**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**

**KHOA KỸ THUẬT ĐIỆN TỬ II**



**BÁO CÁO CUỐI KỲ**

**ĐỒ ÁN THIẾT KẾ HỆ THỐNG NHÚNG**

**NHẬN DIỆN CỬ CHỈ BÀN TAY BẰNG ESP32CAM**

**NIÊN KHÓA: 2020 – 2025**

**Giáo viên hướng dẫn : Thầy Nguyễn Trọng Kiên**

**Sinh viên thực hiện:**

|  |  |  |
| --- | --- | --- |
| Họ và Tên | Mã số sinh viên | Lớp |
| Trần Hào Phóng | N20DCDT016 | D20CQDT01-N |
| Võ Minh Thông | N20DCDT028 | D20CQDT01-N |

*TP Hồ Chí Minh , Tháng 05, năm 2024*

**MỤC LỤC CHƯƠNG**

[Chương 1: Cơ sở lý thuyết – phần cứng 5](#_Toc168689328)

[1.1.Cơ sở lý thuyết 5](#_Toc168689329)

[1.1.1.Kiến thức về AI 5](#_Toc168689330)

[1.1.1.1.Khái niệm cơ bản 5](#_Toc168689331)

[1.1.1.2.Các bước xây dựng mô hình AI 7](#_Toc168689332)

[1.1.2.Kiến thức về xử lí ảnh 8](#_Toc168689333)

[1.1.2.1. Khái niệm Độ Chói (Độ Xám) 8](#_Toc168689334)

[1.1.2.2. Chuyển đổi Ảnh Màu thành Ảnh Độ Xám 9](#_Toc168689335)

[1.2.Cơ sở phần cứng 11](#_Toc168689336)

[1.2.1. ESP32CAM 11](#_Toc168689337)

[1.2.1.1.Thông số kỹ thuật và sơ đồ chân 11](#_Toc168689338)

[1.2.1.2.Những khả năng đáng chú ý của ESP32CAM 13](#_Toc168689339)

[1.2.2. Các linh kiện khác 15](#_Toc168689340)

[1.2.2.1.Điện trở 15](#_Toc168689341)

[1.2.2.2.LED 7 đoạn 17](#_Toc168689342)

[Chương 2: Triển khai đồ án 21](#_Toc168689343)

[2.1.Xử lí ảnh chụp và gửi dữ liệu điểm ảnh 21](#_Toc168689344)

[2.1.1. Hàm setup\_cam() 22](#_Toc168689345)

[2.1.2. Hàm handle\_capture() 23](#_Toc168689346)

[2.1.3. Hàm setup() và loop() 25](#_Toc168689347)

[2.2.Thu thập dữ liệu và huấn luyện mô hình AI 25](#_Toc168689348)

[2.2.1.Thu thập dữ liệu: 25](#_Toc168689349)

[2.2.2. Huấn luyện mô hình AI 29](#_Toc168689350)

[2.3.Nhúng mô hình AI và triển khai sử dụng trên ESP32CAM 35](#_Toc168689351)

[2.3.1.Hàm setup\_camera() 37](#_Toc168689352)

[2.3.2.Hàm handle\_captureAI() 37](#_Toc168689353)

[2.3.3.Hàm setup() và loop(): 43](#_Toc168689354)

[Chương 3: Kết luận 44](#_Toc168689355)

[TÀI LIỆU THAM KHẢO 45](#_Toc168689356)

**MỤC LỤC HÌNH ẢNH**

CHƯƠNG 1:

[Hình 1. 1:Artificial Intelligence 5](#_Toc168344608)

[Hình 1. 2. Sơ đồ cây của Machine Learning 6](#_Toc168344609)

[Hình 1. 3: Sơ đồ cây của NLP 7](#_Toc168344610)

[Hình 1. 4: Các bước xây dựng mô hình AI 7](#_Toc168344611)

[Hình 1. 5: Hình ảnh thực tế về thang độ xám 9](#_Toc168344612)

[Hình 1. 6: Chuyển đổi ảnh xám từ ảnh màu 10](#_Toc168344613)

[Hình 1. 7: Hình ảnh thực tế của ESP32CAM 11](#_Toc168344614)

[Hình 1. 8: Sơ đồ chân của ESP32CAM 12](#_Toc168344615)

[Hình 1. 9: Hình ảnh về điện trở 15](#_Toc168344616)

[Hình 1. 10: Dải màu biểu thị thông số điện trở 16](#_Toc168344617)

[Hình 1. 11: Hình ảnh thực tế LED 7 đoạn 17](#_Toc168344618)

[Hình 1. 12: Cấu tạo chân của LED 7 đoạn 18](#_Toc168344619)

CHƯƠNG 2:

[Hình 2. 1: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 1) 21](#_Toc168346546)

[Hình 2. 2: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 2) 21](#_Toc168346547)

[Hình 2. 3: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 3) 22](#_Toc168346548)

[Hình 2. 4: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 4) 23](#_Toc168346549)

[Hình 2. 5: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 5) 24](#_Toc168346550)

[Hình 2. 6: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 6) 25](#_Toc168346551)

[Hình 2. 7: Thu thập dữ liệu (Phần 1) 26](#_Toc168346552)

[Hình 2. 8: Thu thập dữ liệu (Phần 2) 26](#_Toc168346553)

[Hình 2. 9: Thu thập dữ liệu (Phần 3) 27](#_Toc168346554)

[Hình 2. 10: Thu thập dữ liệu (Phần 4) 27](#_Toc168346555)

[Hình 2. 11: Thu thập dữ liệu (Phần 5) 28](#_Toc168346556)

[Hình 2. 12: Thu thập dữ liệu (Phần 6) 28](#_Toc168346557)

[Hình 2. 13: Huấn luyện mô hình AI (Phần 1) 29](#_Toc168346558)

[Hình 2. 14:Huấn luyện mô hình AI (Phần 2) 29](#_Toc168346559)

[Hình 2. 15: Huấn luyện mô hình AI (Phần 3) 30](#_Toc168346560)

[Hình 2. 16: Huấn luyện mô hình AI (Phần 4) 31](#_Toc168346561)

[Hình 2. 17: Huấn luyện mô hình AI (Phần 5) 31](#_Toc168346562)

[Hình 2. 18: Huấn luyện mô hình AI (Phần 6) 32](#_Toc168346563)

[Hình 2. 19: Huấn luyện mô hình AI (Phần 7) 32](#_Toc168346564)

[Hình 2. 20: Huấn luyện mô hình AI (Phần 8) 33](#_Toc168346565)

[Hình 2. 21: Huấn luyện mô hình AI (Phần 9) 33](#_Toc168346566)

[Hình 2. 22: Huấn luyện mô hình AI (Phần 10) 34](#_Toc168346567)

[Hình 2. 23: Huấn luyện mô hình AI (Phần 11) 34](#_Toc168346568)

[Hình 2. 24: Huấn luyện mô hình AI (Phần 12) 35](#_Toc168346569)

[Hình 2. 25: Huấn luyện mô hình AI (Phần 13) 35](#_Toc168346570)

[Hình 2. 26: Nhúng mô hình AI (Phần 1) 36](#_Toc168346571)

[Hình 2. 27: Nhúng mô hình AI (Phần 2) 36](#_Toc168346572)

[Hình 2. 28: Nhúng mô hình AI (Phần 3) 37](#_Toc168346573)

[Hình 2. 29: Nhúng mô hình AI (Phần 4) 38](#_Toc168346574)

[Hình 2. 30: Nhúng mô hình AI (Phần 5) 38](#_Toc168346575)

[Hình 2. 31: Nhúng mô hình AI (Phần 6) 39](#_Toc168346576)

[Hình 2. 32: Nhúng mô hình AI (Phần 7) 40](#_Toc168346577)

[Hình 2. 33: Nhúng mô hình AI (Phần 8) 40](#_Toc168346578)

[Hình 2. 34: Nhúng mô hình AI (Phần 9) 41](#_Toc168346579)

[Hình 2. 35: Nhúng mô hình AI (Phần 10) 42](#_Toc168346580)

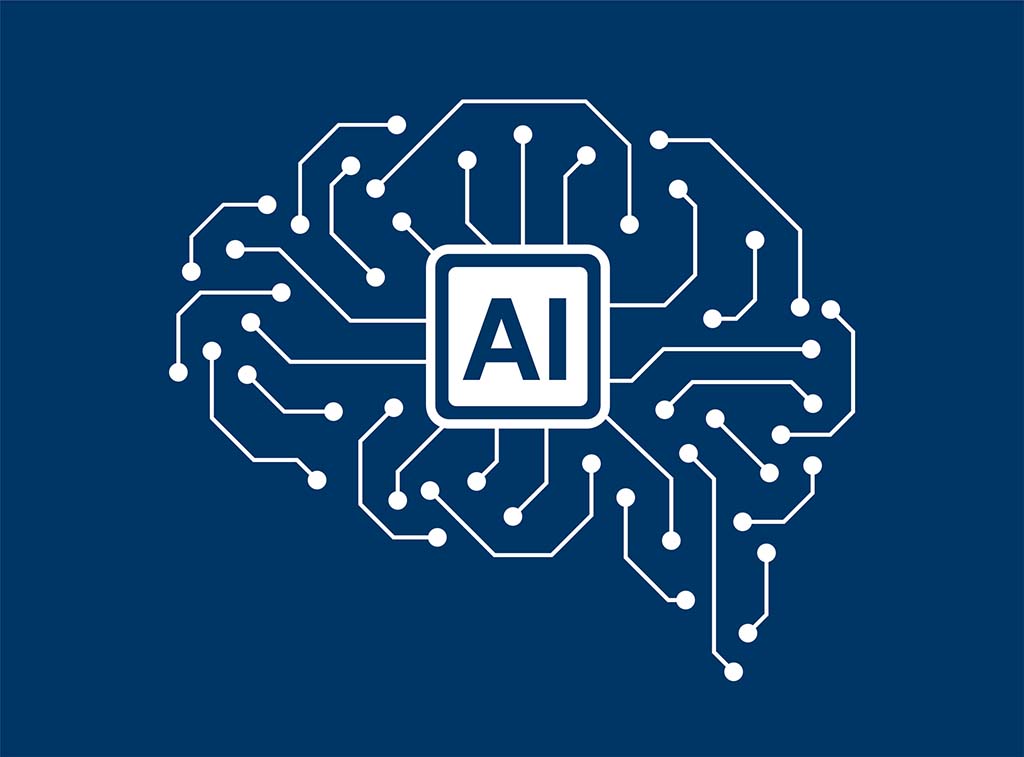
[Hình 2. 36: Nhúng mô hình AI (Phần 11) 43](#_Toc168346581)

