**Báo cáo về phần AI - NCKH**

# Lý thuyết

**Mục tiêu**: Dự đoán sức khỏe của bệnh nhân dựa trên các dấu hiệu sinh tồn

**Dataset**: [Human Vital Sign Dataset](https://www.kaggle.com/datasets/nasirayub2/human-vital-sign-dataset/data)

**Attributes**

| **Attribute** | **Description** | **Type** | **Range / Format** | **Example** |
| --- | --- | --- | --- | --- |
| Patient ID | A unique identifier assigned to each patient. | Integer | — | 1, 2, 3 |
| Heart Rate | The number of heartbeats per minute. | Integer | 60–100 bpm | 72, 85, 90 |
| Respiratory Rate | The number of breaths taken per minute. | Integer | 12–20 breaths/min | 16, 18, 15 |
| Timestamp | The exact time at which the vital signs were recorded. | Datetime | YYYY-MM-DD HH:MM | 2023-07-19 10:15:30 |
| Body Temperature | The body temperature measured in degrees Celsius. | Float | 36.0–37.5°C | 36.7, 37.0, 36.5 |
| Oxygen Saturation | The percentage of oxygen-bound hemoglobin in the blood. | Float | 95–100% | 98.5, 97.2, 99.1 |
| Systolic Blood Pressure | The pressure in the arteries when the heart beats (systolic pressure). | Integer | 110–140 mmHg | 120, 130, 115 |
| Diastolic Blood Pressure | The pressure in the arteries when the heart rests between beats. | Integer | 70–90 mmHg | 80, 75, 85 |
| Age | The age of the patient. | Integer | 18–90 years | 25, 45, 60 |
| Gender | The gender of the patient. | Categorical | Male, Female | Male, Female |
| Weight (kg) | The weight of the patient in kilograms. | Float | 50–100 kg | 70.5, 80.3, 65.2 |
| Height (m) | The height of the patient in meters. | Float | 1.5–2.0 m | 1.75, 1.68, 1.82 |

**Derived Features**

| **Derived Feature** | **Description** | **Type** | **Formula** | **Example** |
| --- | --- | --- | --- | --- |
| Derived\_HRV | A measure of the variation in time between heartbeats. | Float | HRV = Standard Deviation of Heart Rate / Mean Heart Rate | 0.10, 0.12, 0.08 |
| Derived\_Pulse\_Pressure | The difference between systolic and diastolic blood pressure. | Integer | PP = Systolic Blood Pressure − Diastolic Blood Pressure | 40, 45, 30 |
| Derived\_BMI | A measure of body fat based on weight and height. | Float | BMI = Weight (kg) / (Height (m))² | 22.8, 25.4, 20.3 |
| Derived\_MAP | An average blood pressure during a single cardiac cycle. | Float | MAP = Diastolic BP + 1/3 × (Systolic BP − Diastolic BP) | 93.3, 100.0, 88.7 |

**Target Feature**

| **Target Feature** | **Description** | **Type** | **Categories** | **Example** |
| --- | --- | --- | --- | --- |
| Risk Category | Classification of patients into "High Risk" or "Low Risk". | Categorical | High Risk, Low Risk | High Risk, Low Risk |

=====================================================================

**Input**: Các chỉ số sinh học (HR, Body Temperature, Oxygen Saturation, Age, Gender, Weight, Height (7 feature)

**Output**: Risk Category (Low/High Risk)

=====================================================================

# Thực hiện

**[Done]** **Model Development** (tham khảo code tại Github) -> Save model thành 3 file:

* human\_vital\_signs\_model.pth : Trọng số của mô hình PyTorch
* scaler.pkl : Đối tượng chuẩn hóa dữ liệu đầu vào
* label\_encoder.pkl : Mã hóa nhãn (label encoder)

**[In Progress] Model Inference**

Điều kiện: *ESP-IDF, ESP-DL*

1. **Model quantization**: bước này để giảm kích thước mô hình (tham khảo code tại Github). Sau khi chạy code export ra 3 file:

* .espdl: Tệp nhị phân mô hình ESPDL, có thể dùng trực tiếp để suy luận (inference) trên chip.
* .info: Tệp văn bản mô hình ESPDL, dùng để gỡ lỗi và kiểm tra xem mô hình .espdl có được xuất đúng không. Tệp này chứa thông tin về cấu trúc mô hình, trọng số đã lượng tử hóa, dữ liệu đầu vào/đầu ra kiểm thử và các thông tin khác.
* .json: Tệp thông tin lượng tử hóa, dùng để lưu và tải thông tin lượng tử hóa

