Note code:

- Khai báo thư viện và chọn model cam

#include <final\_test\_inferencing.h> // mô hình Edge Impulse (C++)

#include "edge-impulse-sdk/dsp/image/image.hpp" // tiện ích xử lý ảnh EI

#include "esp\_camera.h" // driver camera OV2640

#include <WiFi.h>, <WebServer.h> // WiFi + HTTP server

#include <WiFiClient.h> // client socket cho streaming

- Hằng số và cấu hình

#define EI\_CAMERA\_RAW\_FRAME\_BUFFER\_COLS 320

#define EI\_CAMERA\_RAW\_FRAME\_BUFFER\_ROWS 240

#define EI\_CAMERA\_FRAME\_BYTE\_SIZE 3 // RGB888 → 3 bytes/pixel

// 320x240x3 = 230,400 bytes cho 1 khung--> suy luận

// camera init

static camera\_config\_t camera\_config = {

    .pin\_pwdn = PWDN\_GPIO\_NUM,

    .pin\_reset = RESET\_GPIO\_NUM,

    .pin\_xclk = XCLK\_GPIO\_NUM,

    .pin\_sscb\_sda = SIOD\_GPIO\_NUM,

    .pin\_sscb\_scl = SIOC\_GPIO\_NUM,

    .pin\_d7 = Y9\_GPIO\_NUM,

    .pin\_d6 = Y8\_GPIO\_NUM,

    .pin\_d5 = Y7\_GPIO\_NUM,

    .pin\_d4 = Y6\_GPIO\_NUM,

    .pin\_d3 = Y5\_GPIO\_NUM,

    .pin\_d2 = Y4\_GPIO\_NUM,

    .pin\_d1 = Y3\_GPIO\_NUM,

    .pin\_d0 = Y2\_GPIO\_NUM,

    .pin\_vsync = VSYNC\_GPIO\_NUM,

    .pin\_href = HREF\_GPIO\_NUM,

    .pin\_pclk = PCLK\_GPIO\_NUM,

    .xclk\_freq\_hz = 20000000,

    .ledc\_timer = LEDC\_TIMER\_0,

    .ledc\_channel = LEDC\_CHANNEL\_0,

    .pixel\_format = PIXFORMAT\_JPEG, // Dùng jpeg để nhẹ băng thông khi streaming

    .frame\_size = FRAMESIZE\_QVGA,

    .jpeg\_quality = 12,

    .fb\_count = 1,

    .fb\_location = CAMERA\_FB\_IN\_PSRAM,

    .grab\_mode = CAMERA\_GRAB\_WHEN\_EMPTY,

};

- hàm setup: kết nối uart hiển thị qua serial monitor; khởi tạo cam; kết nối wifi;

void setup() {

Serial.begin(115200);

while (!Serial); // Chờ serial kết nối

Serial.println("Edge Impulse Person Counter Demo with Video Stream");

// Khởi tạo camera

if (ei\_camera\_init() == false) {

Serial.println("Failed to initialize Camera!");

while(1); // Dừng hẳn nếu lỗi camera

}

Serial.println("Camera initialized successfully");

// Kết nối WiFi

WiFi.begin(ssid, password);

Serial.print("Connecting to WiFi");

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println();

Serial.print("Connected! IP Address: ");

Serial.println(WiFi.localIP());

// Khởi tạo web server

server.on("/", handleRoot);

server.on("/stream", handleStream);

server.on("/count", handleCount);

server.begin();

Serial.println("HTTP server started");

Serial.println("\nStarting continuous inference in 2 seconds...");

delay(2000);

}

- hàm loop(): xử lý từng bước --> xử lý web --> chụp ảnh, xử lý ảnh --> tạo signal edge impulse--> chạy mô hình và hiển thị kết quả --> in kết quả, cập nhật số người

Lưu ý: không cập nhật trực tiếp trên giao diện stream mà xem trên monitor

void loop() {

server.handleClient(); // Xử lý request web

if (ei\_sleep(5) != EI\_IMPULSE\_OK) {

return;

}

// Cấp phát buffer cho ảnh RGB888

snapshot\_buf = (uint8\_t\*)malloc(EI\_CAMERA\_RAW\_FRAME\_BUFFER\_COLS \*

EI\_CAMERA\_RAW\_FRAME\_BUFFER\_ROWS \*

EI\_CAMERA\_FRAME\_BYTE\_SIZE);

if (snapshot\_buf == nullptr) {

Serial.println("ERR: Failed to allocate snapshot buffer!");

return;

