HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

...

BÁO CÁO ĐỒ ÁN MÔN HỌC

XÂY DỰNG CÁC HỆ THỐNG NHÚNG

ĐỀ TÀI:

PHÁT HIỆN CHUYỂN ĐỘNG CỦA CON NGƯỜI

GIẢNG VIÊN HƯỚNG DẪN : NGUYỄN TRỌNG KIÊN

- NGUYỄN THÀNH TRUNG – N20DCCN160 – D20CQCNPM02-N
- LÊ VĂN PHÚC – N20DCCN128 – D20CQCNPM02-N
- LÊ TUẤN TRIỆU – N20DCCN157 – D20CQCNPM02-N
- NGUYỄN XUÂN THỊNH – N20DCCN151 – D20CQCNPM02-N
- NGUYỄN HỮU NGHĨA – N20DCCN120 – D20CQCNPM02-N
- NGUYỄN VĂN ĐẠI – N20DCCN093 – D20CQCNPM02-N

LỜI CẢM ƠN

Nhóm em xin gửi lời biết ơn sâu sắc đến thầy Nguyễn Trọng Kiên, người đã hướng dẫn và hỗ trợ em rất nhiều trong quá trình học môn Hệ thống nhúng. Nhờ có sự chỉ dẫn của thầy, chúng em đã hoàn thành được một đề tài khoa học chất lượng.

Đồng thời, chúng em cũng đã học được nhiều kiến thức và kỹ năng quan trọng để tiếp tục nghiên cứu và xây dựng các đề tài khác phục vụ cho công việc sau này. Em cũng xin bày tỏ lòng biết ơn đến Ban lãnh đạo và các giảng viên tại Học Viện Công Nghệ Bưu Chính Viễn Thông vì đã tạo điều kiện và cung cấp cơ sở vật chất cho em có cơ hội và môi trường học tập tốt nhất.

Tuy nhiên, do kiến thức và kinh nghiệm còn hạn chế nên đề tài của chúng em không tránh khỏi những thiếu sót và sai lầm. Em rất mong nhận được những góp ý và nhận xét của thầy để chúng em có thể sửa chữa và hoàn thiện đề tài một cách tốt nhất.

Chúng em xin chân thành cảm ơn!

Thành phố Hồ Chí Minh, tháng 5 năm 2024

Sinh viên thực hiện

Nhóm 6

Mục Lục

Danh mục hình ảnh

PHẦN I: GIỚI THIỆU CHUNG

1. Lý do chọn đề tài

Với nhu cầu ngày càng cao về giám sát sức khỏe, hỗ trợ người già và người khuyết tật, đảm bảo an ninh,... việc phát triển các hệ thống phát hiện chuyển động hiệu quả trở nên cấp thiết hơn bao giờ hết. Hệ thống phát hiện chuyển động bằng cảm biến gia tốc nổi lên như giải pháp tiềm năng bởi những ưu điểm vượt trội như chi phí thấp, kích thước nhỏ gọn, dễ dàng tích hợp và hoạt động hiệu quả trong môi trường thiếu sáng. Ứng dụng các nguyên tắc khoa học về gia tốc kế, xử lý tín hiệu số và học máy, hệ thống có khả năng nhận diện chính xác các chuyển động đi, đứng, nằm của con người. Nhờ vậy, hệ thống có thể được ứng dụng rộng rãi trong nhiều lĩnh vực như: Giám sát sức khỏe: Theo dõi hoạt động thể chất, phát hiện té ngã, hỗ trợ người già và người khuyết tật,... An ninh: Phát hiện đột nhập, theo dõi hành vi của con người,... Nhà thông minh: Điều khiển tự động các thiết bị điện tử trong nhà dựa trên chuyển động của con người,... Robot: Giúp robot nhận biết và tương tác với môi trường xung quanh,... Với tiềm năng ứng dụng to lớn và tính thực tiễn cao, hệ thống phát hiện chuyển động bằng cảm biến gia tốc hứa hẹn mang lại nhiều lợi ích cho xã hội. Hệ thống có thể góp phần nâng cao chất lượng cuộc sống, đảm bảo an toàn cho con người và tạo ra những đột phá trong các lĩnh vực như y tế, an ninh, nhà thông minh,... Hệ thống phát hiện chuyển động của con người sử dụng cảm biến gia tốc là giải pháp công nghệ tiên tiến, hiệu quả và đầy tiềm năng cho tương lai.

2. Mục tiêu

Phát triển hệ thống có khả năng phát hiện chính xác các chuyển động đi, đứng, nằm,… của con người bằng cách sử dụng cảm biến gia tốc. Áp dụng các thuật toán xử lý tín hiệu số và học máy để trích xuất các đặc trưng chuyển động từ dữ liệu gia tốc một cách hiệu quả. Đánh giá hiệu suất của hệ thống thông qua các thí nghiệm thực tế với nhiều đối tượng tham gia.

PHẦN II: PHÂN TÍCH THIẾT KẾ

1. Hành động của con người là gì?

Hành động của con người là những hoạt động do con người thực hiện một cách có ý thức, có chủ đích nhằm đạt được mục đích nhất định. Những hành động này có thể đơn giản hoặc phức tạp, có thể diễn ra trong thời gian ngắn hoặc kéo dài, có thể ảnh hưởng đến bản thân như đi, đứng chạy, nhảy,…

2. Phát hiện chuyển động giúp ích gì?

2.1 Theo dõi hoạt động

Nâng cao ý thức về sức khỏe: Việc theo dõi số bước chân, lượng calo đốt cháy và các hoạt động thể chất khác giúp người dùng nhận thức rõ ràng hơn về mức độ hoạt động của bản thân, từ đó khuyến khích họ duy trì lối sống lành mạnh.

Đặt mục tiêu và theo dõi tiến độ: Người dùng có thể đặt ra mục tiêu cụ thể về số bước chân, lượng calo đốt cháy hoặc thời gian tập luyện và theo dõi tiến độ của mình một cách trực quan. Điều này giúp họ duy trì động lực và cải thiện hiệu quả tập luyện.

Cải thiện sức khỏe tim mạch: Hoạt động thể chất thường xuyên giúp tăng cường sức khỏe tim mạch, giảm nguy cơ mắc các bệnh tim mạch, đột quỵ và tiểu đường.

Kiểm soát cân nặng: Theo dõi lượng calo đốt cháy giúp người dùng điều chỉnh chế độ ăn uống hợp lý để duy trì cân nặng hợp lý, giảm nguy cơ béo phì và các bệnh liên quan.

Nâng cao sức khỏe tinh thần: Tập thể dục thường xuyên giúp giảm căng thẳng, lo âu, trầm cảm và cải thiện tâm trạng, mang lại cho người dùng cảm giác vui vẻ và tràn đầy năng lượng.

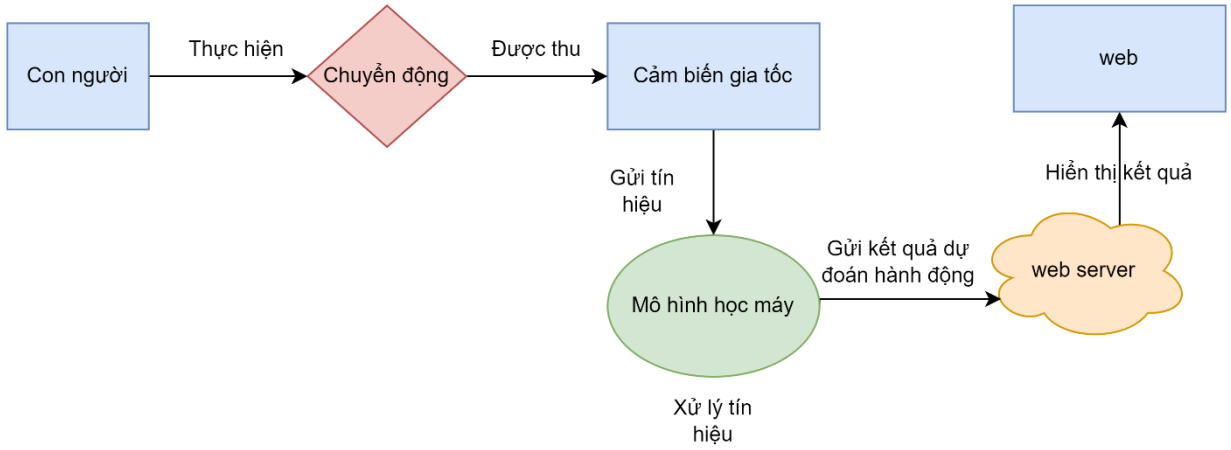
2.2 Phát hiện té ngã

Bảo vệ người già và người có nguy cơ té ngã: Té ngã là một vấn đề nghiêm trọng đối với người già và người có nguy cơ té ngã, có thể dẫn đến gãy xương, chấn thương não và thậm chí tử vong. Việc phát hiện té ngã kịp thời giúp người thân hoặc dịch vụ y tế hỗ trợ người bị ngã nhanh chóng, giảm thiểu nguy cơ biến chứng và cải thiện chất lượng cuộc sống của họ.

Mang lại sự an tâm cho người thân: Khi biết rằng người thân được theo dõi và bảo vệ, người thân của họ sẽ cảm thấy an tâm hơn và giảm bớt lo lắng.

Giúp người bị ngã hồi phục nhanh hơn: Việc hỗ trợ kịp thời sau khi té ngã giúp người bị ngã hồi phục nhanh hơn và giảm nguy cơ tái phát.