1. **Load & test & profile model:** bước này để tải, kiểm tra và phân tích hiệu năng mô hình

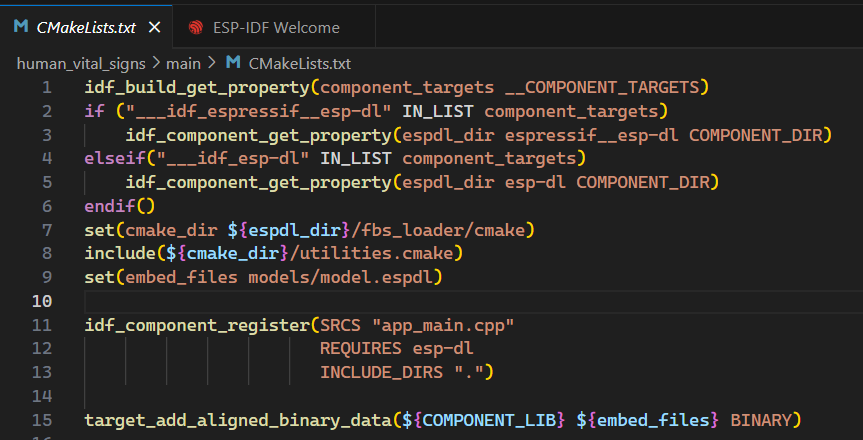
**2.1 Load model**

Ở bước này chọn cách load mô hình từ **rodata**. Load model từ .rodata nghĩa là nạp mô hình từ vùng bộ nhớ chỉ đọc (read-only data section) trên vi điều khiển, thay vì từ thẻ nhớ hoặc file hệ thống.

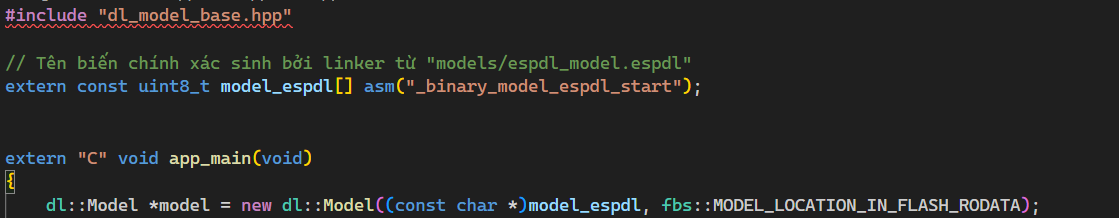
**Vì sao dùng .rodata?**

* Tốc độ truy xuất nhanh: vì mô hình nằm sẵn trong flash
* Không cần hệ thống file: Tiện cho hệ thống nhúng không có SPIFFS, SD
* An toàn, không bị ghi đè khi chạy: Vì chỉ đọc.

*1. Thêm file model vào CMakeLists.txt*



*2. Load model trong chương trình C/C++:*



**2.2 Test model**

Dòng lệnh kiểm tra: ESP\_ERROR\_CHECK(model->test());

Dòng lệnh này sẽ:

* Chạy kiểm tra inference ngay trên chip, sử dụng input/output đã lưu sẵn trong .espdl
* So sánh output thật sự của mô hình với output kỳ vọng
* Trả về lỗi (nếu có sai lệch quá mức cho phép)
* Nếu model->test() thành công, nghĩa là mô hình hoạt động đúng khi nhúng vào thiết bị.

**2.3 Profile model (Hiệu năng model)**

1. *Phân tích mức sử dụng bộ nhớ: model->profile\_memory();*

Kết quả sẽ liệt kê các thành phần bộ nhớ như sau:

| **Tên** | **Giải thích** |
| --- | --- |
| fbs\_model.parameter | **Mô hình flatbuffers**: bao gồm toàn bộ cấu trúc mô hình, các tham số, dữ liệu test input/output, shape, và kiến trúc. |
| parameter\_copy | **Bản sao của tham số mô hình**: khi mô hình nằm trong FLASH (rodata), các tham số được **copy vào PSRAM hoặc RAM nội** để tăng tốc độ suy luận. |
| variable | **Vùng nhớ động (runtime)**: cấp phát để chứa input/output và các giá trị trung gian trong quá trình inference. |
| others | Các vùng nhớ nhỏ khác: như biến thành viên, căn chỉnh bộ nhớ (alignment) khi dùng heap\_caps\_aligned\_alloc. Kích thước thường rất nhỏ. |

1. *Phân tích độ trễ mô hình: model->profile\_module()*

In ra thời gian thực thi (latency) của từng module (layer) trong mô hình theo thứ tự phụ thuộc.

