HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

HỌC PHẦN : IOT VÀ ỨNG DỤNG

BÁO CÁO GIỮA KỲ

Đề tài : Hệ thống phân loại rác thông minh

Giảng viên hướng dẫn : **Kim Ngọc Bách**
Lớp chuyên ngành : D22CNPM02
Nhóm lớp học phần : 05
Nhóm bài tập lớn : 07
Danh sách thành viên : 1. Nguyễn Trí Dũng – B22DCCN135
2. Nguyễn Nhật Thành - B22DCCN795
3. Hoàng Hải Long - B22DCCN496
4. Phạm Quang Minh - B22DCCN544

Hà Nội – 2025

Mục lục

Giới thiệu

Trong những năm gần đây, vấn đề ô nhiễm môi trường và xử lý rác thải sinh hoạt đang trở thành thách thức lớn đối với nhiều quốc gia, đặc biệt là tại các đô thị đông dân. Mỗi ngày, một lượng lớn rác thải được thải ra môi trường, trong đó phần lớn không được phân loại đúng cách, dẫn đến việc tăng chi phí xử lý, giảm hiệu quả tái chế và gây ra ô nhiễm nguồn đất, nước, không khí. Trong bối cảnh đó, việc áp dụng công nghệ thông minh nhằm tự động hóa quy trình phân loại rác là hướng đi cần thiết và mang tính thực tiễn cao.

Đề tài "Hệ thống Phân loại Rác Thông minh" được thực hiện với mục tiêu xây dựng một mô hình thùng rác tự động có khả năng nhận diện và phân loại rác thải dựa trên công nghệ trí tuệ nhân tạo (AI). Hệ thống được tích hợp camera ESP32-CAM để thu nhận hình ảnh rác thải. Sau đó, hình ảnh được xử lý và phân tích bằng mô hình học sâu (Deep Learning), cụ thể là mạng nơ-ron tích chập (Convolutional Neural Network - CNN) được huấn luyện để nhận dạng các loại rác như rác hữu cơ, rác vô cơ, nhựa, kim loại, giấy, v.v.

Khác với các hệ thống AI truyền thống chạy trên máy tính, mô hình của đề tài được tối ưu hóa và triển khai trực tiếp trên vi điều khiển ESP32 thông qua công nghệ TinyML (Tiny Machine Learning) – giúp thiết bị có khả năng nhận dạng độc lập, không phụ thuộc vào máy chủ và tiết kiệm năng lượng. Khi rác được đưa vào, hệ thống sẽ tự động xác định loại rác và điều khiển servo motor mở nắp hoặc chuyển rác đến ngăn chứa phù hợp.

Mục tiêu của đề tài là tạo ra một mô hình thử nghiệm hoàn chỉnh, chứng minh tính khả thi của việc triển khai trí tuệ nhân tạo trên thiết bị nhúng giá rẻ, tiêu thụ ít năng lượng — mở ra hướng đi mới cho các ứng dụng IoT xanh và bền vững trong tương lai.

I. Lý thuyết và công nghệ áp dụng

1. Lý thuyết

a. Internet of Things (IoT)

- Internet of Things (IoT) là một mạng lưới liên kết các thiết bị vật lý, phương tiện, đồ gia dụng và các vật phẩm khác được tích hợp các thành phần điện tử, phần mềm, cảm biến, cơ cấu chấp hành và khả năng kết nối mạng, cho phép chúng thu thập và trao đổi dữ liệu.
- Một hệ thống IoT điển hình bao gồm 3 lớp kiến trúc chính:
    + **Lớp Cảm nhận (Perception Layer):** đây là lớp vật lý, đóng vai trò là giao diện trực tiếp giữa hệ thống IoT và thế giới thực. Chức năng chính của lớp này là thu thập dữ liệu từ môi trường thông qua các cảm biến hoặc thực hiện các hành động vật lý thông qua các cơ cấu chấp hành.
    + **Lớp Mạng (Network Layer):** lớp Mạng được ví như hệ thống thần kinh, chịu trách nhiệm kết nối và truyền tải dữ liệu một cách an toàn, tin cậy giữa Lớp Cảm nhận và Lớp Ứng dụng. Nó bao gồm tất cả các công nghệ, giao thức và cơ sở hạ tầng cần thiết để dữ liệu có thể di chuyển.
    + **Lớp Ứng dụng (Application Layer):** đây là lớp cao nhất trong kiến trúc, nơi dữ liệu được xử lý để mang lại giá trị cụ thể cho người dùng cuối. Lớp này chịu trách nhiệm về toàn bộ logic nghiệp vụ của hệ thống và cung cấp các dịch vụ thông minh dựa trên dữ liệu đã được thu thập.

b. TinyML: Triển khai Trí tuệ Nhân tạo trên Vi điều khiển

- Dự án được xây dựng dựa trên mô hình **TinyML**, một lĩnh vực giao thoa giữa học máy và hệ thống nhúng. Khác với các hệ thống AI biên (Edge AI) truyền thống thường yêu cầu các máy tính bo mạch đơn (SBC) có hiệu năng cao, TinyML cho phép thực thi các thuật toán học máy trực tiếp trên các **vi điều khiển (Microcontroller - MCU)**.
- Việc áp dụng TinyML mang lại những lợi thế chiến lược cho các ứng dụng IoT:
    + **Tiêu thụ năng lượng cực thấp:** Vi điều khiển được thiết kế để hoạt động với nguồn năng lượng hạn chế, cho phép thiết bị hoạt động bằng pin trong thời gian dài.
    + **Độ trễ tối thiểu:** Toàn bộ quá trình từ thu nhận hình ảnh đến đưa ra quyết định phân loại đều diễn ra ngay trên chip, loại bỏ hoàn toàn độ trễ mạng.
    + **Bảo mật và riêng tư:** Dữ liệu hình ảnh được xử lý tại chỗ và không cần truyền ra ngoài, đảm bảo tính riêng tư cho người dùng.
    + **Chi phí thấp và kích thước nhỏ gọn:** Cho phép tích hợp trí thông minh vào những thiết bị nhỏ nhất và có giá thành phải chăng.

c. Thị giác máy tính (Computer Vision) cho bài toán phân loại

- Thị giác máy tính là một lĩnh vực của trí tuệ nhân tạo (AI) cho phép máy tính và hệ thống có thể "nhìn", "hiểu" và diễn giải thông tin từ hình ảnh kỹ thuật số, video và các loại đầu vào hình ảnh khác.
- Bài toán cốt lõi trong đề tài này là **Phân loại ảnh (Image Classification)**. Đây là nhiệm vụ gán một nhãn (label) cho một hình ảnh đầu vào từ một tập hợp các nhãn đã được định nghĩa trước.
    + Input: Một hình ảnh chứa một vật thể rác.
    + Process: Mô hình AI phân tích các đặc trưng trong ảnh (hình dạng, màu sắc, kết cấu...).
    + Output: Một nhãn dự đoán (hữu cơ hoặc vô cơ).