3. Mô hình phát hiện chuyển động



Sơ đồ 3.1 Sơ đồ biểu diễn qui trình phát hiện chuyển động

4. Các thiết bị hỗ trợ

4.1 *Giới thiệu về ESP32*

ESP32 là một vi điều khiển (microcontroller) mạnh mẽ được phát triển bởi Espressif Systems. Nó được thiết kế để hỗ trợ các ứng dụng IoT (Internet of Things) với khả năng kết nối Wi-Fi và Bluetooth tích hợp sẵn. ESP32 nổi bật nhờ hiệu năng cao, tính năng đa dạng và giá thành hợp lý, trở thành một lựa chọn phổ biến trong nhiều dự án điện tử và IoT.

4.1.1 *Các đặc điểm và tính năng chính của ESP32*

Vi xử lý mạnh mẽ: ESP32 được trang bị vi xử lý hai nhân (dual-core) Xtensa LX6 với xung nhịp lên đến 240 MHz, cho phép xử lý nhanh chóng và hiệu quả các tác vụ phức tạp.

Kết nối không dây: ESP32 tích hợp cả Wi-Fi 802.11 b/g/n và Bluetooth 4.2 (cả BLE và Classic), hỗ trợ giao tiếp không dây đa dạng, phù hợp cho các ứng dụng IoT yêu cầu kết nối linh hoạt.

Bộ nhớ: ESP32 có 520 KB RAM và dung lượng bộ nhớ flash từ 4 MB trở lên, đủ để lưu trữ và chạy các chương trình phức tạp.

Giao diện ngoại vi: ESP32 hỗ trợ nhiều giao diện ngoại vi như SPI, I2C, UART, ADC, DAC và PWM, dễ dàng kết nối với các cảm biến và thiết bị ngoại vi khác.

Quản lý năng lượng: ESP32 có khả năng quản lý năng lượng hiệu quả với các chế độ ngủ (sleep modes) khác nhau, giúp tiết kiệm pin và kéo dài thời gian hoạt động của thiết bị.

Bảo mật: ESP32 hỗ trợ các tính năng bảo mật tiên tiến như mã hóa AES, SHA-2, và RSA, đảm bảo an toàn cho các ứng dụng IoT.

4.1.2 *Ứng dụng của ESP32 trong dự án phát hiện chuyển động*

Trong dự án sử dụng ESP32 và MPU6050 để phát hiện các chuyển động cơ bản của con người như đi, đứng, nằm, ngồi, ESP32 đóng vai trò quan trọng với các chức năng chính như sau:

Thu thập dữ liệu từ MPU6050: MPU6050 là một cảm biến 6 trục (gyroscope và accelerometer) cung cấp dữ liệu về gia tốc và quay. ESP32 sẽ giao tiếp với MPU6050 thông qua giao diện I2C để thu thập dữ liệu chuyển động.

Xử lý dữ liệu: Với vi xử lý mạnh mẽ, ESP32 có khả năng xử lý dữ liệu cảm biến từ MPU6050 theo thời gian thực, áp dụng các thuật toán phân tích để nhận diện các trạng thái chuyển động như đi, đứng, nằm, ngồi…

Kết nối và truyền dữ liệu: ESP32 có thể truyền dữ liệu phân tích được qua Wi-Fi hoặc Bluetooth đến một máy chủ trung tâm hoặc thiết bị khác để giám sát và lưu trữ. Điều này cho phép xây dựng các hệ thống theo dõi chuyển động thông minh, phục vụ cho các ứng dụng an ninh, y tế, hoặc nhà thông minh.

Điều khiển thiết bị: Dựa vào kết quả nhận diện hành động, ESP32 có thể kích hoạt các thiết bị khác như cảnh báo, đèn, hoặc gửi thông báo đến người dùng.

4.1.3 Lợi ích của việc sử dụng ESP32

Hiệu năng và tính năng mạnh mẽ: Với khả năng xử lý mạnh mẽ và tính năng kết nối phong phú, ESP32 đáp ứng tốt các yêu cầu của dự án phát hiện chuyển động.

Giá thành hợp lý: ESP32 có giá thành thấp, giúp giảm chi phí tổng thể của dự án mà vẫn đảm bảo chất lượng và hiệu suất.

Cộng đồng hỗ trợ mạnh mẽ: ESP32 có một cộng đồng người dùng và nhà phát triển lớn, cung cấp nhiều tài liệu, thư viện và hỗ trợ kỹ thuật, giúp quá trình phát triển và triển khai dự án dễ dàng hơn.

4.2 *Giới thiệu về MPU6050*

MPU6050 là một cảm biến MEMS 6 trục (6 Degrees of Freedom - DoF) kết hợp giữa gia tốc kế 3 trục (accelerometer) và con quay hồi chuyển 3 trục (gyroscope). Được sản xuất bởi InvenSense (hiện là TDK InvenSense), MPU6050 được sử dụng rộng rãi trong các ứng dụng yêu cầu theo dõi và đo lường chuyển động và tư thế trong không gian ba chiều.

4.2.1 *Các đặc điểm và tính năng chính của MPU6050*

Gia tốc kế 3 trục: Đo lường gia tốc theo ba trục X, Y, Z với dải đo có thể điều chỉnh từ ±2g đến ±16g. Gia tốc kế cung cấp dữ liệu về sự thay đổi vị trí và gia tốc của thiết bị trong không gian.

Con quay hồi chuyển 3 trục: Đo lường tốc độ quay theo ba trục X, Y, Z với dải đo từ ±250 đến ±2000 độ/giây. Con quay hồi chuyển cung cấp thông tin về sự thay đổi góc và hướng của thiết bị.

Giao tiếp I2C: MPU6050 sử dụng giao thức I2C (Inter-Integrated Circuit) để giao tiếp với các vi điều khiển như ESP32. Giao thức I2C đơn giản và hiệu quả, cho phép truyền dữ liệu nhanh chóng và đáng tin cậy.

Bộ lọc số DMP (Digital Motion Processor): MPU6050 tích hợp bộ xử lý chuyển động số (DMP) có khả năng xử lý dữ liệu cảm biến nội bộ, giảm tải cho vi điều khiển chính. DMP có thể thực hiện các phép tính phức tạp như tính toán góc quay và quaternions.

Đo lường nhiệt độ: MPU6050 cũng có cảm biến nhiệt độ tích hợp để theo dõi và điều chỉnh hiệu suất hoạt động.

4.2.2 Chức năng của MPU6050 trong dự án phát hiện chuyển động

Trong dự án phát hiện các chuyển động cơ bản của con người như đi, đứng, nằm, ngồi,… MPU6050 đóng vai trò quan trọng bằng cách cung cấp dữ liệu cảm biến chi tiết và chính xác cho ESP32 để xử lý và phân tích. Các chức năng cụ thể bao gồm:

Đo lường gia tốc: Gia tốc kế của MPU6050 đo lường sự thay đổi vận tốc của cơ thể trong không gian, giúp xác định các chuyển động như bước đi, đứng dậy hoặc ngồi xuống.

Đo lường tốc độ quay: Con quay hồi chuyển đo lường tốc độ quay của cơ thể, cung cấp thông tin về hướng và tốc độ quay khi thực hiện các động tác như xoay người hay ngã.

Tích hợp dữ liệu: MPU6050 có khả năng tích hợp dữ liệu gia tốc và quay để cung cấp thông tin tổng thể về tư thế và chuyển động của cơ thể, giúp nhận diện chính xác các trạng thái như đi, đứng, nằm, ngồi.

Cải thiện độ chính xác: Sử dụng DMP tích hợp, MPU6050 có thể xử lý và lọc dữ liệu cảm biến nội bộ trước khi truyền đến ESP32, giảm thiểu nhiễu và cải thiện độ chính xác của các phép đo.

4.2.3 Ưu điểm của MPU6050

Độ chính xác cao: MPU6050 cung cấp dữ liệu chuyển động và tư thế với độ chính xác cao, phù hợp cho các ứng dụng yêu cầu đo lường chính xác.

Tích hợp DMP: Bộ xử lý chuyển động số giúp giảm tải cho vi điều khiển chính và cải thiện hiệu suất xử lý dữ liệu.

Giao tiếp dễ dàng: Sử dụng giao thức I2C giúp việc giao tiếp với các vi điều khiển như ESP32 trở nên đơn giản và hiệu quả.

Kích thước nhỏ gọn: MPU6050 có kích thước nhỏ gọn, dễ dàng tích hợp vào các thiết bị và hệ thống khác nhau.

4.2.4 Các ứng dụng của MPU6050

Thiết bị đeo thông minh: Sử dụng trong các thiết bị đeo để theo dõi chuyển động và hoạt động của người dùng, như đồng hồ thông minh và vòng đeo tay theo dõi sức khỏe.

Robot và điều khiển tự động: Dùng trong các hệ thống robot để điều khiển và theo dõi chuyển động của robot.

Hệ thống thực tế ảo (VR) và tăng cường (AR): Cung cấp dữ liệu chuyển động chính xác để cải thiện trải nghiệm người dùng trong các ứng dụng VR và AR.

Giám sát sức khỏe và an ninh: Sử dụng trong các hệ thống giám sát để theo dõi và nhận diện các trạng thái và chuyển động của người dùng trong các ứng dụng y tế và an ninh.

5. Các thư viện sử dụng cho ESP32 và MPU6050 trong hệ Arduino

Trong dự án phát hiện chuyển động của con người sử dụng ESP32 và MPU6050, các thư viện dưới đây được sử dụng để hỗ trợ việc thu thập dữ liệu cảm biến, xử lý dữ liệu và nhận diện các trạng thái chuyển động.