*app\_main.cpp:*

#include "dl\_model\_base.hpp"

// Tên biến chính xác sinh bởi linker từ "models/espdl\_model.espdl"

extern const uint8\_t model\_espdl[] asm("\_binary\_model\_espdl\_start");

extern "C" void app\_main(void)

{

dl::Model \*model = new dl::Model((const char \*)model\_espdl, fbs::MODEL\_LOCATION\_IN\_FLASH\_RODATA);

// Test whether on-board model inference is correct

ESP\_ERROR\_CHECK(model->test());

// Profile model memory usage

model->profile();

// Profile model inference latency

model->profile(true);

delete model;

}

Full code *CMakeLists.txt* trong mục main:

idf\_build\_get\_property(component\_targets \_\_COMPONENT\_TARGETS)

if ("\_\_\_idf\_espressif\_\_esp-dl" IN\_LIST component\_targets)

idf\_component\_get\_property(espdl\_dir espressif\_\_esp-dl COMPONENT\_DIR)

elseif("\_\_\_idf\_esp-dl" IN\_LIST component\_targets)

idf\_component\_get\_property(espdl\_dir esp-dl COMPONENT\_DIR)

endif()

set(cmake\_dir ${espdl\_dir}/fbs\_loader/cmake)

include(${cmake\_dir}/utilities.cmake)

set(embed\_files models/model.espdl)

idf\_component\_register(SRCS "app\_main.cpp"

REQUIRES esp-dl

INCLUDE\_DIRS ".")

target\_add\_aligned\_binary\_data(${COMPONENT\_LIB} ${embed\_files} BINARY)