}

// Tạo tín hiệu cho mô hình EI

ei::signal\_t signal;

signal.total\_length = EI\_CLASSIFIER\_INPUT\_WIDTH \* EI\_CLASSIFIER\_INPUT\_HEIGHT;

signal.get\_data = &ei\_camera\_get\_data;

// Chụp ảnh

if (!ei\_camera\_capture(EI\_CLASSIFIER\_INPUT\_WIDTH, EI\_CLASSIFIER\_INPUT\_HEIGHT, snapshot\_buf)) {

Serial.println("Failed to capture image");

free(snapshot\_buf);

snapshot\_buf = nullptr;

return;

}

// Chạy suy luận

ei\_impulse\_result\_t result = { 0 };

EI\_IMPULSE\_ERROR err = run\_classifier(&signal, &result, debug\_nn);

if (err != EI\_IMPULSE\_OK) {

Serial.printf("ERR: Failed to run classifier (%d)\n", err);

free(snapshot\_buf);

snapshot\_buf = nullptr;

return;

}

// In kết quả

printDetectionResults(&result);

// Đếm người

countPeople(&result);

free(snapshot\_buf);

snapshot\_buf = nullptr;

}

-hàm countpeople(): chạy 2s mỗi lần --> duyệt bb --> nếu phát hiện nhãn person với độ tin cậy > 0. 8 thì đếm --> cập nhật biến personCount và in

void countPeople(ei\_impulse\_result\_t \*result) {

unsigned long currentTime = millis();

if (currentTime - lastDetectionTime >= detectionInterval) {

int currentCount = 0;

for (uint32\_t i = 0; i < result->bounding\_boxes\_count; i++) {

ei\_impulse\_result\_bounding\_box\_t bb = result->bounding\_boxes[i];

if (bb.value >= 0.80 &&

strcmp(bb.label, "person") == 0 &&

bb.width >= 15 &&

bb.height >= 15) {

currentCount++;

}

}

if (currentCount > 0) {

personCount = currentCount;

Serial.printf("People detected: %d\n", personCount);

lastDetectionTime = currentTime;

}

}

}

-hàm printDêtectionresults():

In thời gian xử lý.

Nếu là object detection → in bounding boxes có confidence ≥ 0.5.

Nếu là classification → in kết quả > 0.1.

Nếu có anomaly → in ra khi > 0.3.

void printDetectionResults(ei\_impulse\_result\_t \*result) {

Serial.printf("Timing - DSP: %d ms, Classification: %d ms, Anomaly: %d ms\n",

result->timing.dsp, result->timing.classification, result->timing.anomaly);

#if EI\_CLASSIFIER\_OBJECT\_DETECTION == 1

Serial.println("Object detection bounding boxes:");

for (uint32\_t i = 0; i < result->bounding\_boxes\_count; i++) {

ei\_impulse\_result\_bounding\_box\_t bb = result->bounding\_boxes[i];

if (bb.value < 0.5) continue;

Serial.printf(" %s (%.2f) [ x: %u, y: %u, width: %u, height: %u ]\n",

bb.label, bb.value, bb.x, bb.y, bb.width, bb.height);

}

#else

Serial.println("Classification results:");

for (uint16\_t i = 0; i < EI\_CLASSIFIER\_LABEL\_COUNT; i++) {

if (result->classification[i].value > 0.1) {

Serial.printf(" %s: %.5f\n",

ei\_classifier\_inferencing\_categories[i],

result->classification[i].value);

}

}

#endif

#if EI\_CLASSIFIER\_HAS\_ANOMALY

if (result->anomaly > 0.3) {

Serial.printf("Anomaly score: %.3f\n", result->anomaly);

}

#endif

}

- handleRoot() --> trả về html giao diện

void handleRoot() {

server.send(200, "text/html", sendHTML());

}

- handlestream() --> stream

void handleStream() {

WiFiClient client = server.client();

String response = "HTTP/1.1 200 OK\r\n";

response += "Content-Type: multipart/x-mixed-replace; boundary=frame\r\n\r\n";

server.sendContent(response);

while (true) {

if (!client.connected()) break;

camera\_fb\_t \*fb = esp\_camera\_fb\_get();

if (!fb) {

Serial.println("Camera capture failed");

continue;

