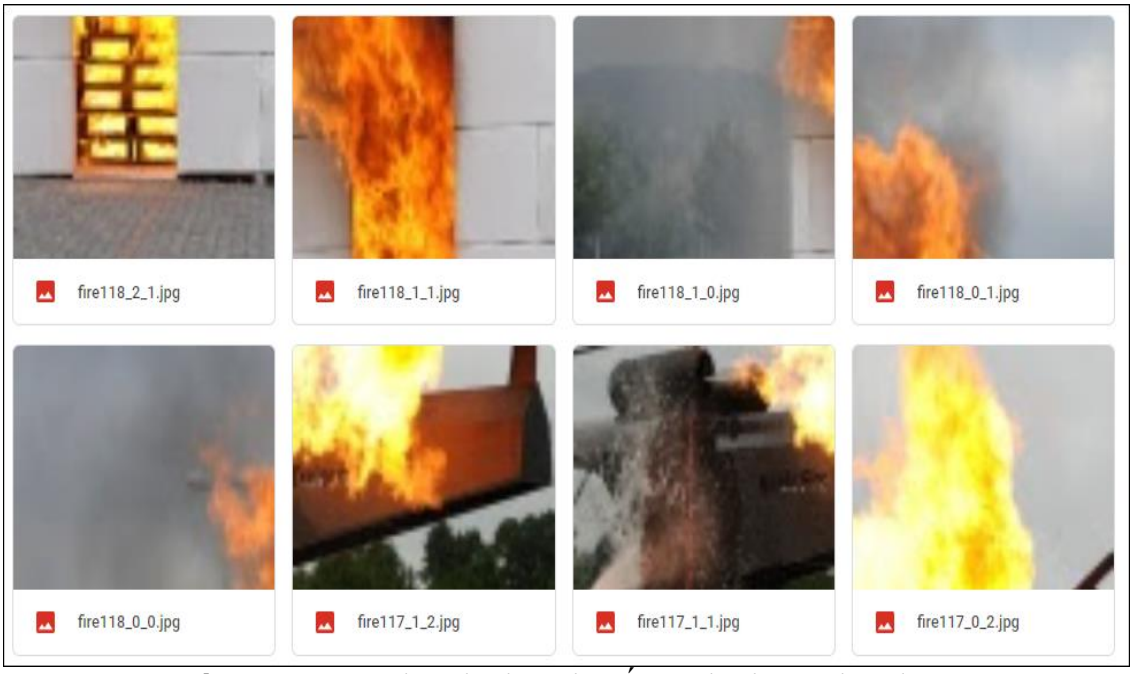
Nhóm đã thu thập được 116 ảnh cháy có màu và 116 ảnh cháy nhị phân tương ứng. Việc tìm được các ảnh này sẽ giúp việc xử lý dữ liệu đầu vào được thuận tiện hơn.

Hình 3-12: *Ảnh cháy bình thường và ảnh cháy nhị phân.*

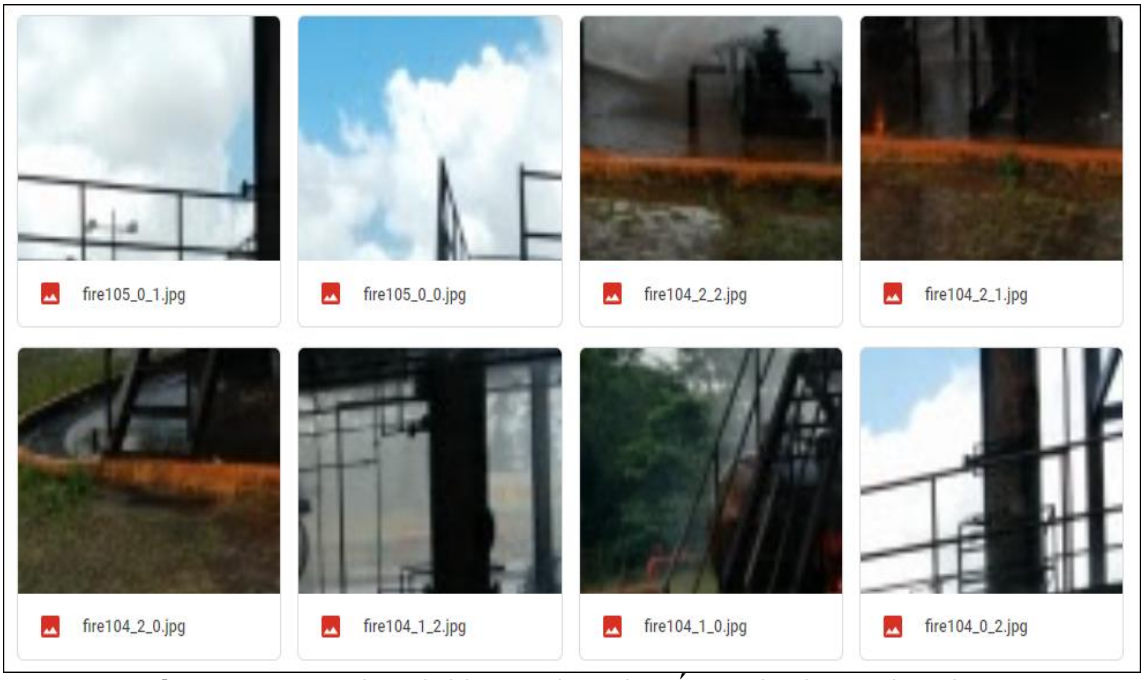
3.3.4 Hiệu chỉnh lại các thuộc tính của ảnh:

Tất cả các ảnh trong tập dữ liệu về cháy này được lấy từ nhiều nguồn nên sẽ có rất nhiều kích thước khác nhau, nên việc đầu tiên ta cần làm là tiến hành resize lại tất cả các ảnh về cùng một kích thước. Do mô hình cuối cùng ta sẽ chạy trên vi điều khiển STM32 bị hạn chế hạn chế về tốc độ và bộ nhớ, nên ta sẽ resize ảnh lại kích thước càng nhỏ càng tốt nhưng nếu ảnh quá nhỏ sẽ làm mất những chi tiết của ảnh. Qua nhiều quá trình thử nghiệm nhóm đã chọn kích thước tối ưu nhất là 120x120 Pixel.

Khi đã resize ảnh màu và ảnh nhị phân ta tiến hành chia mỗi ảnh thành chín phần, mỗi ảnh nhỏ lúc này có kích thước là 40x40 Pixel. Ta tiến hành so sánh và đếm số pixel màu trắng ở ảnh nhị phân, nếu số pixel này lớn hơn 5% thì ta tiến hành phân loại cho ảnh màu tương ứng là cháy và nếu số pixel bé hơn 5% thì là không cháy.



Hình 3-13: *Hình ảnh cháy đã cắt và đã được phân loại.*



Hình 3-14: *Hình ảnh không cháy đã cắt và đã được phân loại.*

Khi đã phân loại được ảnh cháy và không cháy ta tiến hành lưu tất cả tên của ảnh vào một file CSV để thuận tiện cho việc đánh nhãn cho từng ảnh sau này. Trong file CSV gồm ba phần:

- Slice_file_name: chứa các tên File.
- Class: chứa loại ảnh là cháy, không cháy tương ứng với FIRE, NOT_FIRE.
- ClassID: chứa ID của từng loại ảnh với FIRE là ‘1’ và NOT_FIRE là ‘0’.

	A	B	C
1	slice_file_name	class	ClassID
2	fire003_0_1.jpg	NOT_FIRE	0
3	fire003_0_0.jpg	NOT_FIRE	0
4	fire002_0_2.jpg	NOT_FIRE	0
5	fire002_1_0.jpg	NOT_FIRE	0
6	fire002_1_2.jpg	NOT_FIRE	0
7	fire001_2_1.jpg	NOT_FIRE	0
8	fire002_0_1.jpg	NOT_FIRE	0
9	fire002_0_0.jpg	NOT_FIRE	0
10	fire002_2_0.jpg	NOT_FIRE	0
11	fire001_2_2.jpg	NOT_FIRE	0
12	fire002_2_2.jpg	NOT_FIRE	0
13	fire001_1_0.jpg	NOT_FIRE	0
14	fire001_2_0.jpg	NOT_FIRE	0
15	fire000_2_2.jpg	NOT_FIRE	0
16	fire001_0_2.jpg	NOT_FIRE	0
17	fire000_2_1.jpg	NOT_FIRE	0

Hình 3-15: *File CSV của các ảnh đã phân loại.*

Khi đã có dữ liệu đầu vào thì ta tiến hành chuẩn hóa dữ liệu của từng ảnh, với mục đích là làm cho các giá trị của từng điểm ảnh chỉ dao động từ 0 đến 1 để dễ dàng quản lý. Ta tiến hành chuẩn hóa các dữ liệu bằng cách dùng hàm "*max()*" tìm ra phần tử lớn nhất trong ma trận, tiến hành chia các giá trị từng điểm ảnh cho giá trị lớn nhất này. Các giá trị sau khi chuẩn hóa sẽ nằm trong khoảng 0-1, bằng cách này giá trị sau khi chuẩn hóa vẫn đảm bảo được tính tương quan của dữ liệu hình ảnh.

Khi tất cả dữ liệu đã được chuẩn hóa xong, thì ta tiến hành chuyển các giá trị trong một ảnh thành dạng ma trận bằng thư viện 'numpy'. Ở hình 3-11 là kết quả của quá chuẩn hóa dữ liệu và chuyển thành dạng ma trận.

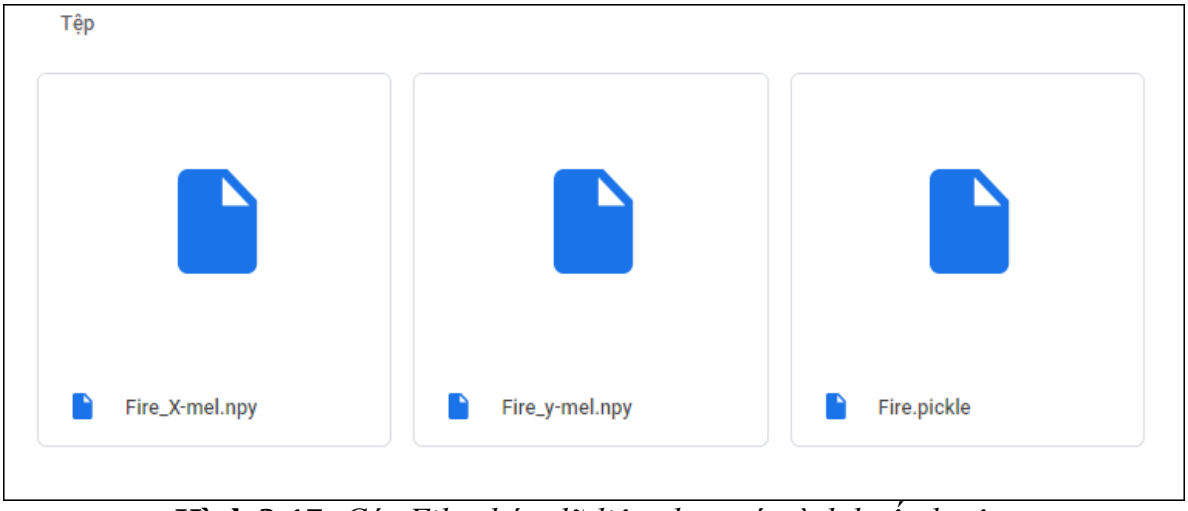
```
Fire
[[[0.         0.00392157 0.11764706]
  [0.         0.         0.11764706]
  [0.         0.         0.14117647]
  ...
  [0.         0.01176471 0.00392157]
  [0.         0.00784314 0.00784314]
  [0.         0.00784314 0.00784314]]

 [[0.         0.02352941 0.24313725]
  [0.         0.04313725 0.26666667]
  [0.00784314 0.03137255 0.2627451 ]
  ...
  [0.         0.00392157 0.00392157]
  [0.         0.00392157 0.00392157]
  [0.         0.00392157 0.00392157]]

 [[0.02352941 0.0745098  0.49019608]
  [0.         0.02352941 0.42352941]
  [0.02352941 0.03921569 0.43921569]
  ...
  [0.00392157 0.         0.00784314]
  [0.00392157 0.         0.00784314]
  [0.00392157 0.         0.00784314]]
```

Hình 3-16: *Dữ liệu sau khi được chuẩn hóa.*

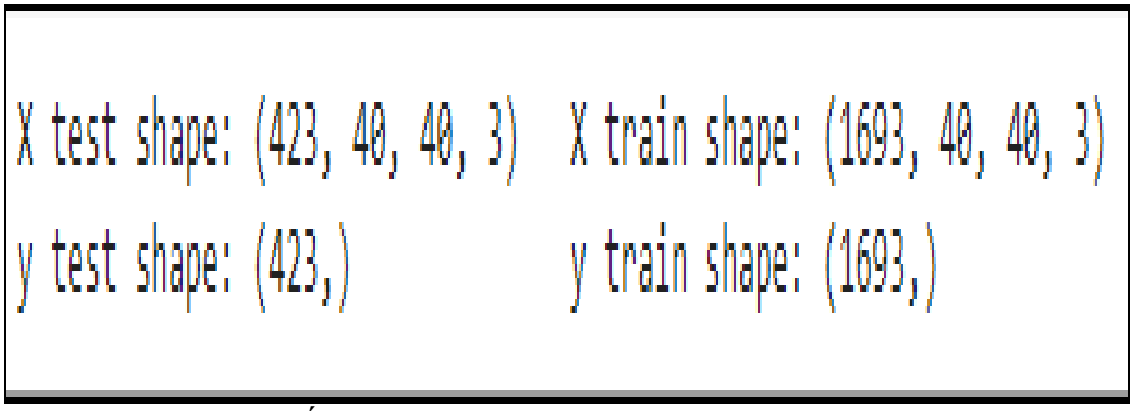
Để thuận tiện cho việc đưa dữ liệu vào mô hình để huấn luyện nhóm đã tiến hành tạo ra một tập dữ liệu gồm ba file. Trong đó file 'Fire_X_mel.npy' để lưa các giá trị của ảnh., file 'Fire_y_mel.npy' để lưu nhãn của từng ảnh và cuối cùng là file 'Fire.pickle' để lưu chiều cao và chiều rộng của ảnh.