*partitions.csv:*

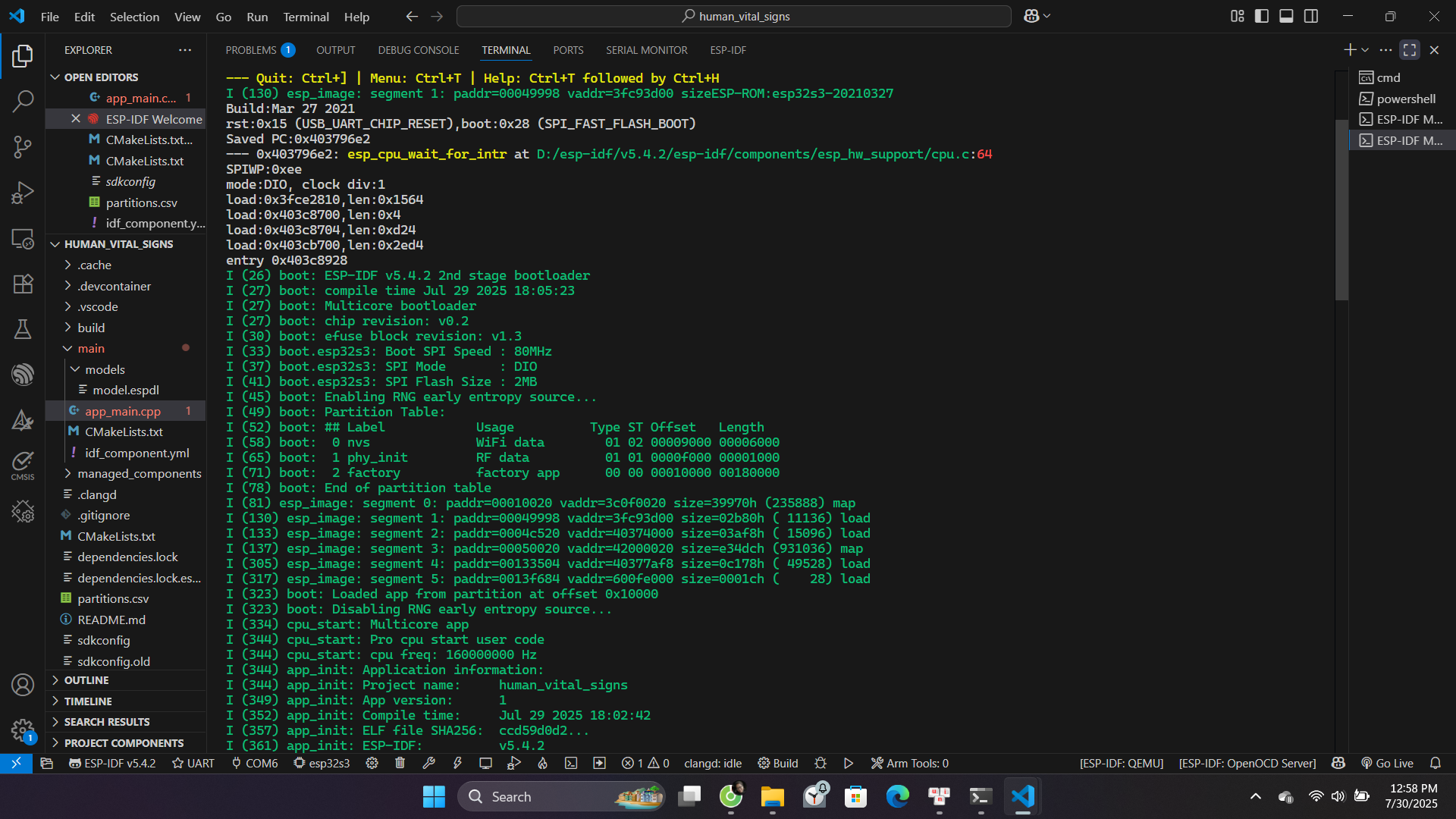
# Name, Type, SubType, Offset, Size