# Chương 1: Cơ sở lý thuyết – phần cứng

## 1.1.Cơ sở lý thuyết

### 1.1.1.Kiến thức về AI

#### 1.1.1.1.Khái niệm cơ bản

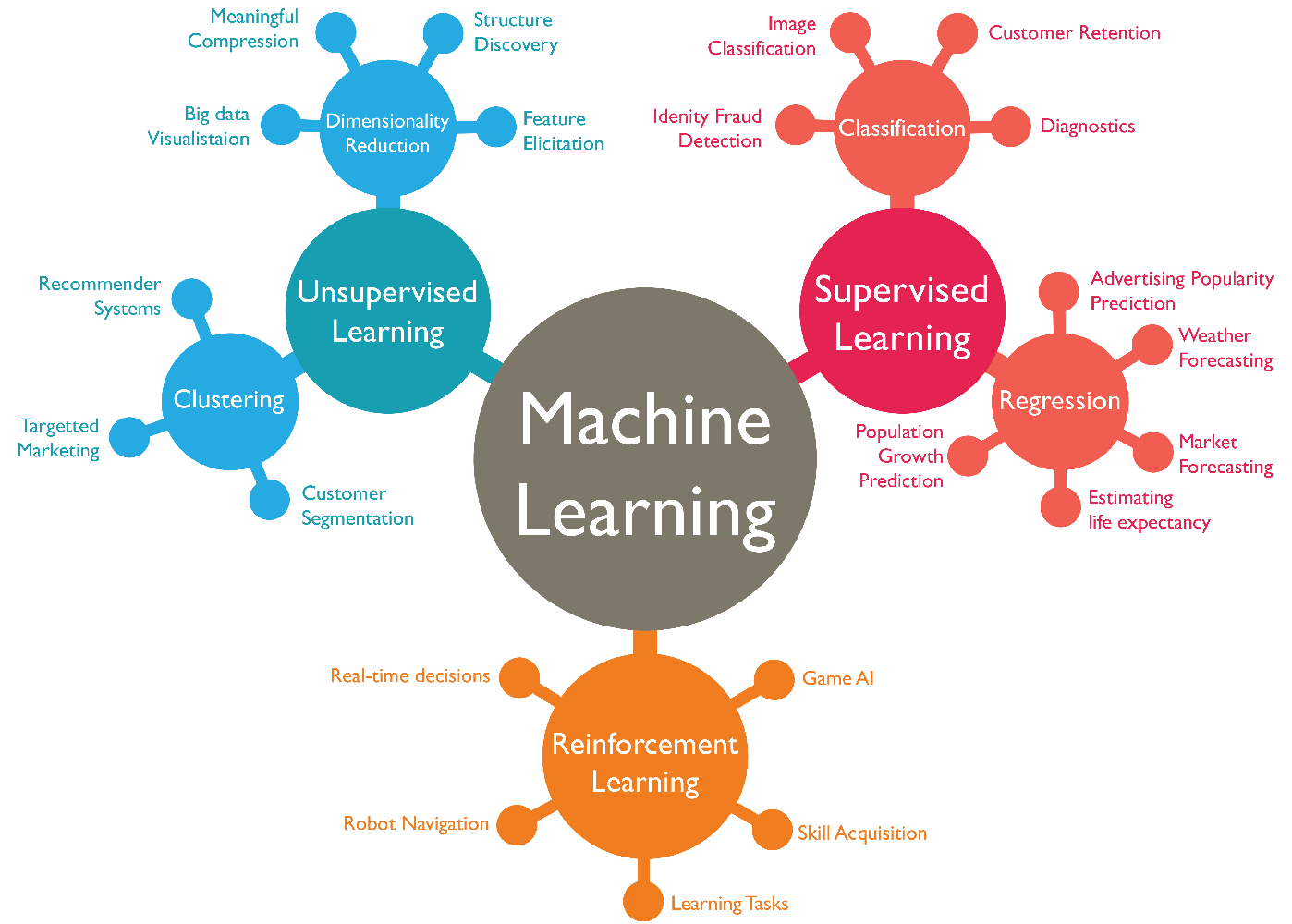


Hình 1. 1:Artificial Intelligence

AI (Artificial Intelligence - Trí tuệ nhân tạo) là một lĩnh vực trong khoa học máy tính, nghiên cứu và phát triển các hệ thống máy tính có khả năng thực hiện các tác vụ thông minh mà bình thường cần có trí tuệ con người để thực hiện. Các hệ thống AI có thể bao gồm từ các chương trình đơn giản như nhận dạng mẫu, xử lý ngôn ngữ tự nhiên, đến các hệ thống phức tạp hơn như học sâu (deep learning) và học tăng cường (reinforcement learning).

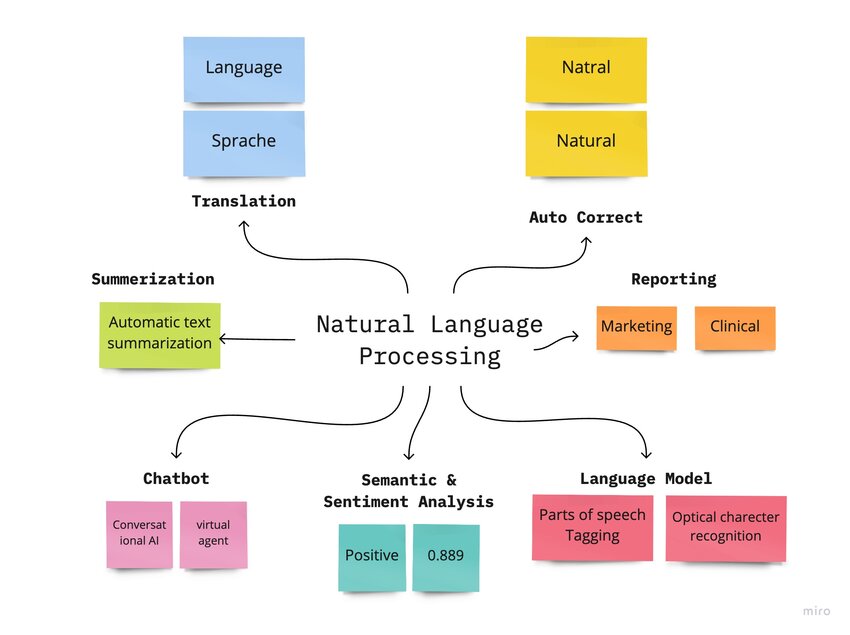
Có một số khái niệm cơ bản về AI bao gồm:

* Machine Learning (Học máy): Một nhánh của AI, tập trung vào việc phát triển các thuật toán cho phép máy tính học từ dữ liệu và cải thiện hiệu suất theo thời gian mà không cần lập trình cụ thể.



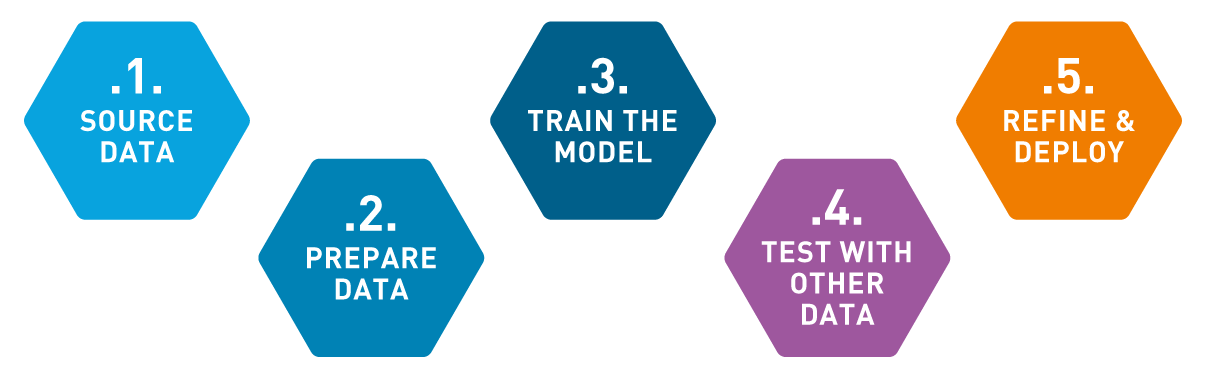
Hình 1. 2. Sơ đồ cây của Machine Learning

* Deep Learning (Học sâu): Một tập con của học máy, sử dụng các mạng nơ-ron nhân tạo với nhiều lớp để mô phỏng cách hoạt động của bộ não con người trong việc xử lý dữ liệu và ra quyết định.
* Natural Language Processing (Xử lý ngôn ngữ tự nhiên): Một lĩnh vực nghiên cứu và ứng dụng của AI, liên quan đến tương tác giữa máy tính và ngôn ngữ tự nhiên của con người.



Hình 1. 3: Sơ đồ cây của NLP

#### 1.1.1.2.Các bước xây dựng mô hình AI



Hình 1. 4: Các bước xây dựng mô hình AI

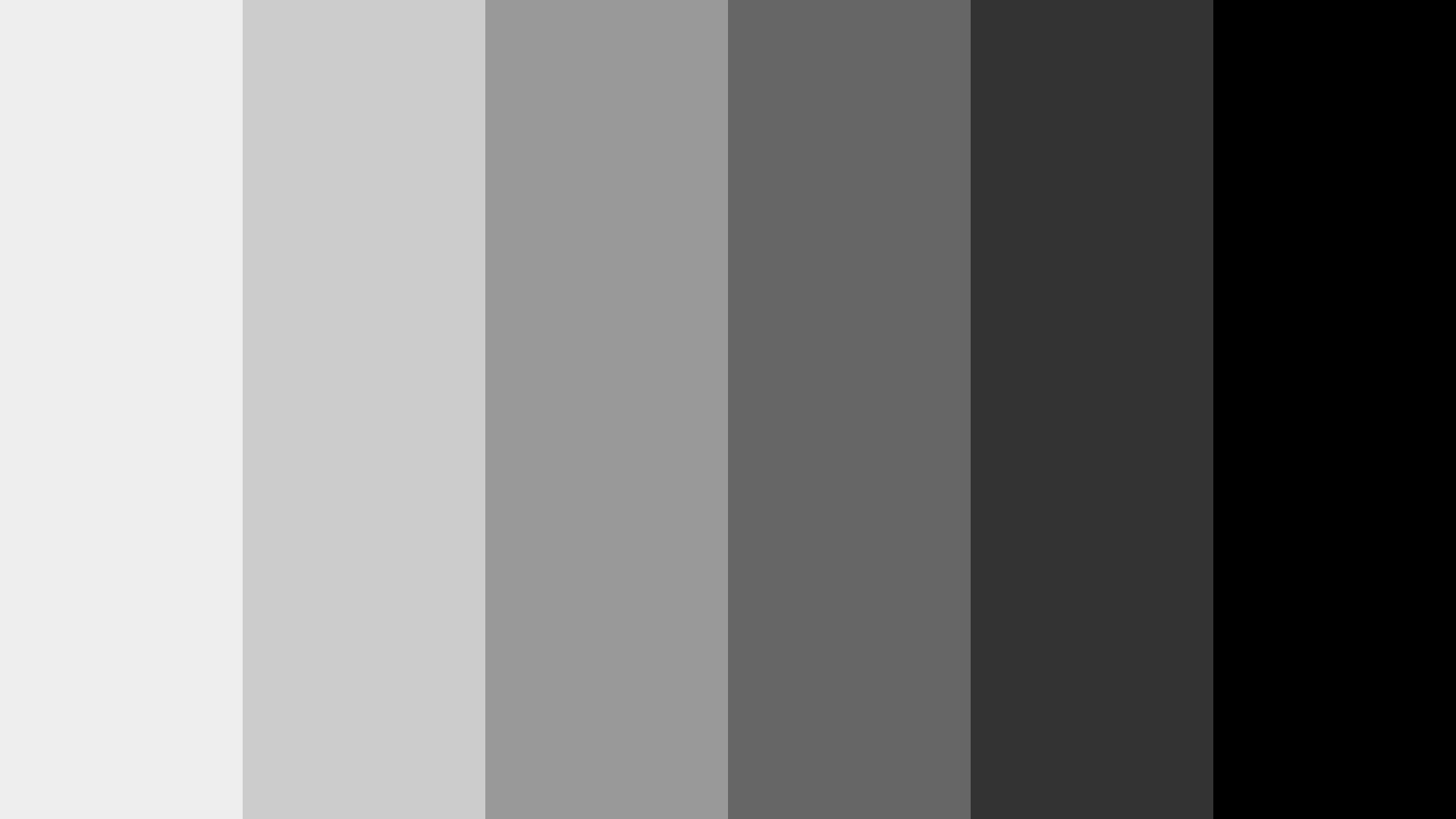
Xây dựng một mô hình AI cơ bản thường bao gồm các bước sau:

* Xác định vấn đề và mục tiêu: Đầu tiên, cần xác định rõ vấn đề cần giải quyết và mục tiêu của mô hình AI. Điều này bao gồm việc hiểu rõ yêu cầu, phạm vi và các tiêu chí đánh giá hiệu quả của mô hình.
* Thu thập và chuẩn bị dữ liệu: Dữ liệu là yếu tố quan trọng nhất trong việc xây dựng mô hình AI. Cần thu thập dữ liệu phù hợp, làm sạch dữ liệu (xử lý các giá trị thiếu, loại bỏ dữ liệu nhiễu,...) và chuẩn hóa dữ liệu (chuyển đổi dữ liệu về cùng định dạng, phân loại,...).
* Lựa chọn và triển khai thuật toán: Dựa trên vấn đề và dữ liệu, lựa chọn thuật toán học máy phù hợp. Các thuật toán phổ biến bao gồm hồi quy tuyến tính, cây quyết định, mạng nơ-ron nhân tạo, và các thuật toán học sâu khác.
* Huấn luyện mô hình: Sử dụng dữ liệu huấn luyện để "dạy" cho mô hình học các mẫu và mối quan hệ trong dữ liệu. Quá trình này bao gồm việc điều chỉnh các tham số của mô hình để tối ưu hóa hiệu suất.
* Đánh giá mô hình: Sau khi huấn luyện, cần đánh giá mô hình bằng cách sử dụng dữ liệu kiểm tra. Điều này giúp kiểm tra xem mô hình hoạt động tốt đến mức nào và có thể phát hiện ra các vấn đề như overfitting hay underfitting.
* Triển khai và giám sát mô hình: Khi mô hình đã được đánh giá và đạt yêu cầu, mô hình sẽ được triển khai vào môi trường thực tế. Sau đó, cần giám sát hiệu suất của mô hình liên tục để đảm bảo nó hoạt động ổn định và cải tiến nếu cần thiết.

### 1.1.2.Kiến thức về xử lí ảnh

#### 1.1.2.1. Khái niệm Độ Chói (Độ Xám)

Độ chói (luminance) hay độ xám (grayscale) của ảnh là một thước đo thể hiện mức độ sáng tối của các điểm ảnh. Một ảnh độ xám chỉ chứa các sắc độ của màu xám, không có màu sắc nào khác. Mỗi điểm ảnh (pixel) trong ảnh độ xám được biểu diễn bằng một giá trị duy nhất, thường nằm trong khoảng từ 0 đến 255.



Hình 1. 5: Hình ảnh thực tế về thang độ xám

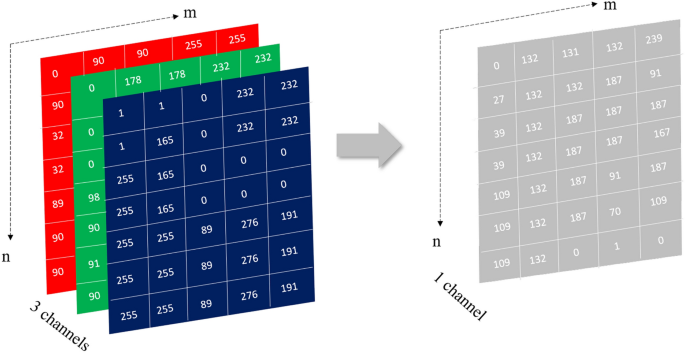
0: màu đen hoàn toàn.

255: màu trắng hoàn toàn.

Các giá trị từ 1 đến 254: các mức xám khác nhau từ đen đến trắng.

#### 1.1.2.2. Chuyển đổi Ảnh Màu thành Ảnh Độ Xám

Ảnh màu thường được biểu diễn bằng hệ màu RGB, trong đó mỗi điểm ảnh có ba giá trị riêng biệt: đỏ (Red), xanh lá (Green), và xanh dương (Blue). Để chuyển đổi ảnh màu thành ảnh độ xám, người ta sử dụng một công thức tính toán dựa trên các giá trị R, G, B của từng điểm ảnh.



Hình 1. 6: Chuyển đổi ảnh xám từ ảnh màu

Một công thức phổ biến để chuyển đổi là:

Gray=0.299×R+0.587×G+0.114×B

Các hệ số 0.299, 0.587 và 0.114 phản ánh mức độ nhạy cảm của mắt người đối với các màu sắc khác nhau, với màu xanh lá có độ nhạy cao nhất và màu xanh dương có độ nhạy thấp nhất.

1.1.2.3. Lợi ích và ứng dụng sử dụng Ảnh Độ Xám

Lợi ích:

* Giảm Tài Nguyên: Ảnh độ xám chỉ cần một giá trị cho mỗi điểm ảnh thay vì ba giá trị như ảnh màu, giúp giảm dung lượng bộ nhớ và tài nguyên tính toán.
* Đơn Giản Hóa Xử Lý: Việc xử lý các thao tác trên ảnh độ xám thường nhanh hơn và đơn giản hơn so với ảnh màu.
* Tập Trung Vào Độ Sáng: Ảnh độ xám giúp tập trung vào thông tin độ sáng, bỏ qua các yếu tố màu sắc có thể không cần thiết trong một số ứng dụng.

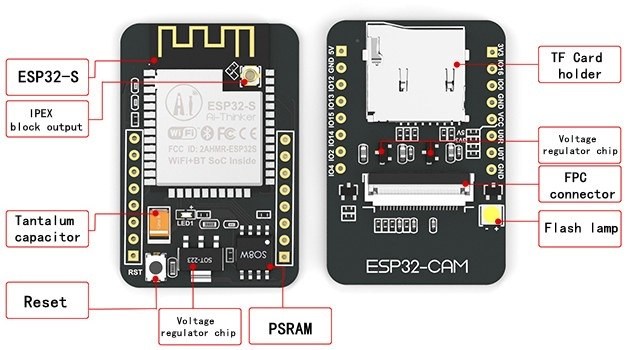
Ứng dụng:

* Xử lý ảnh: Nhiều thuật toán xử lý ảnh, như làm mịn (smoothing), làm sắc nét (sharpening), và phát hiện biên cạnh (edge detection), thường được áp dụng trên ảnh độ xám để giảm độ phức tạp và cải thiện hiệu suất.
* Nhận diện mẫu: Trong các hệ thống nhận diện khuôn mặt, biển số xe, và các ứng dụng khác, ảnh độ xám thường được sử dụng để đơn giản hóa việc phân tích và trích xuất đặc trưng.
* Thị giác máy tính: Các ứng dụng thị giác máy tính như robot và xe tự hành thường sử dụng ảnh độ xám để tiết kiệm tài nguyên tính toán trong khi vẫn duy trì được độ chính xác cao.

## 1.2.Cơ sở phần cứng

### 1.2.1. ESP32CAM

#### 1.2.1.1.Thông số kỹ thuật và sơ đồ chân



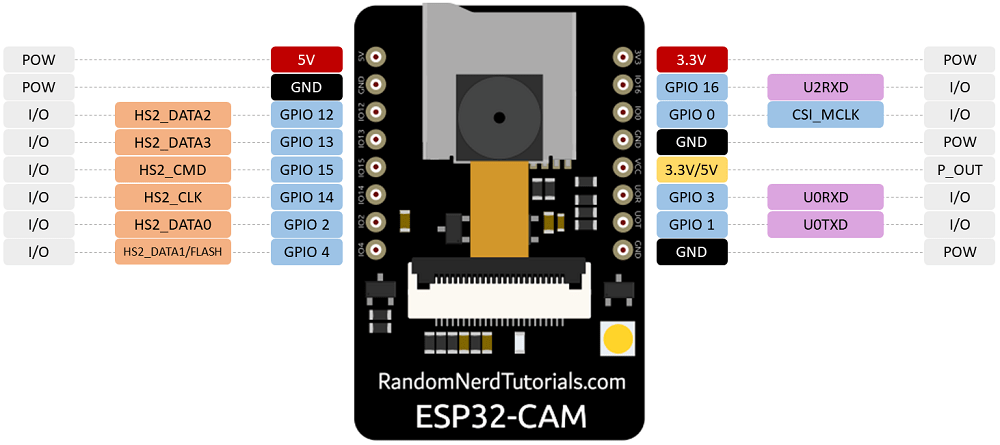
Hình 1. 7: Hình ảnh thực tế của ESP32CAM

ESP32CAM là một module camera nhỏ gọn và mạnh mẽ, dựa trên vi xử lý ESP32 của Espressif Systems. Nó được thiết kế đặc biệt cho các ứng dụng IoT (Internet of Things) và các dự án DIY (Do It Yourself), cung cấp các tính năng vượt trội với giá cả phải chăng.

-Thông số kỹ thuật

* Vi xử lý: ESP32, dual-core với xung nhịp lên đến 240 MHz.
* RAM: 520 KB SRAM.
* Flash: 4 MB.
* Camera: OV2640 với độ phân giải 2MP, hỗ trợ định dạng JPEG.
* Wi-Fi: 802.11 b/g/n, hỗ trợ cả 2.4 GHz.
* Bluetooth: BLE và v4.2 BR/EDR.
* GPIOs: 9 chân GPIO có thể được sử dụng cho các mục đích khác nhau như PWM, ADC, DAC, và các giao thức giao tiếp như I2C, SPI, UART.
* Khe cắm thẻ nhớ: Hỗ trợ thẻ nhớ microSD lên đến 4 GB.
* Giao tiếp: UART, SPI, I2C, PWM.
* Nguồn điện: 5V qua cổng USB hoặc 3.3V qua chân VIN.
* Kích thước: 27 x 40.5 x 4.5 mm.

Sơ đồ chân



Hình 1. 8: Sơ đồ chân của ESP32CAM

ESP32CAM có một số chân quan trọng cần biết để kết nối và sử dụng module hiệu quả:

* 3V3 (VCC): Cung cấp điện áp 3.3V cho module.
* GND: Chân nối đất.
* GPIO0: Chân này thường được sử dụng để flash firmware. Khi kết nối với GND và reset, ESP32CAM sẽ vào chế độ bootloader.
* GPIO1 (U0TXD): Chân truyền dữ liệu UART.
* GPIO3 (U0RXD): Chân nhận dữ liệu UART.
* GPIO2: Chân này thường kết nối với đèn flash của camera.
* GPIO12, GPIO13, GPIO14, GPIO15: Các chân GPIO đa dụng, có thể sử dụng cho các cảm biến, relay, hoặc các thiết bị khác.
* RST (EN): Chân reset, dùng để khởi động lại module.
* SD2 (D2), SD3 (D3), CMD, CLK: Các chân này thường được sử dụng cho giao tiếp với thẻ nhớ microSD

#### 1.2.1.2.Những khả năng đáng chú ý của ESP32CAM

ESP32CAM không chỉ nổi bật với thiết kế nhỏ gọn và chi phí thấp, mà còn được trang bị nhiều khả năng mạnh mẽ, làm cho nó trở thành một lựa chọn lý tưởng cho các dự án IoT và DIY. Dưới đây là một số khả năng đáng chú ý của ESP32CAM:

* Chụp ảnh và quay video

Với camera OV2640 tích hợp, ESP32CAM có thể chụp ảnh và quay video với độ phân giải lên đến 1600x1200 pixel. Camera hỗ trợ định dạng JPEG, giúp giảm thiểu dung lượng lưu trữ cần thiết mà vẫn giữ được chất lượng hình ảnh tốt. Khả năng chụp ảnh và quay video này có thể được tận dụng trong các ứng dụng giám sát, hệ thống an ninh, và các dự án liên quan đến thị giác máy tính.