Hình 3-17: *Các File chứa dữ liệu cho quá trình huấn luyện.*

3.3.5 Xây dựng cấu trúc mạng nơ-ron tích chập (CNN):

Đầu vào mô hình là các ảnh có kích thước 40x40x3 đã được đánh nhãn là lửa và không lửa ở phần trước. Và tập dữ liệu sẽ được chia ngẫu nhiên trong đó nhóm chia cho việc huấn luyện (Training) là 80% dữ liệu và 20% cho việc đánh giá (Validation). Trong đó 423 hình ảnh, nhãn được chọn ngẫu nhiên cho quá trình Validation và 1693 hình ảnh và nhãn cho quá trình Training.



Hình 3-18: *Số lượng hình ảnh cho quá trình Train và Validation.*

Cấu trúc mô hình mà nhóm bao gồm 8 lớp chính:

- Đầu tiên là hai lớp tích chập (Convolution layer) với 16 bộ lọc và kích thước của bộ lọc kernel_size = (3,3).
- Một lớp Max-Pooling với kernel_size = (2,2) để giảm kích thước của các lớp đi một nữa và chỉ giữ lại các trọng số cao nhất.
- Hai lớp tích chập (Convolutation layer) với 32 bộ lọc.
- Một lớp Global Max-Pooling để giữ lại các trọng số cao nhất của mỗi lớp.
- Một lớp đầu ra là Fully connected layer để dàn phẳng dữ liệu đầu ra thành một vector.
- Và sử dụng softmax để phân loại đầu ra là cháy hay không cháy.

Trong quá trình huấn luyện mô hình, nhóm đã sử dụng Keras Checkpoint để lưu lại mô hình tốt nhất quá trình huấn luyện. Đây chính là mô hình có hàm loss thấp nhất trong quá trình huấn luyện. Bên cạnh đó, để tránh trường hợp mô hình rơi vào hiện tượng overfitting nhóm còn sử dụng một số kỹ thuật regularizer được liệt kê như sau:

- Batch Normalization [3]
- Spatial Dropout [4]
- L2 Regularizer [5]

Mô hình sẽ sử dụng thuật toán tối ưu Adam [6] để tối ưu độ chính xác của mô hình và giảm tối thiểu loss của mô hình

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 38, 38, 16)	448
leaky_re_lu (LeakyReLU)	(None, 38, 38, 16)	0
batch_normalization (BatchNo	(None, 38, 38, 16)	64
spatial_dropout2d (SpatialDr	(None, 38, 38, 16)	0
conv2d_1 (Conv2D)	(None, 36, 36, 16)	2320
leaky_re_lu_1 (LeakyReLU)	(None, 36, 36, 16)	0
batch_normalization_1 (Batch	(None, 36, 36, 16)	64
max_pooling2d (MaxPooling2D)	(None, 18, 18, 16)	0
spatial_dropout2d_1 (Spatial	(None, 18, 18, 16)	0
conv2d_2 (Conv2D)	(None, 16, 16, 32)	4640
leaky_re_lu_2 (LeakyReLU)	(None, 16, 16, 32)	0
batch_normalization_2 (Batch	(None, 16, 16, 32)	128
spatial_dropout2d_2 (Spatial	(None, 16, 16, 32)	0
conv2d_3 (Conv2D)	(None, 14, 14, 32)	9248
leaky_re_lu_3 (LeakyReLU)	(None, 14, 14, 32)	0
batch_normalization_3 (Batch	(None, 14, 14, 32)	128
global_max_pooling2d (Global	(None, 32)	0
dense (Dense)	(None, 2)	66

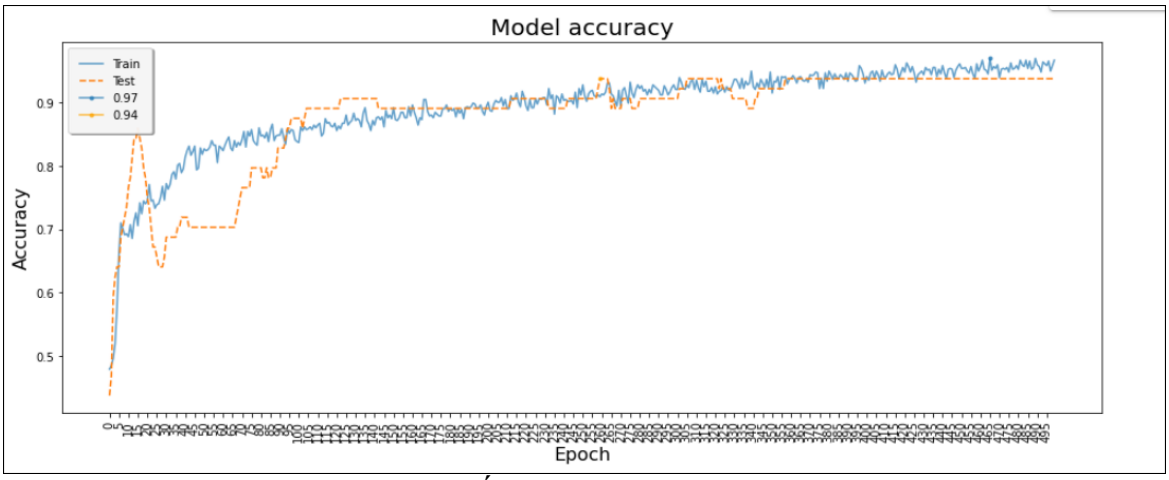
Total params: 17,106
Trainable params: 16,914
Non-trainable params: 192

Hình 3-19: *Cấu trúc của mô hình phân loại cháy.*

3.3.6 Huấn luyện và đánh giá mô hình:

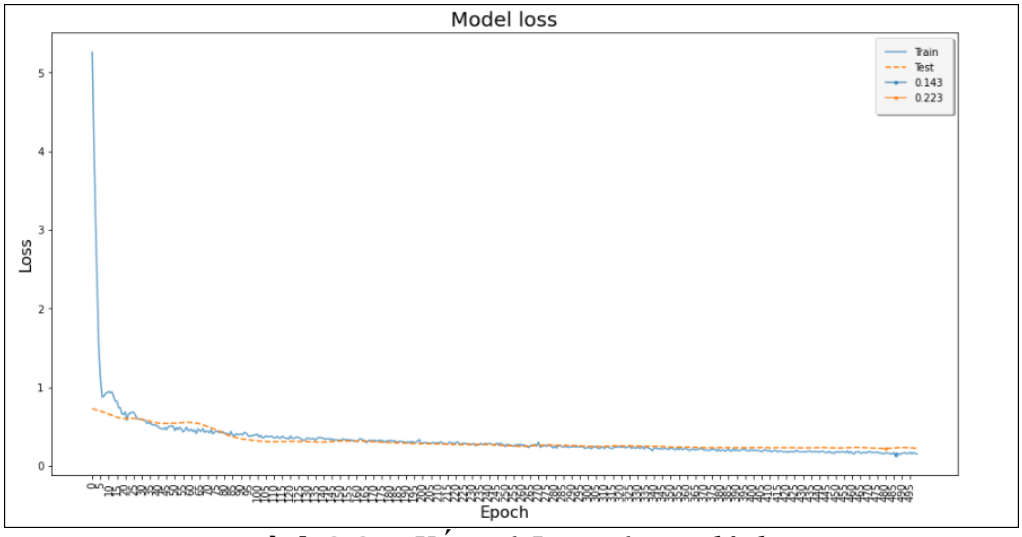
Nhóm huấn luyện mô hình CNN với giá trị epochs = 500 với mục tiêu nhóm đề ra là mô hình có giá Accuracy trên 90% và giá trị loss function càng thấp càng tốt.

Sau quá trình huấn luyện trên Google Colab với mỗi epochs tăng thì Accuracy tăng và loss giảm xuống. Trong đó hàm Accuracy là hàm dùng để đánh giá độ chính xác của mô hình. Kết quả cuối cùng là Accuracy đạt 97 % đối với quá trình huấn luyện (Traning) và 94% đối với quá trình đánh giá (Validation) và giá trị loss ở mức thấp.



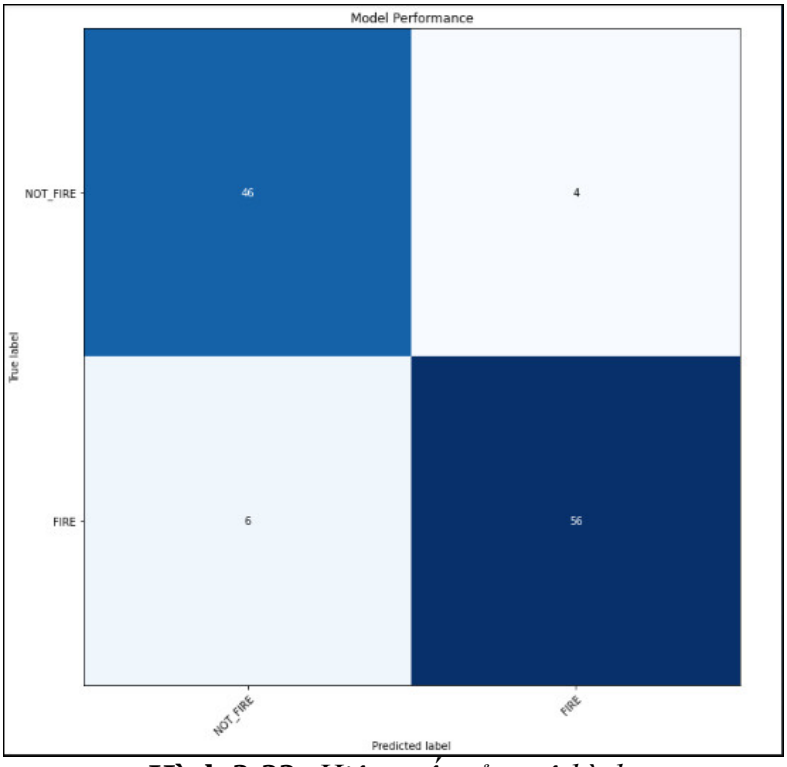
Hình 3-20: *Kết quả Accuracy của mô hình.*

Hàm mất mát (loss functiona) là một hàm thể hiện sự chênh lệch kết quả của mô hình dự đoán với kết quả thực tế. Để có được mô hình có độ chính xác cao thì giá trị của hàm này phải càng nhỏ. Với mô hình của nhóm thì giá trị Loss Function của quá trình Training là 0.143 và Validation là 0.223.



Hình 3-21: *Kết quả Loss của mô hình.*

Trong quá huấn luyện nếu có kết quả accuracy và loss tốt nhất thì chương trình tự động lưu lại dưới dạng file ".h5". Nhóm tiến hành vẽ một ma trận để đánh giá hiệu suất của mô hình như hình 3-18. Trong đó với 56 hình ảnh không lửa thì mô hình đoán đúng 46 ảnh là không lửa và với 72 ảnh có lửa thì mô hình đoán đúng 56 hình là có lửa.

