**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 : 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*

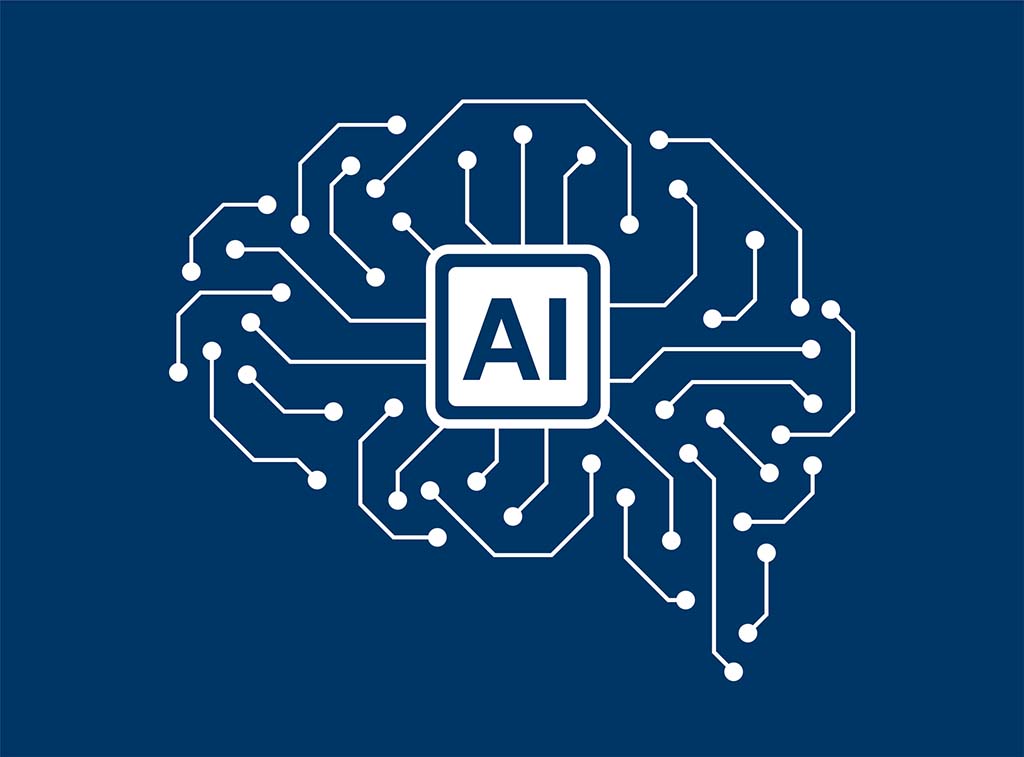
Mục lục

Chương 1: Cơ sở lý thuyết – phần cứng

1.1.Cơ sở lý thuyết

1.1.1.Kiến thức về AI

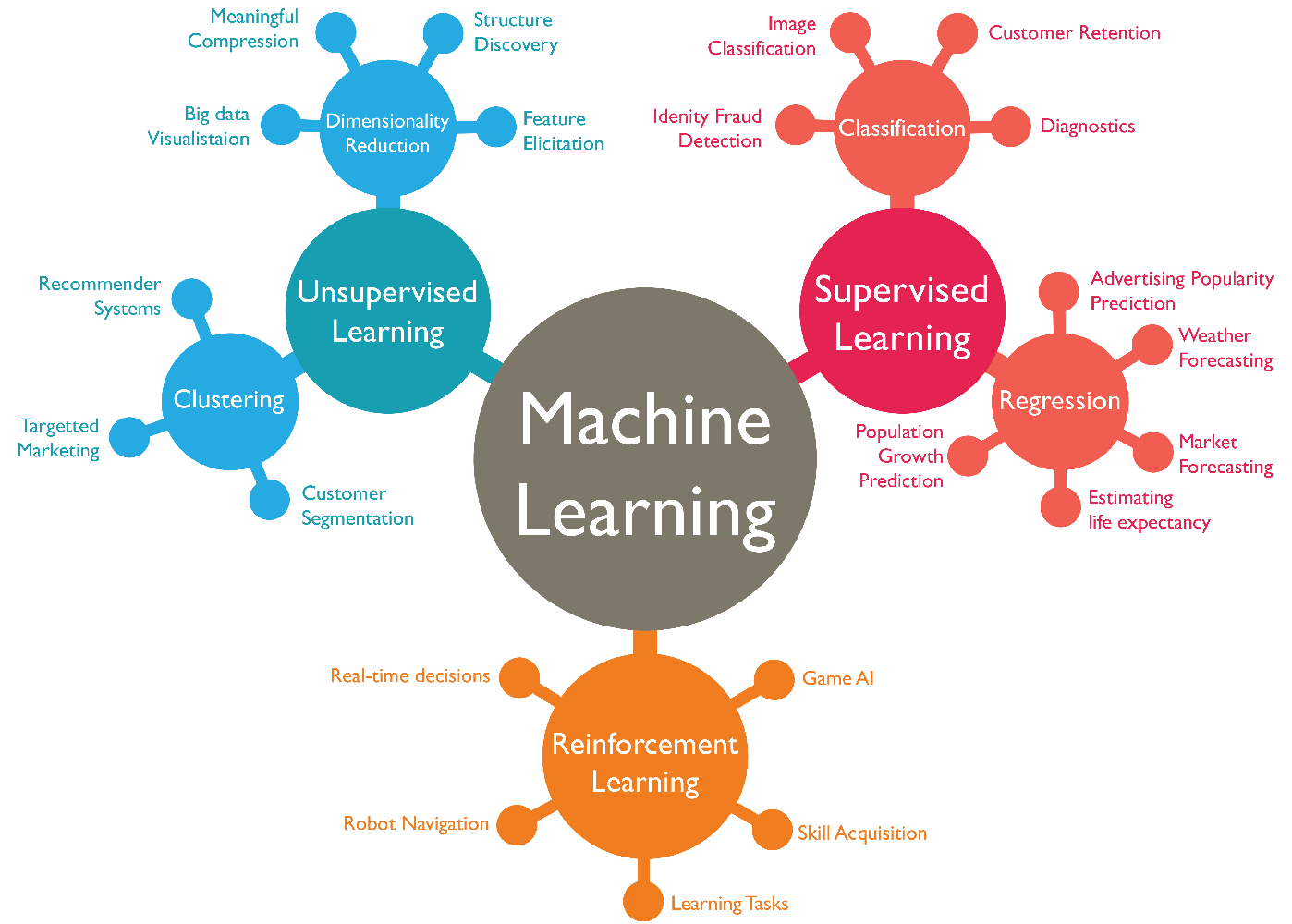
1.1.1.1.Khái niệm cơ bản



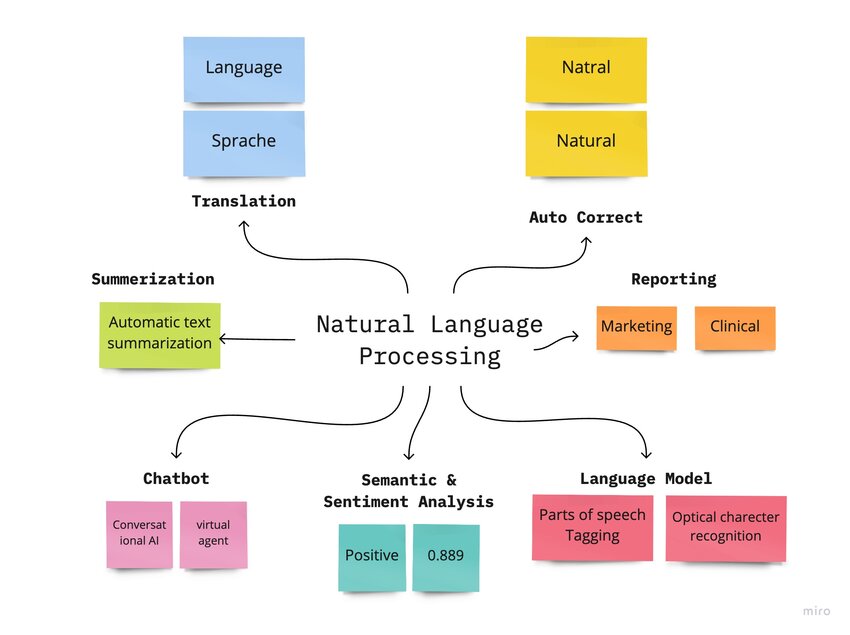
AI (Artificial Intelligence - Trí tuệ nhân tạo) là một lĩnh vực trong khoa học máy tính, nghiên cứu và phát triển các hệ thống máy tính có khả năng thực hiện các tác vụ thông minh mà bình thường cần có trí tuệ con người để thực hiện. Các hệ thống AI có thể bao gồm từ các chương trình đơn giản như nhận 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:

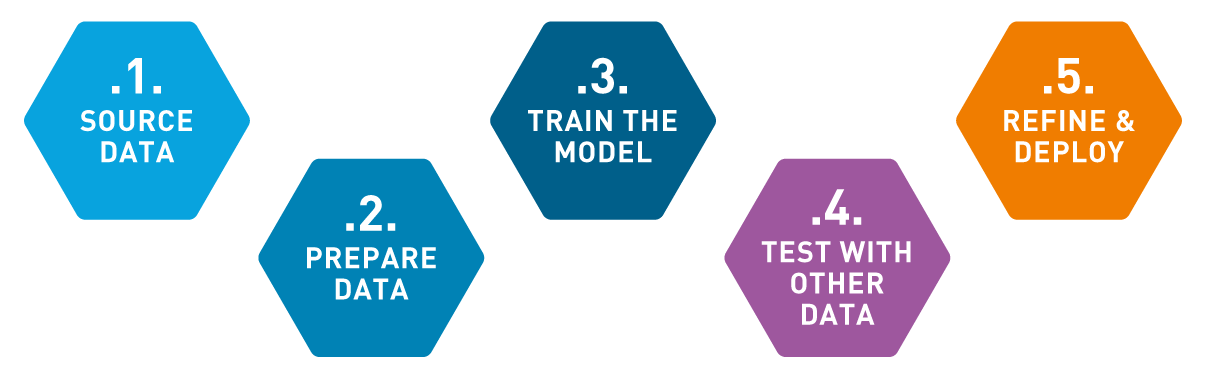