* Truyền phát video trực tiếp

Khả năng kết nối Wi-Fi của ESP32CAM cho phép truyền phát video trực tiếp qua mạng. Điều này rất hữu ích trong các ứng dụng giám sát an ninh, nơi mà hình ảnh hoặc video cần được truyền đến một thiết bị khác như điện thoại thông minh, máy tính, hoặc máy chủ để theo dõi và phân tích. Với tính năng này, ESP32CAM có thể được sử dụng trong các dự án robot, máy bay không người lái, và các thiết bị IoT khác yêu cầu truyền tải dữ liệu video theo thời gian thực.

* Nhận diện và xử lý hình ảnh

Nhờ sức mạnh xử lý của vi xử lý ESP32, ESP32CAM có khả năng thực hiện các tác vụ nhận diện và xử lý hình ảnh ngay trên module. Điều này bao gồm các chức năng như nhận diện khuôn mặt, phát hiện chuyển động, và các thuật toán xử lý hình ảnh khác. Khả năng này mở ra nhiều ứng dụng sáng tạo trong lĩnh vực an ninh, tự động hóa nhà cửa, và các hệ thống tương tác thông minh.

* Lưu trữ dữ liệu

ESP32CAM hỗ trợ thẻ nhớ microSD lên đến 4 GB, giúp lưu trữ hình ảnh và video dễ dàng. Điều này rất hữu ích trong các ứng dụng yêu cầu lưu trữ dữ liệu tại chỗ, chẳng hạn như hệ thống giám sát an ninh độc lập hoặc các dự án ghi nhận dữ liệu trong môi trường không có kết nối mạng liên tục. Khả năng lưu trữ nội bộ này cũng giúp tăng tính linh hoạt trong việc quản lý dữ liệu.

* Khả năng mở rộng

Với các chân GPIO đa dụng, ESP32CAM có thể dễ dàng tích hợp với các cảm biến và module khác để mở rộng chức năng. Người dùng có thể kết nối các cảm biến nhiệt độ, độ ẩm, ánh sáng, chuyển động, hoặc các thiết bị điều khiển như relay, động cơ servo, và nhiều thiết bị khác. Khả năng này giúp ESP32CAM trở thành một giải pháp mạnh mẽ và linh hoạt cho các dự án IoT phức tạp.

### 1.2.2. Các linh kiện khác

#### 1.2.2.1.Điện trở



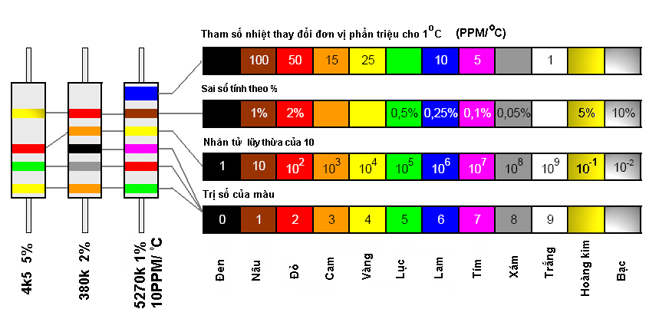
Hình 1. 9: Hình ảnh về điện trở

Điện trở là một trong những linh kiện điện tử cơ bản nhất và quan trọng trong mọi mạch điện tử. Điện trở có chức năng hạn chế dòng điện chạy qua mạch, phân chia điện áp, bảo vệ các linh kiện khác, và tạo ra các tín hiệu điều khiển cần thiết trong các thiết kế mạch. Chúng có nhiều loại khác nhau và được sử dụng trong vô số ứng dụng từ những mạch đơn giản đến các hệ thống phức tạp.

- Đơn vị đo: Điện trở được đo bằng đơn vị Ohm, ký hiệu là Ω. Một Ohm là giá trị điện trở tạo ra khi một dòng điện một Ampere chảy qua và tạo ra một hiệu điện thế một Volt. Đối với các giá trị lớn hơn, người ta sử dụng các đơn vị kiloOhm (kΩ) và megaOhm (MΩ).

- Mã màu: Điện trở thường có mã màu để biểu thị giá trị và dung sai (độ chính xác). Hệ thống mã màu này gồm các dải màu được in trên thân điện trở. Mỗi màu đại diện cho một con số cụ thể, và các dải màu này kết hợp lại để cho biết giá trị điện trở và độ dung sai. Cách đọc mã màu:

* Dải thứ nhất và thứ hai: Đại diện cho hai chữ số đầu tiên của giá trị điện trở.
* Dải thứ ba: Là bội số, nghĩa là số lượng số 0 cần thêm vào sau hai chữ số đầu tiên.
* Dải thứ tư (nếu có): Đại diện cho độ dung sai (ví dụ: vàng là ±5%, bạc là ±10%).



Hình 1. 10: Dải màu biểu thị thông số điện trở

Ví dụ, một điện trở với các dải màu: đỏ (2), tím (7), cam (000), vàng (±5%) có giá trị là 27kΩ ±5%.

- Các loại điện trở

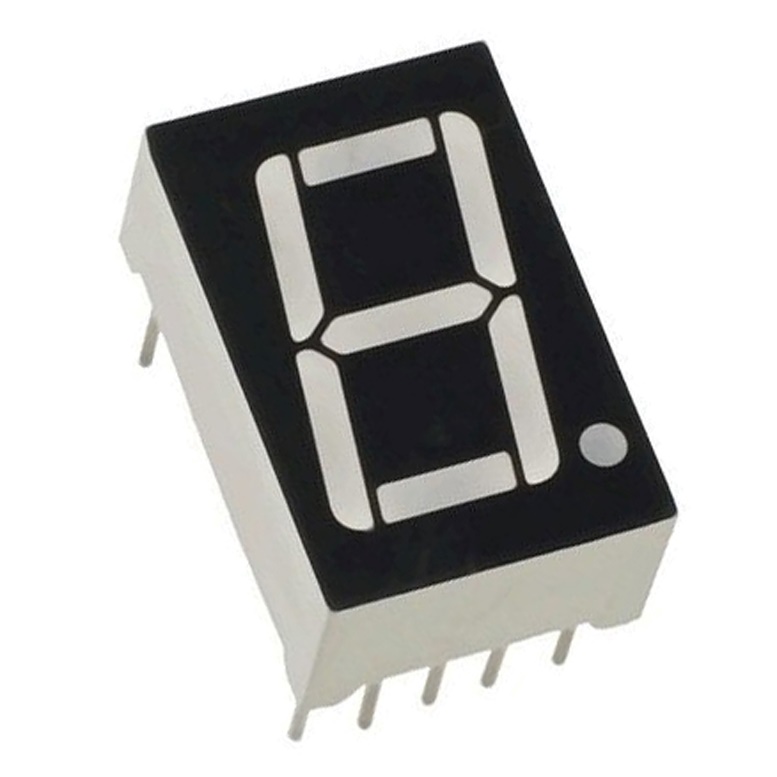
Điện trở cố định: Có giá trị cố định và không thể thay đổi được. Đây là loại điện trở phổ biến nhất.

* Điện trở biến đổi (biến trở): Có thể điều chỉnh được giá trị. Chúng thường được sử dụng trong các ứng dụng cần thay đổi liên tục như điều chỉnh âm lượng.
* Điện trở nhiệt (NTC/PTC): Thay đổi giá trị điện trở theo nhiệt độ. NTC (Negative Temperature Coefficient) giảm điện trở khi nhiệt độ tăng, trong khi PTC (Positive Temperature Coefficient) tăng điện trở khi nhiệt độ tăng.
* Điện trở quang (LDR): Thay đổi giá trị điện trở theo cường độ ánh sáng. Giá trị điện trở giảm khi cường độ ánh sáng tăng.

- Ứng dụng

* Hạn dòng cho đèn LED: Điện trở được sử dụng để hạn chế dòng điện đi qua đèn LED, bảo vệ đèn LED khỏi bị quá tải và cháy nổ.
* Tạo phân áp: Bằng cách kết nối các điện trở theo cấu hình phân áp, có thể chia một điện áp lớn thành các điện áp nhỏ hơn. Đây là một cách thông dụng để cung cấp điện áp thích hợp cho các linh kiện khác nhau trong mạch.
* Lọc tín hiệu: Điện trở kết hợp với tụ điện để tạo ra các mạch lọc (lọc cao tần, lọc thấp tần) nhằm loại bỏ nhiễu hoặc các thành phần tần số không mong muốn khỏi tín hiệu.
* Điều chỉnh độ lợi trong các mạch khuếch đại: Điện trở được sử dụng để thiết lập và điều chỉnh độ lợi của các mạch khuếch đại tín hiệu, chẳng hạn như trong các bộ khuếch đại âm thanh.
* Bảo vệ mạch: Điện trở được dùng để hạn chế dòng điện đến mức an toàn, bảo vệ các linh kiện nhạy cảm như IC, transistor khỏi bị hỏng do quá tải.

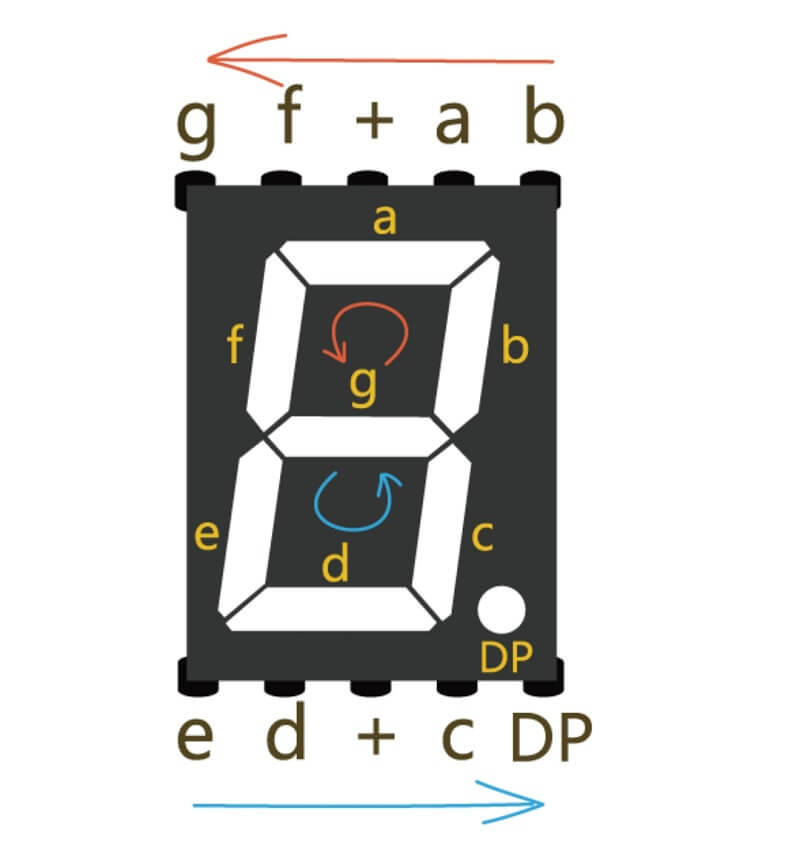
#### 1.2.2.2.LED 7 đoạn



Hình 1. 11: Hình ảnh thực tế LED 7 đoạn

LED 7 đoạn là một linh kiện hiển thị phổ biến, thường được sử dụng để hiển thị các chữ số từ 0 đến 9 trong các thiết bị điện tử. Được sắp xếp thành hình số 8, các LED này có thể được bật hoặc tắt độc lập để tạo ra các con số mong muốn. LED 7 đoạn được sử dụng rộng rãi trong đồng hồ điện tử, bộ đếm, bảng hiển thị, và nhiều thiết bị đo lường khác.