5.1 *Thư viện FreeRTOS*

FreeRTOS là một hệ điều hành thời gian thực (Real-Time Operating System - RTOS) mã nguồn mở được thiết kế cho các vi điều khiển và vi xử lý nhỏ gọn. Trong dự án sử dụng ESP32 để phát hiện các chuyển động cơ bản của con người như đi, đứng, nằm, ngồi, FreeRTOS cung cấp nền tảng quản lý đa nhiệm, giúp tối ưu hóa hiệu suất và độ phản hồi của hệ thống.

5.1.1 *Các đặc điểm và tính năng chính của FreeRTOS*

Đa nhiệm (Multitasking): FreeRTOS cho phép tạo và quản lý nhiều tác vụ đồng thời, giúp phân chia công việc và tối ưu hóa hiệu suất hệ thống.

Quản lý bộ nhớ: Hỗ trợ các cơ chế quản lý bộ nhớ động và tĩnh, giúp sử dụng tài nguyên bộ nhớ một cách hiệu quả.

Đồng bộ hóa và truyền thông giữa các tác vụ: Cung cấp các công cụ như semaphore, mutex, queue, và event groups để đồng bộ hóa và truyền thông giữa các tác vụ.

Hệ thống thời gian thực: Hỗ trợ các chức năng định thời (timers) và trì hoãn (delays), giúp quản lý các sự kiện thời gian thực.

5.1.2 *Chức năng của FreeRTOS trong dự án phát hiện chuyển động*

FreeRTOS giúp quản lý và điều phối các tác vụ liên quan đến thu thập, xử lý dữ liệu cảm biến và truyền thông, đảm bảo hệ thống hoạt động ổn định và hiệu quả. Các chức năng cụ thể bao gồm:

Quản lý tác vụ thu thập dữ liệu: Tạo một tác vụ riêng biệt để liên tục thu thập dữ liệu từ cảm biến MPU6050, đảm bảo dữ liệu được cập nhật kịp thời và không bị gián đoạn.

Xử lý dữ liệu song song: Tạo các tác vụ xử lý dữ liệu riêng biệt để phân tích và nhận diện các trạng thái chuyển động. Việc xử lý song song giúp cải thiện hiệu suất và tốc độ phản hồi của hệ thống.

Quản lý truyền thông: Sử dụng các tác vụ để quản lý truyền thông qua Wi-Fi hoặc Bluetooth, đảm bảo dữ liệu được truyền đến các thiết bị hoặc máy chủ trung tâm một cách hiệu quả và đáng tin cậy.

Đồng bộ hóa và truyền thông: Sử dụng semaphore và queue để đồng bộ hóa giữa các tác vụ, đảm bảo dữ liệu được truyền và xử lý một cách chính xác.

5.1.3 *Ưu điểm của FreeRTOS*

Tính linh hoạt cao: FreeRTOS dễ dàng tùy chỉnh và cấu hình theo nhu cầu cụ thể của dự án, cho phép tối ưu hóa hiệu suất và tài nguyên.

Hiệu quả và nhẹ: Với kích thước nhỏ gọn và hiệu suất cao, FreeRTOS không tốn nhiều tài nguyên hệ thống, phù hợp cho các ứng dụng nhúng và vi điều khiển.

Hỗ trợ rộng rãi: FreeRTOS hỗ trợ nhiều kiến trúc vi xử lý khác nhau và có cộng đồng người dùng lớn, cung cấp nhiều tài liệu và hỗ trợ kỹ thuật.

Tích hợp với ESP32: FreeRTOS được tích hợp sẵn trong SDK của ESP32 (ESP-IDF), giúp việc triển khai và quản lý các tác vụ trên ESP32 trở nên dễ dàng và hiệu quả.

5.2 *Thư viện TensorFlowLite_ESP32.h*

Thư viện TensorFlow Lite for Microcontrollers dành cho ESP32 giúp tích hợp mô hình học máy vào các ứng dụng nhúng. Thư viện này cho phép ESP32 chạy các mô hình TensorFlow Lite để phân tích và nhận diện các chuyển động từ dữ liệu cảm biến. *Chức năng chính:*

Inference: Thực hiện các phép tính suy luận (inference) từ mô hình học máy đã được huấn luyện.

Tích hợp dễ dàng: Dễ dàng tích hợp với phần cứng ESP32 để thực hiện các tác vụ học máy trên thiết bị.

5.3 *Thư viện MPU6050.h*

Thư viện MPU6050 cung cấp các hàm và phương thức để giao tiếp với cảm biến MPU6050. Thư viện này giúp ESP32 thu thập dữ liệu gia tốc và tốc độ quay từ cảm biến. *Chức năng chính:*

Giao tiếp I2C: Thiết lập và quản lý giao tiếp I2C giữa ESP32 và MPU6050.

Đọc dữ liệu cảm biến: Thu thập dữ liệu gia tốc và con quay hồi chuyển theo ba trục X, Y, Z.

Cấu hình cảm biến: Cấu hình các thông số hoạt động của MPU6050 như dải đo, tần số lấy mẫu.

5.4 *Thư viện Wire.h*

Thư viện Wire là thư viện tiêu chuẩn của Arduino để giao tiếp I2C. Thư viện này cung cấp các phương thức để khởi tạo và quản lý giao tiếp I2C giữa ESP32 và các thiết bị ngoại vi như MPU6050. *Chức năng chính:*

Khởi tạo I2C: Thiết lập các thông số cho giao tiếp I2C.

Giao tiếp dữ liệu: Gửi và nhận dữ liệu qua giao thức I2C.

5.5 *Thư viện Tensorflow/lite/micro/all_ops_resolver.h*

Thư viện này thuộc TensorFlow Lite for Microcontrollers và cung cấp tất cả các phép toán (operations) cần thiết cho việc thực hiện mô hình học máy trên các thiết bị vi điều khiển. *Chức năng chính:*

Đăng ký phép toán: Đăng ký các phép toán sử dụng trong mô hình TensorFlow Lite.

Quản lý phép toán: Cung cấp các phương thức để quản lý và tối ưu hóa các phép toán.

5.6 *Thư viện tensorflow/lite/micro/micro_error_reporter.h*

Thư viện này quản lý và báo cáo các lỗi phát sinh trong quá trình thực hiện mô hình TensorFlow Lite trên vi điều khiển. *Chức năng chính:*

Báo cáo lỗi: Ghi nhận và báo cáo các lỗi trong quá trình thực hiện suy luận mô hình.

Quản lý lỗi: Cung cấp thông tin chi tiết về lỗi để hỗ trợ gỡ lỗi và cải thiện hiệu suất.

5.7 *Thư viện Tensorflow/lite/micro/micro_interpreter.h*

Thư viện này thực hiện chức năng của một interpreter cho mô hình TensorFlow Lite, giúp chạy mô hình trên vi điều khiển. *Chức năng chính:*

Thiết lập interpreter: Khởi tạo và cấu hình interpreter để chạy mô hình.

Thực hiện suy luận: Thực hiện các phép tính suy luận dựa trên mô hình đã được tải lên.

5.8 *Thư viện Tensorflow/lite/micro/system_setup.h*

Thư viện này cung cấp các phương thức thiết lập hệ thống, cần thiết cho việc khởi tạo môi trường chạy mô hình TensorFlow Lite trên vi điều khiển. *Chức năng chính:*

Thiết lập hệ thống: Cấu hình và khởi tạo các thành phần hệ thống cần thiết cho TensorFlow Lite.

Tối ưu hóa hiệu suất: Đảm bảo môi trường chạy mô hình được tối ưu hóa.

5.9 *Thư viện Tensorflow/lite/schema/schema_generated.h*

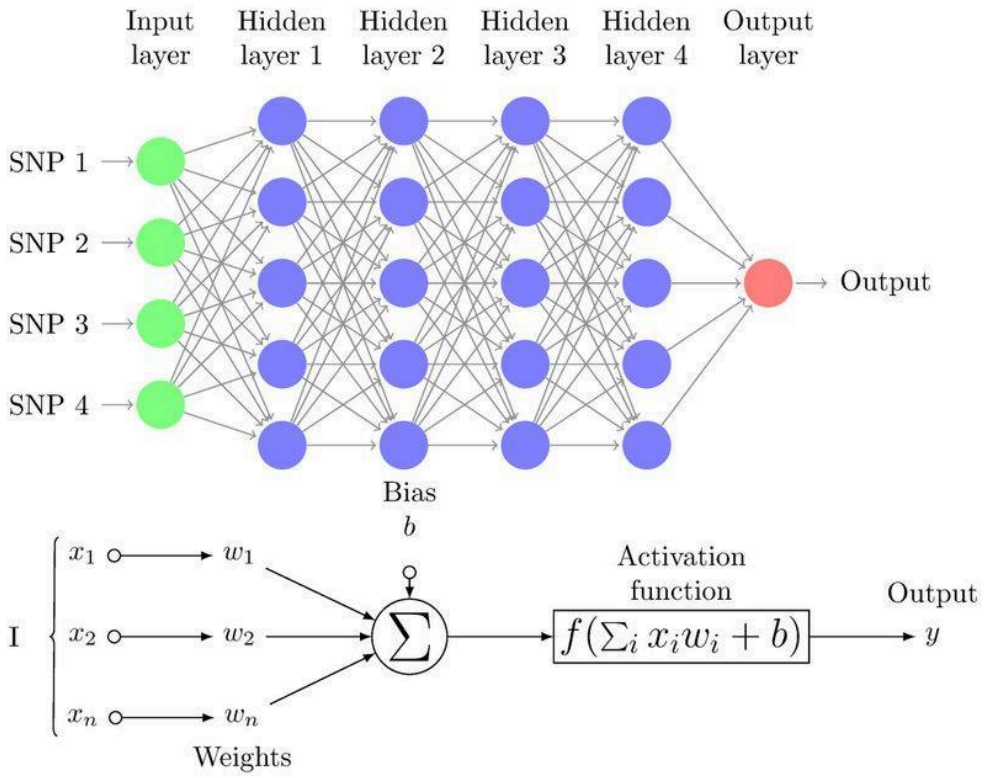
Thư viện này chứa các định nghĩa schema cho mô hình TensorFlow Lite, giúp quản lý và phân tích cấu trúc của mô hình. *Chức năng chính:*

Quản lý schema: Cung cấp các định nghĩa và phương thức để quản lý schema của mô hình TensorFlow Lite.