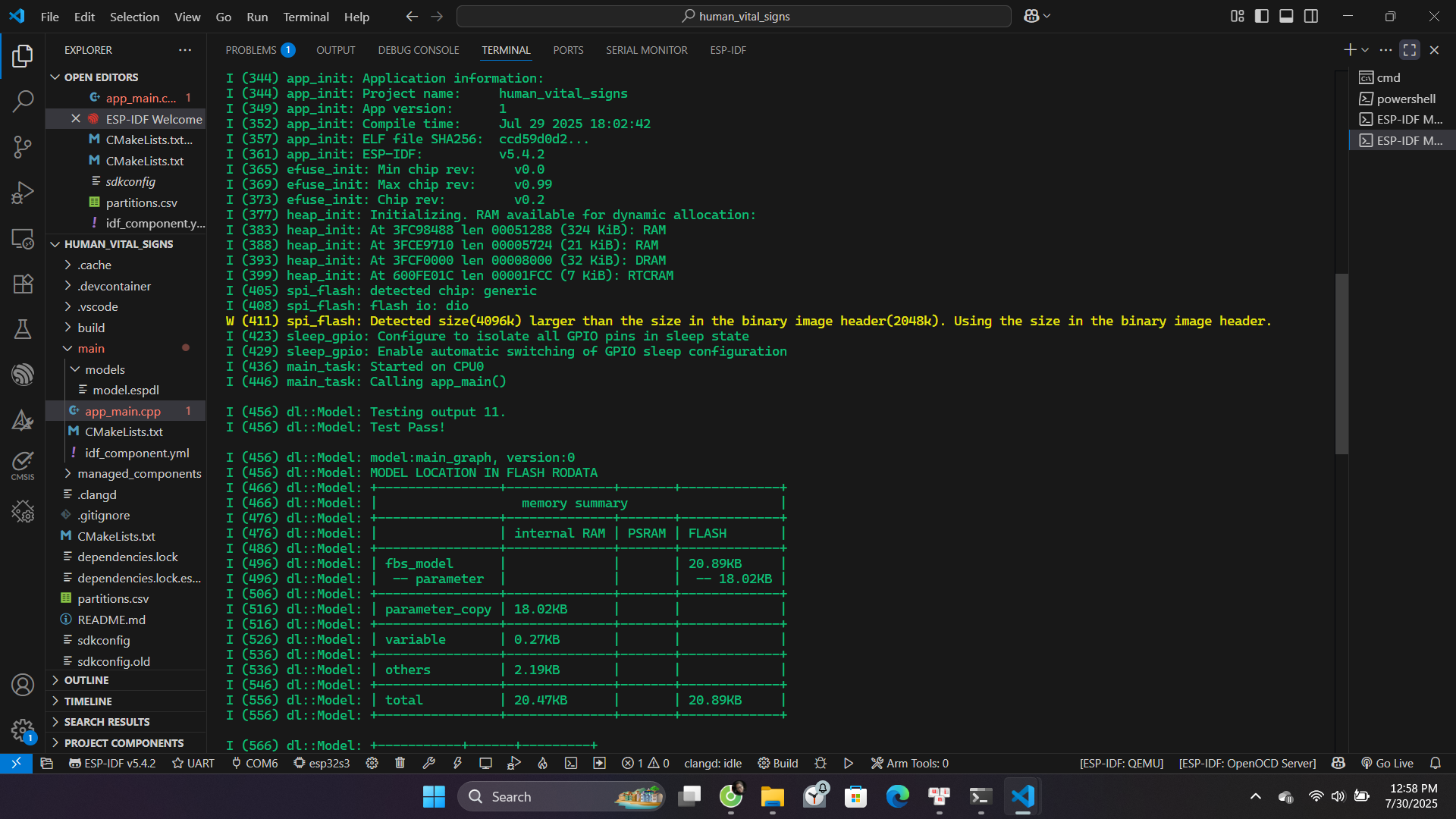
nvs, data, nvs, 0x9000, 0x6000

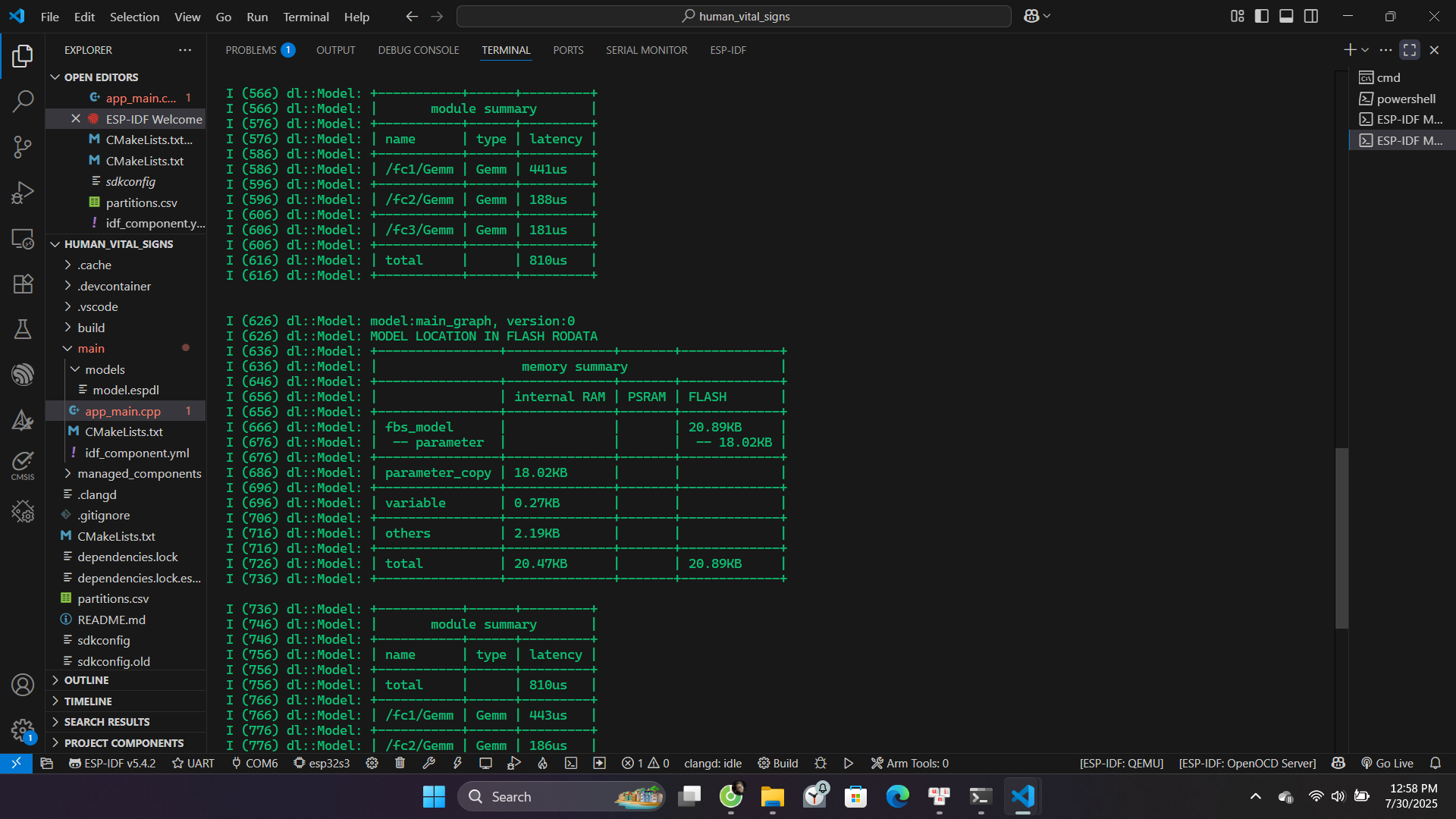