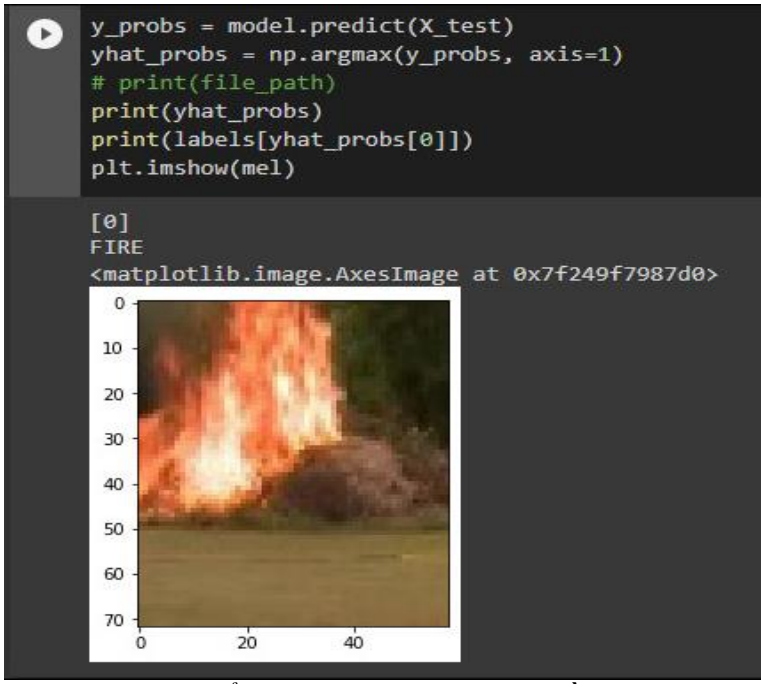


Hình 3-22: *Hiệu suất của mô hình.*

3.3.7 Kiểm tra mô hình trên tập dữ liệu mới:

Nhóm tiến hành chuẩn bị một tập dữ liệu về cháy và không cháy mới từ 53 ảnh gốc, nhóm tiến hành resize tất cả ảnh về kích thước 120x120 pixel và tiến hành cắt và đánh nhãn cho từng ảnh. Tập dữ liệu mới bao gồm 271 ảnh có lửa và 206 ảnh không lửa.

Hình 3-23: *Kiểm tra hình ảnh không lửa bằng mô hình.*



Hình 3-24: *Kiểm tra hình ảnh có lửa bằng mô hình.*

Nhóm tiến hành kiểm tra ngẫu nhiên các ảnh có lửa và không lửa thì được kết quả dự đoán đúng rất cao với tỉ lệ nhận diện đúng khoảng 90%.

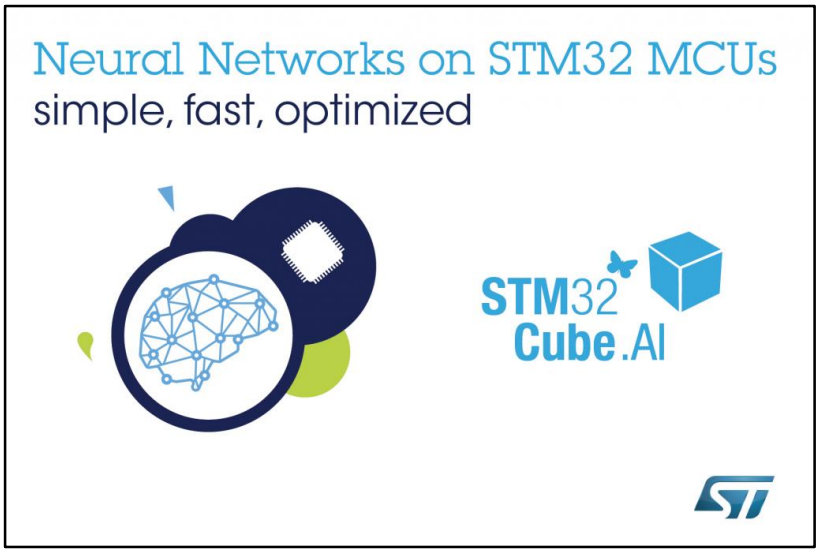
CHƯƠNG 4: TRIỂN KHAI MÔ HÌNH NHẬN DIỆN LỬA LÊN VI ĐIỀU KHIỂN STM32

4.1 Phương pháp nhúng mô hình xuống STM32:

Nhóm sẽ trình bày cách đưa một mô hình AI xuống thiết bị vi điều khiển cho việc nhận dạng lửa và công cụ STM32 Cube AI để chuyển đổi mô hình AI đã huấn luyện trên máy tính xuống vi điều khiển STM32, và thực thi công việc ở đó.

4.1.1 Công cụ STM32 Cube AI:

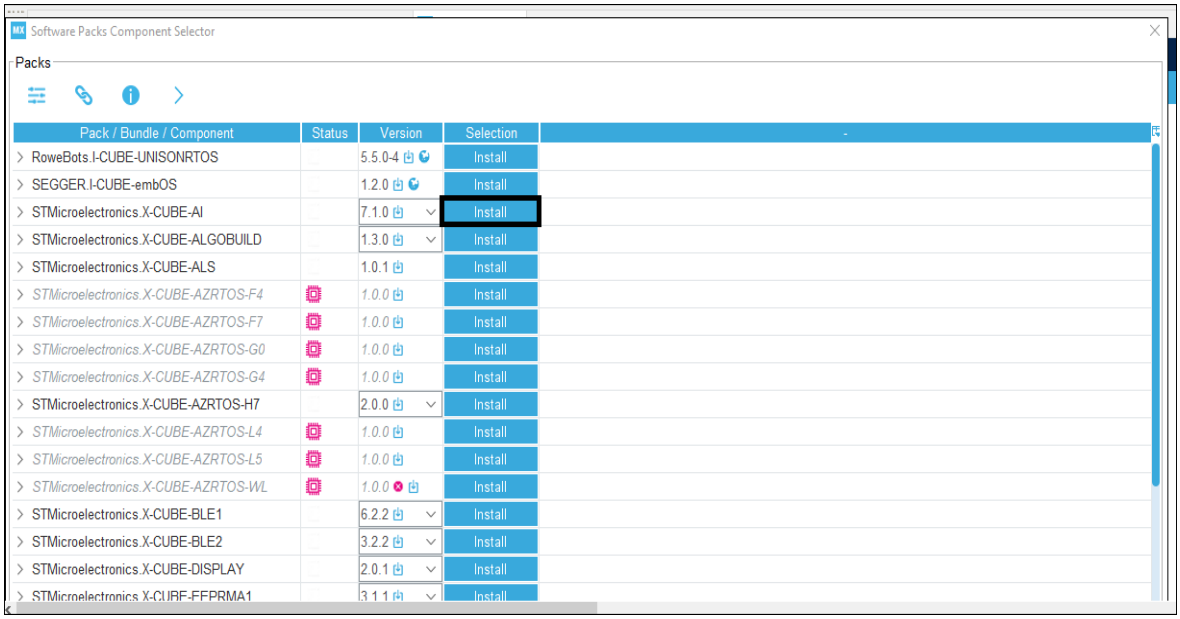
Sau khi đã huấn luyện mô hình nhận dạng lửa trên máy tính thành công, nhóm sẽ triển khai mô hình xuống vi điều khiển STM32. Thông thường thì các mô hình AI sẽ được chạy trên các sever hay trên một hệ thống máy tính có hiệu năng cao. Tuy nhiên ở đây nhóm sử dụng công nghệ Edge AI cho phép triển khai nhúng mô hình xuống vi điều khiển. Một vấn đề đặt ra là tính chất chủa một mô hình AI yêu cầu khá lớn về tài nguyên bộ nhớ cũng như hiệu năng xử lý phải cao, vì vậy việc triển khai mô hình trên vi điều khiển là cực kì khó khan so với việc triển khai trên máy tính thông thường. Chính vì lẽ đó hãng STMicroelectronics đã đưa ra một công cụ là STM32 Cube AI, công cụ này cho phép chúng ta có thể triển khai nhúng mô hình AI xuống vi điều khiển STM32 một cách dễ dàng



Hình 4-1: *Công cụ STM32 Cube AI.*

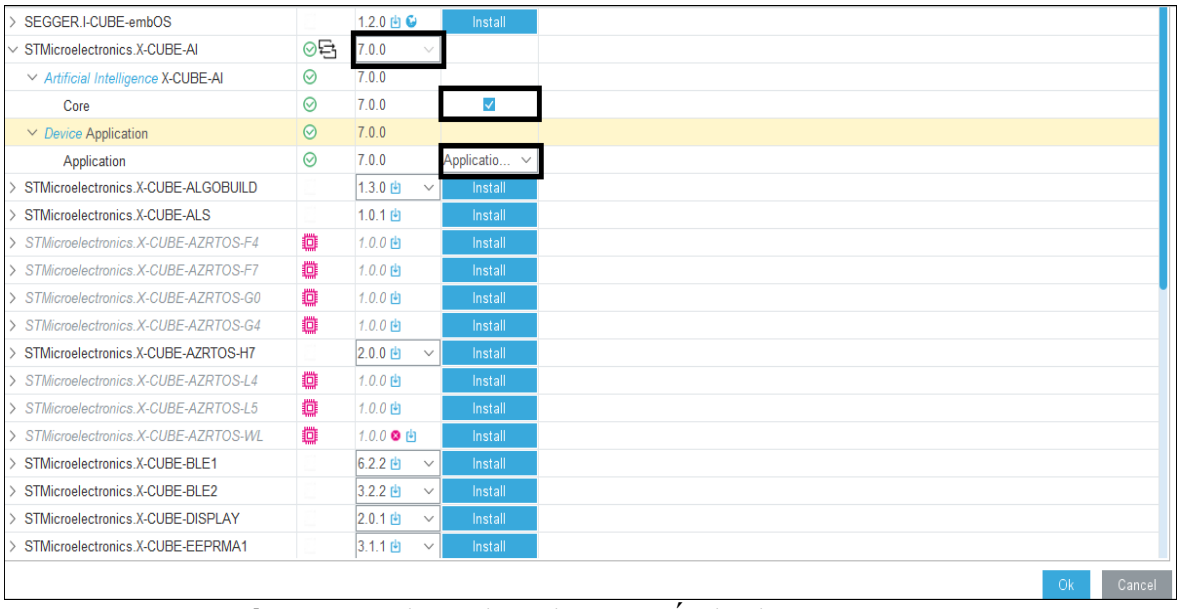
STM32CubeIDE cung cấp các phần mềm để giúp người dùng có thể tích hợp các tính năng mới vào chương trình của mình. Trong đó có phần mềm về AI tên là X-CUBE-AI [7], thư viện này cần đưa vào chương trình để chạy mô hình AI của nhóm.

Để tải gói thư viện X-CUBE-AI đầu tiên ta nhấn vào 'Software packs' chọn 'Select Components', ở cửa sổ Packs chọn phiên bản X-CUBE-AI và nhấn Install như hình 4.2.



Hình 4-2: *Tải thư viện X-CUBE-AI.*

Tiếp theo ta chọn version, kích hoạt Core và Application.

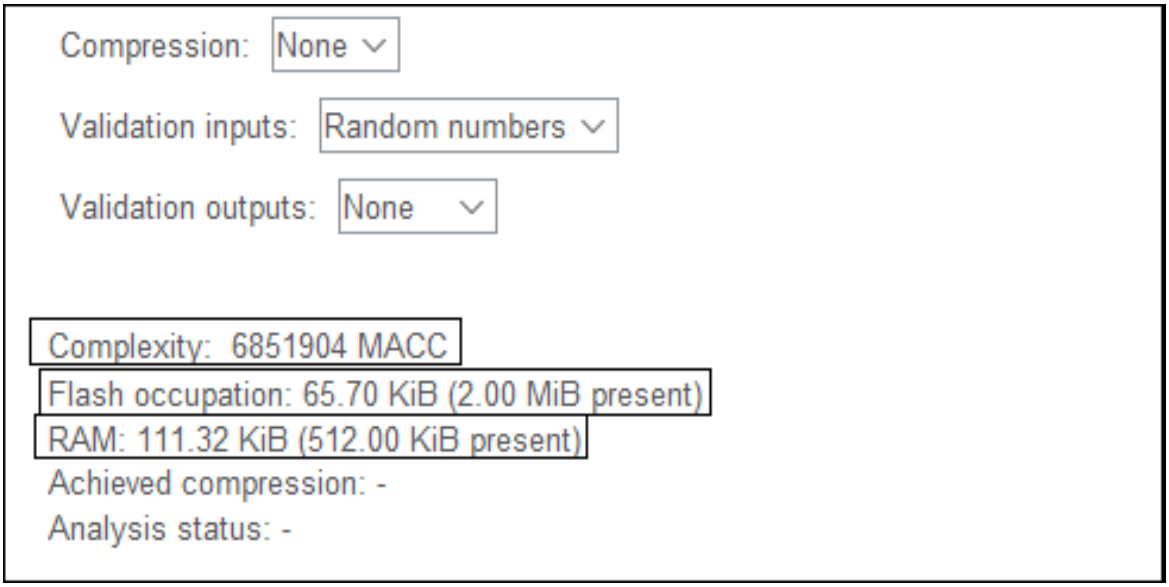


Hình 4-3: *Chọn phiên bản và cấu hình X-CUBE-AI.*

4.1.2 Đánh giá mô hình bằng công cụ STM32Cube AI:

Sau khi có được công cụ này nhóm đã tiến hành tải mô hình đã huấn luyện lên Cube AI để đánh giá. Kết quả mà công cụ STM32 Cube AI đưa ra yêu cầu tài nguyên phần cứng tối thiểu để chạy được mô hình là:

- 111.32 KB RAM.
- 65.70 Kb Flash



Hình 4-4: *Tài nguyên mà mô hình AI sẽ chiếm dụng.*

So sánh với board STM32H743ZI phần cứng của nó là:

- 1 MB RAM
- 2 MB Flash
- Tần số hoạt động tối đa là 480Mhz

Với các thông số trên thì đã đáp ứng được yêu cầu phần cứng để chạy mô hình AI, cũng như chạy một số tác vụ khác

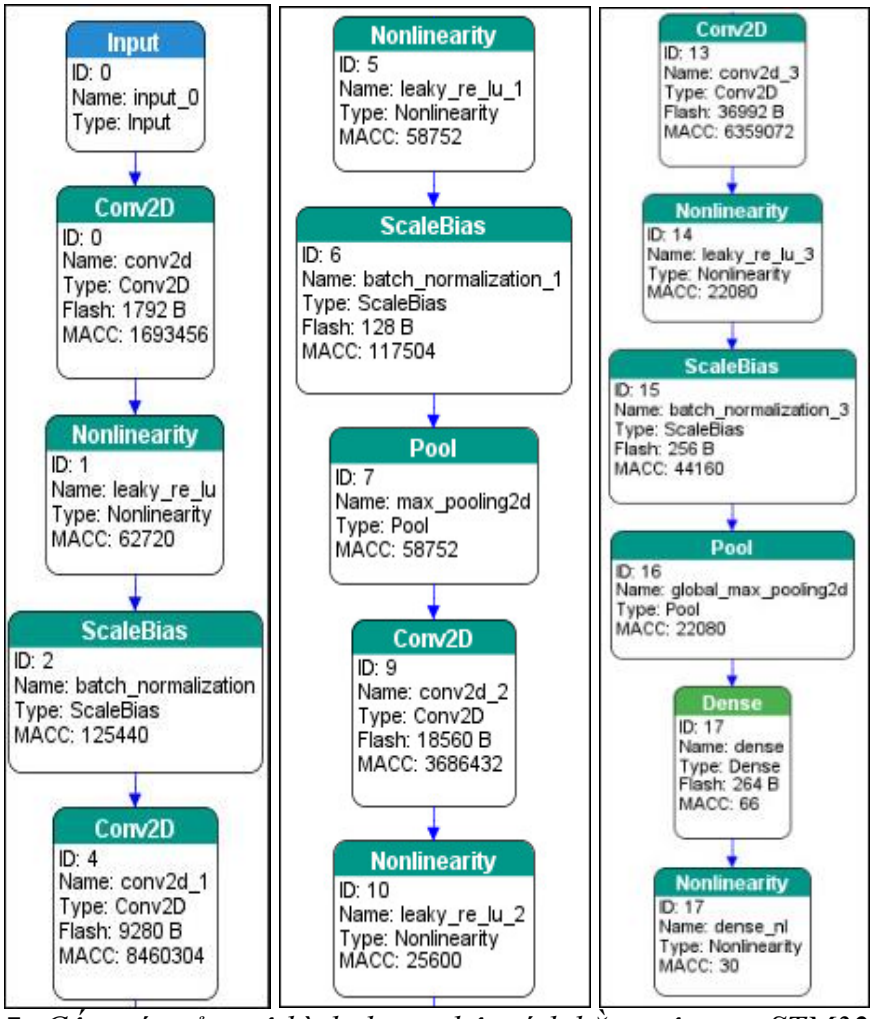
Đồng thời thì công cụ này con giúp chúng ta đánh giá độ phức tạp của mô hình bằng chỉ số MACC = 6.851.904. Với việc sử dụng vi điều khiển STM32H743ZI có lõi ARM cortex M7 thì số cycles trên 1MACC trung bình là 6 cycles/1MACC với độ phức tạp của mô hình này cùng với việc cấu hình vi điều khiển ở tần số fclock = 480Mhz thì thời gian để xử lý từ lúc đưa dữ liệu vào đến lúc đưa ra kết quả là:

$$t = \frac{n \times MACC}{fclock} = \frac{6 \times 6.851.904}{480 \times 10^6} \approx 0.08s$$

- Với n là số cycles trên 1 MACC
- fclock là tần số hoạt động của vi điều khiển

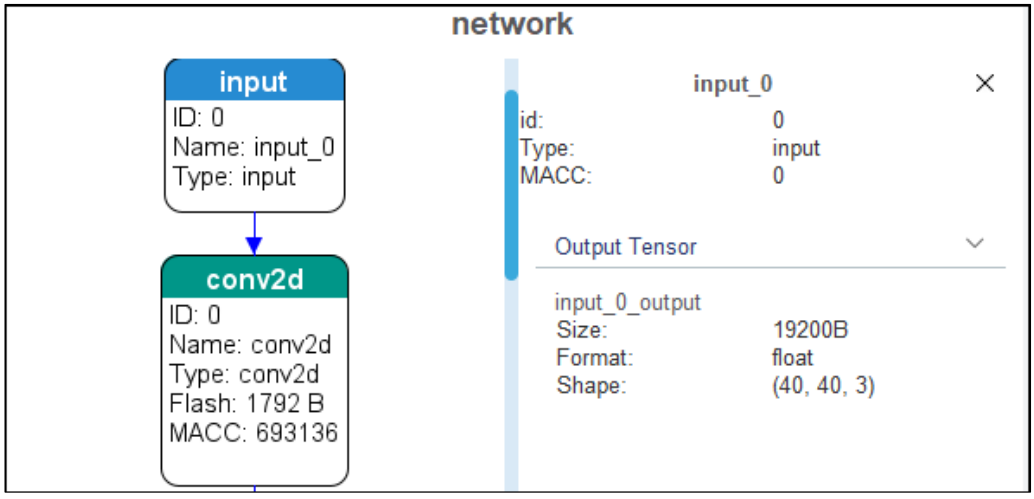
Vì STM32 có tài nguyên giới hạn và phải chạy nhiều chương trình khác ngoài chương trình AI như chương trình thu thập ảnh đầu vào và điều khiển động cơ Servo SG90. Thời gian để chạy chương trình AI chỉ mất 0.08s nên sẽ đáp ứng được tốc độ Realtime của mô hình.

Cấu trúc mô hình sau khi được phân tích bởi công cụ STM32 Cube.AI thông qua lệnh Analyze và Show Graph để hiển thị thông tin đầu vào và đầu ra của mỗi lớp như hình 4.3.



Hình 4-5: *Cấu trúc của mô hình được phân tích bằng công cụ STM32 Cube AI.*

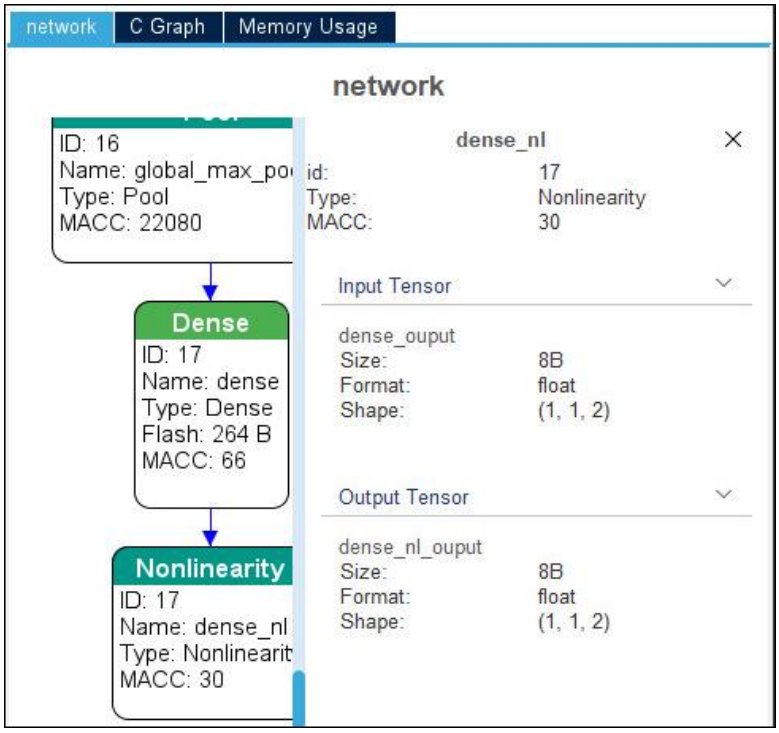
Khi sử dụng công cụ STM32CubeAI để phân tích mô hình ta cần quan tâm đến thông số của lớp đầu vào (Input) và lớp đầu ra (Output). Với mô hình của nhóm đầu vào sẽ là một hình ảnh có kích thước là 40x40x3 và loại dữ liệu là float.



Hình 4-6: Kích thước của ảnh đầu vào mô hình.

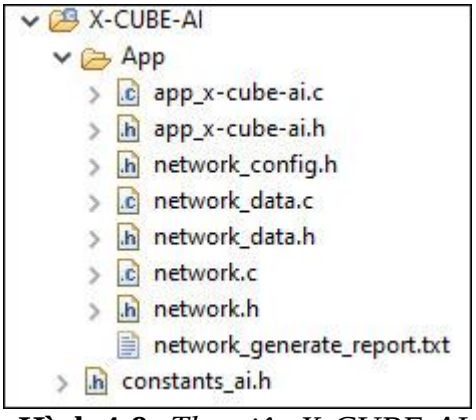
Với dữ liệu đầu ra là một mảng có hai phần tử có kích thước là 8 byte và ở loại dữ liệu là float, trong đó phần tử đầu tiên sẽ giá trị của lửa và phần tử thứ hai là không lửa.

Output = {Fire, Not_fire};



Hình 4-7: *Dữ liệu ra của mô hình.*

Khi hoàn tất việc đánh giá mô hình thì cũng đồng nghĩa với việc triển khai mô hình thành công. Ta tiến hành hoàn tất cấu hình các ngoại vi và tiến hành sinh ra thư viện X-CUBE-AI để cho người dùng sử dụng.



Hình 4-8: *Thư viện X-CUBE-AI.*

Ở trong các thư viện X-CUBE-AI có rất nhiều thư viện giúp người dùng triển khai được mô hình AI của mình trên vi điều khiển STM32. Trong đó ta chú ý đến một hàm 'ai_run' được viết trong thư viện 'app_x-cube-ai.c' như hình 4.9. Ở hàm này có hai tham số là dữ liệu vào 'data_in' và dữ liệu ra 'data_out'. Với dữ liệu vào ta cần xử lý cho

đúng yêu cầu của mô hình AI như hình 4.6 và đối với dữ liệu ra hàm sẽ trả về xác xuất có lửa và không có lửa.

```
154 int ai_run(void *data_in, void *data_out)
155 {
156   ai_i32 batch;
157
158   ai_buffer *ai_input = network_info.inputs;
159   ai_buffer *ai_output = network_info.outputs;
160
161   ai_input[0].data = AI_HANDLE_PTR(data_in);
162   ai_output[0].data = AI_HANDLE_PTR(data_out);
163
164   batch = ai_network_run(network, ai_input, ai_output);
165   if (batch != 1) {
166     ai_log_err(ai_network_get_error(network),
167         "ai_network_run");
168     return -1;
169   }
170
171   return 0;
172 }
173
```

Hình 4-9: *Hàm ai_run.*

4.2 Thu thập dữ liệu hình ảnh trong thực tế:

Để lấy được các hình ảnh từ môi trường bên ngoài nhóm đã tiến hành sử dụng OPENMV Cam H7 để thu thập dữ liệu và gửi về vi điều khiển STM32 để chạy mô hình AI. OPENMV có một công cụ đi kèm là OPENMV IDE cho người dùng lập trình OPENMV của mình bằng ngôn ngữ MicroPython [8].