Hỗ trợ phân tích mô hình: Giúp phân tích và hiểu cấu trúc của mô hình học máy.

6. Mô hình học máy

6.1 *Giới thiệu mô hình Mạng Neural Đa lớp (Multilayer Perceptron - MLP)*



Sơ đồ 6.1 Sơ đồ biểu diễn mô hình học máy Multilayer Perceptron - MLP

Mạng Neural Đa lớp, hay còn gọi là MLP (Multilayer Perceptron), là một loại mạng neural feedforward bao gồm ít nhất ba lớp: lớp đầu vào (input layer), một hoặc nhiều lớp ẩn (hidden layers), và lớp đầu ra (output layer).

6.1.1 *Cấu trúc của MLP*

Lớp đầu vào (Input Layer): Đây là lớp đầu tiên trong MLP, nhận đầu vào trực tiếp từ dữ liệu. Số lượng neuron trong lớp này tương ứng với số lượng đặc trưng (features) của dữ liệu đầu vào. Ví dụ, trong trường hợp dữ liệu từ cảm biến MPU6050, mỗi mẫu dữ liệu có 6 giá trị (gia tốc và con quay hồi chuyển trên ba trục), vì vậy lớp đầu vào sẽ có 6 neuron.

Lớp ẩn (Hidden Layers): Đây là các lớp giữa lớp đầu vào và lớp đầu ra. Mỗi lớp ẩn thường có một số lượng neuron tùy ý và sử dụng các hàm kích hoạt phi tuyến tính (như ReLU) để tạo ra các quan hệ phi tuyến tính trong dữ liệu. MLP có thể có một hoặc nhiều lớp ẩn,

và số lượng neuron trong mỗi lớp cũng có thể khác nhau. Các lớp ẩn giúp mạng học các đặc trưng phức tạp từ dữ liệu đầu vào.

Lớp đầu ra (Output Layer): Đây là lớp cuối cùng trong MLP, tạo ra dự đoán cuối cùng. Số lượng neuron trong lớp này tương ứng với số lượng lớp (classes) hoặc giá trị liên tục mà mô hình cần dự đoán. Nếu làm bài toán phân loại, lớp đầu ra thường sử dụng hàm kích hoạt softmax để tạo ra xác suất của mỗi lớp.

6.1.2 *Hoạt động của MLP*

MLP hoạt động theo cơ chế truyền thẳng (feedforward):

Truyền tiến (Forward Propagation): Dữ liệu đầu vào được truyền từ lớp đầu vào qua các lớp ẩn, và cuối cùng đến lớp đầu ra. Tại mỗi neuron, các đầu vào được tính toán trọng số, sau đó áp dụng hàm kích hoạt để tạo ra đầu ra.

Hàm kích hoạt (Activation Function): Hàm kích hoạt được sử dụng tại mỗi neuron để tạo ra các quan hệ phi tuyến tính. Các hàm kích hoạt phổ biến bao gồm:

- ReLU (Rectified Linear Unit): $f(x)=\max(0,x)$
- Sigmoid $f(x) = \frac{1}{1+e^{-x}}$
- Tanh $f(x) = \tanh(x)$
- Softmax: Được sử dụng ở lớp đầu ra cho bài toán phân loại đa lớp.

6.1.3 *Huấn luyện MLP*

Quá trình huấn luyện MLP bao gồm các bước sau:

Truyền tiến (Forward Propagation): Tính toán đầu ra của mạng dựa trên đầu vào.

Tính toán lỗi (Loss Calculation): Sử dụng một hàm mất mát (loss function) để đo lường sự khác biệt giữa đầu ra dự đoán và giá trị thực tế. Các hàm mất mát phổ biến bao gồm:
Mean Squared Error (MSE) cho bài toán hồi quy.
Cross-Entropy Loss cho bài toán phân loại.

Lan truyền ngược (Backpropagation): Tính toán gradient của hàm mất mát đối với các trọng số của mạng và cập nhật các trọng số này bằng cách sử dụng các thuật toán tối ưu hóa như Gradient Descent. Phổ biến hơn là sử dụng các biến thể của Gradient Descent như Adam hoặc RMSprop.

Cập nhật trọng số (Weight Update): Dựa trên gradient tính toán, các trọng số được cập nhật để giảm lỗi dự đoán.

6.1.4 *Ưu điểm và Nhược điểm của MLP*

Ưu điểm:

Có khả năng mô hình hóa các quan hệ phi tuyến tính phức tạp. Linh hoạt và có thể được sử dụng cho nhiều loại bài toán khác nhau, bao gồm phân loại và hồi quy.

Khi có đủ dữ liệu và cấu trúc mạng phù hợp, MLP có thể đạt hiệu suất cao.

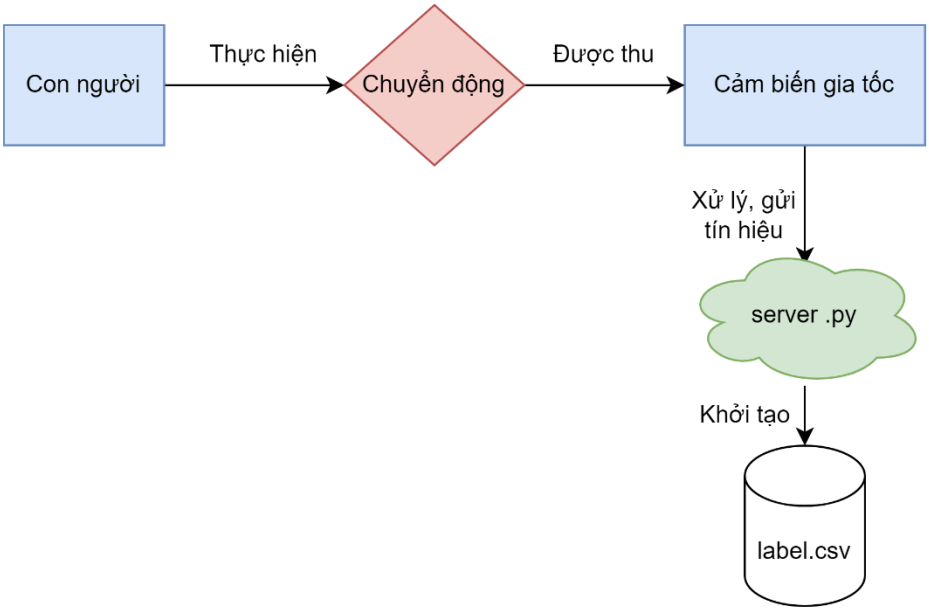
Nhược điểm:

Dễ bị overfitting nếu số lượng lớp ẩn và số lượng neuron quá lớn.

Yêu cầu tài nguyên tính toán lớn khi số lượng tham số tăng lên.

Không hiệu quả khi làm việc với dữ liệu có tính thời gian hoặc dữ liệu tuần tự mà không sử dụng các biến thể như LSTM hoặc GRU.

7. Bộ dữ liệu xây dựng mô hình học máy (bộ Dataset)



Sơ đồ 7.1 Sơ đồ biểu diễn quy trình thu dữ liệu

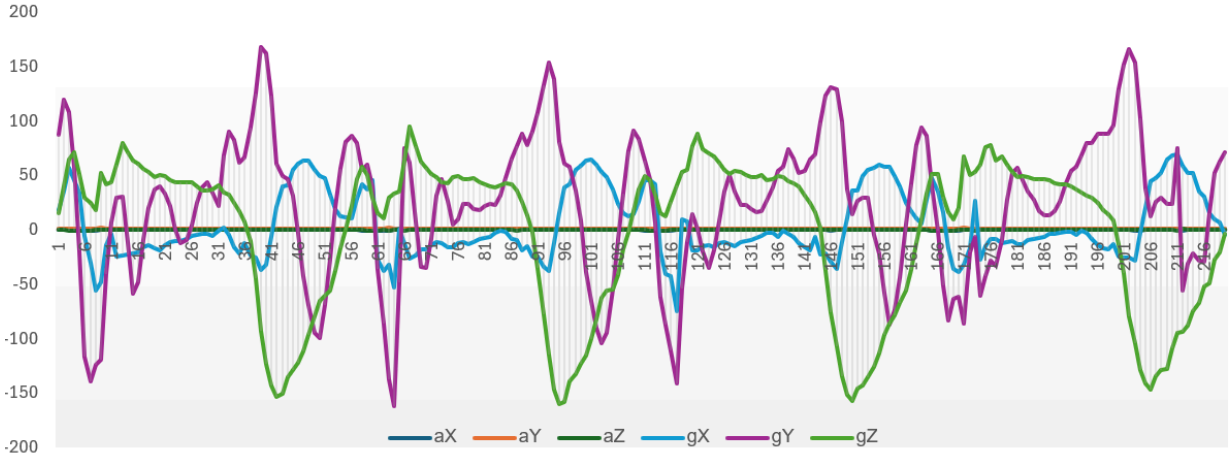
Bộ dataset được xây dựng thông qua việc ghi lại các hành động của các thành viên thực hiện các hành động khác nhau như đi, chạy, nằm, đứng và ngồi. Việc thu thập dữ liệu được thực hiện bằng cách sử dụng cảm biến MPU6050 kết hợp với mạch ESP32. Các cảm biến này được gắn cố định trên chân trái, cách đầu gối 5cm. Trong quá trình thu thập, các cảm biến ghi lại các giá trị gia tốc và hướng theo ba trục Ox, Oy và Oz. Các dữ liệu này phản ánh chính xác các chuyển động và tư thế của người thực hiện, cho phép xây dựng một bộ dữ liệu đầy đủ và chính xác về các hành động cơ bản của con người