* Machine Learning (Học máy): Một nhánh của AI, tập trung vào việc phát triển các thuật toán cho phép máy tính học từ dữ liệu và cải thiện hiệu suất theo thời gian mà không cần lập trình cụ thể.



* 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.



1.1.1.2.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

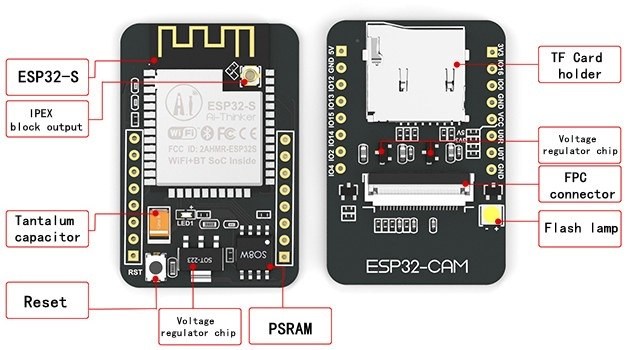
1.1.2.1.

1.1.2.2.

1.2.Cơ sở phần cứng

1.2.1. ESP32CAM

1.2.1.1.Thông số kỹ thuật và sơ đồ chân

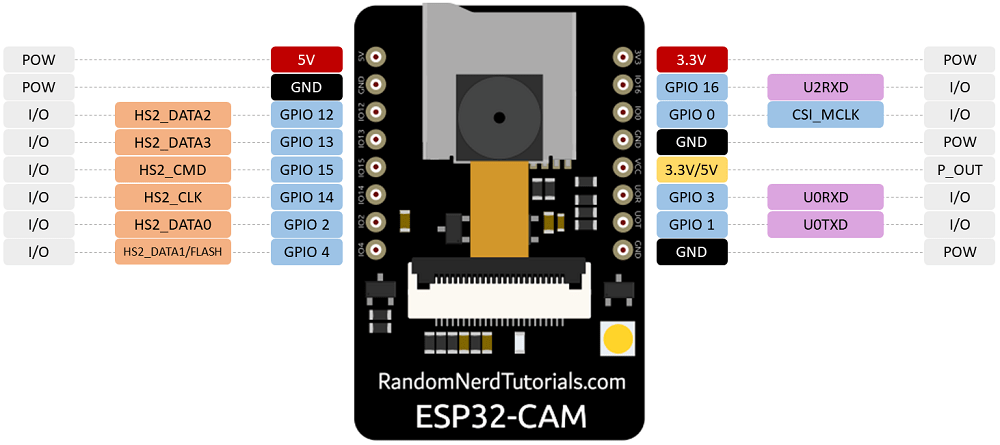


ESP32CAM là một module camera nhỏ gọn và mạnh mẽ, dựa trên vi xử lý ESP32 của Espressif Systems. Nó được thiết kế đặc biệt cho các ứng dụng IoT (Internet of Things) và các dự án DIY (Do It Yourself), cung cấp các tính năng vượt trội với giá cả phải chăng.