- Cấu tạo: Một LED 7 đoạn bao gồm 7 thanh LED nhỏ (thường gọi là các đoạn) được sắp xếp thành hình chữ nhật, với một đoạn thứ 8 dùng để hiển thị dấu chấm thập phân (nếu có). Các đoạn này thường được ký hiệu từ a đến g, với đoạn dấu chấm thập phân được ký hiệu là dp.



Hình 1. 12: Cấu tạo chân của LED 7 đoạn

* Đoạn a: Đỉnh trên cùng.
* Đoạn b: Bên phải phía trên.
* Đoạn c: Bên phải phía dưới.
* Đoạn d: Đáy dưới cùng.
* Đoạn e: Bên trái phía dưới.
* Đoạn f: Bên trái phía trên.
* Đoạn g: Thanh ngang ở giữa.
* Đoạn dp: Dấu chấm thập phân (nếu có).

-Màu sắc: Các LED 7 đoạn có thể có nhiều màu khác nhau như đỏ, xanh lá cây, xanh dương, vàng, hoặc trắng. Màu đỏ là phổ biến nhất do độ sáng và tính dễ nhìn trong nhiều điều kiện ánh sáng.

-Điều khiển: Mỗi đoạn của LED 7 đoạn có thể được điều khiển độc lập. Điều này thường được thực hiện thông qua các vi điều khiển (microcontroller) hoặc các mạch logic đơn giản. Các vi điều khiển có thể gửi tín hiệu điện áp đến từng đoạn LED để bật hoặc tắt chúng, tạo ra các con số mong muốn.

-Cấu hình kết nối:

* Cathode chung (Common Cathode): Trong cấu hình này, các cực âm của tất cả các LED được nối chung và được nối với GND. Các cực dương của các LED được nối với các chân điều khiển của vi điều khiển. Để bật một đoạn LED, chân điều khiển tương ứng cần được đặt mức cao (HIGH).
* Anode chung (Common Anode): Trong cấu hình này, các cực dương của tất cả các LED được nối chung và được nối với nguồn điện dương (Vcc). Các cực âm của các LED được nối với các chân điều khiển của vi điều khiển. Để bật một đoạn LED, chân điều khiển tương ứng cần được đặt mức thấp (LOW).
* Mạch quét (Multiplexing): Để tiết kiệm chân điều khiển của vi điều khiển, các LED 7 đoạn thường được điều khiển bằng phương pháp quét. Phương pháp này cho phép nhiều LED 7 đoạn được điều khiển bởi cùng một bộ vi điều khiển bằng cách bật từng LED trong một khoảng thời gian ngắn theo thứ tự tuần hoàn, tạo ra hiệu ứng hiển thị liên tục.

-Ứng dụng

LED 7 đoạn có nhiều ứng dụng thực tiễn trong các thiết bị điện tử, bao gồm:

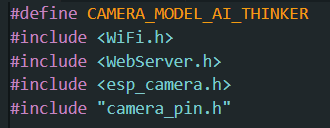
Hiển thị số trong các thiết bị điện tử: LED 7 đoạn thường được sử dụng để hiển thị số trong các thiết bị như đồng hồ kỹ thuật số, bộ đếm, và các thiết bị đo lường.

* Đồng hồ kỹ thuật số: Sử dụng LED 7 đoạn để hiển thị giờ, phút, và giây.
* Bộ đếm: Trong các ứng dụng công nghiệp và thương mại, LED 7 đoạn được sử dụng để hiển thị số liệu đếm như sản phẩm sản xuất, số lượt truy cập, hoặc các chỉ số thống kê khác.
* Bảng hiển thị: Trong các biển quảng cáo, máy bán hàng tự động, và các thiết bị tương tự, LED 7 đoạn được sử dụng để hiển thị giá cả, mã sản phẩm, hoặc các thông tin quan trọng khác.
* Thiết bị đo lường: Trong các thiết bị như vôn kế, ampe kế, và nhiệt kế điện tử, LED 7 đoạn được sử dụng để hiển thị các giá trị đo đạc.

# Chương 2: Triển khai đồ án

## 2.1.Xử lí ảnh chụp và gửi dữ liệu điểm ảnh

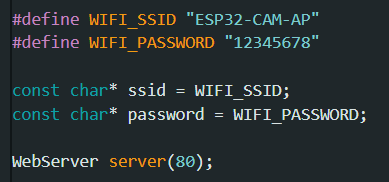
Đầu tiên khai báo thư viện sử dụng:



Hình 2. 1: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 1)

* CAMERA\_MODEL\_AI\_THINKER: Định nghĩa mô hình camera, thường được sử dụng để chọn cấu hình chân cắm (pin) phù hợp.
* WiFi.h: Thư viện để sử dụng các chức năng WiFi của ESP32.
* WebServer.h: Thư viện để chạy một máy chủ web trên ESP32.
* esp\_camera.h: Thư viện hỗ trợ chức năng camera trên ESP32.
* camera\_pin.h: Tệp tiêu đề chứa cấu hình pin của camera

Tiếp theo là cấu hình những thông số cần thiết cho chương trình:



Hình 2. 2: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 2)

* WIFI\_SSID: Định nghĩa tên mạng WiFi mà ESP32 sẽ phát.
* WIFI\_PASSWORD: Định nghĩa mật khẩu cho mạng WiFi đó.
* ssid và password: Chứa tên mạng và mật khẩu WiFi.
* server: Đối tượng máy chủ web chạy trên cổng 80.

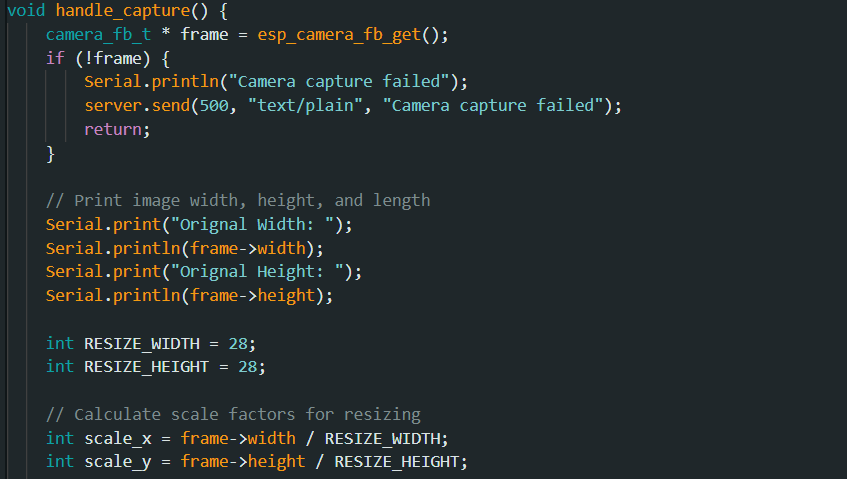
### 2.1.1. Hàm setup\_cam()



Hình 2. 3: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 3)

* camera\_config\_t config: Cấu hình camera, bao gồm các thông số về kênh, bộ đếm thời gian, các chân GPIO được sử dụng và các thông số khác.
* pixel\_format: Định dạng ảnh, ở đây là PIXFORMAT\_GRAYSCALE (ảnh đen trắng).
* frame\_size: Kích thước khung hình.
* jpeg\_quality: Chất lượng ảnh JPEG (1 là tốt nhất, 63 là kém nhất).
* fb\_count: Số khung hình trong bộ đệm.

### 2.1.2. Hàm handle\_capture()



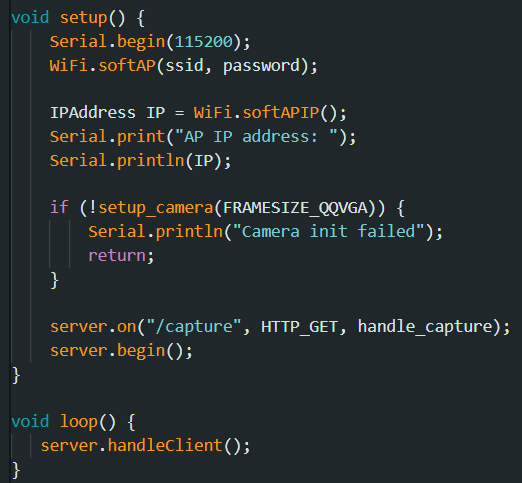
Hình 2. 4: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 4)



Hình 2. 5: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 5)

* esp\_camera\_fb\_get(): Lấy khung hình từ camera.
* frame: Con trỏ tới khung hình lấy được từ camera.
* RESIZE\_WIDTH và RESIZE\_HEIGHT: Kích thước mục tiêu để thu nhỏ ảnh (28x28).
* scale\_x và scale\_y: Tỷ lệ thu nhỏ tính toán từ kích thước gốc của ảnh.
* json: Chuỗi JSON chứa dữ liệu ảnh thu nhỏ.
* esp\_camera\_fb\_return(frame): Giải phóng bộ nhớ của khung hình sau khi xử lý.

### 2.1.3. Hàm setup() và loop()



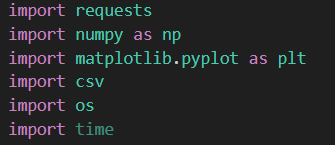
Hình 2. 6: Xử lí ảnh chụp và gửi dữ liệu điểm ảnh (Phần 6)

* WiFi.softAP(ssid, password): Thiết lập ESP32 làm điểm truy cập WiFi.
* WiFi.softAPIP(): Lấy địa chỉ IP của điểm truy cập.
* setup\_camera(FRAMESIZE\_QQVGA): Khởi tạo camera với kích thước khung hình QQVGA.
* server.on("/capture", HTTP\_GET, handle\_capture): Định nghĩa URL /capture để xử lý yêu cầu GET bằng hàm handle\_capture.
* server.begin(): Bắt đầu máy chủ web.
* server.handleClient(): Xử lý các yêu cầu từ máy khách.

## 2.2.Thu thập dữ liệu và huấn luyện mô hình AI

### 2.2.1.Thu thập dữ liệu:

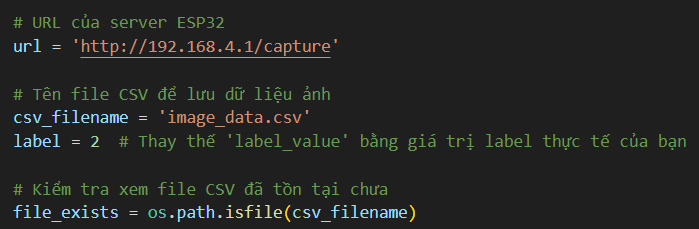
Đầu tiên khai báo thư viện sử dụng:



Hình 2. 7: Thu thập dữ liệu (Phần 1)

* requests: Được sử dụng để gửi yêu cầu HTTP đến server ESP32.
* numpy: Được sử dụng để xử lý dữ liệu mảng.
* matplotlib.pyplot: Được sử dụng để hiển thị ảnh.
* csv: Được sử dụng để ghi dữ liệu vào tệp CSV.
* os: Được sử dụng để kiểm tra sự tồn tại của tệp CSV.
* time: Được sử dụng để tạo độ trễ trước khi chụp ảnh tiếp theo.