Bảng 7.1 Thống kê dữ liệu đã thu từ bộ cảm biến

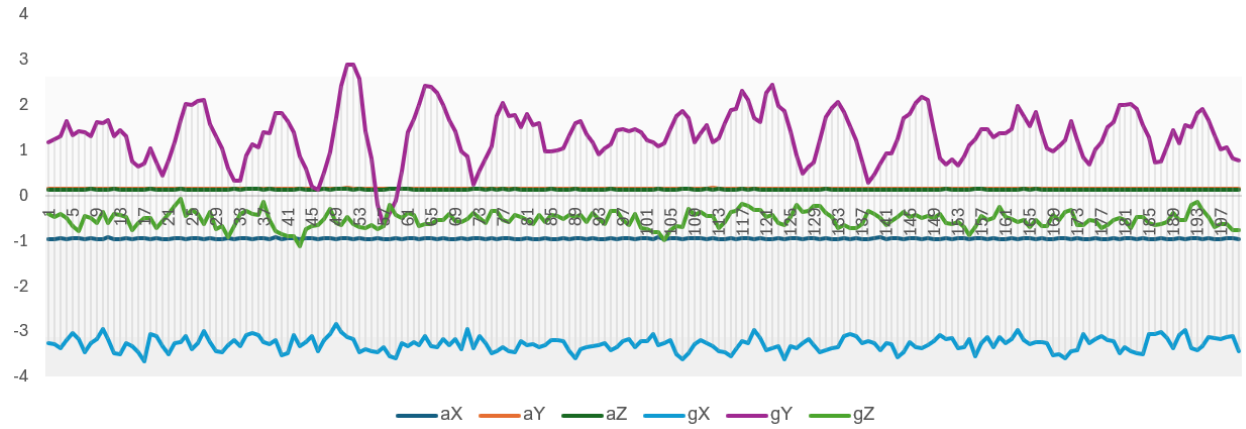
STT	**Hành động**	**Số mẫu(Mỗi mẫu có 60 mốc thời gian)**
1	Walking	75
2	Sitting	16
3	Running	104
4	Jumping	172
5	Standing	16

Bộ data được thu trên 3 người khỏe mạnh, không dị tật với số lần thu tùy thuộc vào điểm đặc trưng của hành động.

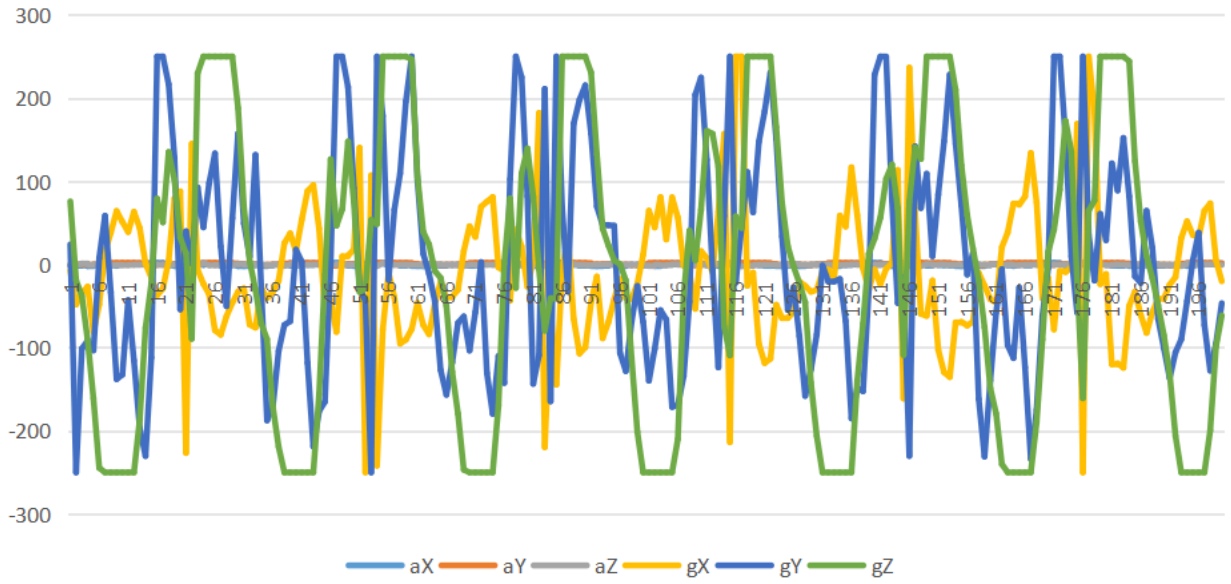
Các sơ đồ hành động trong 200 mẫu



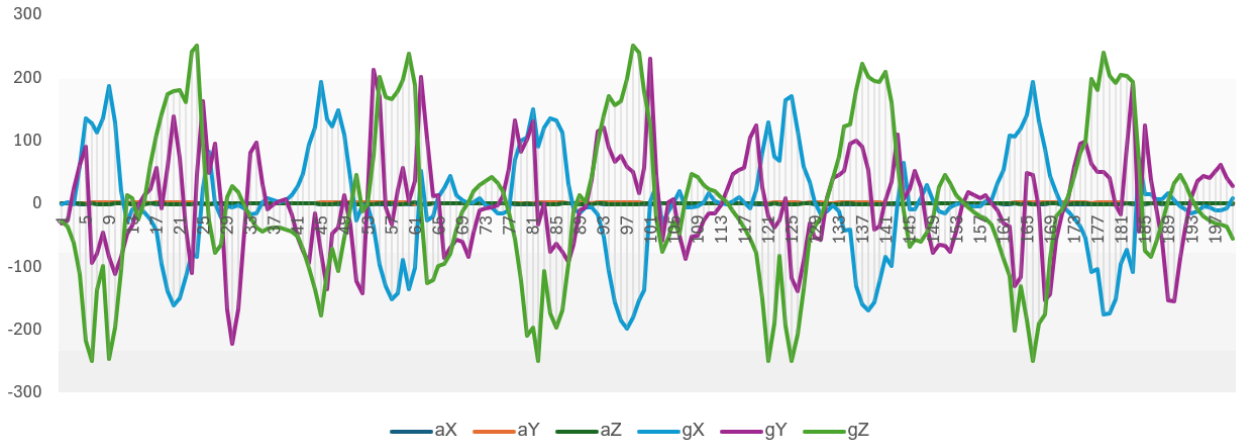
Biểu đồ 7.2 Biểu đồ biểu diễn hành động Walking



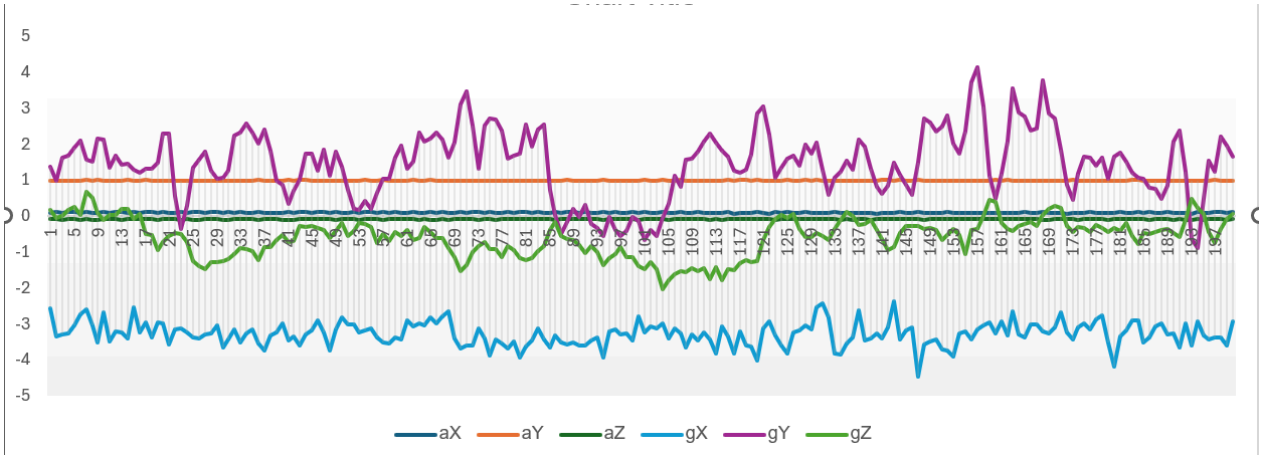
Biểu đồ 7.3 Biểu đồ biểu diễn hành động Sitting