-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



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

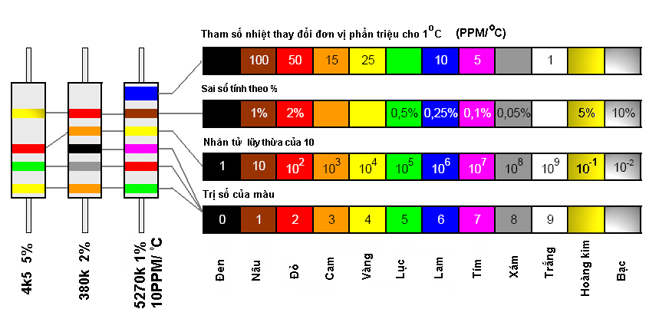


Điện trở là một trong những linh kiện điện tử cơ bản nhất và quan trọng trong mọi mạch điện tử. Điện trở có chức năng hạn chế dòng điện chạy qua mạch, phân chia điện áp, bảo vệ các linh kiện khác, và tạo ra các tín hiệu điều khiển cần thiết trong các thiết kế mạch. Chúng có nhiều loại khác nhau và được 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%).



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

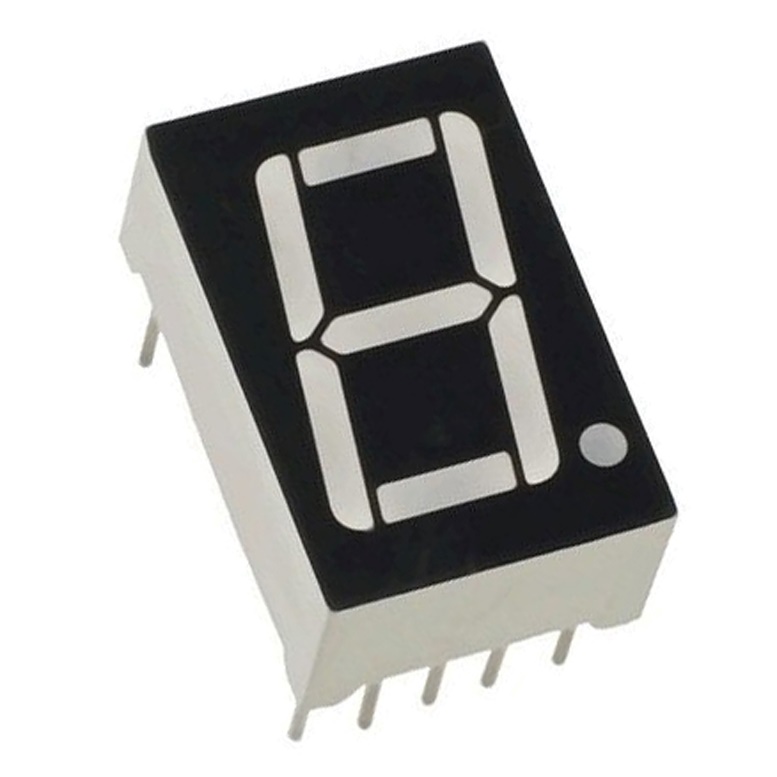
Điện trở cố định: Có giá trị cố định và không thể thay đổi được. Đây là loại điện trở phổ biến nhất.

* 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

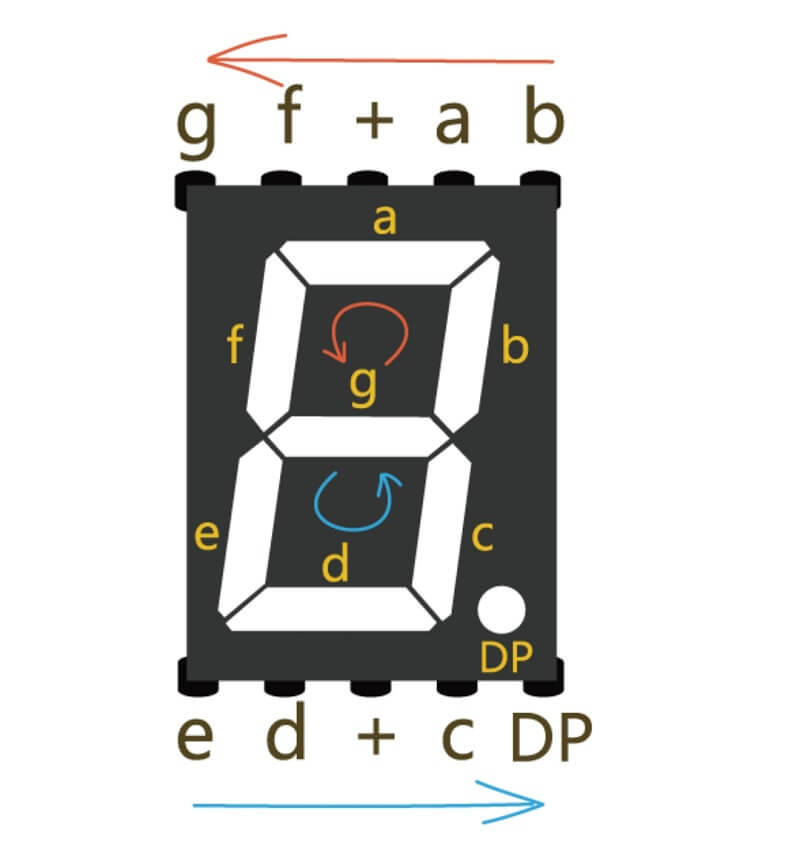
* Hạn dòng cho đèn LED: Điện trở được sử dụng để hạn chế dòng điện đi qua đèn LED, bảo vệ đèn LED khỏi bị quá tải và cháy nổ.
* Tạo phân áp: Bằng cách kết nối các điện trở theo cấu hình phân áp, có thể chia một điện áp lớn thành các điện áp nhỏ hơn. Đây là một cách thông dụng để cung cấp điện áp thích 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



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.



* 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.

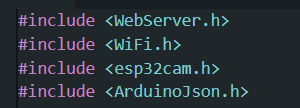
* Đồng hồ kỹ thuật số: Sử dụng LED 7 đoạn để hiển thị giờ, phút, và giây.
* Bộ đếm: Trong các ứng dụng công nghiệp và thương mại, LED 7 đoạn được sử dụng để hiển thị số liệu đếm như sản phẩm sản xuất, số lượt truy cập, hoặc các chỉ số thống kê khác.
* Bảng hiển thị: Trong các biển quảng cáo, máy bán 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

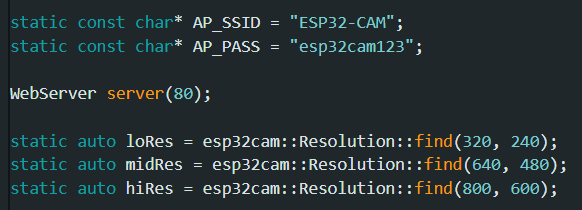
2.1. Triển khai theo hướng dùng phần cứng ngoài hỗ trợ

2.1.1.Tạo access point và gửi ảnh từ ESP32CAM

Đầu tiên khai báo các thư viện cần thiết cho chương trình. WebServer.h và WiFi.h là để tạo một web server và kết nối WiFi. esp32cam.h là thư viện để điều khiển camera trên ESP32-CAM. ArduinoJson.h được sử dụng để xử lý dữ liệu JSON.