d. Tối ưu hóa mô hình với TensorFlow Lite for Microcontrollers

- Mạng Neural tích chập (CNN) là một lớp chuyên biệt của mạng neural nhân tạo, được thiết kế để xử lý hiệu quả dữ liệu có cấu trúc dạng lưới, tiêu biểu là hình ảnh. CNN là phương pháp tiên tiến và đạt hiệu quả cao nhất hiện nay cho các bài toán nhận dạng đối tượng và phân loại ảnh. Cấu trúc của một mạng CNN tiêu chuẩn bao gồm ba loại lớp chính:
    + **Lớp Tích chập (Convolutional Layer):** Sử dụng các bộ lọc (kernels) để thực hiện phép toán tích chập trên ảnh đầu vào, qua đó trích xuất các đặc trưng (features) ở các cấp độ khác nhau, từ các đặc trưng cơ bản như cạnh, góc, màu sắc cho đến các đặc trưng phức tạp hơn như hình dạng của vật thể.
    + **Lớp Gộp (Pooling Layer):** Thực hiện thao tác giảm chiều dữ liệu (downsampling) trên các bản đồ đặc trưng (feature maps) do lớp tích chập tạo ra. Mục đích của lớp này là giảm số lượng tham số tính toán, giúp mô hình có khả năng khái quát hóa tốt hơn và giảm nguy cơ quá khớp (overfitting).
    + **Lớp Kết nối đầy đủ (Fully Connected Layer):** Nằm ở cuối kiến trúc mạng, lớp này nhận đầu vào là một vector đặc trưng đã được làm phẳng và thực hiện nhiệm vụ phân loại cuối cùng dựa trên các đặc trưng đã được trích xuất từ các lớp trước đó.
- Để tối ưu hóa hiệu năng trên thiết bị nhúng có tài nguyên hạn chế, đề tài lựa chọn triển khai một kiến trúc CNN gọn nhẹ như MobileNet, vốn được thiết kế để cân bằng giữa độ chính xác và hiệu quả tính toán.

e. Tối ưu hóa mô hình với TensorFlow Lite for Microcontrollers

- Để triển khai một mô hình CNN lên một vi điều khiển có bộ nhớ RAM chỉ vài trăm kilobyte, cần phải sử dụng một bộ công cụ chuyên biệt là **TensorFlow**

Lite for Microcontrollers (TFLite Micro). Đây là một thư viện C++ được tối ưu hóa cao, đóng vai trò là một "bộ thông dịch" (interpreter) để chạy các mô hình AI trên các nền tảng không có hệ điều hành.

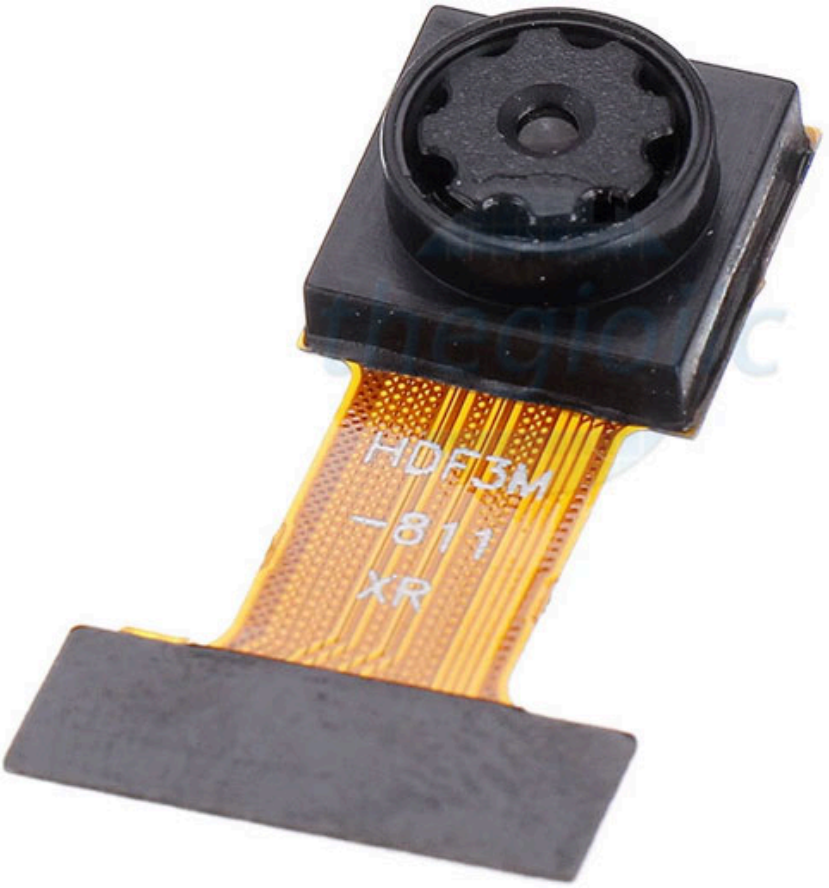
- Quy trình chuẩn bị mô hình bao gồm các bước sau:
    + **Huấn luyện (Training):** Một mô hình CNN được thiết kế và huấn luyện trên máy tính bằng framework TensorFlow/Keras với bộ dữ liệu hình ảnh rác đã được gán nhãn.
    + Chuyển đổi và Lượng tử hóa (Conversion & Quantization):
        - Mô hình đã huấn luyện được chuyển đổi sang định dạng **.tflite** bằng công cụ TensorFlow Lite Converter.
        - Trong quá trình này, kỹ thuật **Lượng tử hóa toàn bộ số nguyên (Full Integer Quantization)** được áp dụng. Kỹ thuật này chuyển đổi tất cả các trọng số và giá trị kích hoạt của mạng từ dạng số thực dấu phẩy động 32-bit (float32) sang số nguyên 8-bit (int8). Quá trình này giúp **giảm kích thước mô hình xuống 4 lần** và cho phép thực thi các phép toán trên phần cứng số nguyên của vi điều khiển nhanh hơn đáng kể, trong khi chỉ ảnh hưởng tối thiểu đến độ chính xác chung của mô hình.
    + Triển khai (Deployment): Mô hình **.tflite** sau khi lượng tử hóa được chuyển đổi thành một mảng byte C/C++ (C byte array) trong một tệp tiêu đề (.h). Tệp này sau đó được biên dịch trực tiếp cùng với mã nguồn của ứng dụng và nạp vào bộ nhớ flash của vi điều khiển.

2. Công nghệ áp dụng

a. Công nghệ phần cứng

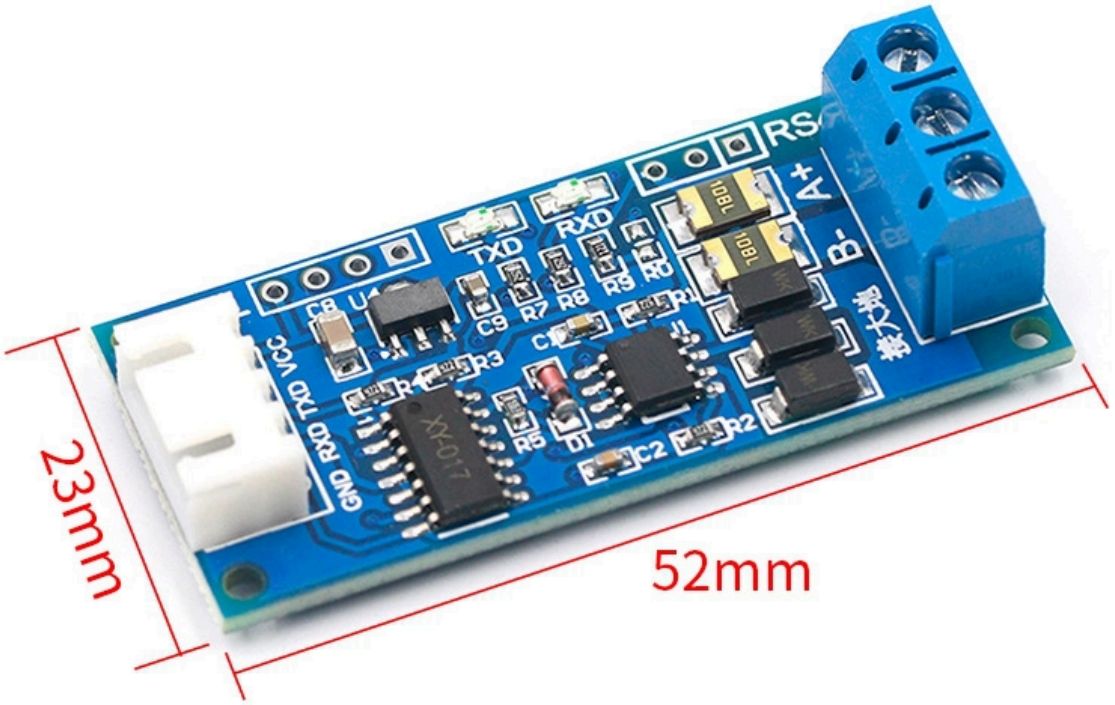
- **Vi điều khiển trung tâm và Camera: Module ESP32-CAM:** Hệ thống sử dụng module ESP32-CAM, một giải pháp tích hợp nhỏ gọn bao gồm:
+ **Vi điều khiển ESP32-S:** Đây là "bộ não" của module, chứa một **CPU lõi kép Xtensa LX6 32-bit** có thể hoạt động ở tốc độ lên tới 240MHz. Kiến trúc lõi kép cho phép một lõi chuyên xử lý các tác vụ mạng (Wi-Fi/Bluetooth) trong khi lõi còn lại tập trung hoàn toàn vào việc thực thi mô hình AI và logic ứng dụng. Chip ESP32 cũng tích hợp sẵn 520KB SRAM và các ngoại vi phần cứng phong phú.