}

response = "--frame\r\n";

response += "Content-Type: image/jpeg\r\n\r\n";

server.sendContent(response);

client.write(fb->buf, fb->len);

server.sendContent("\r\n");

esp\_camera\_fb\_return(fb);

delay(100); // khoảng 10 fps

}

}

- sendHTML() --> hiển thị giao diện web

String sendHTML() {

String ptr = "<!DOCTYPE html> <html>\n";

ptr += "<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";

ptr += "<title>ESP32-CAM Video Stream</title>\n";

ptr += "<style>body { margin: 0; background-color: #000; color: #fff;}</style>\n";

ptr += "</head>\n<body>\n";

ptr += "<h1>ESP32-CAM Video Stream</h1>\n";

ptr += "<img src=\"/stream\" style=\"width:100%;max-width:640px;\">\n";

//ptr += "<h2>Person Count: <span id=\"count\">" + String(personCount) + "</span></h2>\n";

ptr += "<script>\n";

ptr += "setInterval(function() {\n";

ptr += " fetch('/count')\n";

ptr += " .then(response => response.json())\n";

ptr += " .then(data => {\n";

ptr += " document.getElementById('count').textContent = data.count;\n";

ptr += " });\n";

ptr += "}, 2000);\n";

ptr += "</script>\n</body>\n</html>\n";

return ptr;

}

- ei\_camera\_init() --> khởi tạo cam

bool ei\_camera\_init(void) {

if (is\_initialised) return true;

esp\_err\_t err = esp\_camera\_init(&camera\_config);

if (err != ESP\_OK) {

Serial.printf("Camera init failed with error 0x%x\n", err);

return false;

}

sensor\_t \*s = esp\_camera\_sensor\_get();

if (s->id.PID == OV3660\_PID) {

s->set\_vflip(s, 1);

s->set\_brightness(s, 1);

s->set\_saturation(s, 0);

}

is\_initialised = true;

return true;

}

-hàm ei\_camera\_capture() --> lấy ảnh jpeg từ cam sau đó xử lý (resize/crop) chuyển sang rgb888

bool ei\_camera\_capture(uint32\_t img\_width, uint32\_t img\_height, uint8\_t \*out\_buf) {

if (!is\_initialised) {

Serial.println("Camera not initialized");

return false;

}

camera\_fb\_t \*fb = esp\_camera\_fb\_get();

if (!fb) {

Serial.println("Camera capture failed");

return false;

}

bool converted = fmt2rgb888(fb->buf, fb->len, PIXFORMAT\_JPEG, out\_buf);

esp\_camera\_fb\_return(fb);

if (!converted) {

Serial.println("RGB conversion failed");

return false;

}

if (img\_width != EI\_CAMERA\_RAW\_FRAME\_BUFFER\_COLS ||

img\_height != EI\_CAMERA\_RAW\_FRAME\_BUFFER\_ROWS) {

ei::image::processing::crop\_and\_interpolate\_rgb888(

out\_buf,

EI\_CAMERA\_RAW\_FRAME\_BUFFER\_COLS,

EI\_CAMERA\_RAW\_FRAME\_BUFFER\_ROWS,

out\_buf,

img\_width,

img\_height);

}

return true;

}

- hàm ei\_camera\_get\_data() --> call edge impulse để đọc pixel (BGR --> RGB) --> đóng gói thành 24bit trong float --> đưa vô mô hình suy luận

static int ei\_camera\_get\_data(size\_t offset, size\_t length, float \*out\_ptr) {

if (snapshot\_buf == nullptr) {

return -1;

}

size\_t pixel\_ix = offset \* 3;

size\_t out\_ptr\_ix = 0;

while (length > 0) {

out\_ptr[out\_ptr\_ix] = (snapshot\_buf[pixel\_ix + 2] << 16) +

(snapshot\_buf[pixel\_ix + 1] << 8) +

snapshot\_buf[pixel\_ix];

out\_ptr\_ix++;

pixel\_ix += 3;

length--;

}

return 0;

}

Pipeline hoạtt động:

setup()

- Mở Serial

- Khởi tạo camera

- Kết nối WiFi

- Bật WebServer

|

v

loop()

1. server.handle()

(xử lý request)

2. Chụp ảnh

-> ei\_camera\_

capture()

3. Chạy mô hình

Edge Impulse

-> run\_classifier

4. In kết quả

-> printDetection

Results()

5. Đếm người

-> countPeople()

6. Giải phóng bộ nhớ

Flow xử lý của camera

↓ esp\_camera\_fb\_get() [JPEG]

↓ fmt2rgb888() [RGB888]

↓ ei\_camera\_capture()

↓ ei\_camera\_get\_data() [callback]

↓ Edge Impulse model

↓ run\_classifier()

↓ Kết quả bounding boxes

↓ countPeople() → personCount