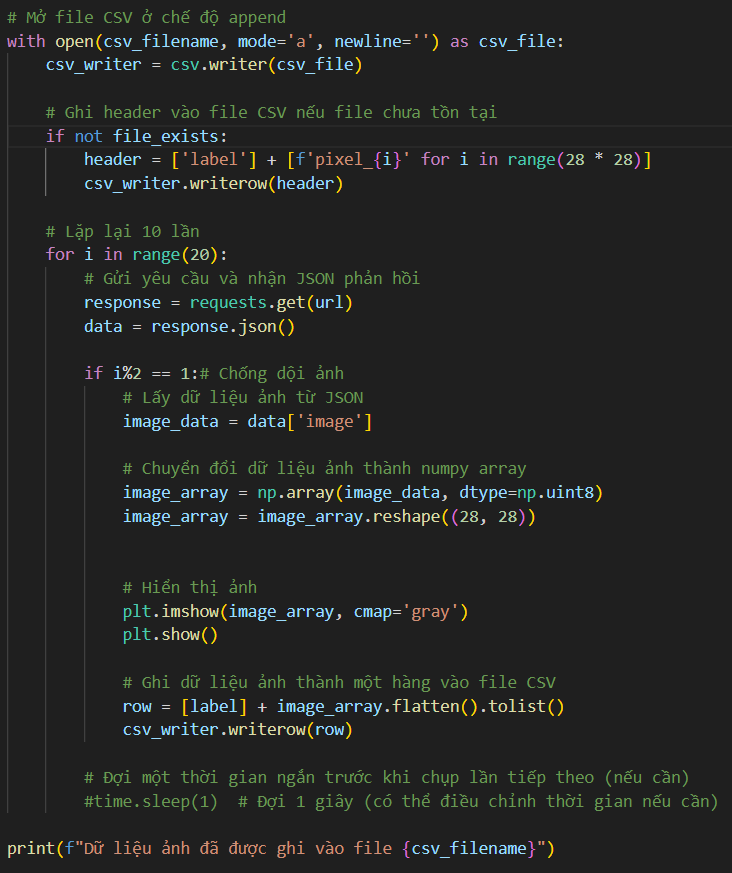
Tiếp theo là cấu hình thông số và thiết lập cần thiết cho chương trình



Hình 2. 8: Thu thập dữ liệu (Phần 2)

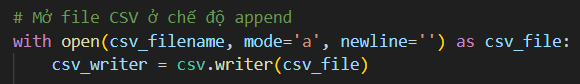
* url: URL của server ESP32 để gửi yêu cầu chụp ảnh.
* csv\_filename: Tên tệp CSV để lưu dữ liệu ảnh.
* label: Nhãn (label) của dữ liệu ảnh, có thể thay đổi tùy theo yêu cầu
* os.path.isfile(csv\_filename): Kiểm tra xem tệp CSV đã tồn tại hay không.

Vòng lặp chính:



Hình 2. 9: Thu thập dữ liệu (Phần 3)

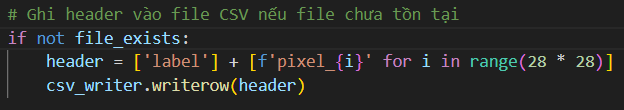
* Tạo đối tượng ghi file csv:



Hình 2. 10: Thu thập dữ liệu (Phần 4)

open(csv\_filename, mode='a', newline=''): Mở tệp CSV trong chế độ append để ghi dữ liệu mới vào cuối tệp.

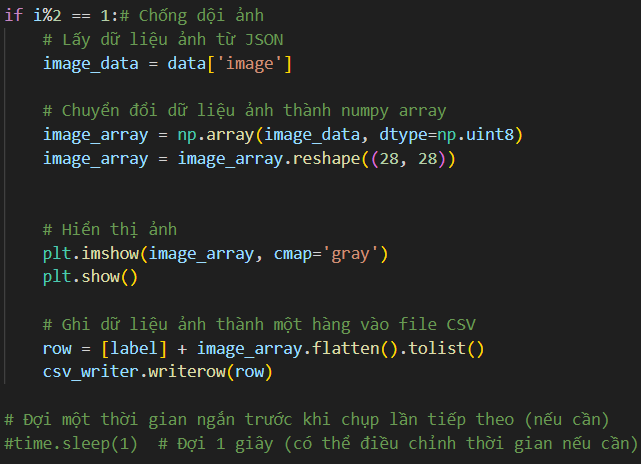
* (Tùy chọn) Tạo header cho file csv:



Hình 2. 11: Thu thập dữ liệu (Phần 5)

Nếu tệp CSV không tồn tại, một header mới được tạo và ghi vào tệp CSV. Header này bao gồm nhãn (label) và các pixel của ảnh.

* Lặp lại quá trình gửi yêu cầu và xử lý dữ liệu ảnh:

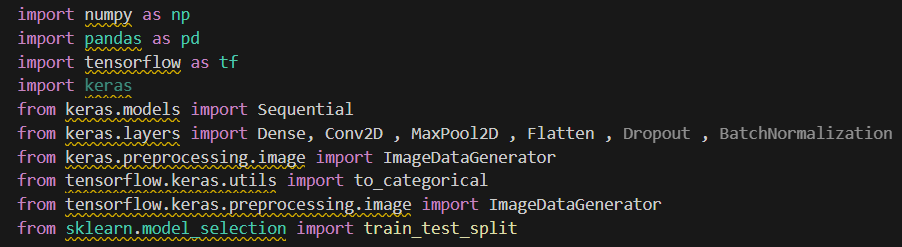


Hình 2. 12: Thu thập dữ liệu (Phần 6)

* Trong vòng lặp, yêu cầu được gửi đến server ESP32 để chụp ảnh và dữ liệu ảnh được nhận dưới dạng JSON.
* Sau đó, ảnh được hiển thị bằng thư viện matplotlib.
* Dữ liệu ảnh được chuyển đổi thành một hàng của tệp CSV và được ghi vào tệp.

### 2.2.2. Huấn luyện mô hình AI

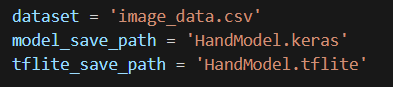
Đầu tiên khai báo thư viện cần thiết:



Hình 2. 13: Huấn luyện mô hình AI (Phần 1)

* numpy: Thư viện hỗ trợ các phép tính toán trên mảng.
* pandas: Thư viện để xử lý và phân tích dữ liệu.
* tensorflow và keras: Thư viện để xây dựng và huấn luyện mô hình học sâu.
* sklearn: Thư viện hỗ trợ các công cụ học máy, ở đây dùng để chia dữ liệu thành tập huấn luyện và kiểm tra.

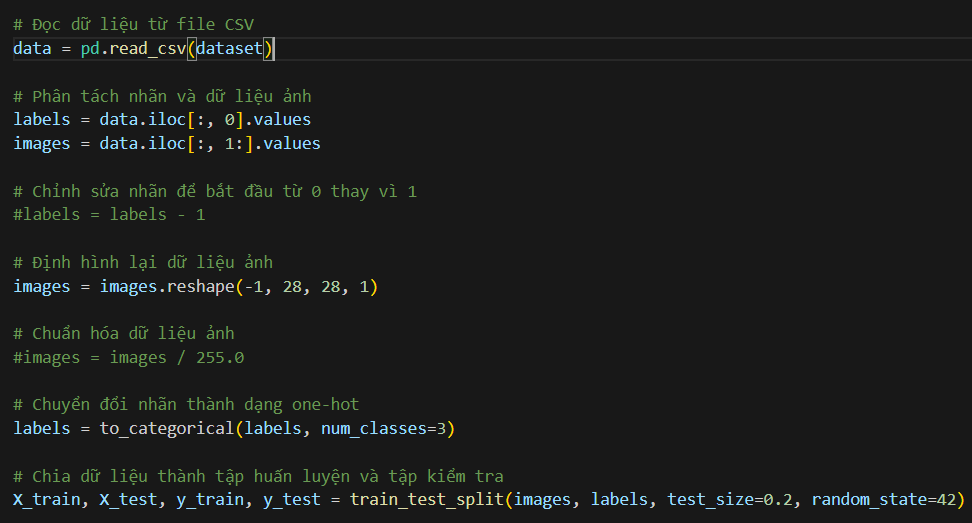
Khai báo địa chỉ và tên file sử dụng trong chương trình



Hình 2. 14:Huấn luyện mô hình AI (Phần 2)

* dataset: Đường dẫn tới file dữ liệu.
* model\_save\_path: Đường dẫn lưu mô hình sau khi huấn luyện.
* tflite\_save\_path: Đường dẫn lưu mô hình đã được chuyển đổi sang định dạng TensorFlow Lite.

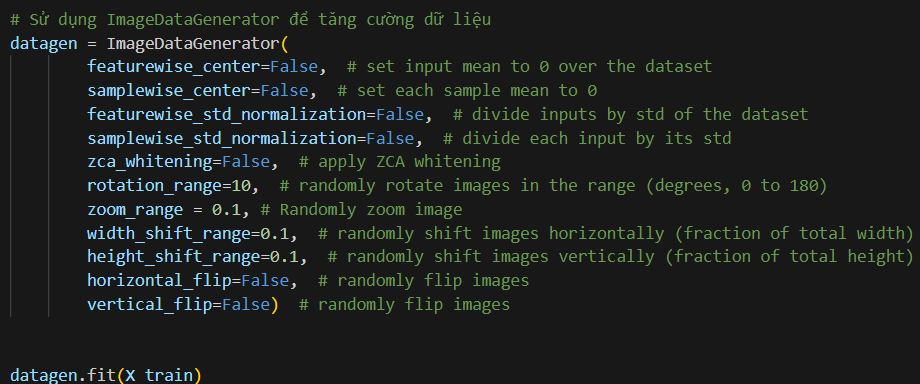
Đọc và xử lý dữ liệu



Hình 2. 15: Huấn luyện mô hình AI (Phần 3)

* Đọc dữ liệu từ file CSV.
* Tách nhãn và dữ liệu ảnh.
* Chuyển đổi dữ liệu ảnh thành định dạng phù hợp.
* Chuyển đổi nhãn sang dạng one-hot encoding.
* Chia dữ liệu thành tập huấn luyện và kiểm tra.

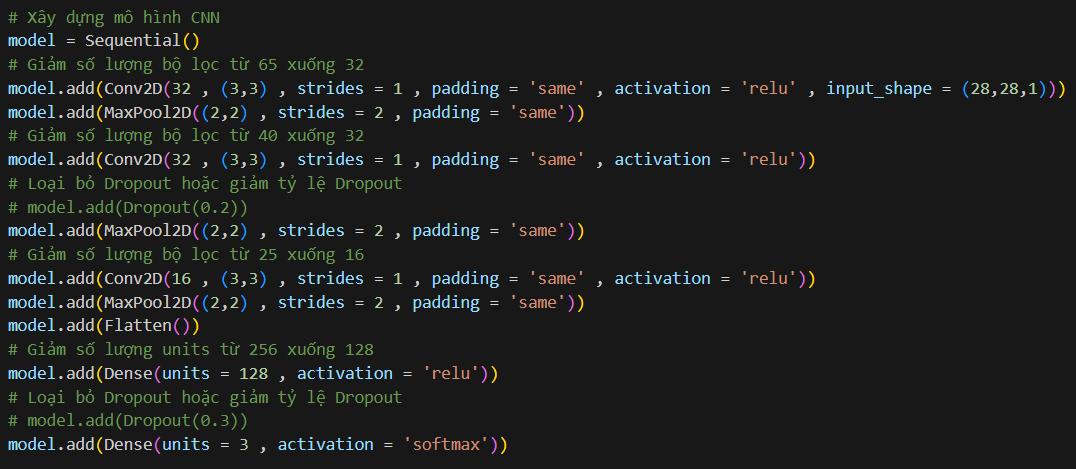
Tăng cường dữ liệu



Hình 2. 16: Huấn luyện mô hình AI (Phần 4)

* Sử dụng ImageDataGenerator để tăng cường dữ liệu bằng các phép biến đổi như xoay, zoom, dịch chuyển,…

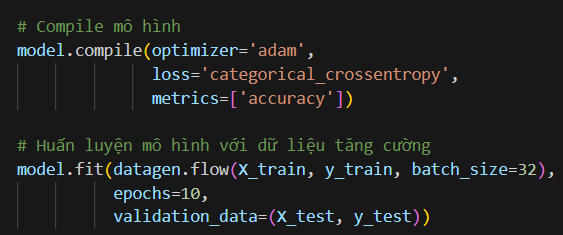
Xây dựng mô hình CNN



Hình 2. 17: Huấn luyện mô hình AI (Phần 5)

* Xây dựng một mô hình tuần tự gồm các lớp tích chập (Conv2D), pooling (MaxPool2D), và lớp fully connected (Dense).