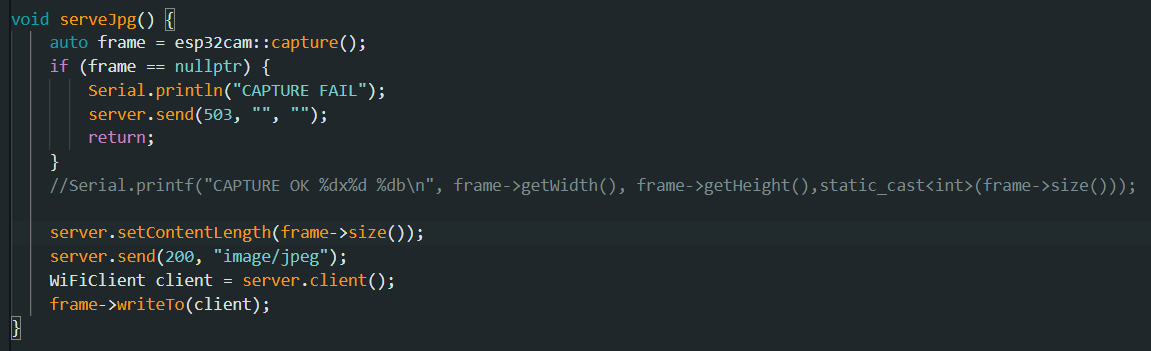
Tiếp theo là cấu hình những thông số cần cho chương trình:



* Cấu hình tên và mật khẩu cho Access Point (AP) mà ESP32-CAM sẽ tạo ra để cho phép các thiết bị khác kết nối và tương tác với nó qua WiFi.
* Tạo một đối tượng WebServer lắng nghe các yêu cầu HTTP trên cổng 80.
* Định nghĩa ba độ phân giải ảnh: low resolution (320x240), medium resolution (640x480), và high resolution (800x600).

Triển khai các hàm con phục vụ chương trình:

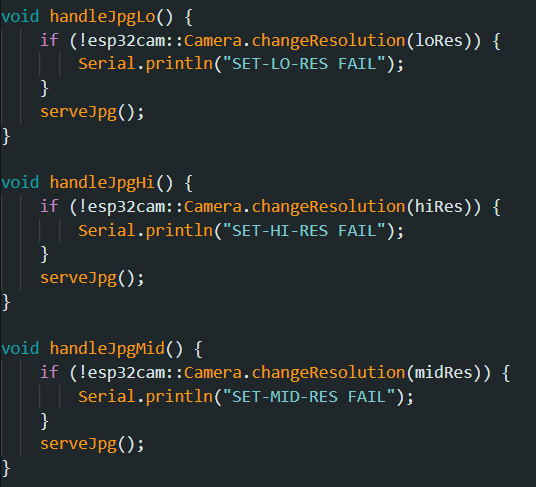
2.1.1.1.Hàm chụp và gửi ảnh qua server



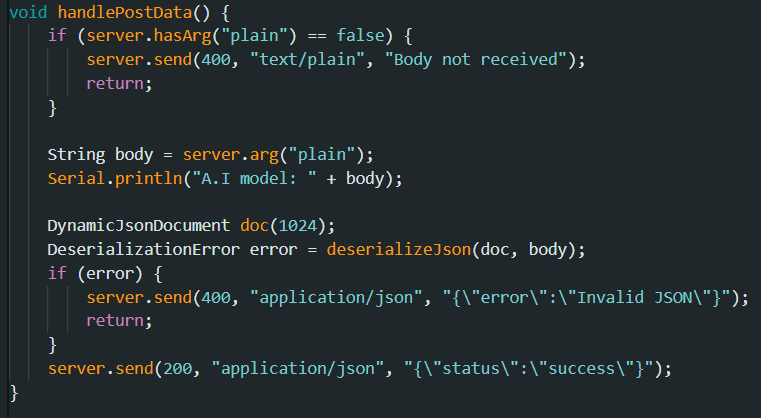
Đoạn code bao gồm:

* esp32cam::capture(): Hàm này được gọi để chụp một ảnh từ camera ESP32-CAM. Kết quả trả về là một con trỏ đến dữ liệu khung hình (frame).
* if (frame == nullptr): Kiểm tra xem việc chụp ảnh có thành công hay không. Nếu không, in ra thông báo "CAPTURE FAIL" trên giao tiếp serial và gửi mã lỗi 503 (Service Unavailable) đến client.
* server.setContentLength(frame->size()): Thiết lập độ dài (content length) của dữ liệu ảnh mà server sẽ gửi đi. Độ dài này được lấy từ kích thước của khung hình (frame).
* server.send(200, "image/jpeg"): Gửi mã phản hồi 200 (OK) và khai báo rằng dữ liệu sẽ được gửi dưới dạng ảnh JPEG.
* WiFiClient client = server.client(): Tạo một kết nối mới với client thông qua đối tượng WiFiClient.
* frame->writeTo(client): Ghi dữ liệu khung hình (frame) vào client, tức là gửi ảnh đi qua kết nối WiFi đến client.

2.1.1.2. Các hàm chuyển độ phân giải (tùy chọn)



2.1.1.3. Hàm nhận phản hồi về esp32cam:



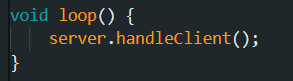
* Hàm server.hasArg("plain") kiểm tra xem có đối số nào với tên "plain" được gửi kèm theo yêu cầu POST hay không.
* Nếu không có, hàm server.send(400, "text/plain", "Body not received") gửi phản hồi HTTP với mã trạng thái 400 (Bad Request) và thông báo rằng không nhận được nội dung.
* Lệnh String body = server.arg("plain"); lấy nội dung của đối số "plain" và lưu nó vào biến body.
* Serial.println("A.I model: " + body); in nội dung của biến body ra cổng serial để debug.
* DynamicJsonDocument doc(1024); khởi tạo một đối tượng DynamicJsonDocument với kích thước bộ nhớ đệm là 1024 byte.
* DeserializationError error = deserializeJson(doc, body); thực hiện việc giải mã JSON từ chuỗi body và lưu kết quả vào doc.
* Nếu có lỗi trong quá trình giải mã JSON, server.send(400, "application/json", "{\"error\":\"Invalid JSON\"}"); gửi phản hồi HTTP với mã trạng thái 400 và thông báo lỗi JSON không hợp lệ.
* Nếu JSON hợp lệ và được giải mã thành công, gửi phản hồi HTTP với mã trạng thái 200 (OK) và thông báo rằng yêu cầu đã thành công.