phy\_init, data, phy, 0xf000, 0x1000

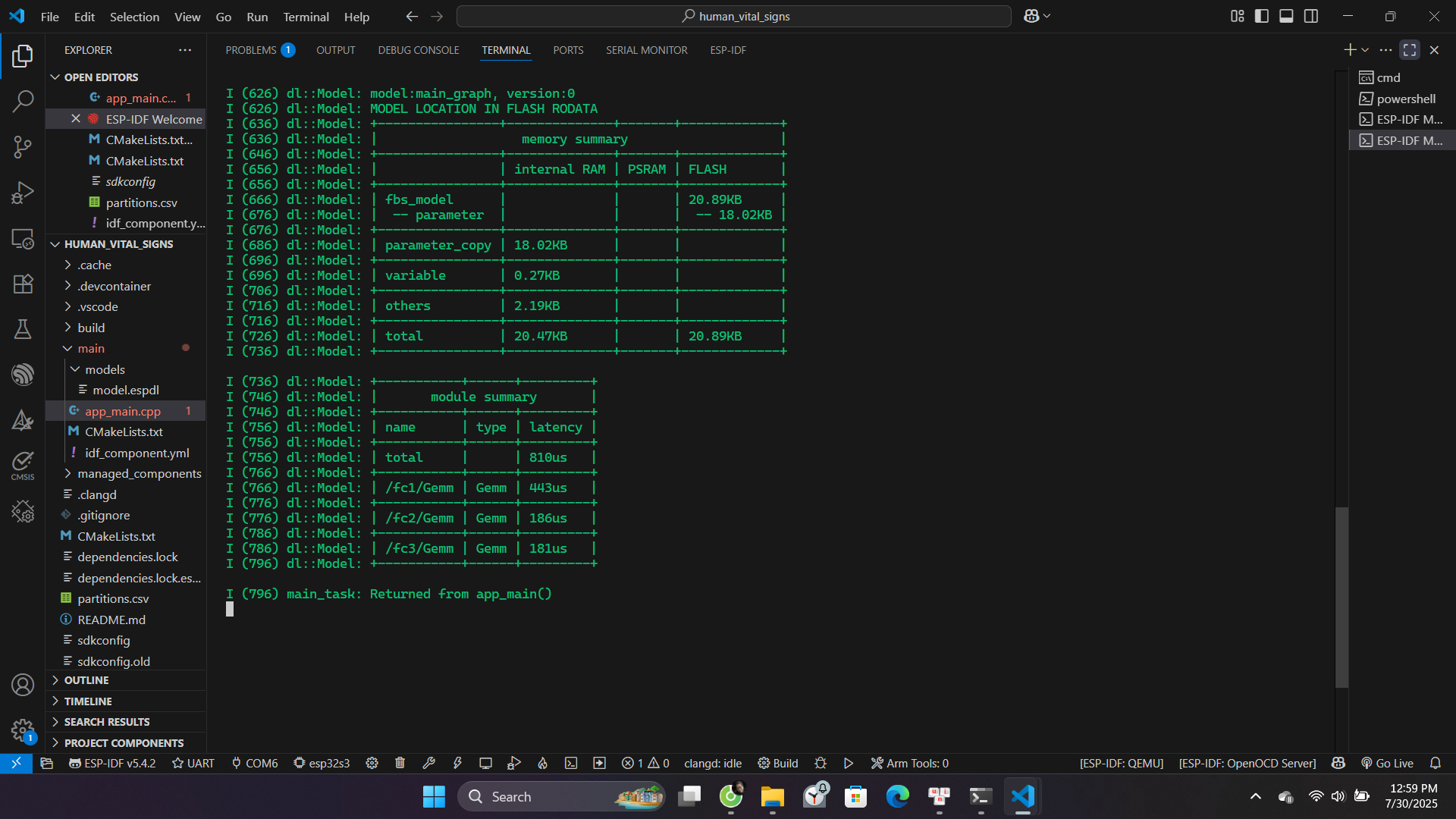
factory, app, factory, 0x10000, 0x180000