+ **Cảm biến ảnh OV2640:** Đây là một cảm biến hình ảnh CMOS 2 megapixel được kết nối với ESP32. Nó có khả năng chụp ảnh ở nhiều độ phân giải khác nhau và hỗ trợ các định dạng đầu ra như YUV, RGB và JPEG. Trong dự án, camera được cấu hình để chụp ảnh ở độ phân giải thấp (ví dụ: 96x96 pixel) để giảm gánh nặng tính toán cho vi điều khiển.



- **Cơ cấu chấp hành: Động cơ Servo SG90:** Cơ cấu gạt phân loại được điều khiển bởi Động cơ Servo SG90. Servo này được điều khiển bằng tín hiệu Điều chế độ rộng xung (Pulse Width Modulation - PWM). Vi điều khiển ESP32 có các bộ điều khiển ngoại vi phần cứng chuyên dụng (LEDC Peripheral), cho phép tạo ra tín hiệu PWM rất ổn định và chính xác mà không cần sự can thiệp liên tục của CPU. Bằng cách thay đổi độ rộng của xung tín hiệu (thường từ 1ms đến 2ms trong một chu kỳ 20ms), trục của servo có thể được điều khiển để quay đến một góc xác định.

- **Mạch nạp và Gỡ lỗi: Giao tiếp UART:** Do ESP32-CAM không tích hợp mạch USB, việc giao tiếp với máy tính để nạp chương trình và theo dõi log gỡ lỗi được thực hiện thông qua giao thức nối tiếp UART (Universal Asynchronous Receiver-Transmitter). Một mạch chuyển đổi USB-to-UART bên ngoài (sử dụng chip CP2102 hoặc CH340) được sử dụng làm cầu nối, chuyển đổi tín hiệu từ cổng USB của máy tính thành tín hiệu mức logic TTL mà các chân TXD, RXD của ESP32 có thể hiểu được



b. Công nghệ phần mềm

Toàn bộ firmware cho thiết bị được phát triển bằng ngôn ngữ **C++**. Việc sử dụng C++ cho phép lập trình viên kiểm soát trực tiếp và hiệu quả việc cấp phát bộ nhớ, một yếu tố cực kỳ quan trọng trên các hệ thống có RAM hạn chế. **Arduino IDE** cùng với bộ công cụ **ESP32 Core** được sử dụng làm môi trường phát triển. Nền tảng này cung cấp một tập hợp các thư viện và API cấp cao, giúp trừu tượng hóa các thao tác phần cứng phức tạp, đẩy nhanh quá trình phát triển.

- **TensorFlow Lite for Microcontrollers:** Thư viện cốt lõi, cung cấp các hàm để khởi tạo bộ thông dịch, nạp mô hình từ mảng byte, cung cấp dữ liệu ảnh đầu vào và thực hiện quá trình suy luận (inference) để nhận về kết quả phân loại.
- **Thư viện Camera Driver (esp_camera.h)**: Đây là thư viện do Espressif (nhà sản xuất ESP32) cung cấp, chịu trách nhiệm cấu hình các thanh ghi của cảm biến ảnh OV2640, khởi tạo giao tiếp I2S, và thu nhận các khung hình (frames) từ camera.

- **Thư viện điều khiển Servo (ESP32Servo)**: Thư viện này đơn giản hóa việc điều khiển động cơ servo bằng cách cung cấp các hàm API trực quan như attach() để liên kết một chân GPIO với servo và write() để ra lệnh cho servo quay đến một góc cụ thể.

II. Giới thiệu: mục tiêu, phạm vi.

1. Bối cảnh và Tổng quan Dự án

Trong bối cảnh đô thị hóa nhanh chóng và sự gia tăng của rác thải sinh hoạt, vấn đề xử lý và tái chế rác đang trở thành một thách thức môi trường cấp bách. Việc phân loại rác tại nguồn được xem là giải pháp nền tảng, giúp tối ưu hóa quy trình tái chế, giảm thiểu lượng rác thải cần chôn lấp và bảo vệ môi trường. Tuy nhiên, trên thực tế, quá trình này phần lớn vẫn phụ thuộc vào ý thức thủ công của con người, dẫn đến hiệu quả không cao và thiếu đồng bộ.

Trước thực trạng đó, dự án **"Hệ thống Phân loại Rác thông minh"** được đề xuất nhằm ứng dụng các công nghệ tiên tiến của Cách mạng Công nghiệp 4.0, cụ thể là **Internet of Things (IoT)** và **Trí tuệ Nhân tạo tại biên (Edge AI)**, để tự động hóa quy trình phân loại rác cơ bản.

Dự án tập trung vào việc xây dựng một mô hình thử nghiệm (prototype) của thùng rác thông minh. Thiết bị này có khả năng sử dụng thị giác máy tính để tự nhận dạng và phân loại rác thải thành các nhóm cơ bản ngay tại thời điểm rác được bỏ vào, sau đó tự động điều khiển cơ cấu để đưa rác vào đúng ngăn chứa. Đặc biệt, dự án áp dụng hướng tiếp cận **TinyML (Tiny Machine Learning)**, cho phép triển khai mô hình AI trực tiếp trên vi điều khiển chi phí thấp, tiêu thụ ít năng lượng, chứng minh tiềm năng nhân rộng của giải pháp trong thực tế.

2. Mục tiêu

Dự án được thực hiện nhằm đạt được các mục tiêu chính sau đây:

- **Mục tiêu 1: Thiết kế và Chế tạo Phần cứng**
    - Xây dựng một mô hình cơ khí hoàn chỉnh cho thùng rác, bao gồm các ngăn chứa riêng biệt và cơ cấu cần gạt phân loại.
    - Tích hợp và kết nối các thành phần điện tử cốt lõi, bao gồm vi điều khiển trung tâm ESP32-CAM, cảm biến hình ảnh, và cơ cấu chấp hành là động cơ servo.
- **Mục tiêu 2: Xây dựng và Tối ưu hóa Mô hình AI**
    - Xây dựng một bộ dữ liệu hình ảnh gồm hai lớp rác thải chính: hữu cơ và vô cơ.
    - Huấn luyện một mô hình mạng Neural tích chập (CNN) cho bài toán phân loại ảnh.

    - Tối ưu hóa mô hình đã huấn luyện bằng TensorFlow Lite for Microcontrollers, thực hiện lượng tử hóa để giảm kích thước và phù hợp với tài nguyên hạn chế của vi điều khiển.
- **Mục tiêu 3: Phát triển Firmware Tích hợp**
    - Phát triển firmware (chương trình nhúng) bằng ngôn ngữ C++ trên nền tảng Arduino.
    - Firmware có khả năng điều khiển camera để thu nhận hình ảnh, thực thi mô hình AI để đưa ra dự đoán, và điều khiển chính xác động cơ servo dựa trên kết quả phân loại.
- **Mục tiêu 4: Kiểm thử và Đánh giá**
    - Triển khai hệ thống hoàn chỉnh trên mô hình thực tế.
    - Thực hiện các kịch bản kiểm thử để đánh giá độ chính xác phân loại và độ trễ của hệ thống trong điều kiện hoạt động mô phỏng.