2.1.1.4. Hàm setup và loop của chương trình:



* Esp32cam Configuration: Cấu hình ESP32-CAM bao gồm:
  + cfg.setPins(pins::AiThinker): Xác định các chân kết nối của module camera.
  + cfg.setResolution(hiRes): Thiết lập độ phân giải ảnh cao nhất.
  + cfg.setBufferCount(2): Đặt số lượng buffer ảnh sẽ được sử dụng.
  + cfg.setJpeg(80): Thiết lập chất lượng ảnh JPEG.
* Camera.begin(cfg): Khởi động camera với các cấu hình đã được thiết lập. Trạng thái khởi động được in ra thông qua giao tiếp serial.
* WiFi.mode(WIFI\_AP): Thiết lập chế độ hoạt động của WiFi là Access Point (AP).
* WiFi.softAP(AP\_SSID, AP\_PASS): Tạo một Access Point với tên và mật khẩu đã định nghĩa.
* In địa chỉ IP của Access Point và các endpoint: In địa chỉ IP của Access Point ra giao tiếp serial để người dùng biết. Các đường dẫn của các endpoint cũng được in ra để hướng dẫn người dùng sử dụng.
* Gán các endpoint và các hàm xử lý tương ứng cho server: Các hàm xử lý đã được định nghĩa trước đó sẽ được gán vào các endpoint tương ứng để xử lý các yêu cầu từ client.
* server.begin(): Khởi động server để bắt đầu lắng nghe các yêu cầu từ client.

Tiếp theo là hàm loop():

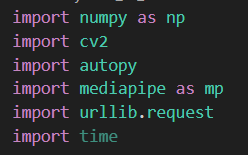


Chương trình sẽ liên tục gọi hàm server.handleClient() để xử lý các yêu cầu mới từ client:

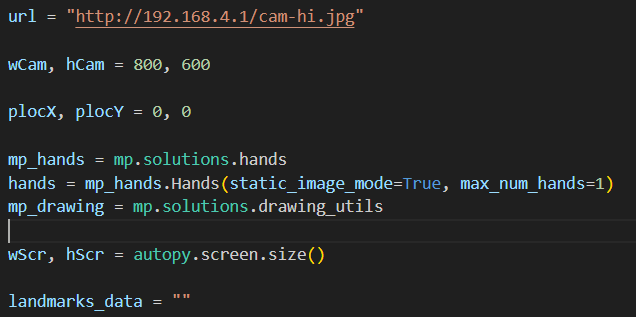
2.1.2.Thu thập dữ liệu và huấn luyện mô hình AI tại phần cứng ngoài

2.1.2.1.Thu thập dữ liệu

Đầu tiên khai báo các thư viện cần thiết: numpy (để làm việc với mảng), cv2 (OpenCV để xử lý hình ảnh), autopy (để điều khiển chuột và bàn phím), mediapipe (để phát hiện landmark trên tay trong hình ảnh), urllib.request (để gửi yêu cầu HTTP đến một URL).

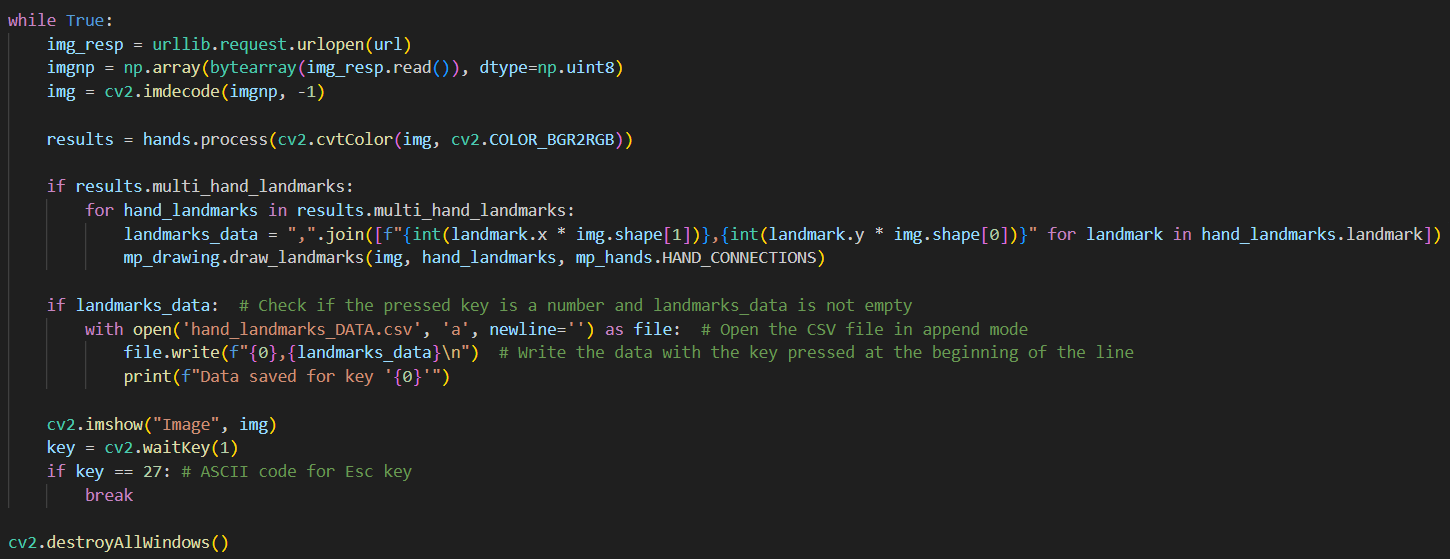


Tiếp theo là cấu hình những thông số cần cho chương trình:

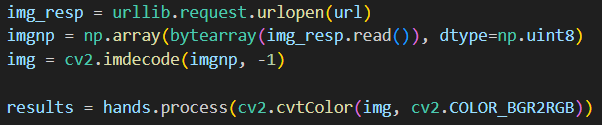


* Địa chỉ URL của camera IP hoặc bất kỳ nguồn hình ảnh nào mà chương trình sẽ lấy dữ liệu từ.
* Độ phân giải mà hình ảnh từ camera sẽ được đọc và xử lý.
* Tọa độ (X, Y) của tâm của bàn phím. Ban đầu được đặt ở vị trí (0, 0).
* Khởi tạo các đối tượng cho việc phát hiện landmark trên tay bằng MediaPipe Hands.
* Lấy kích thước của màn hình hiển thị.
* Chuỗi để lưu các landmark của tay.

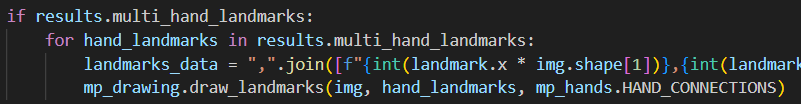
Vòng lặp chính của chương trình:



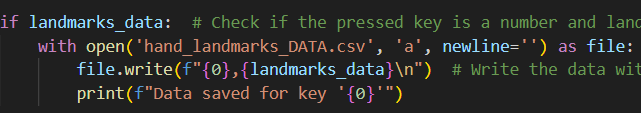
* Triển khai đọc hình ảnh từ server (qua URL của esp32cam), giải mã thành hình ảnh OpenCV, sau đo phát hiện các thông số landmark thông qua thư viện Mediapipe.



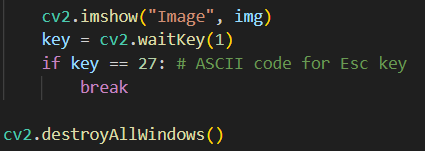
* Nếu có ít nhất một tay được phát hiện trong hình ảnh, lặp qua các landmark của tay và lưu tọa độ của mỗi landmark vào biến landmarks\_data, sau đó vẽ các landmark lên hình ảnh.