**Build và flash vào chip:**









1. Run model (đang thực hiện)

Full Code Test:  
#include "dl\_model\_base.hpp"

#include <map>

#include <vector>

#include <string>

#include <iostream>

#include <cmath>

#define NUM\_SAMPLES 1

#define NUM\_FEATURES 7

// Tên biến chính xác sinh bởi linker từ "models/espdl\_model.espdl"

extern const uint8\_t model\_espdl[] asm("\_binary\_model\_espdl\_start");

// Hàm sigmoid

float sigmoid(float x) {

return 1.0f / (1.0f + expf(-x));

}

// Dữ liệu thô

float input\_data[NUM\_SAMPLES][NUM\_FEATURES] = {

{96, 36.85263343f, 97.12412468f, 72, 0, 71.75897167f, 1.603887873f}

//{83, 36.044191f, 98.584497f, 84, 0, 79.295332f, 1.672735f}

// {63, 37.052, 98.50, 68, 1, 90.31, 1.77}

//{79, 36.885, 95.9871, 22, 0, 79.8699, 1.92233}

};

const float means[NUM\_FEATURES] = {79.53374663, 36.74835291, 97.50437243, 53.44627537, 0.49946505, 74.99641903, 1.75003102};

const float stds[NUM\_FEATURES] = {11.55286498, 0.43328918, 1.44259433, 20.78674961, 0.49999971, 14.4714659, 0.14455348};

float standard\_scaler(float x, float u, float s) {

float z = (x - u) / s;

return z;

}

void run\_hvsd\_model\_batch()

{

// Load model từ flash

dl::Model \*model = new dl::Model((const char \*)model\_espdl, fbs::MODEL\_LOCATION\_IN\_FLASH\_RODATA);

ESP\_ERROR\_CHECK(model->test());

// Lấy input/output tensor

dl::TensorBase \*model\_input = model->get\_inputs().begin()->second;

dl::TensorBase \*model\_output = model->get\_outputs().begin()->second;

// Chuẩn bị dữ liệu input chuẩn hóa

float input\_scaled[NUM\_SAMPLES][NUM\_FEATURES];

for (int i = 0; i < NUM\_SAMPLES; ++i) {

printf("Sample %d:\n", i);

for (int j = 0; j < NUM\_FEATURES; ++j) {

input\_scaled[i][j] = (input\_data[i][j] - means[j]) / stds[j];

printf("Feature %d: raw=%.4f, mean=%.4f, std=%.4f, normalized=%.4f\n", j, input\_data[i][j], means[j], stds[j], input\_scaled[i][j]);

}

}

// Gán dữ liệu cho input tensor

dl::TensorBase\* input\_tensor = new dl::TensorBase({NUM\_SAMPLES, NUM\_FEATURES}, nullptr, 0, dl::DATA\_TYPE\_FLOAT);

memcpy(input\_tensor->data, input\_scaled, sizeof(input\_scaled));

model\_input->assign(input\_tensor);

// Chạy mô hình

model->run();

// Lấy và đọc output

printf("Model output shape: [%zu, %zu]\n", model\_output->shape[0], model\_output->shape[1]);

float\* raw\_output\_data = (float\*)model\_output->data;

for (int i = 0; i < NUM\_SAMPLES; ++i) {

printf("Raw model output[%d] = %f\n", i, raw\_output\_data[i]);

}

dl::TensorBase\* output\_tensor = new dl::TensorBase({NUM\_SAMPLES, 1}, nullptr, 0, dl::DATA\_TYPE\_FLOAT);

output\_tensor->assign(model\_output);

float \*output\_ptr = (float \*)output\_tensor->data;

for (int i = 0; i < NUM\_SAMPLES; ++i) {

float logit = output\_ptr[i];

float prob = sigmoid(logit);

int class\_label = (prob >= 0.5f) ? 1 : 0;

const char\* risk = (class\_label == 1) ? "Low Risk" : "High Risk";

printf("Sample %d: logit = %.4f, prob = %.4f -> %s\n", i, logit, prob, risk);