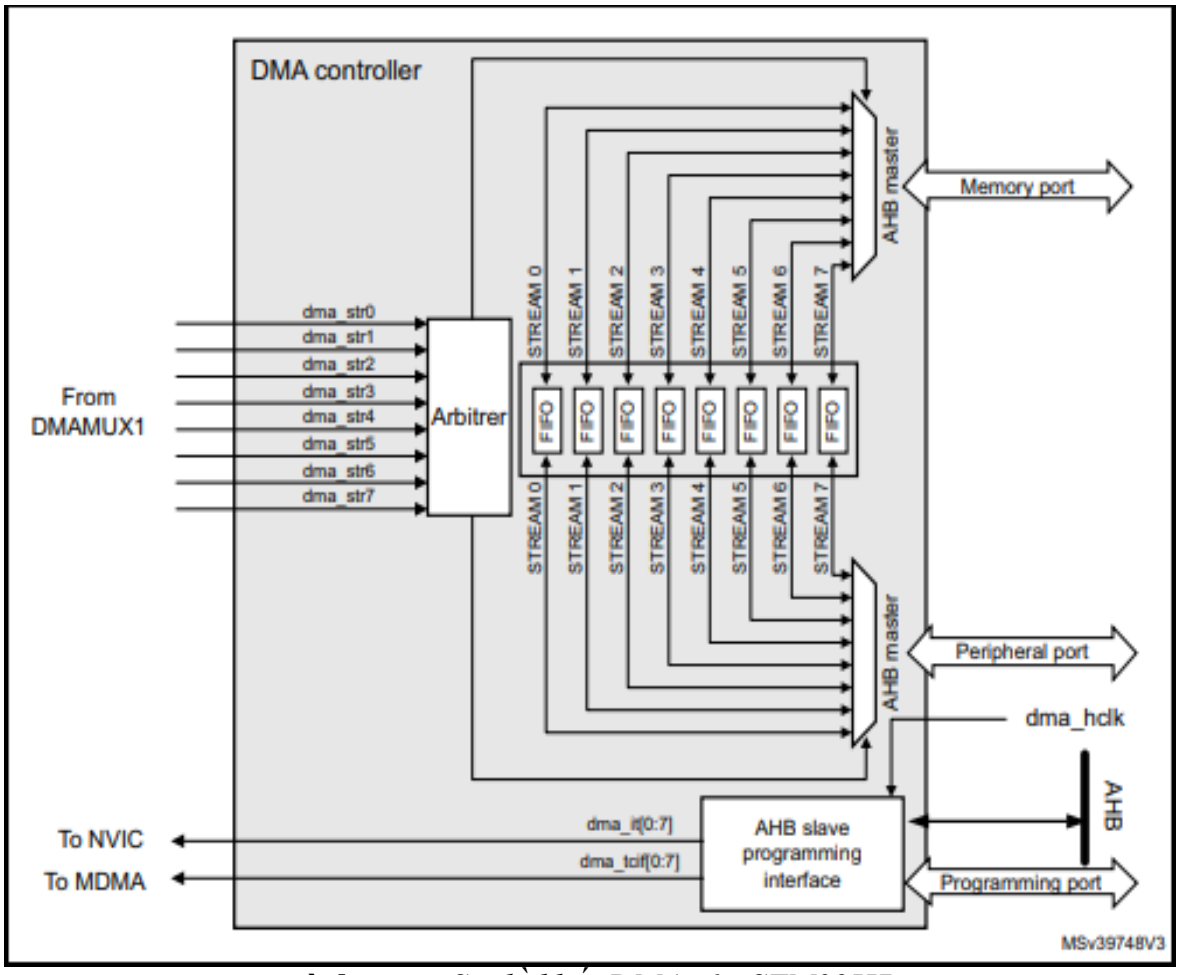
Nhóm tiến hành lập trình trên OpenMV IDE để cấu hình chế độ hoạt động của camera với định dạng ảnh RGB565, kích thước ảnh là 120x120 pixel. Cấu hình UART để giao tiếp với STM32 ở tốc độ 115200 bit/s.

Hình 4-10: *Chương trình truyền hình ảnh cho STM32 trên OPENMV IDE.*

Để nhận dữ liệu hình ảnh liên tục từ OPENMV CAM nhóm đã tiến hành cấu hình nhận DMA. Quá trình nhận DMA (Direct memory access) được sử dụng để nhận dữ liệu liên tục từ ngoại vi USART đến bộ nhớ hoặc ngược lại mà không cần sự tác động của CPU. Việc này giúp cho CPU được rảnh rỗi và thực hiện các công việc khác của chương trình như xử lý dữ liệu, chạy chương trình AI.

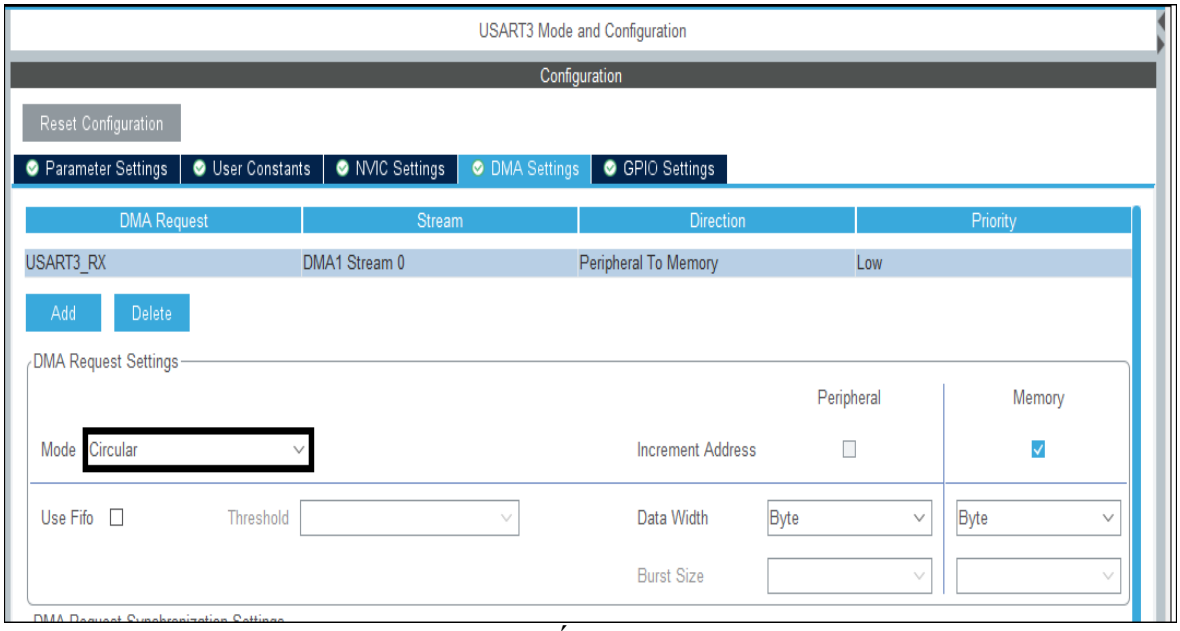
DMA có hai chế độ hoạt động Normal mode và Circular mode:

- Normal mode: Khi chế độ này hoạt động thì dữ liệu sẽ được DMA truyền đến giới hạn đã khai báo trước đó và dừng lại. Muốn nhận dữ liệu lại phải khai báo hàm để nhận lại, khi nhận lại thì dữ liệu sẽ lưu lại từ đầu.
- Circular mode: Ở chế độ này thì DMA sẽ truyền hết một lượng dữ liệu đã cho giới hạn trước đó, khi có dữ liệu mới nó sẽ lưu tiếp vào dữ liệu trước đó và khi bộ đệm nhận dữ liệu đầy thì sẽ quay lại từ đầu.



Hình 4-11: *Sơ đồ khối DMA của STM32H7.*

Nhóm tiến hành cấu hình USART3 trên STM32 với tốc độ baud rate là 115200bit/s, nhận DMA ở chế độ Circular để nhận dữ liệu liên tục.



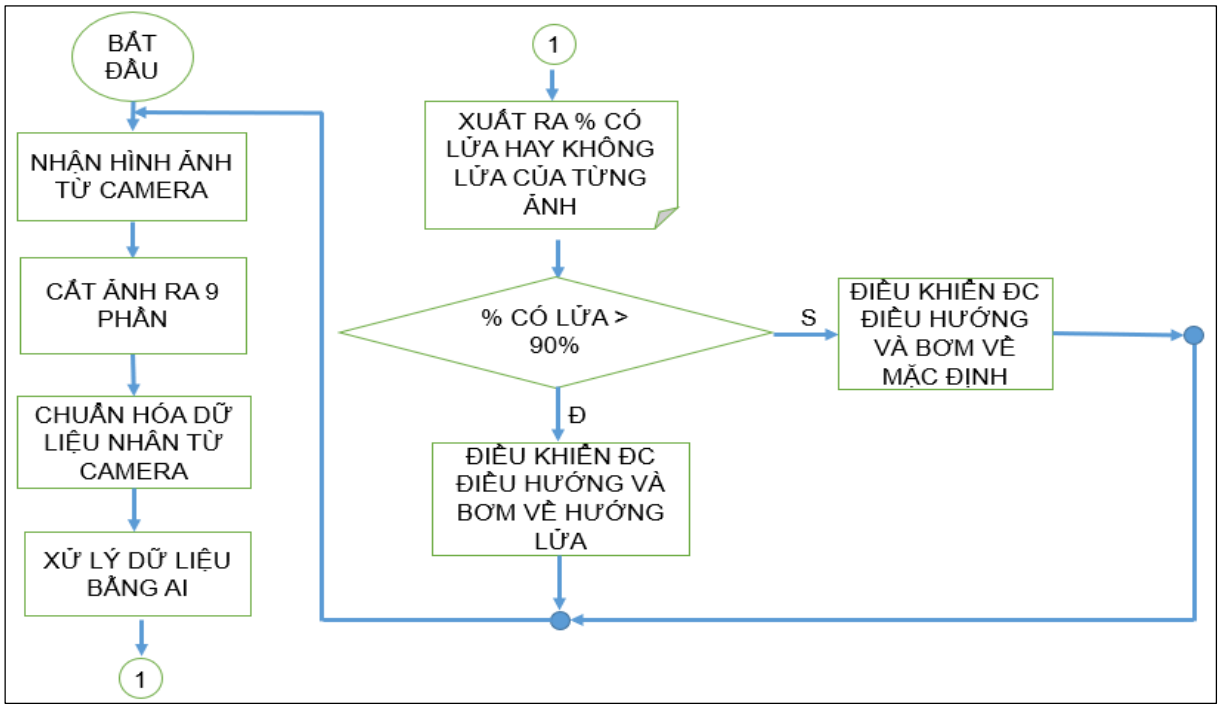
Hình 4-12: *Cấu hình nhận DMA.*

4.3 Tiến hành chạy chương trình AI trên vi điều khiển STM32:

4.3.1 Xây dựng lưu đồ thuật toán trên STM32:

Khi đã Train được mô hình AI nhận diện lửa thành công thì nhóm tiến hành sử dụng công cụ STM32Cube AI để chuyển thành mã C. Sử dụng các hàm AI và viết thêm các chương trình để nhận dữ liệu hình ảnh từ camera và điều khiển các động cơ khác.

Lưu đồ thuật toán trên vi điều khiển STM32:



Hình 4-13: *Lưu đồ thuật toán của vi điều khiển STM32.*

4.3.2 Xử lý dữ liệu và phương pháp để xác định vị trí nguồn nhiệt trên STM32:

Hình ảnh sau khi được thu thập bằng OpenMV Cam H7 sẽ có dạng một ma trận kích thước 120x120 pixel, ta tiến hành lưu ma trận này vào một mảng 2 chiều và sau đó chia các giá trị của ma trận này thành 9 phần tương ứng với 9 ảnh nhỏ.

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
    uint8_t i, j;

    // img 1
    for(i = 0; i<40; i++){
        for(j = 0; j<40; j++){
            buf1[i][j] = buf[i][j];
        }
    }
    transfer(buf1, bf1);

    // img 2
    for(i = 0; i<40; i++){
        for(j = 40; j<80; j++){
            buf2[i][j-40] = buf[i][j];
        }
    }
    transfer(buf2, bf2);

    // img 3
    for(i = 0; i<40; i++){
        for(j = 80; j<120; j++){
            buf3[i][j-80] = buf[i][j];
        }
    }
    transfer(buf3, bf3);
```

```
    // img 4
    for(i = 40; i<80; i++){
        for(j = 0; j<40; j++){
            buf4[i-40][j] = buf[i][j];
        }
    }
    transfer(buf4, bf4);

    // img 5
    for(i = 40; i<80; i++){
        for(j = 40; j<80; j++){
            buf5[i-40][j-40] = buf[i][j];
        }
    }
    transfer(buf5, bf5);

    // img 6
    for(i = 40; i<80; i++){
        for(j = 80; j<120; j++){
            buf6[i-40][j-80] = buf[i][j];
        }
    }
    transfer(buf6, bf6);
```

```
    // img 7
    for(i = 80; i<120; i++){
        for(j = 0; j<40; j++){
            buf7[i-80][j] = buf[i][j];
        }
    }
    transfer(buf7, bf7);

    // img 8
    for(i = 80; i<120; i++){
        for(j = 40; j<80; j++){
            buf8[i-80][j-40] = buf[i][j];
        }
    }
    transfer(buf8, bf8);