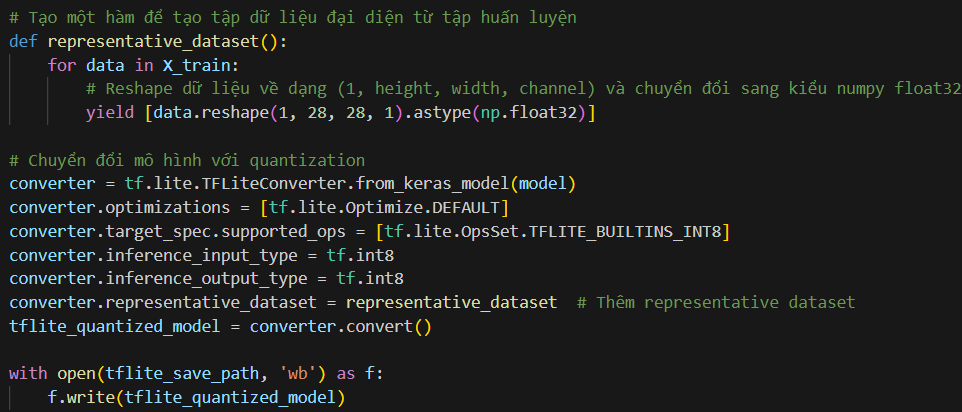
Biên dịch và huấn luyện mô hình



Hình 2. 18: Huấn luyện mô hình AI (Phần 6)

* Biên dịch mô hình với trình tối ưu Adam và hàm mất mát categorical\_crossentropy.
* Huấn luyện mô hình với dữ liệu tăng cường.

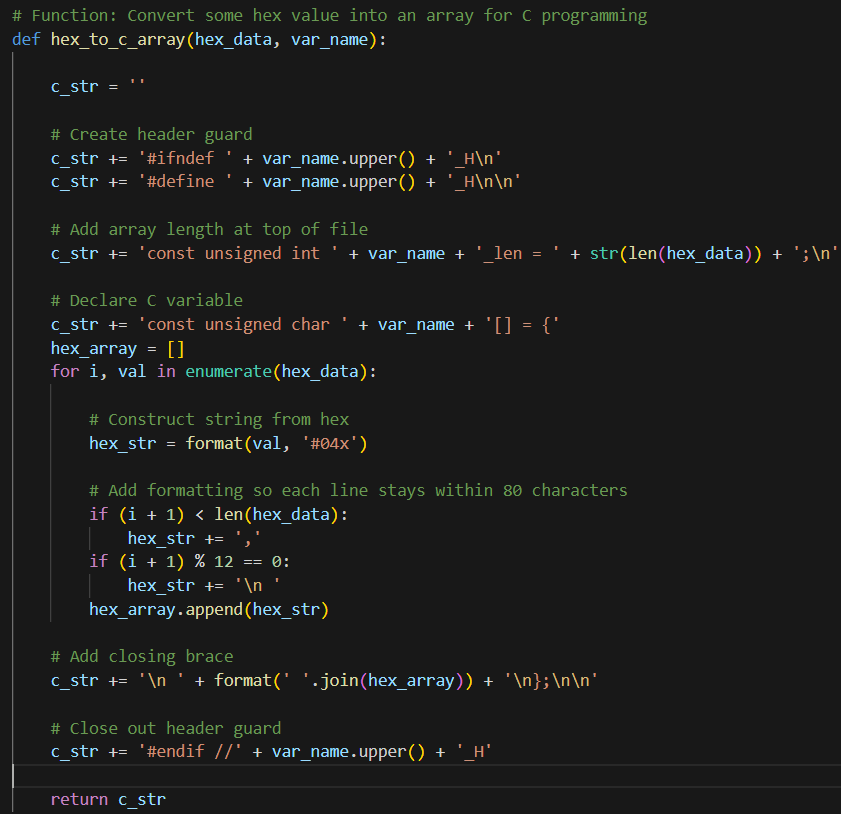
Chuyển đổi mô hình sang định dạng TensorFlow Lite



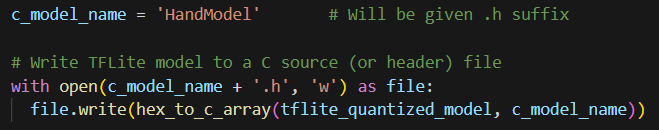
Hình 2. 19: Huấn luyện mô hình AI (Phần 7)

* Định nghĩa hàm representative\_dataset để cung cấp tập dữ liệu đại diện cho quá trình quantization.
* Chuyển đổi mô hình sang định dạng TensorFlow Lite với quantization để tối ưu hóa hiệu suất và kích thước.

Tạo file hex để nạp vào vi điều khiển

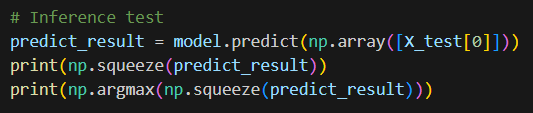


Hình 2. 20: Huấn luyện mô hình AI (Phần 8)



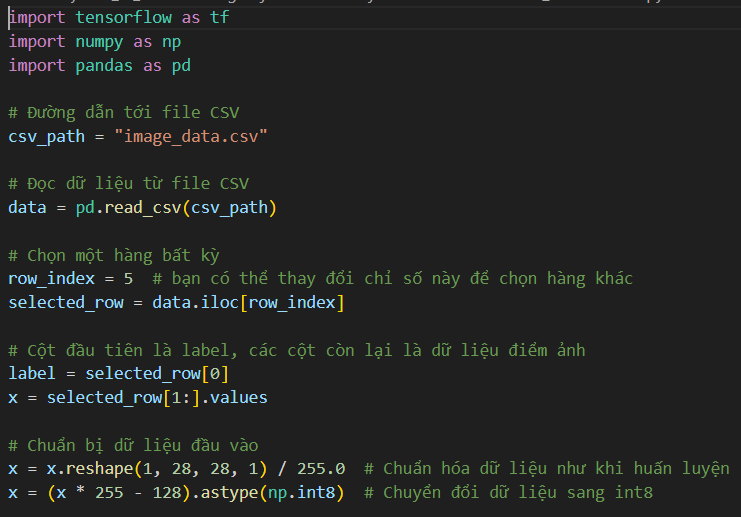
Hình 2. 21: Huấn luyện mô hình AI (Phần 9)

(Tùy chọn) Thực hiện dự đoán với một mẫu từ tập kiểm tra.

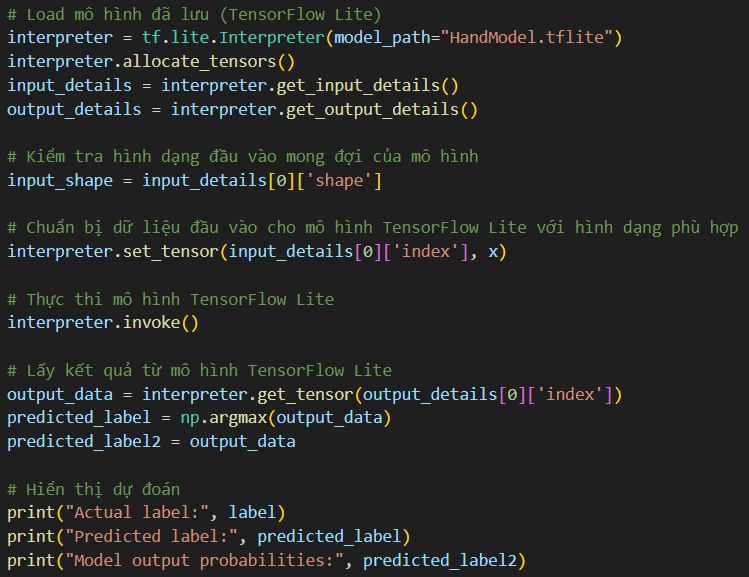


Hình 2. 22: Huấn luyện mô hình AI (Phần 10)

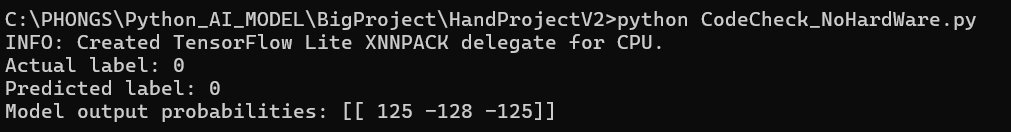
Hoặc sử dụng code sau để kiểm tra mô hình đã train



Hình 2. 23: Huấn luyện mô hình AI (Phần 11)



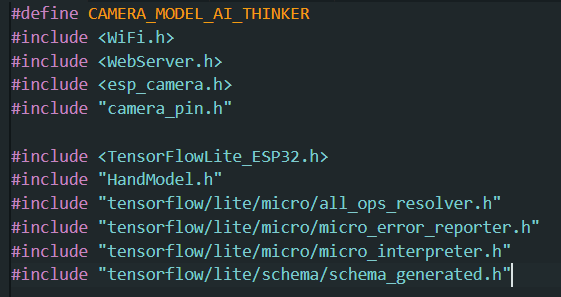
Hình 2. 24: Huấn luyện mô hình AI (Phần 12)



Hình 2. 25: Huấn luyện mô hình AI (Phần 13)

## 2.3.Nhúng mô hình AI và triển khai sử dụng trên ESP32CAM

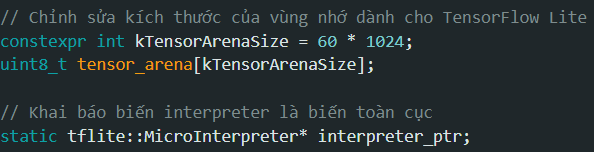
Đầu tiên khai báo thư viện:



Hình 2. 26: Nhúng mô hình AI (Phần 1)

* CAMERA\_MODEL\_AI\_THINKER: Định nghĩa cho biết loại camera đang sử dụng là AI Thinker.
* Bao gồm các thư viện cần thiết: WiFi, WebServer, ESP-Camera, TensorFlow Lite cho ESP32, và các thư viện liên quan đến TensorFlow Lite như model, resolver, error reporter, interpreter, và schema.