Biểu đồ 7.4 Biểu đồ biểu diễn hành động Running

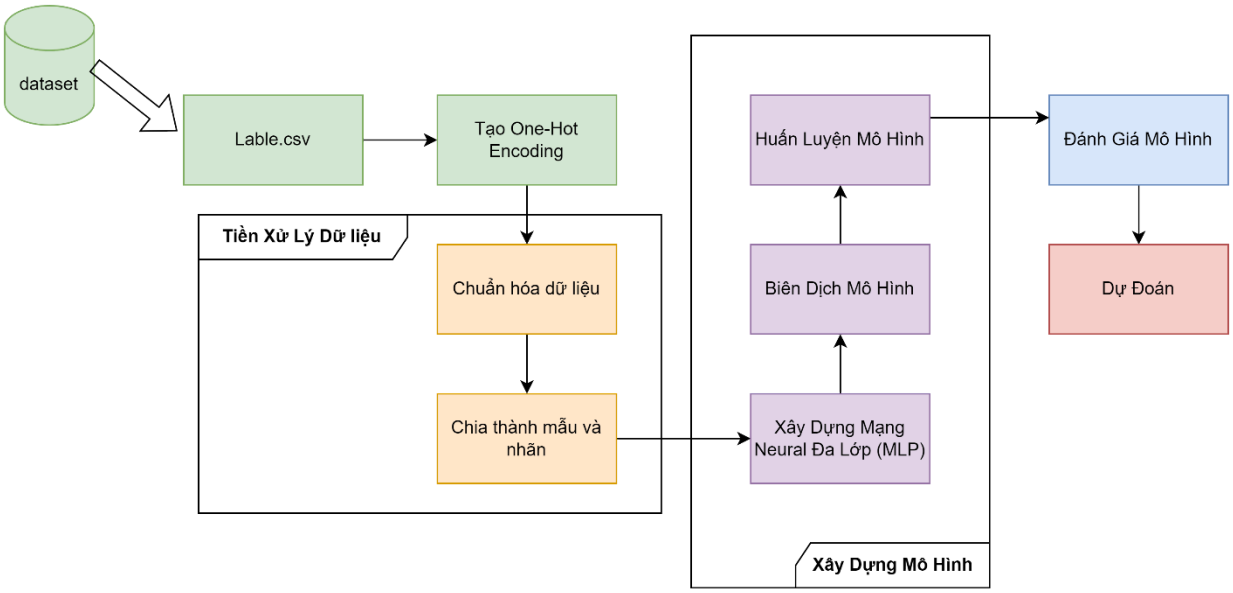


Biểu đồ 7.5 Biểu đồ biểu diễn hành động Jumping



Biểu đồ 7.6 Biểu đồ biểu diễn hành động Standing

8. Huấn luyện mô hình học máy



Sơ đồ 8.1 Sơ đồ khối trình bày quy trình xây dựng mô hình học máy

Xây dựng mô hình

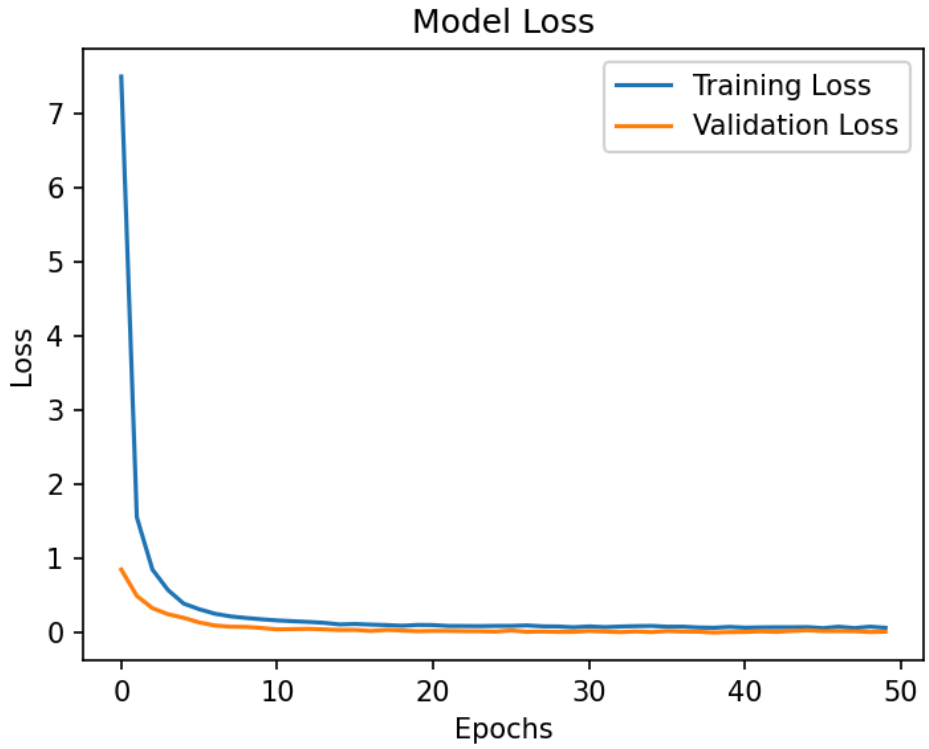
```
model = Sequential()
model.add(keras.layers.Flatten(input_shape=(SAMPLES_PER_GESTURE, SENSOR_NUM)))
model.add(keras.layers.Dense(16, activation='relu'))
model.add(keras.layers.Dropout(0.3))
model.add(keras.layers.Dense(NUM_CLASSESS, activation='softmax'))

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
model.summary()

history = model.fit(x_train,y_train, epochs=50, validation_split =0.1)

model.save('gesture_model.tflite')
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 360)               0

 dense (Dense)               (None, 16)                5776

 dropout (Dropout)           (None, 16)                0

 dense_1 (Dense)             (None, 5)                 85

=================================================================
Total params: 5,861
Trainable params: 5,861
Non-trainable params: 0
```

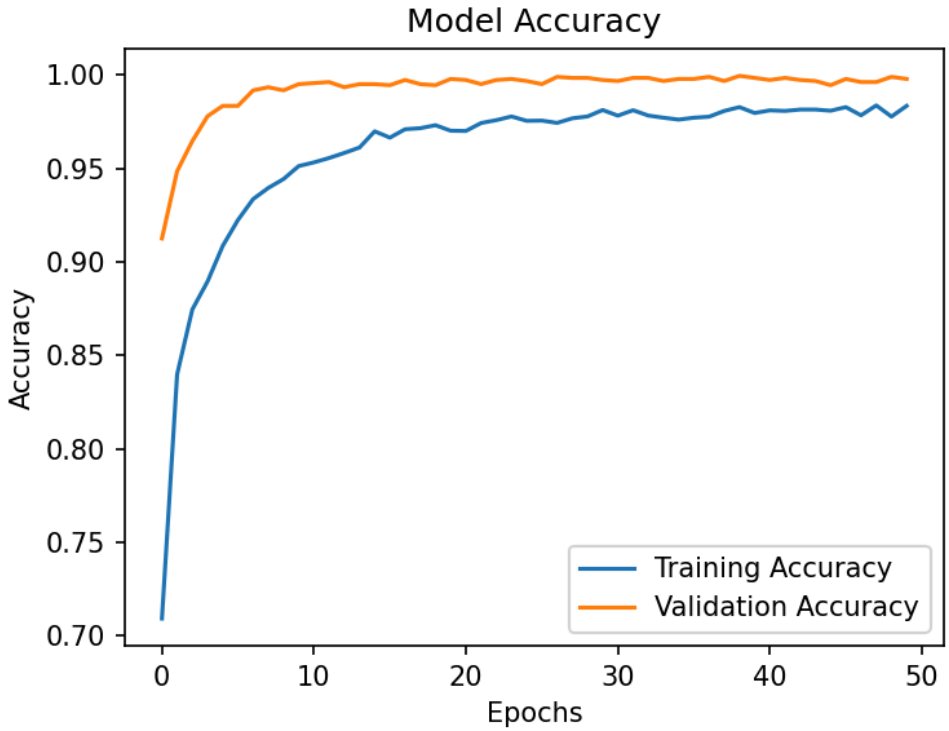
Xây dựng Model MLP train AI

```
Epoch 41/50
508/508 [==============================] - 1s 2ms/step - loss: 0.0730 - accuracy: 0.9810 - val_loss: 0.0153 - val_accuracy: 0.9972
Epoch 42/50
508/508 [==============================] - 1s 2ms/step - loss: 0.0767 - accuracy: 0.9807 - val_loss: 0.0224 - val_accuracy: 0.9983
Epoch 43/50
508/508 [==============================] - 1s 2ms/step - loss: 0.0788 - accuracy: 0.9814 - val_loss: 0.0159 - val_accuracy: 0.9972
Epoch 44/50
508/508 [==============================] - 1s 2ms/step - loss: 0.0789 - accuracy: 0.9814 - val_loss: 0.0288 - val_accuracy: 0.9967
Epoch 45/50
508/508 [==============================] - 1s 2ms/step - loss: 0.0806 - accuracy: 0.9808 - val_loss: 0.0380 - val_accuracy: 0.9945
Epoch 46/50
508/508 [==============================] - 1s 2ms/step - loss: 0.0666 - accuracy: 0.9827 - val_loss: 0.0257 - val_accuracy: 0.9978
Epoch 47/50
508/508 [==============================] - 1s 1ms/step - loss: 0.0862 - accuracy: 0.9783 - val_loss: 0.0262 - val_accuracy: 0.9961
Epoch 48/50
508/508 [==============================] - 1s 1ms/step - loss: 0.0672 - accuracy: 0.9836 - val_loss: 0.0246 - val_accuracy: 0.9961
Epoch 49/50
508/508 [==============================] - 1s 1ms/step - loss: 0.0867 - accuracy: 0.9777 - val_loss: 0.0138 - val_accuracy: 0.9989
Epoch 50/50
508/508 [==============================] - 1s 1ms/step - loss: 0.0716 - accuracy: 0.9834 - val_loss: 0.0190 - val_accuracy: 0.9978
```

Quá trình train AI



Biểu đồ Training Loss qua các Epochs



Biểu đồ Training Accuracy qua các Epochs

Nhận xét: Biểu đồ cho thấy mất mát huấn luyện và kiểm tra giảm dần theo thời gian, chứng tỏ mô hình đang học và cải thiện. Mất mát huấn luyện và kiểm tra dần hội tụ, cho thấy mô hình không bị overfitting nhiều. Mất mát kiểm tra dao động nhiều hơn nhưng điều này bình thường. Sau một số epoch, cả hai mất mát đều ổn định, cho thấy mô hình có thể đã hội tụ. Việc bỏ qua 100 epoch đầu tiên giúp tập trung vào giai đoạn sau, nơi sự thay đổi mất mát rõ ràng hơn. Nhìn chung, mô hình học tốt và ổn định sau một số epoch.