3. Phạm vi

Để đảm bảo tính khả thi và tập trung vào các mục tiêu cốt lõi, dự án được giới hạn trong phạm vi sau:

a. Trong phạm vi:

- Chức năng phân loại: Hệ thống chỉ thực hiện phân loại rác thải thành hai lớp cơ bản:
    - **Rác hữu cơ:** Các loại rác dễ phân hủy sinh học (ví dụ: vỏ trái cây, rau củ thừa).
    - **Rác vô cơ:** Các loại rác khó phân hủy hoặc có thể tái chế (ví dụ: chai nhựa, vỏ lon, hộp giấy).
- Đối tượng nhận dạng: Hệ thống được thiết kế để nhận dạng từng vật thể rác riêng lẻ được đưa vào tại một thời điểm
- Xử lý tại biên: Toàn bộ quá trình xử lý hình ảnh và suy luận AI được thực hiện 100% cục bộ trên vi điều khiển ESP32-CAM.
- Phần cứng: Hệ thống là một mô hình thử nghiệm độc lập, hoạt động trong môi trường phòng thí nghiệm với điều kiện ánh sáng được kiểm soát.
- Kết quả đầu ra: Hệ thống điều khiển cần gạt vật lý để hướng rác vào một trong hai ngăn chứa tương ứng.

b. Ngoài phạm vi

- Phân loại đa lớp phức tạp: Hệ thống không phân loại chi tiết các loại rác vô cơ (ví dụ: không phân biệt giữa nhựa, kim loại, thủy tinh, giấy).
- Kết nối đám mây (Cloud): Phiên bản hiện tại không bao gồm chức năng gửi dữ liệu thống kê lên các nền tảng IoT Cloud.
- Giao diện người dùng: Hệ thống không phát triển ứng dụng di động hay giao diện web để người dùng tương tác.

- Xử lý các trường hợp phức tạp: Hệ thống không được thiết kế để xử lý rác thải dạng lỏng, rác bị che khuất, hoặc nhiều loại rác được bỏ vào cùng một lúc.
- Thương mại hóa: Mô hình không phải là một sản phẩm thương mại hoàn chỉnh, không bao gồm các yếu tố về độ bền công nghiệp, khả năng chống nước, hay tối ưu hóa cho sản xuất hàng loạt.

III. Mô tả tổng quan : Môi trường, ràng buộc và giả định

1. Môi trường

- Hoạt động trong môi trường phòng thí nghiệm / trong nhà.
- Nguồn cấp 5V – 12V DC.
- Điều kiện ánh sáng được kiểm soát, nhiệt độ 0–50°C.

2. Ràng buộc

- Độ chính xác AI ≥ 85%.
- Thời gian xử lý ≤ 2 giây.
- Servo hoạt động góc 0°–180°, quay đúng vị trí trong ±5°.
- Không mở đồng thời hai ngăn để tránh va chạm cơ khí.

3. Giả định

- Mô hình AI đã được huấn luyện sẵn (CNN → TensorFlow Lite → .tflite → mảng byte nhúng vào firmware).
- Cơ cấu gạt vật lý có khả năng di chuyển linh hoạt.
- Người dùng bỏ rác từng vật thể và không liên tục quá nhanh.

IV. Yêu cầu chức năng

1. Các chức năng chính

1.1. Phát hiện vật thể (Rác được đưa vào)

- Hệ thống sử dụng camera OV2640 tích hợp trên ESP32-CAM để giám sát vùng trước miệng thùng rác.
- Khi phát hiện có vật thể xuất hiện trong vùng quan sát (có thể dùng cảm biến IR hoặc đơn giản là chụp liên tục), hệ thống tự động chụp ảnh để xử lý.
- Ảnh được lấy mẫu ở độ phân giải thấp (khoảng 96×96 pixel) nhằm giảm tải tính toán cho vi điều khiển.

1.2. Xử lý hình ảnh và phân loại rác

- Ảnh chụp được truyền trực tiếp tới mô hình AI TinyML đã được huấn luyện trước (CNN, triển khai bằng TensorFlow Lite for Microcontrollers).
- Mô hình phân tích các đặc trưng của vật thể (màu sắc, hình dạng, kết cấu, kích thước, v.v.) và xác định loại rác thuộc một trong hai nhóm:
    - Rác hữu cơ: các vật dễ phân hủy như vỏ trái cây, rau, thức ăn thừa.
    - Rác vô cơ: các vật khó phân hủy như chai nhựa, lon nhôm, hộp giấy.

- Quá trình nhận dạng được thực hiện hoàn toàn cục bộ trên ESP32, không cần kết nối mạng.
- Kết quả phân loại được trả về dưới dạng một nhãn (label) hoặc mã số.

1.3. Điều khiển công tắc / Servo thực hiện phân loại

- Sau khi nhận được kết quả từ AI, vi điều khiển ra lệnh cho động cơ servo SG90 thông qua tín hiệu PWM (Pulse Width Modulation) để điều khiển cơ cấu gạt hoặc mở nắp ngăn rác tương ứng.
- Hoạt động cụ thể của servo:
    - Nếu kết quả là rác hữu cơ, servo quay sang trái (ví dụ 0°) để gạt vật thể vào ngăn rác hữu cơ.
    - Nếu kết quả là rác vô cơ, servo quay sang phải (ví dụ 90°) để gạt vật thể vào ngăn rác vô cơ.
    - Nếu hệ thống không xác định được loại rác, servo giữ nguyên vị trí trung lập (ví dụ 45°) và không kích hoạt cơ cấu gạt.

1.4. Tự động đóng lại / Reset vị trí servo

- Sau khi thực hiện thao tác phân loại (gạt trái hoặc phải), servo sẽ tự động quay trở lại vị trí trung lập sau một khoảng thời gian định sẵn (thường là 3–5 giây).
- Việc trở về vị trí trung tâm giúp hệ thống sẵn sàng cho lần nhận dạng tiếp theo và tránh tình trạng kẹt cơ khí.

1.5. Xử lý lỗi và tình huống không xác định

- Nếu mô hình AI không thể xác định được loại rác (ví dụ hình ảnh mờ, bị che khuất, hoặc không rõ đặc trưng), hệ thống sẽ:
    - Giữ servo ở trạng thái trung lập, không thực hiện gạt.
    - Ghi nhận lỗi hoặc số lần nhận dạng thất bại để phục vụ cho thống kê và đánh giá hiệu năng.
- Trong trường hợp lỗi phần cứng (mất tín hiệu camera, servo không phản hồi), hệ thống có thể báo lỗi qua LED hoặc log nội bộ.

1.6. Ghi log kết quả hoạt động (tuỳ chọn)

- Mỗi lần phân loại thành công, hệ thống có thể lưu thông tin vào bộ nhớ flash:
    - Thời gian thực hiện.
    - Kết quả phân loại (hữu cơ / vô cơ).
    - Thời gian xử lý hoặc số lần thất bại.
- Dữ liệu này có thể được sử dụng để đánh giá độ chính xác và hiệu suất của hệ thống trong các đợt kiểm thử.

1.7. Kiểm tra thiết bị khi khởi động (Self-Test)

Khi hệ thống được bật nguồn, firmware sẽ tự động kiểm tra tình trạng của các thành phần chính:

- Camera hoạt động bình thường.
- Mô hình AI được nạp thành công.

- Servo có phản hồi khi gửi tín hiệu PWM.

Nếu một trong các thành phần không sẵn sàng, hệ thống sẽ không tiến hành nhận dạng và có thể hiển thị cảnh báo.