* Nếu biến landmarks\_data không rỗng (tức là có tay được phát hiện và có dữ liệu landmark), lưu các tọa độ landmark vào tệp tin CSV với nhãn tương ứng (ở đây là 0) ở đầu dòng.

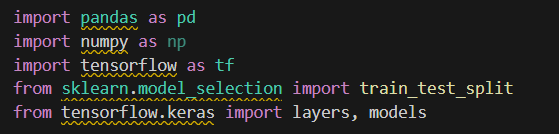


* Hiển thị hình ảnh đã xử lý, và chờ người dùng nhấn phím (ấn Esc để thoát), sau đó đóng tất cả các cửa sổ hình ảnh mở ra bởi OpenCV.



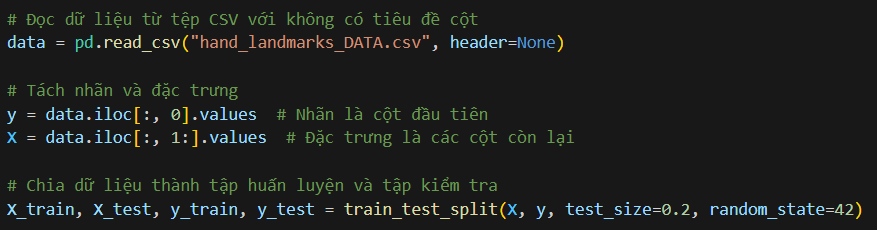
2.1.2.2.Huấn luyện mô hình AI

Đầu tiên khai báo các thư viện cần cho việc huấn luyện AI:



* pandas: Thư viện dùng để làm việc với dữ liệu dạng bảng.
* numpy: Thư viện dùng để làm việc với mảng và ma trận.
* tensorflow: Thư viện machine learning, trong đoạn này sử dụng TensorFlow để xây dựng và huấn luyện mô hình.
* train\_test\_split: Hàm trong sklearn để chia dữ liệu thành tập huấn luyện và tập kiểm tra.
* layers, models: Các module trong tensorflow.keras để xây dựng mô hình neural network.

Đọc và sắp xếp dữ liệu đưa vào huấn luyện:



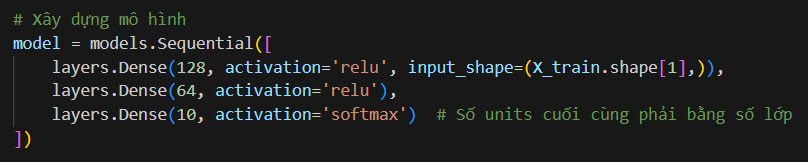
Đọc dữ liệu từ tệp CSV hand\_landmarks\_DATA.csv vào DataFrame data mà không có header cột.

y chứa nhãn (label), là cột đầu tiên của DataFrame data.

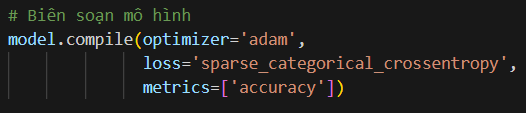
X chứa các đặc trưng (features), là tất cả các cột còn lại của DataFrame data.

Chia dữ liệu thành tập huấn luyện và tập kiểm tra với tỉ lệ 80/20.

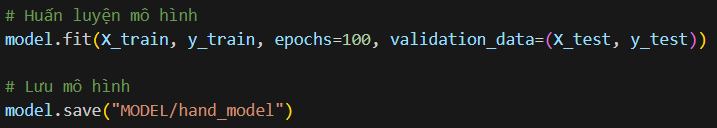
Xây dựng và tiến hành huấn luyện mô hình AI:



* Mô hình neural network được xây dựng với 3 lớp:
* Lớp input có 128 node với hàm kích hoạt relu.
* Lớp ẩn có 64 node với hàm kích hoạt relu.
* Lớp output có 10 node (tương ứng với số lớp) với hàm kích hoạt softmax.

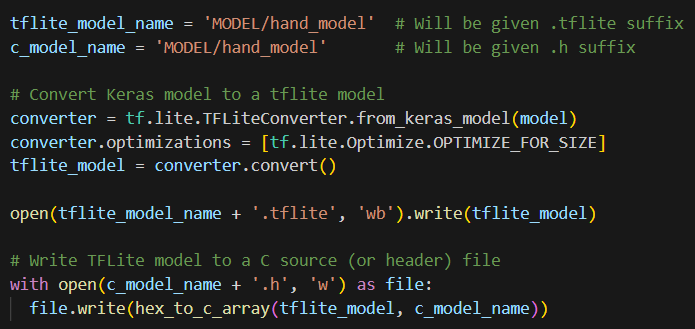


* Biên soạn mô hình với optimizer là 'adam', loss function là 'sparse\_categorical\_crossentropy', và metrics để đánh giá là 'accuracy'.



* Huấn luyện mô hình trên tập huấn luyện X\_train, y\_train trong 100 epochs, và sử dụng tập kiểm tra X\_test, y\_test để đánh giá mô hình.
* Lưu mô hình vào thư mục "MODEL" dưới tên "hand\_model".

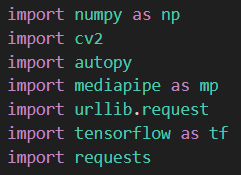
Lưu mô hình dưới dạng tensorflowlite và dang hex (tùy chọn)



* Chuyển đổi mô hình từ định dạng Keras sang định dạng TFLite và lưu vào tệp tin ".tflite".
* Chuyển đổi mô hình TFLite thành một mảng hex và lưu vào một tệp tin C ".h".

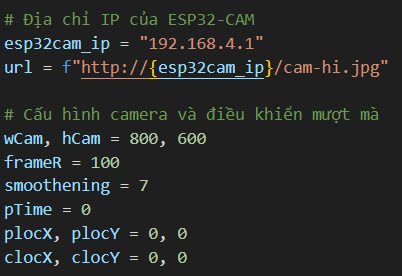
2.1.3.Sử dụng mô hình AI và gửi kết quả về ESP32CAM

Đầu tiên khai báo thư viến sử dụng:

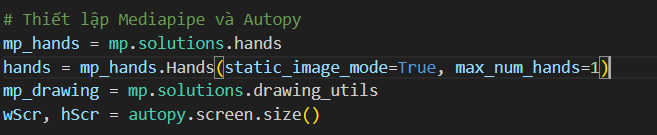


* numpy, cv2, autopy: Các thư viện để làm việc với mảng, xử lý hình ảnh, và điều khiển chuột.
* mediapipe: Thư viện sử dụng để phát hiện landmark trên tay trong hình ảnh.
* urllib.request: Thư viện để tạo yêu cầu HTTP đến ESP32-CAM để lấy hình ảnh.
* tensorflow: Thư viện machine learning, được sử dụng để dự đoán hành động tay.
* requests: Thư viện để gửi yêu cầu HTTP đến ESP32-CAM để gửi dữ liệu dự đoán.