PHẦN III: HƯỚNG DẪN CÀI ĐẶT CHƯƠNG TRÌNH

1. IDE và Server hỗ trợ

1.1 *Arduino*

Arduino là một nền tảng mã nguồn mở dựa trên phần cứng và phần mềm dễ sử dụng. Arduino bao gồm các bảng mạch vi điều khiển và một môi trường phát triển tích hợp (IDE) để viết mã và tải nó lên bảng. Dưới đây là một số điểm nổi bật:

Phần cứng: Arduino sử dụng các bảng mạch vi điều khiển như Arduino Uno, Mega, Nano, v.v., có nhiều chân vào/ra kỹ thuật số và analog để kết nối với các cảm biến, động cơ, đèn LED và các thiết bị khác.

Phần mềm: Arduino IDE hỗ trợ viết mã trong ngôn ngữ lập trình Arduino (dựa trên C++), biên dịch và tải mã lên bảng mạch thông qua cổng USB.

Ứng dụng: Arduino được sử dụng rộng rãi trong giáo dục, các dự án cá nhân, nguyên mẫu thiết bị điện tử, và IoT.

1.2 *Pycharm / Jupyter Notebook*

PyCharm: Một môi trường phát triển tích hợp (IDE) mạnh mẽ dành cho Python, phát triển bởi JetBrains. Hỗ trợ viết, chạy, gỡ lỗi và quản lý mã

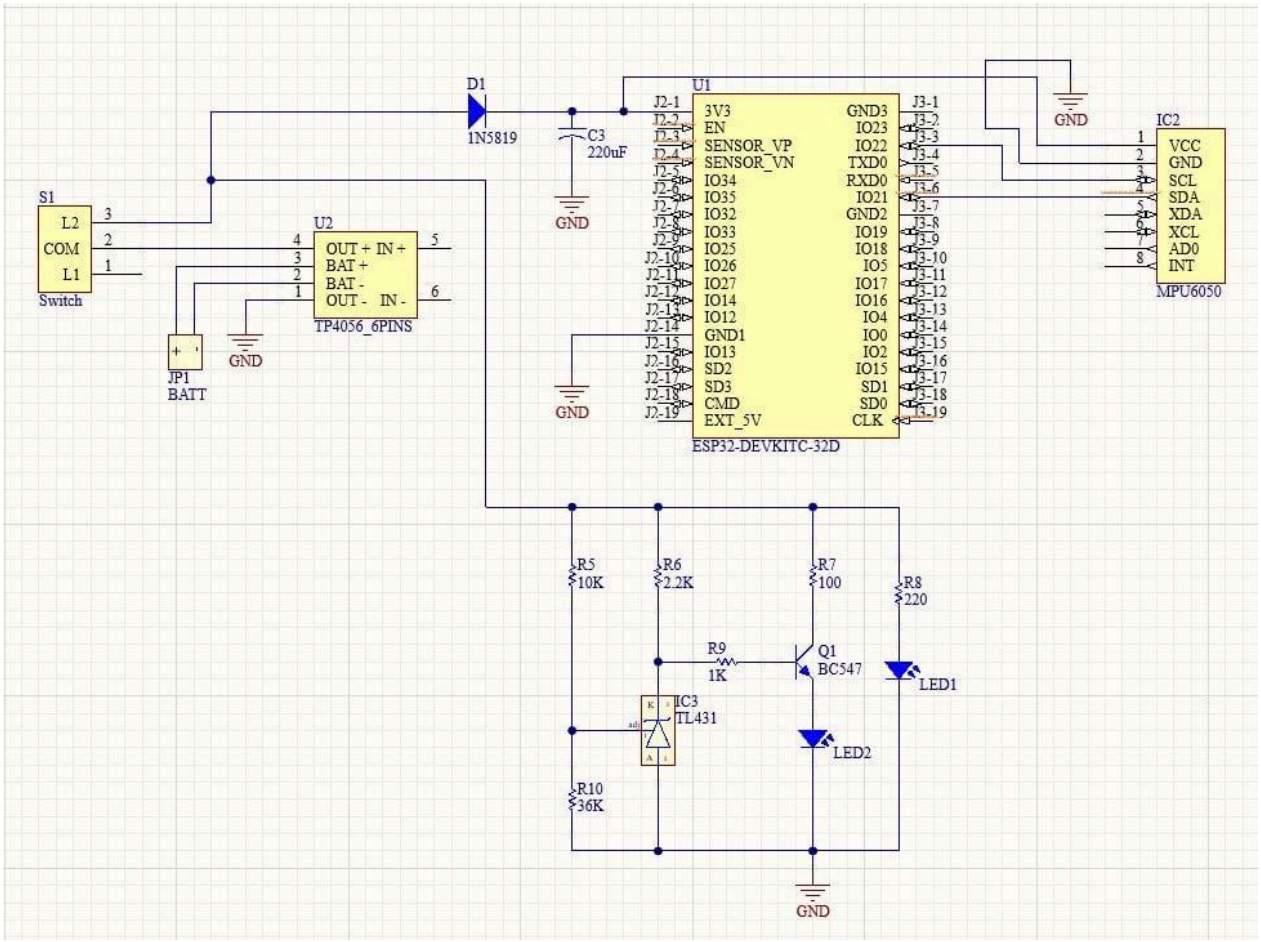
nguồn Python với các tính năng như tự động hoàn thành mã, phát hiện lỗi, quản lý môi trường ảo, tích hợp với các hệ thống kiểm soát phiên bản (Git), và nhiều công cụ hỗ trợ khác.

Jupyter Notebook: Một ứng dụng web mã nguồn mở cho phép tạo và chia sẻ các tài liệu có chứa mã nguồn, phương trình, trực quan hóa và văn bản. Hỗ trợ nhiều ngôn ngữ lập trình (đặc biệt là Python), cho phép chạy mã trực tiếp trong trình duyệt, dễ dàng kết hợp với các thư viện trực quan hóa dữ liệu như Matplotlib, Seaborn, và các công cụ học máy như TensorFlow, scikit-learn.

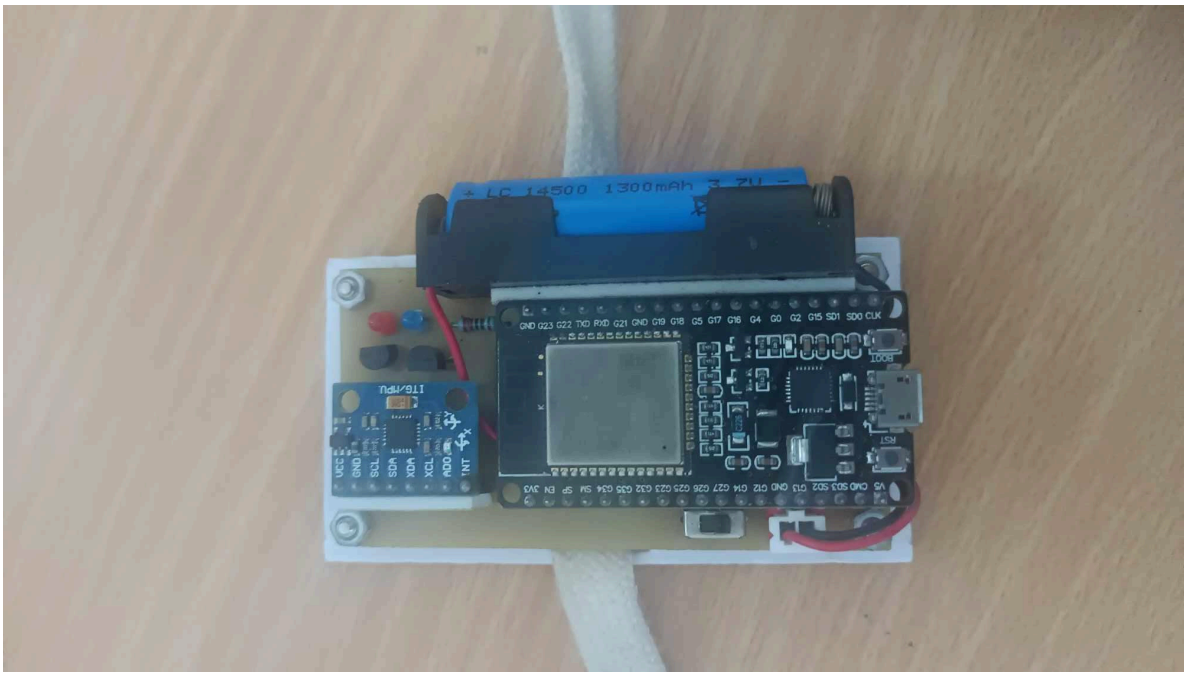
1.3 *Xampp*

XAMPP là một phần mềm mã nguồn mở giúp thiết lập môi trường phát triển web trên máy tính cá nhân. Bao gồm Apache (máy chủ web), MariaDB (hệ quản trị cơ sở dữ liệu), PHP và Perl (ngôn ngữ lập trình). Dễ dàng cài đặt và cấu hình, cung cấp một môi trường hoàn chỉnh để phát triển và kiểm thử các ứng dụng web mà không cần phải triển khai lên máy chủ thực. Ứng dụng Phát triển các trang web và ứng dụng web động sử dụng PHP và MySQL, thử nghiệm các ứng dụng trước khi triển khai lên môi trường sản xuất.