2. Đặc tả luồng công việc

Tác nhân liên quan :

- Người dùng: Là người bỏ rác vào thùng.
- Camera (OV2640): Thiết bị thu nhận hình ảnh của vật thể rác.
- Bộ xử lý (ESP32-CAM): Vi điều khiển thực hiện xử lý AI và điều khiển servo.
- Mô hình AI (TinyML – CNN): Thành phần phân tích hình ảnh và xác định loại rác.
- Cơ cấu servo: Thực hiện thao tác gạt hoặc mở nắp ngăn rác tương ứng.

Tiền điều kiện :

- Hệ thống được cấp nguồn và khởi động thành công.
- Camera đã sẵn sàng chụp ảnh, mô hình AI đã được nạp vào bộ nhớ.
- Servo ở vị trí trung lập, sẵn sàng thực hiện lệnh điều khiển.

Luồng hoạt động chính :

1. Phát hiện rác được đưa vào:
   Khi người dùng bỏ rác vào vùng nhận dạng, camera OV2640 phát hiện vật thể (có thể được kích hoạt bởi cảm biến hoặc thuật toán phát hiện chuyển động).
2. Chụp và xử lý hình ảnh:
   Hệ thống tự động chụp ảnh của vật thể, chuyển ảnh về định dạng đầu vào phù hợp (ví dụ 96×96 pixel, RGB).
   Mô hình AI TinyML được chạy trực tiếp trên ESP32-CAM để phân tích đặc trưng hình ảnh.
3. Phân loại rác:
   Mô hình CNN xác định loại rác dựa trên dữ liệu học:
   - Nếu hình ảnh phù hợp với đặc trưng rác hữu cơ, trả về nhãn “hữu cơ”.
   - Nếu phù hợp với đặc trưng vô cơ, trả về nhãn “vô cơ”.
   - Nếu hình ảnh không đủ thông tin, mô hình trả về “không xác định”.
4. Điều khiển servo thực hiện gạt:
   Sau khi nhận được kết quả phân loại, ESP32 gửi tín hiệu PWM đến servo SG90 để quay đến góc tương ứng:
   - Rác hữu cơ → servo quay sang trái (0°) để gạt vật thể vào ngăn hữu cơ.
   - Rác vô cơ → servo quay sang phải (90°) để gạt vật thể vào ngăn vô cơ.
   - Nếu không xác định, servo giữ nguyên vị trí trung tâm (45°), không gạt.
5. Đóng lại vị trí ban đầu:
   Sau 3–5 giây, servo tự động quay về vị trí trung lập (45°) để sẵn sàng cho lần phân loại kế tiếp.

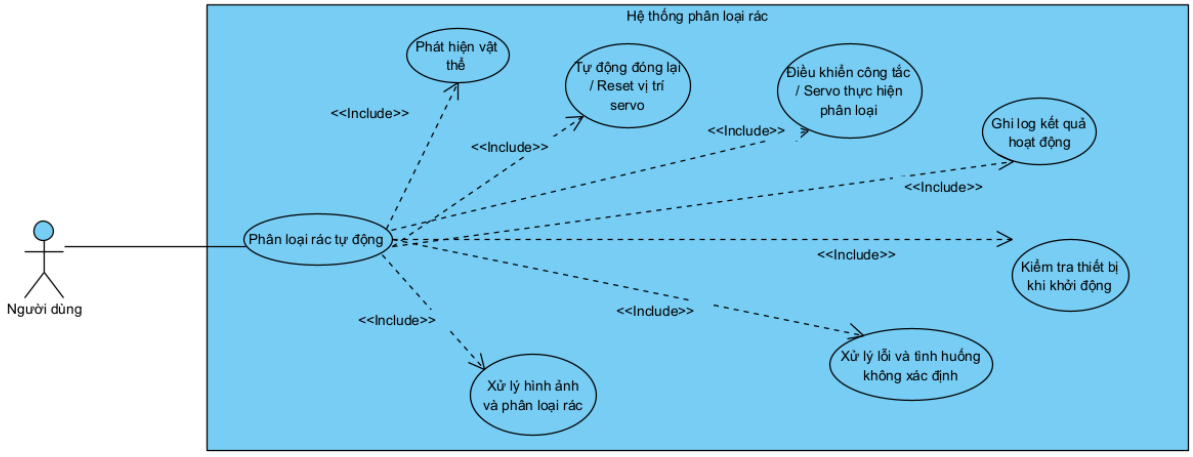
6. Lưu kết quả và hoàn tất chu trình:
   Hệ thống có thể ghi lại thông tin phân loại (thời gian, loại rác, độ tin cậy của mô hình) vào bộ nhớ trong.
   Chu trình kết thúc, hệ thống chờ vật thể tiếp theo.

Luồng phụ (Trường hợp lỗi hoặc bất thường) :

- Nếu camera không chụp được hình ảnh, hệ thống bỏ qua lần phân loại đó và báo lỗi qua log hoặc LED cảnh báo.
- Nếu mô hình AI không nhận dạng được loại rác, servo sẽ không hoạt động để tránh sai lệch cơ học.
- Nếu servo bị kẹt hoặc không phản hồi, hệ thống ngừng hoạt động và chờ kiểm tra lại.

Hậu điều kiện

- Rác được gạt đúng ngăn tương ứng (hữu cơ hoặc vô cơ).
- Servo quay về vị trí trung lập, sẵn sàng cho chu kỳ kế tiếp.
- Hệ thống lưu thông tin kết quả để phục vụ đánh giá hoặc thống kê hiệu suất.



V. Yêu cầu phi chức năng: hiệu năng, bảo mật, tin cậy, mở rộng.

1. Hiệu năng (Performance Requirements)

- Hệ thống cần đảm bảo **phân loại rác nhanh chóng và ổn định** để đáp ứng thời gian thực của ứng dụng IoT biên. Ví dụ, trong các hệ thống IoT, độ trễ (latency) và thông lượng (throughput) là các chỉ số quan trọng để đánh giá hiệu năng của thiết bị và mạng.

- Theo TinyML, mô hình AI được triển khai ngay trên vi điều khiển sẽ **giảm độ trễ đáng kể và tiêu thụ điện năng thấp**, cho phép **phản hồi gần thời gian thực.** Do đó, các yêu cầu hiệu năng bổ sung có thể bao gồm:
    + **Độ trễ xử lý:** Thời gian từ khi cảm biến chụp ảnh đến khi **kết quả phân loại được trả về nên ≤3 giây.** Với TinyML trên vi điều khiển, thời gian phản hồi này có thể thấp hơn, đảm bảo **tính kịp thời của hệ thống.**
    + **Thông lượng:** Hệ thống phải hỗ trợ **tối thiểu 20 lượt phân loại mỗi phút** (tương đương ≈0.33 phân loại/giây) mà không xảy ra nghẽn cổ chai. Yêu cầu này tương ứng với năng lực xử lý dữ liệu đủ để đáp ứng nhu cầu sử dụng bình thường, đảm bảo rằng thiết bị có thể duy trì luồng dữ liệu ổn định (không bị quá tải)
    + **Tài nguyên hệ thống:**
        - Vi điều khiển cần duy trì mức sử dụng **CPU và RAM dưới 80% trong điều kiện cao điểm**, để tránh quá tải và đảm bảo tính ổn định của các quá trình nền (AI inference, giao tiếp mạng).
        - Thêm vào đó, do thiết bị có thể chạy bằng nguồn năng lượng hạn chế (pin), yêu cầu về **tiêu thụ điện năng cũng cần được xem xét** – ví dụ, thiết bị nên tối ưu để hoạt động liên tục trong thời gian dài mà không cần sạc thường xuyên
    + **Độ chính xác mô hình AI:**
        - Mô hình phân loại rác đạt độ chính xác ≥ 90% trên tập dữ liệu kiểm thử là phù hợp nhưng cũng cần bổ sung các **chỉ tiêu về duy trì độ chính xác khi điều kiện thực tế thay đổi (ánh sáng, góc chụp)**. Có thể đặt thêm yêu cầu về **precision và recall** tối thiểu để đảm bảo chất lượng phân loại (ví dụ: precision và recall ≥ 90% cho mỗi loại rác).