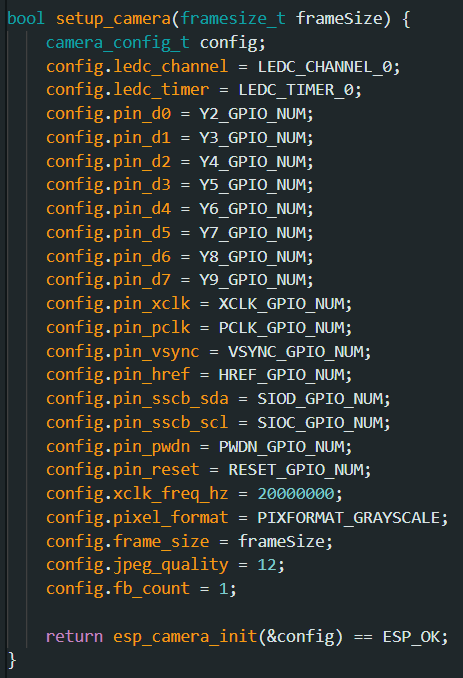
Khai báo vùng nhớ và biến toàn cục:



Hình 2. 27: Nhúng mô hình AI (Phần 2)

* kTensorArenaSize: Kích thước vùng nhớ dành cho TensorFlow Lite.
* tensor\_arena: Mảng byte dùng làm vùng nhớ.
* interpreter\_ptr: Con trỏ tới đối tượng interpreter của TensorFlow Lite.

### 2.3.1.Hàm setup\_camera()

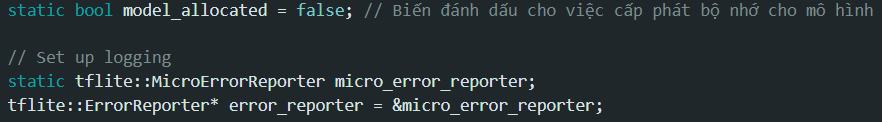


Hình 2. 28: Nhúng mô hình AI (Phần 3)

* Cấu hình và khởi tạo camera với các thiết lập cụ thể như định dạng pixel, kích thước khung hình, tần số xung nhịp, và các chân GPIO kết nối.

### 2.3.2.Hàm handle\_captureAI()

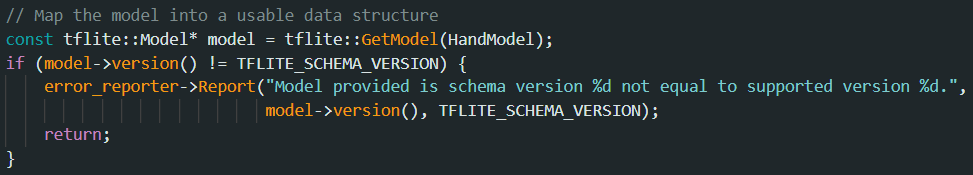
Khai báo các biến cần cho mô hình AI



Hình 2. 29: Nhúng mô hình AI (Phần 4)

* model\_allocated: Biến tĩnh đánh dấu để đảm bảo rằng bộ nhớ chỉ được cấp phát cho mô hình một lần duy nhất trong suốt quá trình chạy của chương trình.
* micro\_error\_reporter: Đối tượng để báo cáo lỗi trong quá trình xử lý TensorFlow Lite.
* error\_reporter: Con trỏ đến đối tượng micro\_error\_reporter để sử dụng trong quá trình báo cáo lỗi.

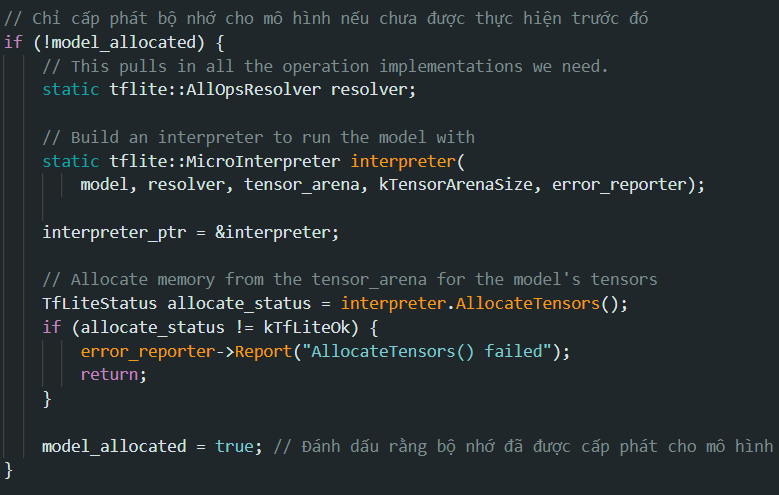
Nạp mô hình TensorFlow Lite



Hình 2. 30: Nhúng mô hình AI (Phần 5)

* tflite::GetModel(HandModel): Lấy mô hình từ dữ liệu nhị phân được biên dịch sẵn (HandModel).
* Kiểm tra phiên bản của mô hình, nếu không khớp với phiên bản hỗ trợ của TensorFlow Lite, báo lỗi và thoát khỏi hàm.

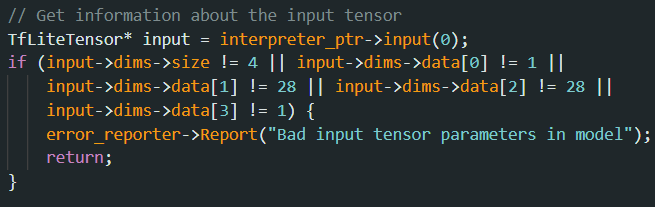
Cấp phát bộ nhớ cho mô hình



Hình 2. 31: Nhúng mô hình AI (Phần 6)

* tflite::AllOpsResolver: Lớp cung cấp các triển khai của tất cả các phép toán cần thiết để chạy mô hình.
* tflite::MicroInterpreter: Đối tượng interpreter được tạo để chạy mô hình, sử dụng mô hình, resolver, vùng nhớ (tensor\_arena), và bộ báo lỗi.
* interpreter\_ptr = &interpreter: Lưu trữ con trỏ tới interpreter để sử dụng sau này.
* interpreter.AllocateTensors(): Cấp phát bộ nhớ cho các tensor của mô hình. Nếu thất bại, báo lỗi và thoát khỏi hàm.
* model\_allocated = true: Đánh dấu rằng bộ nhớ đã được cấp phát cho mô hình.

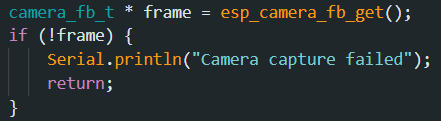
Lấy thông tin về tensor đầu vào



Hình 2. 32: Nhúng mô hình AI (Phần 7)

* interpreter\_ptr->input(0): Lấy tensor đầu vào của mô hình.
* Kiểm tra các kích thước của tensor đầu vào. Nếu không khớp với yêu cầu (1, 28, 28, 1), báo lỗi và thoát khỏi hàm.

Chụp ảnh



Hình 2. 33: Nhúng mô hình AI (Phần 8)

* esp\_camera\_fb\_get(): Chụp một khung hình từ camera.
* Nếu không chụp được ảnh, in thông báo lỗi và thoát khỏi hàm.

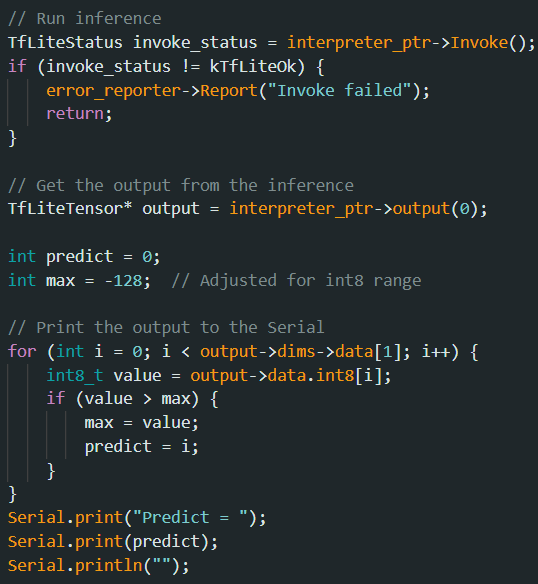
Chỉnh kích thước và chuẩn hóa giá trị pixel



Hình 2. 34: Nhúng mô hình AI (Phần 9)

* RESIZE\_WIDTH và RESIZE\_HEIGHT: Kích thước mong muốn của ảnh đầu vào (28x28).
* scale\_x và scale\_y: Tỷ lệ thu nhỏ ảnh từ kích thước gốc về kích thước 28x28.
* Chỉnh kích thước và chuẩn hóa: Tính toán giá trị trung bình của các pixel trong mỗi ô tỷ lệ, chuẩn hóa về khoảng giá trị [0, 255] rồi chuyển đổi sang khoảng giá trị [-128, 127] để phù hợp với đầu vào int8 của mô hình.
* esp\_camera\_fb\_return(frame): Trả lại bộ nhớ của khung hình cho camera sau khi đã xử lý xong.

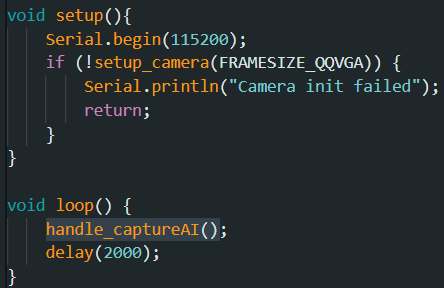
Thực hiện suy luận và in ra serial:



Hình 2. 35: Nhúng mô hình AI (Phần 10)

* interpreter\_ptr->Invoke(): Thực hiện suy luận trên dữ liệu đầu vào đã chuẩn bị. Nếu thất bại, báo lỗi và thoát khỏi hàm.
* interpreter\_ptr->output(0): Lấy tensor đầu ra của mô hình.
* Tìm giá trị lớn nhất trong đầu ra: Lặp qua các giá trị đầu ra để tìm giá trị lớn nhất, lưu lại chỉ số của giá trị đó (predict).
* In kết quả: In kết quả suy luận ra Serial.

### 2.3.3.Hàm setup() và loop():



Hình 2. 36: Nhúng mô hình AI (Phần 11)

* Serial.begin(115200): Khởi động giao tiếp Serial với tốc độ 115200 baud.
* setup\_camera(FRAMESIZE\_QQVGA): Khởi tạo camera với kích thước khung hình QQVGA.
* Gọi hàm handle\_captureAI để chụp ảnh và xử lý.
* delay(2000): Đợi 2 giây trước khi chụp và xử lý ảnh tiếp theo.

# Chương 3: Kết luận

Trong đồ án báo cáo này, chúng em đã thực hiện đồ án nhận diện cử chỉ bằng ESP32CAM. Bằng việc tự nghiệm thu kết quả đồ án của nhóm, chúng em cũng rút ra được những ưu- nhược điểm của đồ án:

* Ưu điểm
  + Toàn bộ quá trình nhận diện cử chỉ được thực hiện hoàn toàn trên ESP32CAM.
  + Tự triển khai mô hình AI phù hợp cho phần cứng cá nhân
* Hạn chế đồ án
  + Quá trình nhận diện cử chỉ làm tiêu hao quá nhiều bộ nhớ của ESP32CAM
  + Số lượng cử chỉ nhận diện còn khá ít, chưa đạt được sự đa dạng.
* Hướng phát triển của đề tài
  + Cải thiện hiệu suất của quá trình nhận diện cử chỉ
  + Mở rộng số lượng cử chỉ nhận diện được.
  + Bổ sung thêm những tính năng khác như lưu ảnh trên thẻ SD,..

# TÀI LIỆU THAM KHẢO

* Sign Language MNIST

<https://www.kaggle.com/datasets/datamunge/sign-language-mnist>

* Arduino Forum

<https://forum.arduino.cc/t/esp32_cam-acces-and-process-image/677628>

* Project GitHub

<https://github.com/jonathanrandall/American-Sign-language>

Sinh viên thực hiện Giảng viên hướng dẫn

Trần Hào Phóng Nguyễn Trọng Kiên

Võ Minh Thông