}

delete input\_tensor;

delete output\_tensor;

delete model;

printf("Inference done.\n");

}

// ===== Load & Test & Profile Model =====

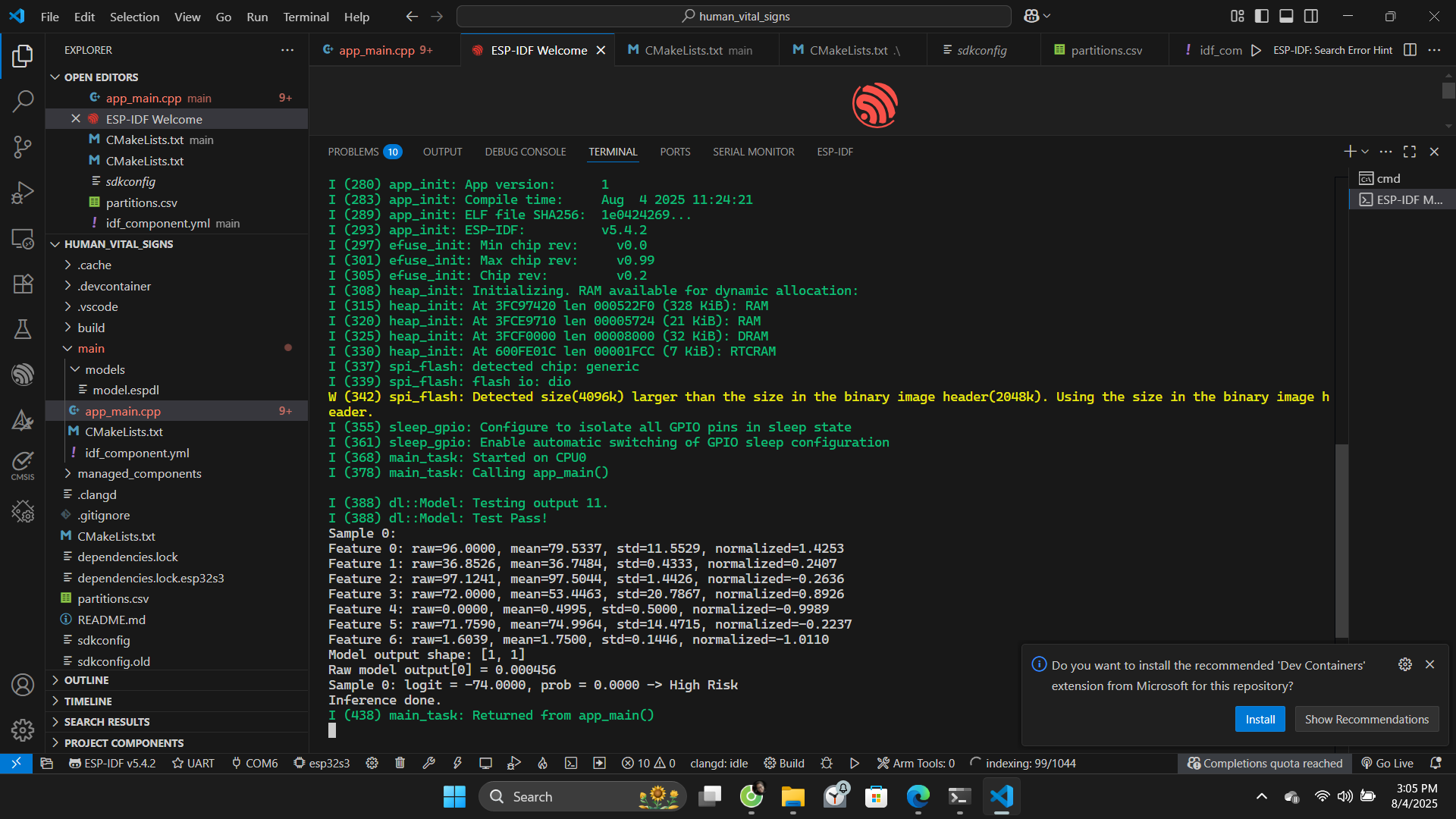
extern "C" void app\_main(void)

{

run\_hvsd\_model\_batch();

}

Kết quả:



# Bước tiếp theo:

* Kết hợp với team Phần Cứng