2. Bảo mật (Security Requirements)

- Với hệ thống IoT **thu thập hình ảnh và dữ liệu nhạy cảm**, các biện pháp bảo mật bắt buộc phải được áp dụng toàn diện. Các yêu cầu bảo mật bổ sung bao gồm:
    + **Xác thực và phân quyền:**
        - Mỗi người dùng (đặc biệt là Admin) cần đăng nhập với chứng thực mạnh (ví dụ: xác thực hai yếu tố) và được gán **vai trò/phân quyền rõ ràng**. **Chỉ Admin mới có quyền truy cập các chức năng nhạy cảm (huấn luyện mô hình, xem thống kê).**
        - Việc quản lý thiết bị cũng cần có cơ chế xác thực chặt chẽ; mỗi thùng rác thông minh mang **một device ID duy nhất và cặp khóa (API Key, certificate)** để tránh giả mạo thiết bị

+ **Mã hóa dữ liệu:** Mọi dữ liệu truyền từ thiết bị IoT lên server (ảnh chụp và kết quả phân loại) phải được **mã hóa sử dụng giao thức an toàn như HTTPS (TLS) hoặc MQTTs**. Điều này đảm bảo tính riêng tư và toàn vẹn dữ liệu trong khi truyền qua mạng công cộng, **ngăn ngừa nghe lén hoặc sửa đổi dữ liệu.**
+ **Bảo vệ mô hình AI và dữ liệu nhạy cảm:**
    - File mô hình (.tflite) và các dữ liệu huấn luyện phải được lưu trữ ở **dạng mã hóa hoặc có kiểm soát truy cập (trên cloud hoặc thiết bị)** để tránh sao chép hay đánh cắp trí tuệ.
    - Giải pháp có thể bao gồm mã hóa nội dung file hoặc sử dụng khoá bảo mật chuyên dụng. Các chứ**ng chỉ và khóa bí mật (như khóa API của thiết bị) cũng phải được lưu trữ an toàn** (ví dụ: sử dụng secure enclave của MCU)
+ **Cập nhật và khởi động an toàn:** Đảm bảo cập nhật firmware/mô hình AI một cách an toàn (có xác thực chữ ký số) và bật chế độ khởi động tin cậy (secure boot) trên ESP32 để **ngăn chặn phần mềm độc hại can thiệp khi thiết bị khởi động.**

3. Độ tin cậy (Reliability Requirements)

- Hệ thống cần hoạt động ổn định trong điều kiện thực tế có thể không hoàn hảo (mạng không ổn định, nguồn điện gián đoạn). Một số yêu cầu độ tin cậy bổ sung:
    + **Độ chịu lỗi và khôi phục:**
        - Thiết bị IoT phải có khả năng xử lý **ngắt mạng một cách mềm dẻo.** Ví dụ, khi mất kết nối Wi-Fi, thiết bị sẽ lưu trữ tạm thời ảnh và kết quả phân loại trong bộ nhớ đệm (hoặc sơ đồ hàng đợi) và tự động gửi lại khi kết nối được khôi phục.
        - Nếu phần mềm gặp sự cố, hệ thống cần có **watchdog timer hoặc cơ chế tự khởi động lại (auto-reboot)** để đảm bảo thiết bị không bị treo lâu dài.
    + **Thời gian hoạt động (Uptime) cao:**
        - Hệ thống backend (server, database, dashboard) nên thiết kế để **đạt ít nhất 99% uptime**, bằng cách sử dụng kiến trúc phân tán và dự phòng (ví dụ: server dự phòng, cân bằng tải).
        - Các thiết bị IoT cũng nên có **thời gian hoạt động liên tục lâu dài** (ví dụ: ≥ 95% thời gian trong điều kiện chuẩn) để giảm thiểu gián đoạn.
    + **Giám sát và logging:** Tất cả lỗi và sự kiện quan trọng phải được **ghi log rõ ràng trên cả thiết bị lẫn server.** Điều này bao gồm log kết nối, log inference (phân loại), và log **lỗi phần cứng/phần mềm.** Việc này giúp

kỹ thuật viên **dễ dàng kiểm tra, khắc phục và cải thiện hệ thống theo thời gian.**

+ **Thông báo lỗi:** Khi mô hình AI hoặc thiết bị gặp lỗi (ví dụ ảnh không hợp lệ, model crash), hệ thống **phải trả về thông báo lỗi có ý nghĩa cho người quản trị**, tránh để thiết bị treo hoặc hành động sai. Ví dụ, nếu phân loại thất bại, giao diện giám sát cần hiển thị trạng thái "Lỗi phân loại" cùng nguyên nhân (nếu có)

4. Khả năng mở rộng (Scalability Requirements)

- Hệ thống IoT thường được mở rộng khi tăng số lượng thiết bị hoặc người dùng. Các yêu cầu mở rộng bổ sung gồm:
  + **Mở rộng thiết bị:**
    - Kiến trúc backend (server, database) phải hỗ trợ tự động xử lý dữ liệu từ **nhiều thùng rác thông minh mới mà không cần thay đổi cấu trúc hệ thống hiện tại.**
    - Ví dụ, khi thêm 10 hay 100 thiết bị mới, hệ thống vẫn cân bằng tải tốt, nhờ kiến trúc phân tán và queue message (ví dụ sử dụng MQTT broker hoặc REST API có khả năng scale out)
  + **Mở rộng mô hình AI:**
    - Hệ thống phải cho phép huấn luyện và cập nhật mô hình phân loại để **thêm các nhãn mới (như rác điện tử, pin) mà không làm gián đoạn dịch vụ.**
    - Điều này bao gồm khả năng lưu trữ và triển khai lại mô hình mới qua không gian mạng (OTA update) một cách liền mạch.
  + **Mở rộng người dùng:**
    - Hệ thống quản lý (dashboard) cần **hỗ trợ đa người dùng cùng lúc với các cấp quyền khác nhau (multi-user).** Ví dụ, nhiều quản trị viên có thể truy cập giám sát đồng thời mà không làm giảm hiệu năng hệ thống.
    - Điều này đòi hỏi server có **cơ chế phân chia tải hợp lý và giao diện ứng dụng phi chức năng có khả năng scale** (có thể sử dụng cơ chế load balancing, cache, v.v.)
  + **Tích hợp mở:**
    - Hệ thống có thể tích hợp với các ứng dụng hoặc dịch vụ khác qua API RESTful. Ví dụ, dữ liệu thống kê phân loại có thể được đẩy sang hệ thống quản lý môi trường hoặc dashboard bên ngoài.
    - Giao diện RESTful cần tuân thủ chuẩn API để dễ dàng mở rộng, cho phép các đối tác bên ngoài kết nối và xử lý dữ liệu một cách an toàn.

VI. Ràng buộc kỹ thuật và môi trường

1. Ràng buộc phần cứng

- **Bo mạch ESP32-CAM:**
  - **Tích hợp chip ESP32 (đôi nhân 240MHz)** với bộ nhớ trong hạn chế (khoảng 520KB SRAM, 4MB Flash) và có thêm 4MB PSRAM mở rộng.
  - Thiết bị **không có bộ xử lý dấu phẩy động (FPU)** tích hợp, chỉ xử lý số nguyên; vì vậy mọi mô hình AI phải được tối ưu (giảm độ chính xác) để phù hợp với khả năng tính toán và bộ nhớ.
- **Mạng CNN lượng tử hóa:**
  - Mô hình CNN phải được lượng tử hóa (precision 8-bit) để giảm kích thước và nhu cầu bộ nhớ.
  - Ví dụ, nghiên cứu trên ESP32-CAM đã áp dụng CNN lượng tử hóa để đạt hiệu suất cao trong giới hạn bộ nhớ
- **Cảm biến hình ảnh:**
  - Sử dụng camera OV2640 2MP, độ phân giải tối đa 1600×1200 pixel. Tuy nhiên tốc độ khung hình của camera khi truyền hình ảnh chỉ **khoảng 5–6 fps**, giới hạn khả năng phân loại thời gian thực liên tục ở tốc độ cao.
  - Thiết kế phần cứng phải đảm bảo **lấy ảnh đủ rõ và ổn định** trong môi trường phòng thí nghiệm.
- **Nguồn cấp:**
  - Hệ thống hoạt động với nguồn **5V (qua USB hoặc adapter)** và chuyển sang 3.3V cho ESP32.
  - Nguồn điện phải ổn định (trong phòng thí nghiệm có nguồn liên tục) để tránh sụt áp hoặc dao động làm gián đoạn hoạt động thiết bị.