    // img 9
    for(i = 80; i<120; i++){
        for(j = 80; j<120; j++){
            buf9[i-80][j-80] = buf[i][j];
        }
    }
    transfer(buf9, bf9);
```

Hình 4-14: *Chương trình cắt ảnh thành 9 phần.*

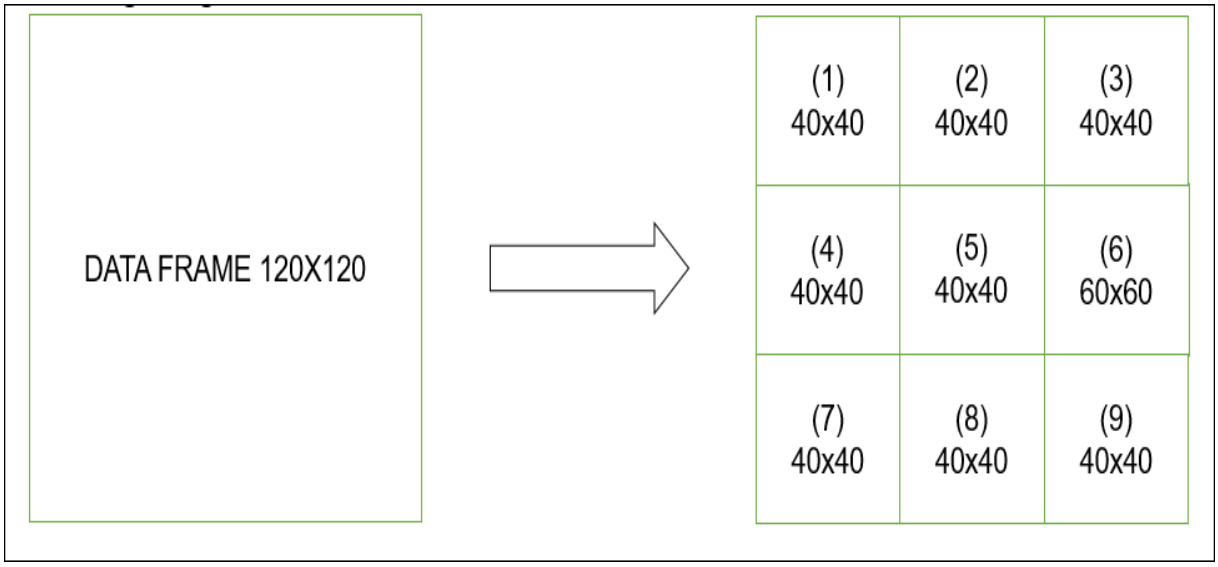
Khi đã có dữ liệu ta tiến hành chuẩn hóa các giá trị của dữ liệu lại tương tự như mục (3.3.2) đã trình bày, các giá trị nhận được sau khi đã chuẩn hóa nằm trong khoảng 0 tới 1 để đưa vào mô hình AI.

Expression	Type	Value
bufD	float [19008]	[19008]
[0..99]		[19008]
(x)= bufD[0]	float	0.03125
(x)= bufD[1]	float	0.0625
(x)= bufD[2]	float	0
(x)= bufD[3]	float	0.03125
(x)= bufD[4]	float	0.0625
(x)= bufD[5]	float	0
(x)= bufD[6]	float	0.0625
(x)= bufD[7]	float	0.0625
(x)= bufD[8]	float	0
(x)= bufD[9]	float	0.09375
(x)= bufD[10]	float	0.078125
(x)= bufD[11]	float	0.03125
(x)= bufD[12]	float	0.09375
(x)= bufD[13]	float	0.09375
(x)= bufD[14]	float	0.03125
(x)= bufD[15]	float	0.125
(x)= bufD[16]	float	0.109375
(x)= bufD[17]	float	0.03125
(x)= bufD[18]	float	0.125
(x)= bufD[19]	float	0.09375

Hình 4-15: *Dữ liệu đã được chuẩn hóa theo yêu cầu đầu vào.*

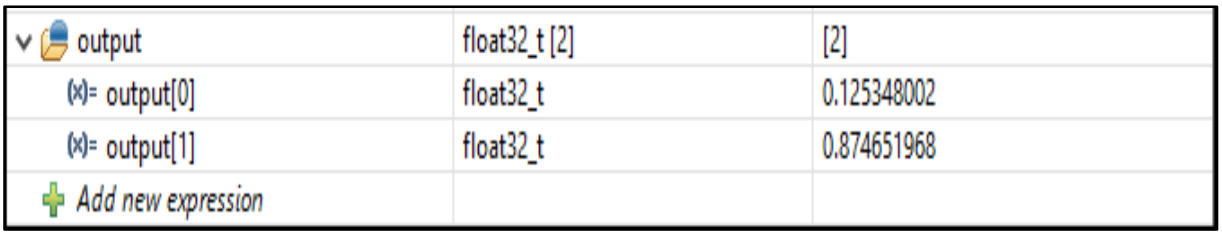
Phương pháp để xác định vị trí nguồn nhiệt

Khi sử dụng công cụ STM32Cube AI để chuyển đổi mô hình thành mã C, thì công cụ này sinh ra các thư viện AI phục vụ cho quá trình nhận dạng hình ảnh ở vi điều khiển. Ở đây ta quan tâm hàm '*ai_run(input, output)*' trong đó với tham số '*input*' là một dữ liệu của một hình ảnh đã được chuẩn hóa. Và tham số '*output*' gồm mảng có hai giá trị chứa xác xuất là lửa và không lửa, tổng của hai giá trị này sẽ bằng 1.



Hình 4-16: *Ảnh đầu vào và các ảnh nhỏ sau khi cắt.*

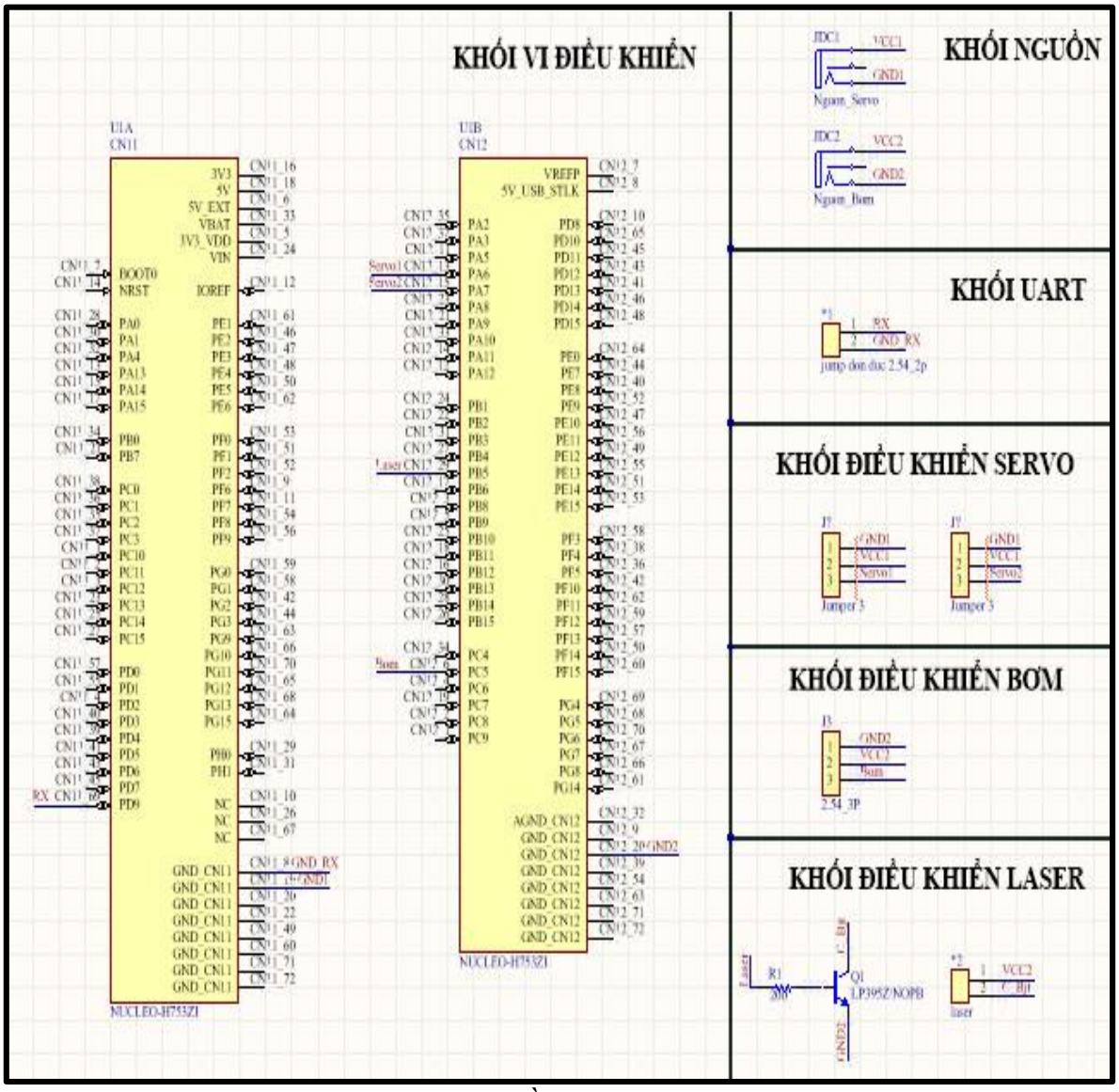
Từ một ảnh đầu vào có kích thước 120x120 pixel ta tiến hành chia ảnh này thành 9 ảnh nhỏ mỗi ảnh có kích thước 40x40 pixel, ta tiến hành đưa lần lượt 9 ảnh vào hàm AI mỗi ảnh lúc này tương ứng với một vị trí, hàm AI sẽ trả về phần trăm xác xuất có lửa hay không có lửa của từng ảnh, khi hàm trả về giá trị ở output[0] > 90% thì ảnh đó có lửa và lưu vị trí đó lại vào biến "poss", và ngược lại. Biến "poss" này sẽ được gán vị trí thông qua ảnh từ 1 đến 9 và cũng phải thỏa mãn điều kiện hàm AI trả về output[0] > 90%, nếu hàm AI trả về output[0] < 90% thì chúng ta gán nó bằng 0. Ví dụ, khi ta đưa ảnh thứ 2 vào hàm AI nếu hàm AI trả về kết quả output[0] > 90% thì ta gán biến poss = 2 tương đương với vị trí là số 2, và nếu output[0] < 90% thì gán biến poss = 0. Khi chúng ta có các vị trị chúng ta tiến hành điều khiển động cơ điều hướng đến vị trí tương ứng



Hình 4-17: *Xem xác xuất của một hình ảnh qua Debug.*

4.4 Thi công hệ thống:

Nhóm tiến hành vẽ một bảng mạch bằng Altium, sơ đồ nguyên lý của mạch bao gồm vi điều khiển STM32 và các chân để điều các động cơ như Servo, Bơm.

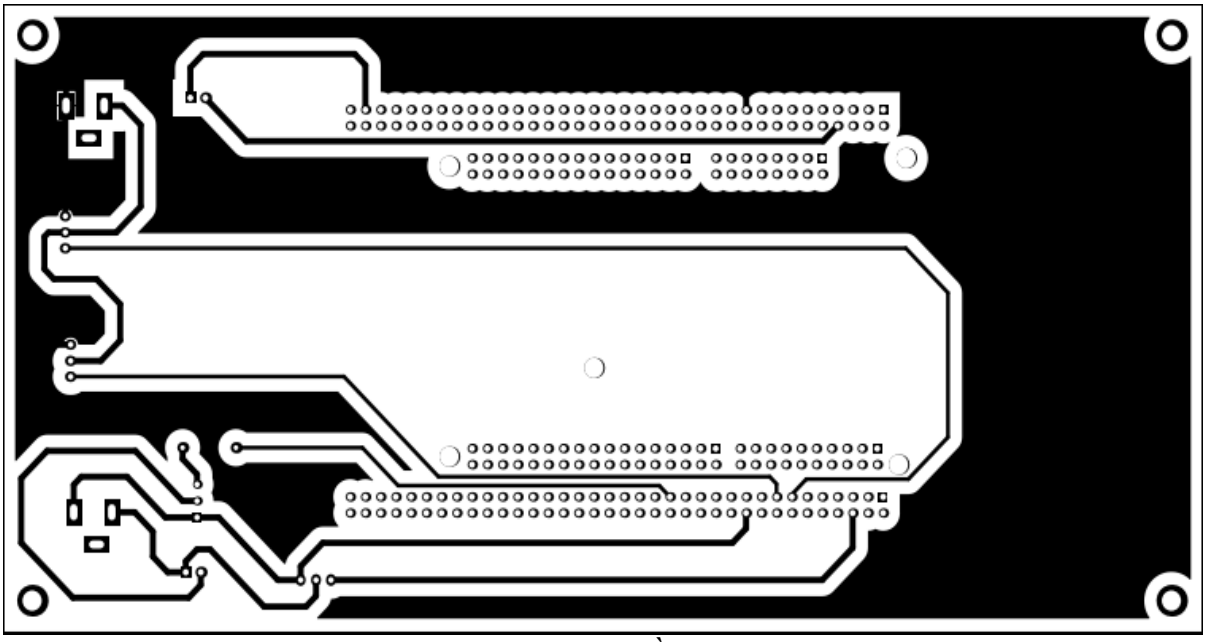


Hình 4-18: *Sơ đồ nguyên lý của mạch.*

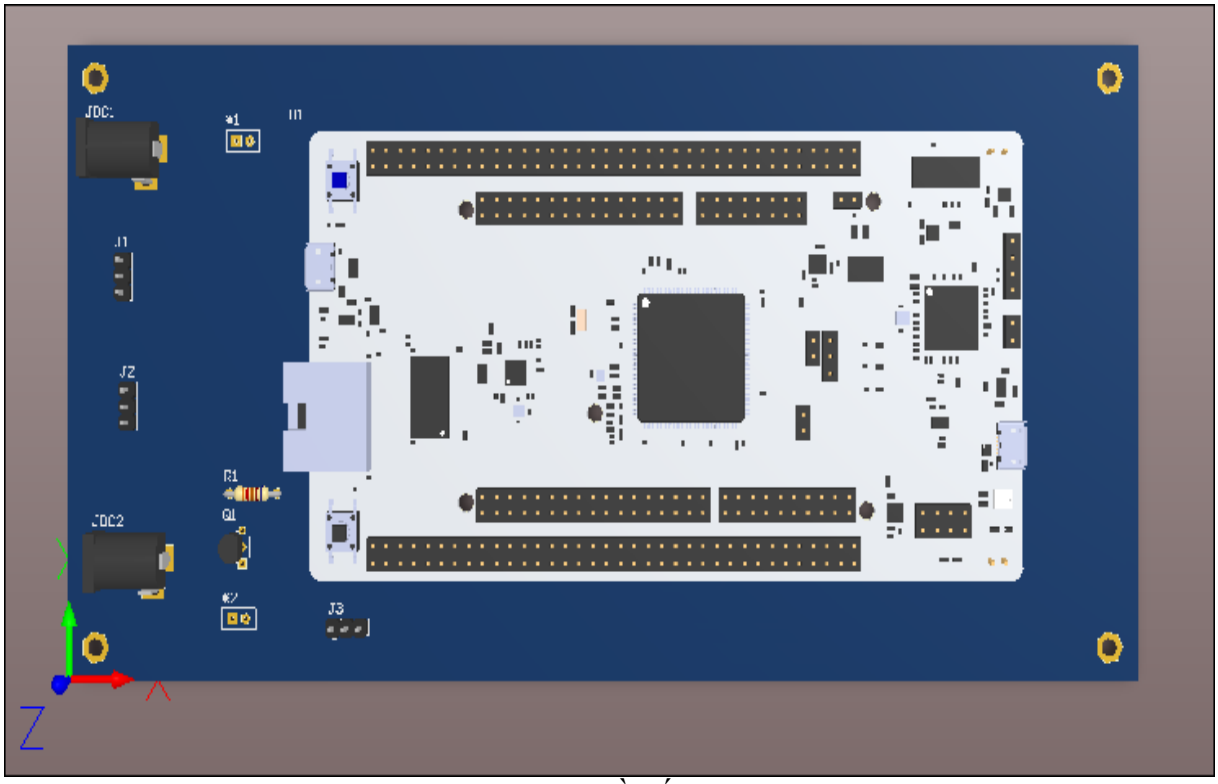
Trong đó khối vi điều khiển gồm vi điều khiển STM32H7 có các chân nối với các các khối khác để điều khiển và giao tiếp.

- Khối nguồn để cấp nguồn cho các động cơ như bơm, servo, laser.
- Khối UART để giao tiếp với OpenMV CAM H7 để nhận ảnh từ camera.
- Khối Servo gồm hai Servo dùng để điều khiển động cơ điều hướng đến vị trí đang có cháy.
- Khối điều khiển bơm dùng đề bơm nước dập lửa.
- Khối Laser dùng để chiếu đến vị trí đang có cháy.

Khi đã hoàn thành sơ đồ nguyên lý ta tiến hành bố trí linh kiện có trên mạch. Nhóm tiến hành đi dây và phủ đồng cho mạch.



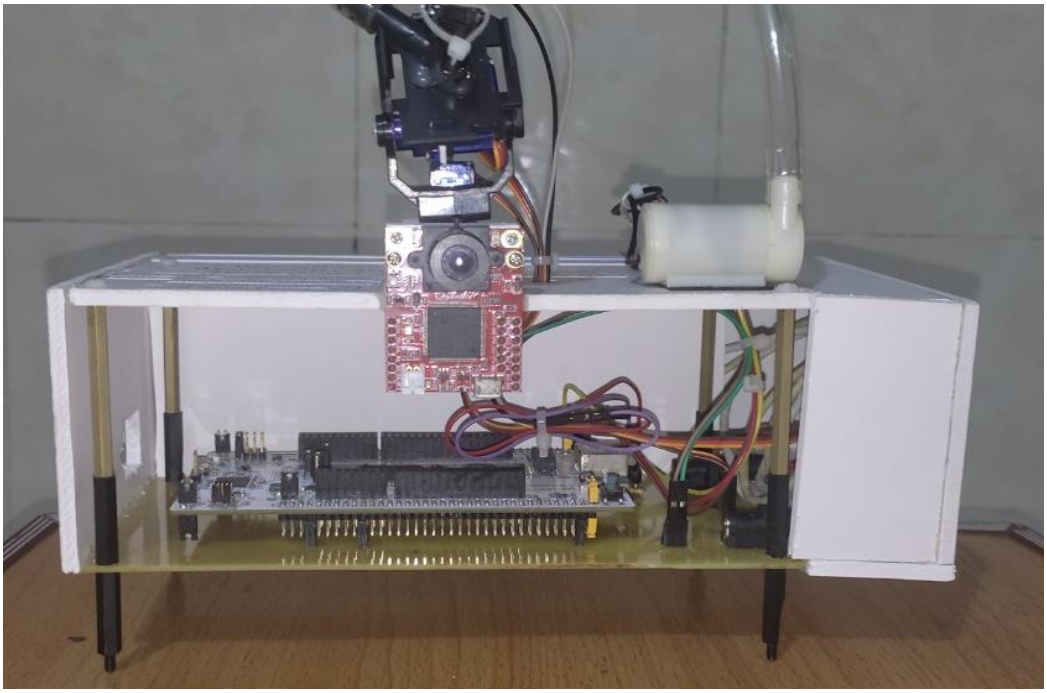
Hình 4-19: *Sơ đồ mạch in.*



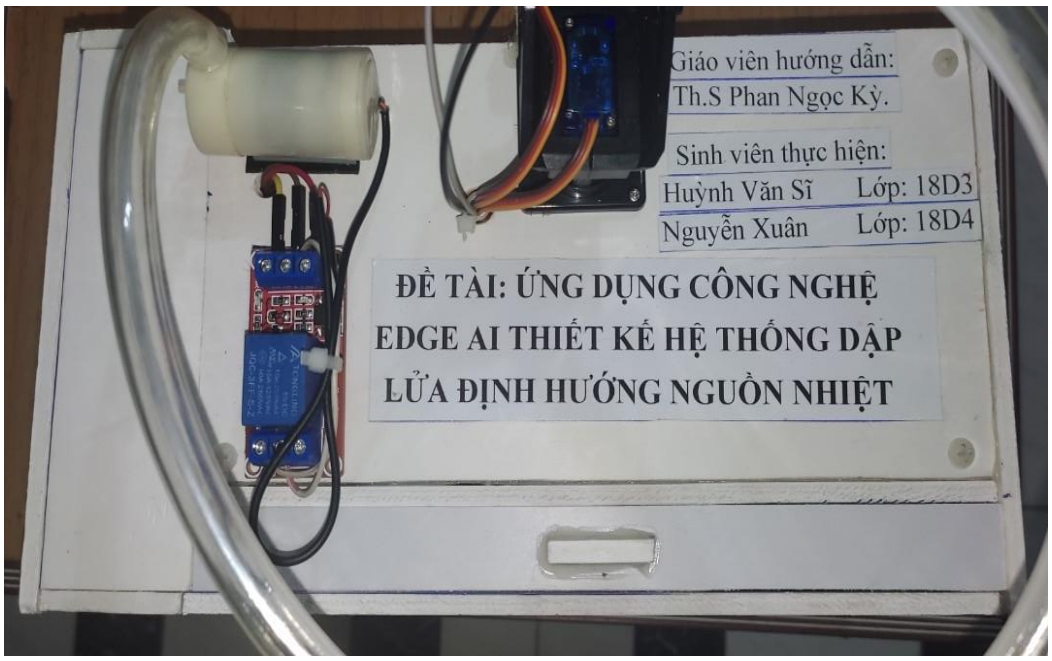
Hình 4-20: *Sơ đồ bố trí linh kiện*

4.5 Kết quả:

Sau khi hoàn tất các chương trình điều khiển thì nhóm tiến hành nạp chương trình xuống vi điều khiển STM32. Tiến hành đấu nối các phần cứng khác như động cơ bơm nước, động cơ điều hướng bao gồm hai Servo, và module laser để chiếu đến vị trí đang có cháy.



Hình 4-21: *Mặt trước mô hình.*