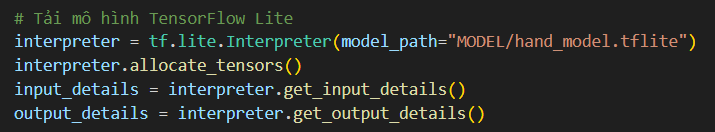
Khai báo thông số và thiết lập cơ bản:



* Định nghĩa địa chỉ IP của ESP32-CAM và URL để lấy hình ảnh.
* Cấu hình kích thước hình ảnh và các biến khác cho việc xử lý hình ảnh và tracking.



* Khởi tạo đối tượng để sử dụng MediaPipe Hands để phát hiện landmark trên tay.
* Lấy kích thước của màn hình sẽ được sử dụng bởi Autopy (tùy chọn)

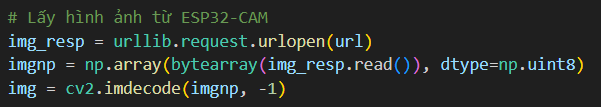


* Tải mô hình TensorFlow Lite đã huấn luyện trước từ file hand\_model.tflite.
* Khởi tạo một Interpreter để thực thi mô hình.

Vòng lặp chính của chương trình:

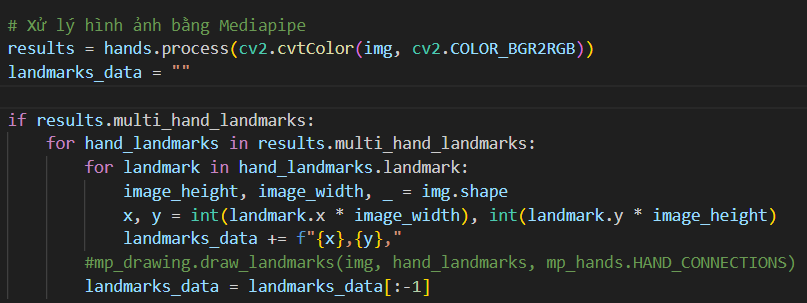


Lấy dữ liệu ảnh từ ESP32CAM:



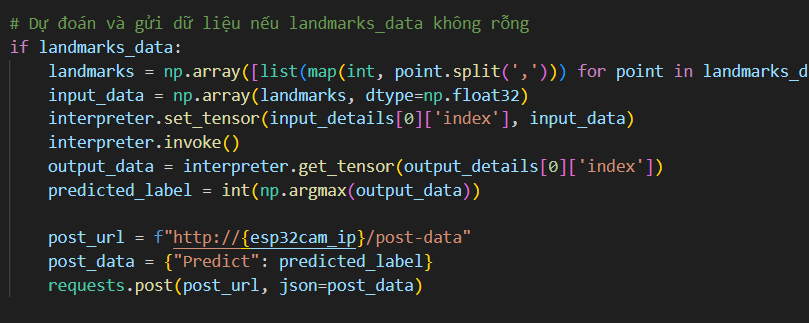
* Sử dụng urllib.request để gửi yêu cầu HTTP đến ESP32-CAM và nhận hình ảnh.
* Chuyển đổi dữ liệu nhận được thành một mảng numpy sử dụng np.array.
* Sử dụng OpenCV (cv2.imdecode) để giải mã và đọc hình ảnh từ mảng nhận được.

Lấy dữ liệu landmark bằng Mediapipe để làm dữ liệu đầu vào mô hình AI:



* Sử dụng Mediapipe Hands để phát hiện landmark trên tay trong hình ảnh. Mỗi landmark được biểu diễn bằng tọa độ (x, y).
* Tạo một chuỗi landmarks\_data chứa tất cả các tọa độ của các landmark.

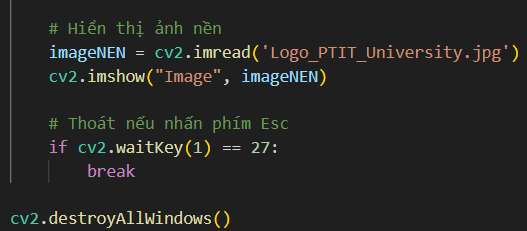
Sử dụng mô hình AI với dữ liệu có được, nhận diện cử chỉ:



Nếu landmarks\_data không rỗng, chuyển đổi chuỗi tọa độ thành một mảng numpy, sau đó truyền vào mô hình TensorFlow Lite để dự đoán hành động tương ứng với tọa độ của các landmark.

Dự đoán xong, gửi dữ liệu dự đoán lên ESP32-CAM thông qua giao thức HTTP.

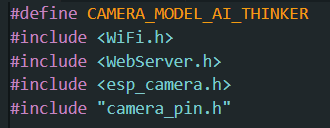
(Tùy chọn) Tạo ảnh nền giao diện và phím thoát chương trình



2.2. Triển khai theo hướng nhúng vào phần cứng

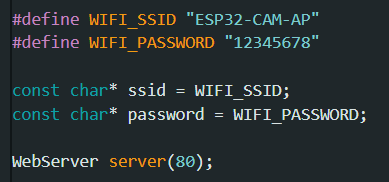
2.2.1.Xử lí ảnh chụp và gửi dữ liệu điểm ảnh

Đầu tiên khai báo thư viện sử dụng:



* 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:



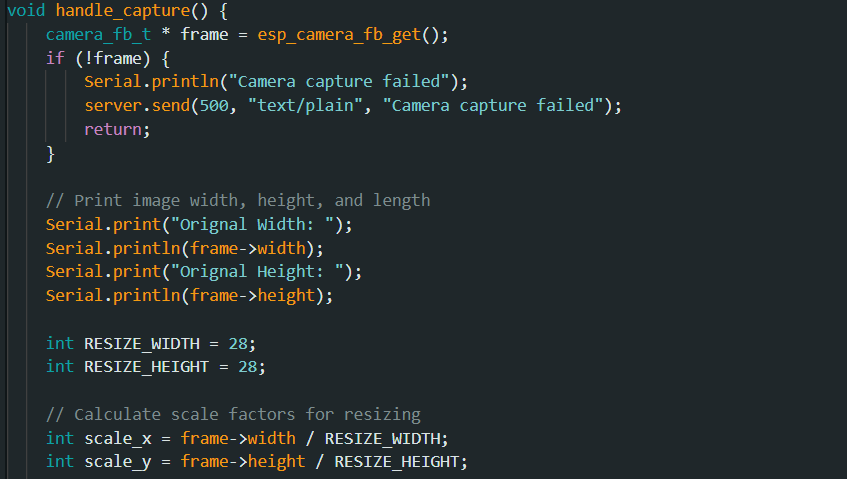
* WIFI\_SSID: Định nghĩa tên mạng WiFi mà ESP32 sẽ phát.
* WIFI\_PASSWORD: Định nghĩa mật khẩu cho mạng WiFi đó.
* ssid và password: Chứa tên mạng và mật khẩu WiFi.
* server: Đối tượng máy chủ web chạy trên cổng 80.