2. Ràng buộc phần mềm

- **Thư viện TinyML:**
  - Sử dụng TensorFlow Lite cho các vi điều khiển (TensorFlow Lite Micro) để chạy mô hình CNN trên ESP32. Thư viện này được thiết kế cho phần cứng hạn chế, có thể chạy với chỉ khoảng **20KB bộ nhớ Flash và 4KB SRAM**. Tuy nhiên, phần mềm vẫn cần tinh chỉnh kỹ lưỡng để phù hợp tài nguyên thiết bị (giảm bớt tính năng không cần thiết, nén code…).
- **Mô hình và hệ điều hành:**

    + Mô hình CNN đã được huấn luyện trước và **lượng tử hóa (8-bit)** để phù hợp với giới hạn bộ nhớ. Việc phân loại (inference) được thực hiện hoàn toàn trên thiết bị (on-device), không yêu cầu kết nối với đám mây.
    + Phần mềm phát triển bằng **C/C++ (thông qua ESP-IDF hoặc Arduino IDE)**, chạy trực tiếp trên firmware đơn giản hoặc dùng FreeRTOS tối thiểu. Điều này tránh việc sử dụng các framework nặng hoặc ngôn ngữ thông dịch không phù hợp với tài nguyên giới hạn.
- **Đa nhiệm và giao tiếp:**
    + Ứng dụng có thể bao gồm giao diện web nội bộ hoặc giao tiếp WiFi để giám sát kết quả, nhưng các tác vụ này phải chạy đồng thời với quá trình chụp ảnh và suy luận. Việc này đòi hỏi quản lý đa nhiệm chặt chẽ để tránh tràn bộ đệm và nghẽn tài nguyên

3. Ràng buộc môi trường hoạt động

- **Thử nghiệm trong phòng:** Hệ thống được chạy trong **môi trường phòng học** với điều kiện ánh sáng và nhiệt độ ổn định. Không có các yếu tố bên ngoài khắc nghiệt (mưa, bụi, ánh sáng thay đổi đột ngột…), giúp chất lượng ảnh thu được nhất quán và quá trình phân loại đáng tin cậy.
- **Hoạt động cục bộ (offline):**
    + Hệ thống phân loại ảnh hoàn toàn cục bộ trên ESP32, không phụ thuộc kết nối Internet hay máy chủ đám mây. Điều này mang lại độ trễ rất thấp và bảo mật dữ liệu (ảnh) cao hơn. Kết quả có thể được truy cập qua giao diện nội bộ hoặc máy tính nối cổng USB/WiFi của ESP32 mà không cần đường truyền bên ngoài.

4. Ràng buộc tài nguyên hệ thống

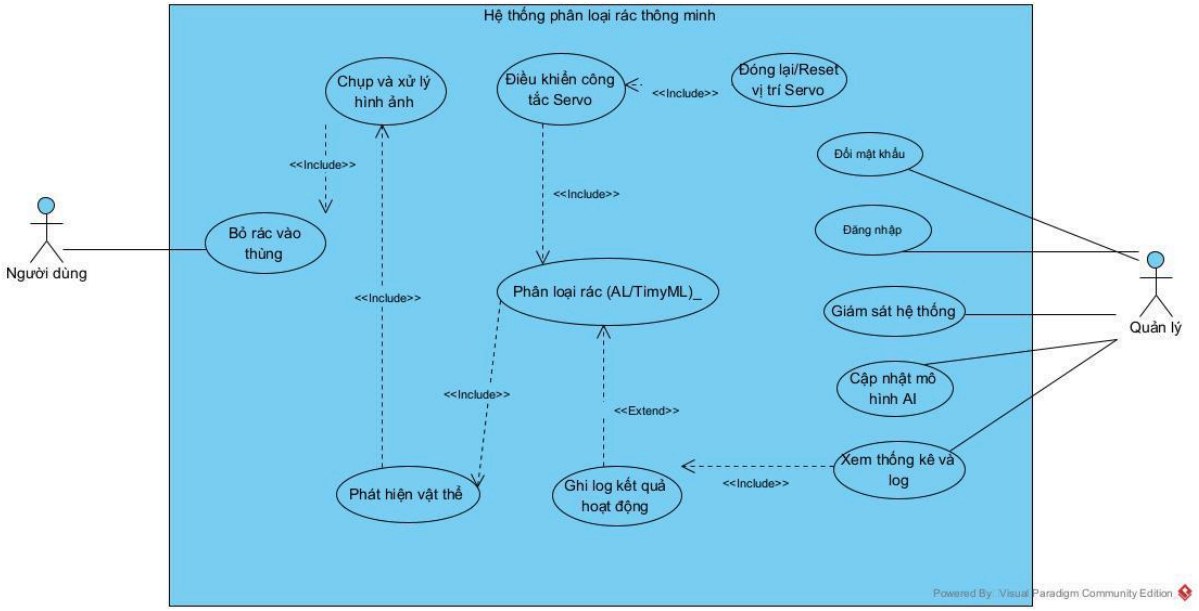
- Thời gian phát triển:
    + Dự án được thực hiện trong một học kỳ **(khoảng 2–3 tháng)**, do đó phạm vi chức năng và độ phức tạp mô hình AI được giới hạn để đảm bảo tiến độ.
- Nhân lực:
    + Đội nhóm gồm **3–5 sinh viên**, đảm nhiệm các mảng: phần cứng, phần mềm, mô hình AI, và thử nghiệm.
- Bộ nhớ:
    + **Bộ nhớ Flash 4MB** dùng để lưu chương trình điều khiển và mô hình CNN đã lượng tử hóa. **SRAM nội bộ ~520KB** (thực tế khả dụng khoảng dưới 500KB) dùng cho bộ đệm, biến trung gian và ngăn xếp chạy chương trình. Mô hình và thuật toán phải nhỏ hơn giới hạn này; việc cấp phát bộ nhớ động cần cẩn trọng để tránh tràn vùng nhớ.

+ Nếu cần không gian bổ sung, có thể sử dụng **PSRAM 4MB tích hợp**, nhưng do PSRAM chậm và có độ trễ cao, chủ yếu ưu tiên lưu dữ liệu hình ảnh, còn mô hình nên để trong bộ nhớ chính.