Hình 4-22: Mặt trên mô hình.

Nguyên lý hoạt động của mô hình:

Khi ta đưa một ngọn lửa vào trước camera, thì camera chuyển hình ảnh này xuống vi điều khiển STM32H7 đang chạy chương trình AI sử lý ảnh. Lúc này STM32H7 tiến hành phân tích vị trí có lửa và điều khiển động cơ điều hướng đến vị trí có lửa, kích hoạt máy bơm nước để dập lửa. Nếu hết lửa thì động cơ quay về vị trí mặc định và khi phát hiện có lửa lại thì động cơ tiếp tục xác định vị trí và dập lửa.

Chạy mô hình trong thực tế:

Khi đã hoàn thiện các phần về phần cứng và phần mềm nhóm tiến hành chạy chương trình trong thực tế. Nhóm thiết kế một bảng bao gồm chín đế tương ứng với chín vị trí khi nhóm đưa một nguồn nhiệt vào vị trí số bảy thì hệ thống sẽ tiến phân tích vị của lửa là đang ở vị trí số bảy, tiến hành điều hướng vòi xịt tới vị trí có lửa chiếu đèn laser tới vị trí đang có lửa và khởi động động cơ bơm phun nước để dập lửa.

Hình 4-23: *Khi mô hình phát hiện lửa ở vị trí số bảy.*

Nhóm tiến hành di chuyển vị trí ngọn lửa sang vị trí số năm thì mô hình lúc này sẽ phát hiện lửa đang ở vị trí khác, vi điều khiển sẽ tiến hành điều khiển động cơ điều hướng sang vị trí số năm chiếu đèn laser và bật máy bơm để dập lửa.

Hình 4-24: *Khi mô hình phát hiện lửa ở vị trí số năm.*

Khi nhóm tắt lửa thì mô hình sẽ phát hiện ra không vị trí nào đang có cháy, mô hình điều khiển động cơ điều hướng về vị trí mặc định và tắt động cơ bơm và đèn laser.

Hình 4-25: *Khi mô hình không phát hiện cháy.*

KẾT LUẬN

Đề tài dã đạt được mục tiêu mà nhóm đã đề ra ban đầu.

Với mô hình phát hiện và chữa cháy theo nguồn nhiệt này sẽ giúp ta chữa cháy sớm tránh lửa lan rộng gây thiệt hại to lớn về của cải và tính mạng con người. Mô hình được triển khai trên vi điều khiển STM32 giúp ta giảm được rất nhiều chi phí so với việc triển khai trên máy tính hoặc siêu máy tính. Với việc mô hình có chi phí rẻ hơn sẽ thuận tiện cho việc nhân rộng sản phẩm.

Hướng phát triển của đề tài:

Để mô hình có thể chạy nhanh hơn ta có sử dụng một vi điều khiển STM32 thế hệ mới có Dual core. Nó sẽ giúp mô hình có khả năng đa nhiệm, lúc này STM32 sẽ dụng một core để thu thập dữ liệu và một core để chạy mô hình AI, việc này sẽ giúp mô hình chạy nhanh hơn đáp ứng được thời gian thực.

Để mô hình có thể chạy với độ chính xác cao hơn hoạt động ổn định ở nhiều môi trường khác nhau thì ta có thể thu thập dữ liệu nhiều hơn phong phú hơn nhiều môi trường hơn, từ đó đưa vào huấn luyện mô hình huấn luyện AI thì mô hình sẽ nhận dạng một cách chính xác và ổn định hơn.

Có thể sử dụng nhiều động cơ điều hướng và động cơ bơm để khi trong trường hợp nếu có nhiều hơn một đốm lửa thì hệ thống có thể xác định và dập nhiều vị trí lửa.

TÀI LIỆU THAM KHẢO

[1] *STM32H7 Nucleo-144 boards (User manual).*

https://www.st.com/resource/en/user_manual/dm00499160-stm32h7-nucleo144-boards-mb1364-stmicroelectronics.pdf

[2] "*Convolutional neural network*".
https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

[3] S. Ioffe and C. Szegedy, "*Batch normalization: Accelerating deep network training by reducing internal covariate shift*," arXiv preprint arXiv:1502.03167, 2015.

[4] Y. Gal and Z. Ghahramani, *"Dropout as a bayesian approximation: Representing model uncertainty in deep learning*," international conference on machine learning, pp. 1050–, 2016.

[5] Y. Gal and Z. Ghahramani, "*Dropout as a bayesian approximation: Representing model*," international conference on machine learning (2016).

[6] "*Loss function and Optimization, Lecture 3 | Loss Functions and Optimization*."
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture3.pdf.

[7] "*Getting started with X-CUBE-AI Expansion Package for Artificial Intelligence (AI)*"
https://www.st.com/resource/en/user_manual/dm00570145-getting-started-with-xcubeai-expansion-package-for-artificial-intelligence-ai-stmicroelectronics.pdf

[8] *MicroPython Documenttion.*
https://docs.openmv.io/index.html

PHỤ LỤC

Code cho vi điều khiển STM32:

```
/* USER CODE BEGIN Header */
/**

******************************************************************************
  * @file           : main.c
  * @brief          : Main program body

******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2022 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *

******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "app_x-cube-ai.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
//#include "arm_math.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
```

```
/* USER CODE BEGIN PD */
#define dai 120
#define rong 120
#define dai1 40
#define rong1 40
#define tong dai1*rong1*3
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

CRC_HandleTypeDef hcrc;

TIM_HandleTypeDef htim3;

UART_HandleTypeDef huart3;
DMA_HandleTypeDef hdma_usart3_rx;

/* USER CODE BEGIN PV */
// buf > mảng tổng
// buf1...9 > 9 cái ảnh
// bf1...9 > 9 cái ảnh chuyển v?? 0 -> 1

uint16_t buf[dai][rong] = {0};

uint16_t buf1[dai1][rong1] = {0};
uint16_t buf2[dai1][rong1] = {0};
uint16_t buf3[dai1][rong1] = {0};
uint16_t buf4[dai1][rong1] = {0};
uint16_t buf5[dai1][rong1] = {0};
uint16_t buf6[dai1][rong1] = {0};
uint16_t buf7[dai1][rong1] = {0};
uint16_t buf8[dai1][rong1] = {0};
uint16_t buf9[dai1][rong1] = {0};

//buf đã chuyển đổi 0->1 // **
float bf1[tong] = {0};
float bf2[tong] = {0};
float bf3[tong] = {0};
float bf4[tong] = {0};
float bf5[tong] = {0};
```

```
float bf6[tong] = {0};
float bf7[tong] = {0};
float bf8[tong] = {0};
float bf9[tong] = {0};


float cl1[2] = {0};
float cl2[2] = {0};
float cl3[2] = {0};
float cl4[2] = {0};
float cl5[2] = {0};
float cl6[2] = {0};
float cl7[2] = {0};
float cl8[2] = {0};
float cl9[2] = {0};
int8_t dem1 = 0;
int8_t dem2 = 0;
//uint8_t dem3 = 1;
//uint8_t dem4 = 1;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_USART3_UART_Init(void);
static void MX_CRC_Init(void);
static void MX_TIM3_Init(void);
/* USER CODE BEGIN PFP */
void transfer(uint16_t buff[dai1][rong1], float bfId[tong]);
uint8_t findMax2(float cl1[2], float cl2[2], float cl3[3], float cl4[2], float cl5[2], float
cl6[2], float cl7[2], float cl8[2], float cl9[2]);
void conServo(uint8_t pos);
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
	uint8_t i, j;

	// img 1
	for(i = 0; i<40; i++){
		for(j = 0; j<40; j++){
			buf1[i][j] = buf[i][j];
		}
```

```
}
transfer(buf1, bf1);

// img 2
for(i = 0; i<40; i++){
	for(j = 40; j<80; j++){
		buf2[i][j-40] = buf[i][j];
	}
}
transfer(buf2, bf2);

// img 3
for(i = 0; i<40; i++){
	for(j = 80; j<120; j++){
		buf3[i][j-80] = buf[i][j];
	}
}
transfer(buf3, bf3);

// img 4
for(i = 40; i<80; i++){
	for(j = 0; j<40; j++){
		buf4[i-40][j] = buf[i][j];
	}
}
transfer(buf4, bf4);

// img 5
for(i = 40; i<80; i++){
	for(j = 40; j<80; j++){
		buf5[i-40][j-40] = buf[i][j];
	}
}
transfer(buf5, bf5);

// img 6
for(i = 40; i<80; i++){
	for(j = 80; j<120; j++){
		buf6[i-40][j-80] = buf[i][j];
	}
}
transfer(buf6, bf6);

// img 7
for(i = 80; i<120; i++){
```

```
            for(j = 0; j<40; j++){
                buf7[i-80][j] = buf[i][j];
            }
        }
        transfer(buf7, bf7);

        // img 8
        for(i = 80; i<120; i++){
            for(j = 40; j<80; j++){
                buf8[i-80][j-40] = buf[i][j];
            }
        }
        transfer(buf8, bf8);

        // img 9
        for(i = 80; i<120; i++){
            for(j = 80; j<120; j++){
                buf9[i-80][j-80] = buf[i][j];
            }
        }
        transfer(buf9, bf9);
}
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* Enable I-Cache---------------------------------------------------------*/
  SCB_EnableICache();

  /* Enable D-Cache---------------------------------------------------------*/
  SCB_EnableDCache();

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();
```

```
  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_USART3_UART_Init();
  MX_CRC_Init();
  MX_TIM3_Init();
  MX_X_CUBE_AI_Init();
  /* USER CODE BEGIN 2 */
  HAL_UART_Receive_DMA(&huart3, (uint8_t *)buf, (dai*rong)*2);
  HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2);
//  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, SET);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        ai_run(bf1, cl1);
        ai_run(bf2, cl2);
        ai_run(bf3, cl3);
        ai_run(bf4, cl4);
        ai_run(bf5, cl5);
        ai_run(bf6, cl6);
        ai_run(bf7, cl7);
        ai_run(bf8, cl8);
        ai_run(bf9, cl9);
        conServo(findMax2(cl1, cl2, cl3, cl4, cl5, cl6, cl7, cl8, cl9));
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

```
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Supply configuration update enable
  */
  HAL_PWREx_ConfigSupply(PWR_LDO_SUPPLY);
  /** Configure the main internal regulator output voltage
  */

__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE0);

  while(!__HAL_PWR_GET_FLAG(PWR_FLAG_VOSRDY)) {}
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_DIV1;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 4;
  RCC_OscInitStruct.PLL.PLLN = 60;
  RCC_OscInitStruct.PLL.PLLP = 2;
  RCC_OscInitStruct.PLL.PLLQ = 2;
  RCC_OscInitStruct.PLL.PLLR = 2;
  RCC_OscInitStruct.PLL.PLLRGE = RCC_PLL1VCIRANGE_3;
  RCC_OscInitStruct.PLL.PLLVCOSEL = RCC_PLL1VCOWIDE;
  RCC_OscInitStruct.PLL.PLLFRACN = 0;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType                                      =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2
```

```
                        |RCC_CLOCKTYPE_D3PCLK1|RCC_CLOCKTYPE_D1PCLK1;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.SYSCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB3CLKDivider = RCC_APB3_DIV2;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_APB1_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_APB2_DIV2;
  RCC_ClkInitStruct.APB4CLKDivider = RCC_APB4_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) !=
HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief CRC Initialization Function
  * @param None
  * @retval None
  */
static void MX_CRC_Init(void)
{

  /* USER CODE BEGIN CRC_Init 0 */

  /* USER CODE END CRC_Init 0 */

  /* USER CODE BEGIN CRC_Init 1 */

  /* USER CODE END CRC_Init 1 */
  hcrc.Instance = CRC;
  hcrc.Init.DefaultPolynomialUse = DEFAULT_POLYNOMIAL_ENABLE;
  hcrc.Init.DefaultInitValueUse = DEFAULT_INIT_VALUE_ENABLE;
  hcrc.Init.InputDataInversionMode = CRC_INPUTDATA_INVERSION_NONE;
  hcrc.Init.OutputDataInversionMode                                    =
CRC_OUTPUTDATA_INVERSION_DISABLE;
  hcrc.InputDataFormat = CRC_INPUTDATA_FORMAT_BYTES;
  if (HAL_CRC_Init(&hcrc) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN CRC_Init 2 */

  /* USER CODE END CRC_Init 2 */
```

```
}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 4799;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 999;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) !=
HAL_OK)
```

```
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_2) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
  HAL_TIM_MspPostInit(&htim3);

}

/**
  * @brief USART3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART3_UART_Init(void)
{

  /* USER CODE BEGIN USART3_Init 0 */

  /* USER CODE END USART3_Init 0 */

  /* USER CODE BEGIN USART3_Init 1 */

  /* USER CODE END USART3_Init 1 */
  huart3.Instance = USART3;
  huart3.Init.BaudRate = 115200;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
  huart3.Init.Mode = UART_MODE_TX_RX;
```

```
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart3.Init.ClockPrescaler = UART_PRESCALER_DIV1;
  huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart3) != HAL_OK)
  {
    Error_Handler();
  }
  if                                    (HAL_UARTEx_SetTxFifoThreshold(&huart3,
UART_TXFIFO_THRESHOLD_1_8) != HAL_OK)
  {
    Error_Handler();
  }
  if                                    (HAL_UARTEx_SetRxFifoThreshold(&huart3,
UART_RXFIFO_THRESHOLD_1_8) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_UARTEx_DisableFifoMode(&huart3) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART3_Init 2 */

  /* USER CODE END USART3_Init 2 */

}

/**
  * Enable DMA controller clock
  */
static void MX_DMA_Init(void)
{

  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Stream0_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Stream0_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Stream0_IRQn);

}
```

```
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_RESET);

  /*Configure GPIO pin : PC5 */
  GPIO_InitStruct.Pin = GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pin : PB5 */
  GPIO_InitStruct.Pin = GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */
void transfer(uint16_t buff[dai1][rong1], float bfId[tong]){
	uint16_t l = 0;
	uint8_t i, j;

	for(i = 0; i<dai1; i++){
		for(j = 0; j<rong1; j++){
```

```
                bfId[l++] = (buff[i][j] >> 11) / 32.0f; //R
                bfId[l++] = ((buff[i][j] >> 5) & 0x3f) / 64.0f; //G
                bfId[l++] = (buff[i][j] & 0x1f) / 32.0f; // B
            }
        }
}

uint8_t findMax2(float cl1[2], float cl2[2], float cl3[3], float cl4[2], float cl5[2], float cl6[2], float cl7[2], float cl8[2], float cl9[2]){
        uint8_t i;
        float max = 0.0f;
        if((cl1[0] > cl1[1]) | (cl2[0] > cl2[1]) | (cl3[0] > cl3[1]) | (cl4[0] > cl4[1]) | (cl5[0] > cl5[1]) | (cl6[0] > cl6[1]) | (cl7[0] > cl7[1]) | (cl8[0] > cl8[1]) | (cl9[0] > cl9[1])){
            if(cl1[0] > max){
                max = cl1[0];
                i = 1;
            }if(cl2[0] > max){
                max = cl2[0];
                i = 2;
            }if(cl3[0] > max){
                max = cl3[0];
                i = 3;
            }if(cl4[0] > max){
                max = cl4[0];
                i = 4;
            }if(cl5[0] > max){
                max = cl5[0];
                i = 5;
            }if(cl6[0] > max){
                max = cl6[0];
                i = 6;
            }if(cl7[0] > max){
                max = cl7[0];
                i = 7;
            }if(cl8[0] > max){
                max = cl8[0];
                i = 8;
            }if(cl9[0] > max){
                max = cl9[0];
                i = 9;
            }
            return i;
        }
        else{
            return 0;
```

```
	}
}

void conServo(uint8_t pos){
	switch(pos){
	case 1:
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 69);
//		HAL_Delay(200);
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 96);
//		HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, SET);
		break;
	case 2:
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 69);
//		HAL_Delay(200);
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 87);
//		HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, SET);
		break;
	case 3:
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 69);
//		HAL_Delay(200);
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 78);
//		HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, SET);
		break;
	case 4:
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 78);
//		HAL_Delay(200);
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 96);
//		HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, SET);
		break;
	case 5:
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 78);
//		HAL_Delay(200);
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 87);
//		HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
		HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, SET);
		break;
		//
	case 6:
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 78);
//		HAL_Delay(200);
		__HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 78);
```

```
//            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
              HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, SET);
              break;
        case 7:
              __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 87);
//            HAL_Delay(200);
              __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 96);
//            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
              HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, SET);
              break;
        case 8:
              __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 87);
//            HAL_Delay(200);
              __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 87);
//            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
              HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, SET);
              break;
        case 9:
              __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 87);
//            HAL_Delay(200);
              __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 78);
//            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, SET);
              HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, SET);
              break;
        case 0:
              __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 65);
//            HAL_Delay(200);
              __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 87);
//            HAL_GPIO_WritePin(GPIOC, GPIO_PIN_5, RESET);
              HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, RESET);
              break;
        }
}
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
```

```
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

Code chương trình OpenMV gửi ảnh sang STM32H7:

```
import sensor, image, time
import pyb
import utime

uart = pyb.UART(3, 115200, timeout_char = 1000)

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QQVGA)
sensor.set_windowing(120,120)
sensor.skip_frames(time = 2000)

while(True):
   img = sensor.snapshot()
   data = bytearray(img)
   uart.write(data)
   utime.sleep_ms(2000)
```