2.2.1.1. Hàm setup\_cam()



* 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.

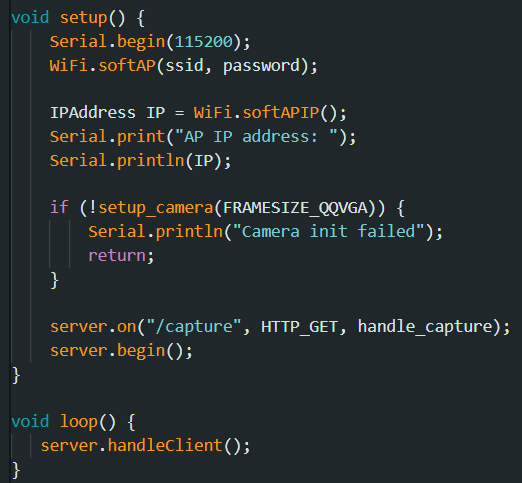
2.2.1.2. Hàm handle\_capture()





* 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.

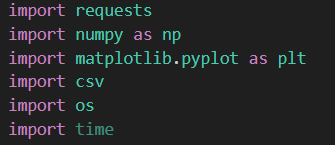
2.2.1.3. Hàm setup() và loop()



* 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.

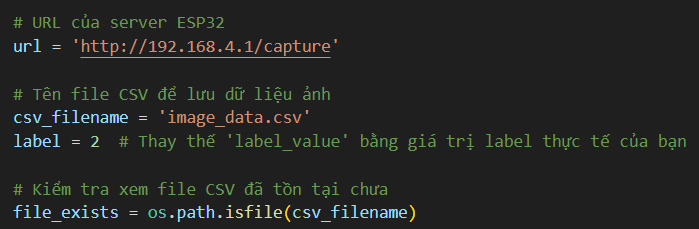
2.2.2.Thu thập dữ liệu và huấn luyện mô hình AI

Đầu tiên khai báo thư viện sử dụng:



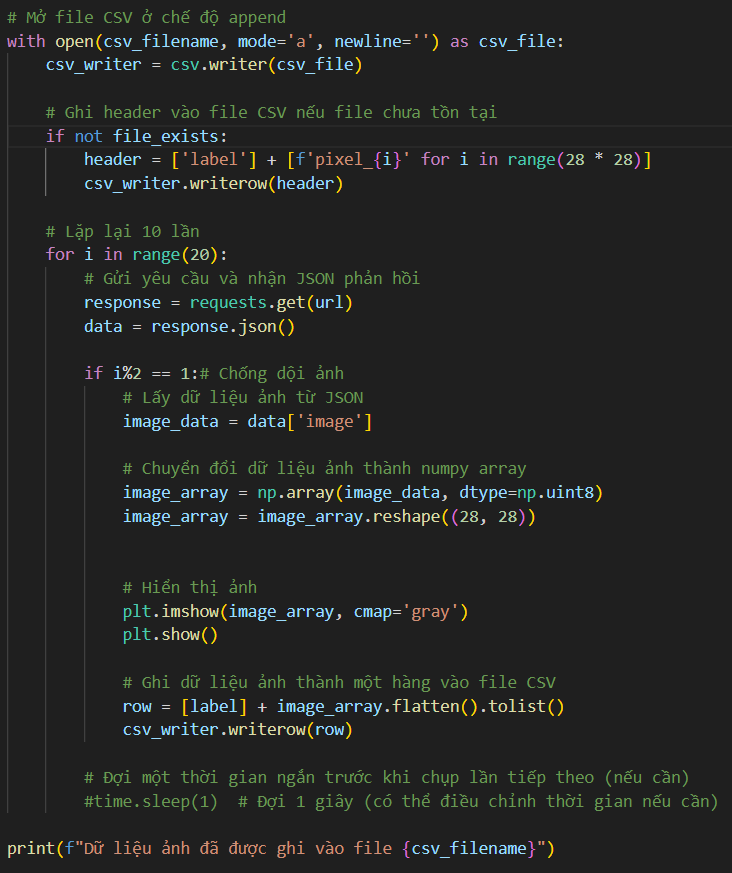
* requests: Được sử dụng để gửi yêu cầu HTTP đến server ESP32.
* numpy: Được sử dụng để xử lý dữ liệu mảng.
* matplotlib.pyplot: Được sử dụng để hiển thị ảnh.
* csv: Được sử dụng để ghi dữ liệu vào tệp CSV.
* os: Được sử dụng để kiểm tra sự tồn tại của tệp CSV.
* time: Được sử dụng để tạo độ trễ trước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

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

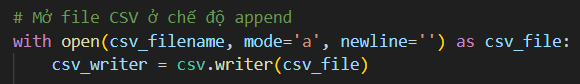


* url: URL của server ESP32 để gửi yêu cầu chụp ảnh.
* csv\_filename: Tên tệp CSV để lưu dữ liệu ảnh.
* label: Nhãn (label) của dữ liệu ảnh, có thể thay đổi tùy theo yêu cầu
* os.path.isfile(csv\_filename): Kiểm tra xem tệp CSV đã tồn tại hay không.

Vòng lặp chính:

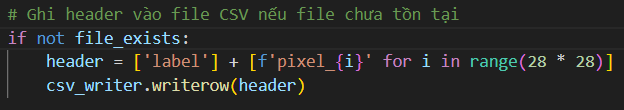


* Tạo đối tượng ghi file csv:



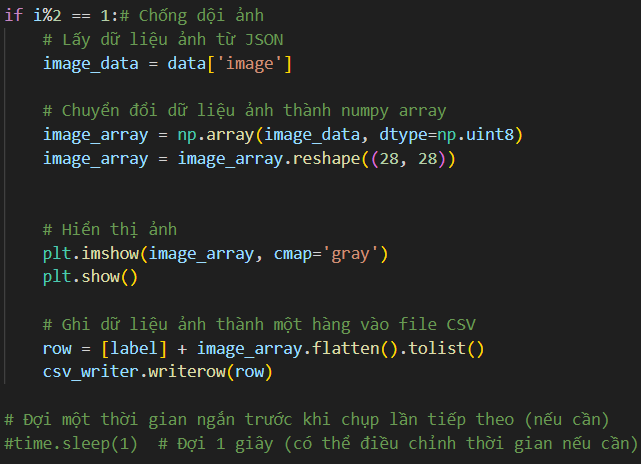
open(csv\_filename, mode='a', newline=''): Mở tệp CSV trong chế độ append để ghi dữ liệu mới vào cuối tệp.

* (Tùy chọn) Tạo header cho file csv:



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:



* 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.

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

Chương 3: Kết luận

3.1.Những điều đã làm được trong đồ án

3.2.Hướng đi tiếp theo cho đồ án