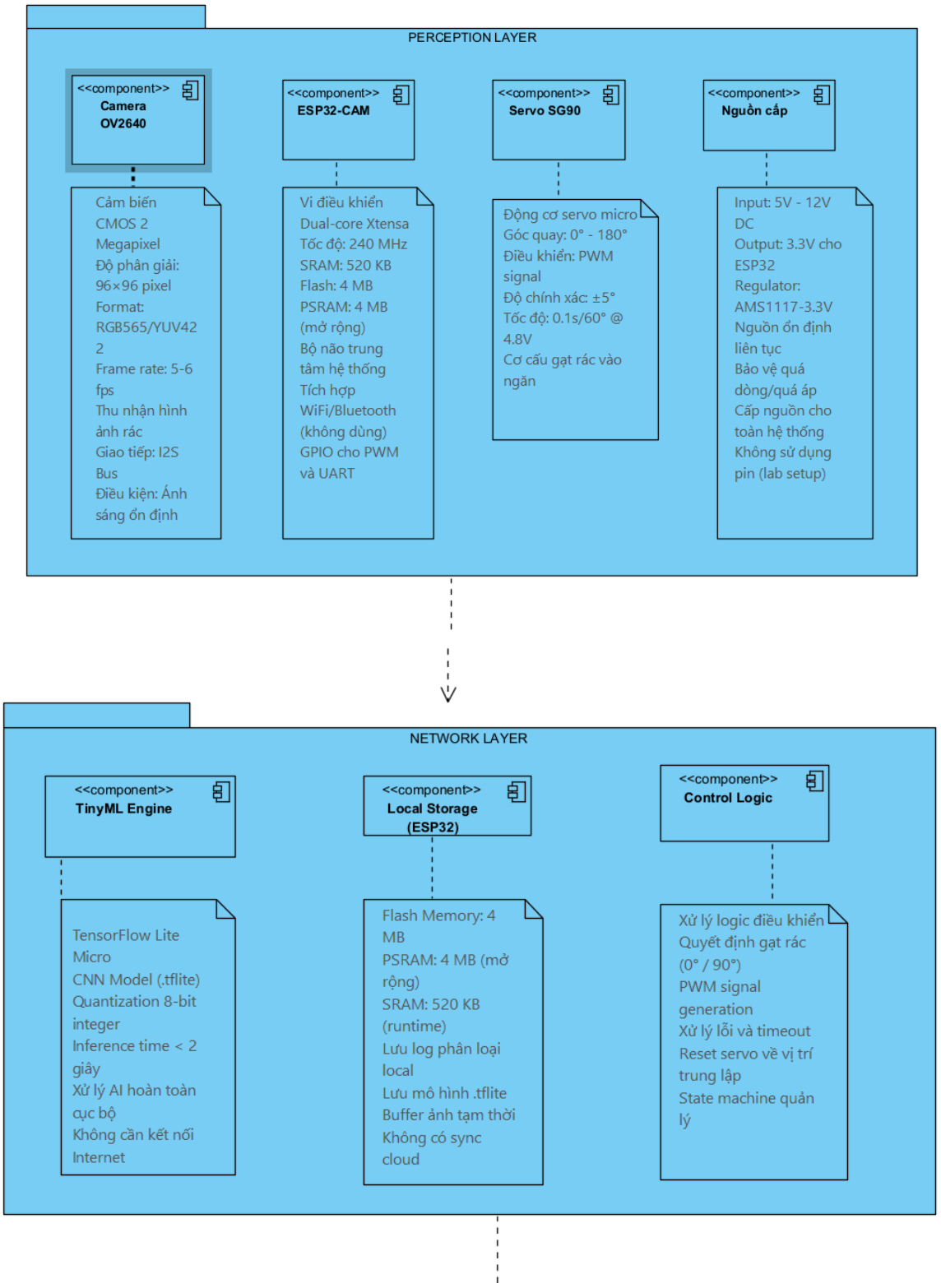
- **Tính toán:**
  + ESP32 dùng bộ xử lý **Xtensa dual-core 240MHz (không có FPU)** nên tính toán CNN chủ yếu trên số nguyên. Cần tối ưu hóa việc thực thi (đơn luồng hoặc đơn giản hóa mạng) để đáp ứng thời gian thực.
  + Hệ thống phải điều phối luồng dữ liệu và các tác vụ (chụp ảnh, chạy CNN, cập nhật giao diện) một cách hợp lý, đảm bảo xử lý suy luận nhanh và không bị nghẽn do tranh chấp tài nguyên.
- **Công suất và phần cứng phụ trợ:** Mặc dù hệ thống có nguồn ổn định trong phòng, vẫn cần quan tâm đến tiêu thụ điện và tản nhiệt của ESP32 khi chạy liên tục. Ngoài ra, các linh kiện phụ trợ như **kết nối USB, module WiFi hay đèn LED** cũng chiếm một phần nhỏ tài nguyên I/O, cần tính toán để không ảnh hưởng quá nhiều đến hiệu năng tổng thể.

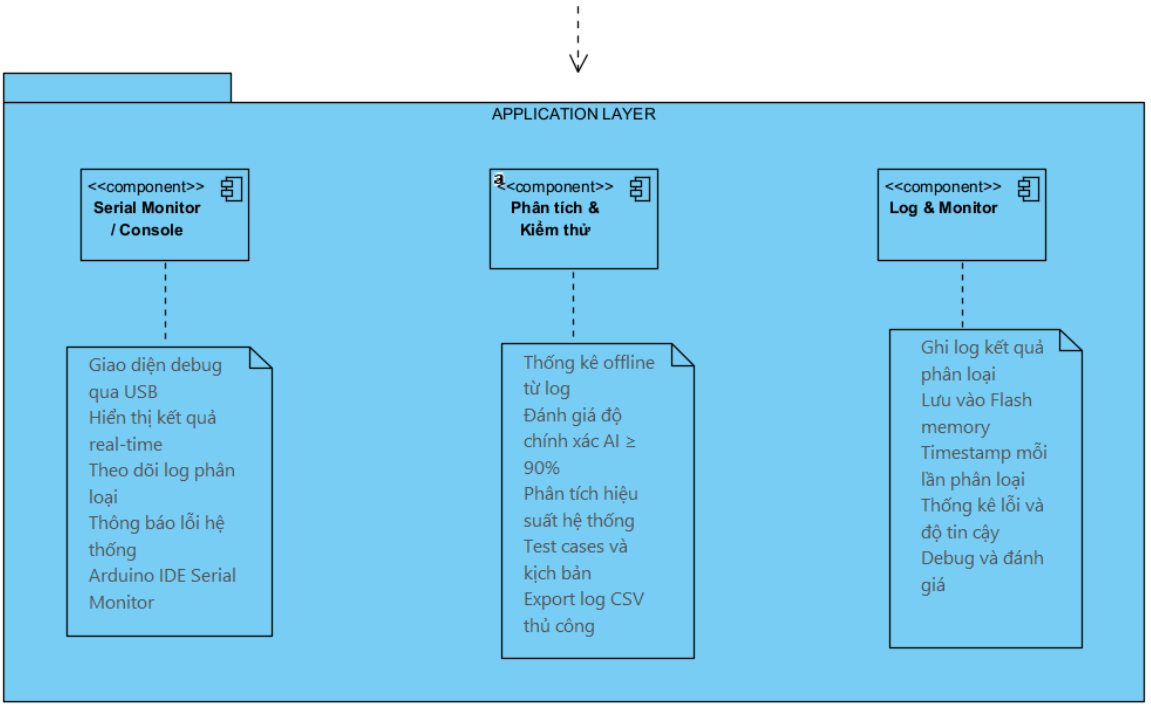
VII. Mô hình yêu cầu

1. Biểu đồ Usecase toàn hệ thống :



2. Biểu đồ mô hình 3 lớp IoT (Perception – Network – Application) :




APPLICATION LAYER
<<component>>
Serial Monitor
/ Console
Giao diện debug qua USB
Hiển thị kết quả real-time
Theo dõi log phân loại
Thông báo lỗi hệ thống
Arduino IDE Serial Monitor
<<component>>
Phân tích & Kiểm thử
Thống kê offline từ log
Đánh giá độ chính xác AI ≥ 90%
Phân tích hiệu suất hệ thống
Test cases và kịch bản
Export log CSV thủ công
<<component>>
Log & Monitor
Ghi log kết quả phân loại
Lưu vào Flash memory
Timestamp mỗi lần phân loại
Thống kê lỗi và độ tin cậy
Debug và đánh giá

Tài liệu tham khảo

1. TensorFlow Lite for Microcontrollers - v4.1 - Gecko Platform API Documentation Silicon Labs
2. Microsoft Word - Bendable EL Wire
3. What is TinyML? Tiny Machine Learning - GeeksforGeeks
4. Giới thiệu module ESP32 và hướng dẫn cài trình biên dich trên Arduino Ide. | Công đồng Arduino Việt Nam
5. IoT and Garbage Monitoring System - GeeksforGeeks