2. Bộ thiết bị cảm biến chuyển động



Sơ đồ 2.1 Sơ đồ nguyên lý thiết kế bộ cảm biến



Hình 2.2 Bộ cảm biến chuyển động

3. Thu dataset từ ESP32, MPU6050 ở IDE Arduino xử lý trên Server.py

Quy trình thu data cho label Running

Bước 1: Khởi tạo Server

```
HOST = '10.252.14.243' # Địa chỉ IP của máy tính chạy server
PORT = 8090            # Cổng mà server lắng nghe
csv_file_path = 'Running.csv' # Tên file lưu dữ liệu nhận được
```

Bước 2: Tách chuỗi thành danh sách

```
def split_string_to_list(string):
    parts = string.split("#")
    result = []
    for part in parts:
        sub_list = part.split(",")
        sub_list = [float(item) for item in sub_list]
        result.append(sub_list)
    return result
```

Hàm này tách chuỗi dữ liệu nhận được từ client thành các danh sách con. Mỗi danh sách con chứa các giá trị float từ cảm biến.

Bước 3: Xử lý kết nối từ client

```python
def handle_connection(conn, addr):
    print(f"Connected by {addr}")
    sampleData = ''
    while True:
        data = conn.recv(10240000)
        if not data:
            break
        decoded_data = data.decode('utf-8')
        sampleData += decoded_data
    print("sampleData: " + sampleData + "\n")
    data_list = split_string_to_list(sampleData.strip(",").strip("#"))

    with open(csv_file_path, mode='a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['aX', 'aY', 'aZ', 'gX', 'gY', 'gZ'])
        writer.writerows(data_list)

    print("Dữ liệu đã được lưu vào tệp " + csv_file_path + " thành công.")
    conn.close()
```

Hàm handle_connection xử lý từng kết nối từ client. Nó nhận dữ liệu, giải mã và tách dữ liệu thành các danh sách con. Sau đó, ghi dữ liệu này vào tệp CSV.

Bước 4 : Chạy server

```python
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    print(f"Listening on {HOST}:{PORT}")
    while True:
        conn, addr = s.accept()
        handle_connection(conn, addr)
```

Server được khởi động và lắng nghe các kết nối từ client. Mỗi khi có kết nối mới, hàm handle_connection sẽ được gọi để xử lý và lưu dữ liệu vào tệp CSV.

Bước 5 : Cấu hình kết nối client (Arduino) đến Server

```
// Tên và mật khẩu mạng WiFi
const char *ssid = "PTIT.HCM_SV";
const char *password = "";

// Thông tin kết nối server
const char *host = "10.252.14.243";
const int port = 8090;

// Các thông số cấu hình
const int numFeature = 6;
int16_t numSample = 0;
const int maxNumSample = 480;
```

```
// Kiểm tra kết nối với MPU6050
if (!mpu.testConnection()) {
  Serial.println("MPU6050 connection failed!");
  while (1);
}

// Kết nối WiFi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(100);
  Serial.print(".");
}
Serial.println("\nWiFi connected");
Serial.print("IP address: ");
Serial.println(WiFi.localIP());
```

```
// Kết nối đến server
if (client.connect(host, port)) {
  Serial.println("Connected to server");
} else {
  Serial.println("Connection to server failed");
}
```

Bước 7 : Thu các số liệu chuẩn bị gửi lên Server

```
void loop() {
  delay(timedelay);
  if (shouldStart) {
    float data[numFeature];

    // Thu thập dữ liệu từ cảm biến và lưu vào mảng
    int16_t ax, ay, az;
    int16_t gx, gy, gz;
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    // Chuyển đổi dữ liệu thành các giá trị float và lưu vào mảng
    data[0] = ax / 16384.0;
    data[1] = ay / 16384.0;
    data[2] = az / 16384.0;
    data[3] = gx / 131.0;
    data[4] = gy / 131.0;
    data[5] = gz / 131.0;
```

```
  // Lưu dữ liệu vào mảng buffer
  for (int i = 0; i < numFeature; i++) {
    dataBuffer[numSample][i] = data[i];
    Serial.print(dataBuffer[numSample][i]);
    Serial.print(" ");
  }

  Serial.println();
  numSample++;

  // Nếu đã đủ mẫu, gửi dữ liệu lên server và đặt lại các biến
  if (numSample >= maxNumSample) {
    delay(1000);
    sendDataToServer();
    shouldStart = false;
  }
```

Bước 8: Gửi lên Server

```
// Gửi dữ liệu từ mảng buffer lên server
for (int i = 0; i < maxNumSample; i++) {
  for (int j = 0; j < numFeature; j++) {
    data_send += String(dataBuffer[i][j]);
    if (j < numFeature - 1) {
      data_send += ",";
    }
  }
  if (i < maxNumSample - 1) {
    data_send += "#";
  }
}
```

4. **Nhúng Model vào bộ thiết bị cảm biến**

Các thư viện cần thiết

```
#include <TensorFlowLite_ESP32.h>
#include <MPU6050.h>
#include <WiFi.h>
#include <Wire.h>
#include "tensorflow/lite/micro/all_ops_resolver.h"
#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/micro/system_setup.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "gesture_model.h"
```

Khởi Tạo Bộ Nhớ và TensorFlow Lite

```
// Tạo một bộ nhớ tĩnh cho TensorFlow Lite
constexpr int tensorArenaSize = 8 * 1024;
byte tensorArena[tensorArenaSize] __attribute__((aligned(16)));

// Khởi tạo các biến toàn cục cho TensorFlow Lite
tflite::MicroErrorReporter tflErrorReporter;
tflite::AllOpsResolver tflOpsResolver;
const tflite::Model* tflModel = nullptr;
tflite::MicroInterpreter* tflInterpreter = nullptr;
TfLiteTensor* tflInputTensor = nullptr;
TfLiteTensor* tflOutputTensor = nullptr;
```

Khởi tạo TensorFlow Lite

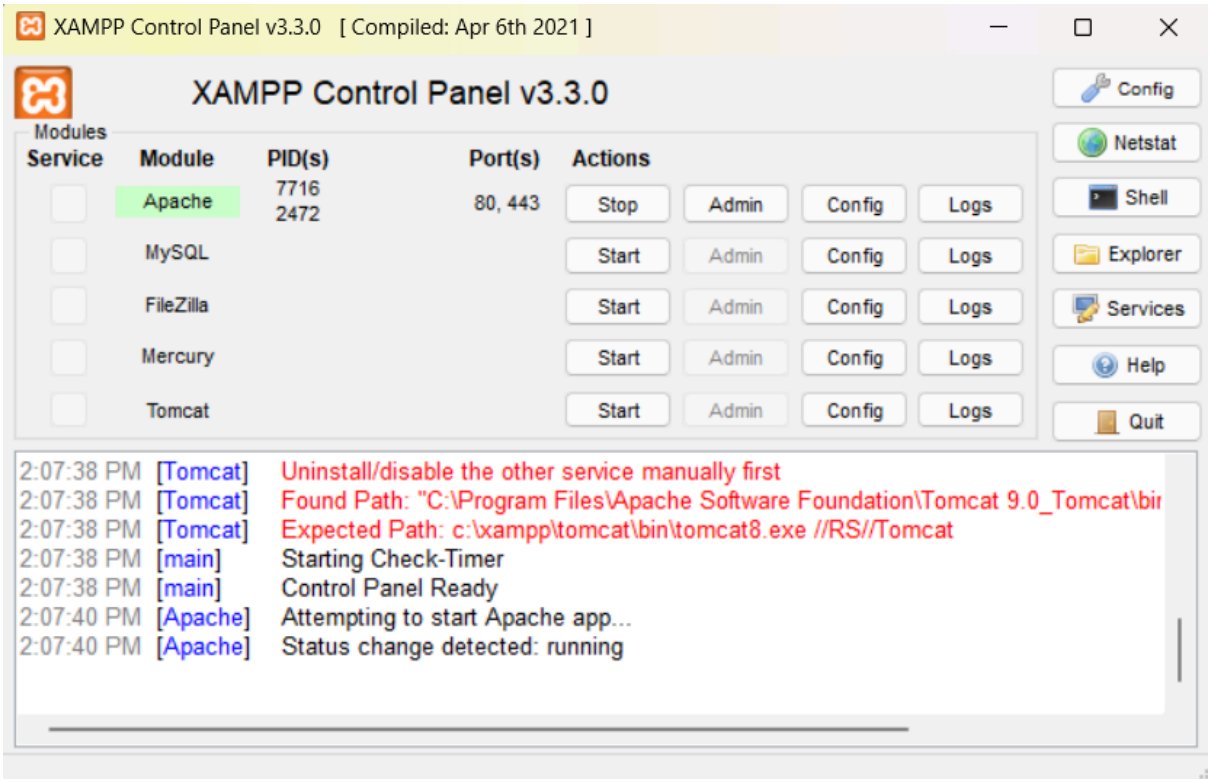
```
// Create an interpreter to run the model
tflInterpreter = new tflite::MicroInterpreter(tflModel, tflOpsResolver, tensorArena, tensorArenaSize, &tflErrorReporter);

// Allocate memory for the model's input and output tensors
tflInterpreter->AllocateTensors();

// Get pointers for the model's input and output tensors
tflInputTensor = tflInterpreter->input(0);
tflOutputTensor = tflInterpreter->output(0);
```

Xử Lý Dữ Liệu và Dự Đoán

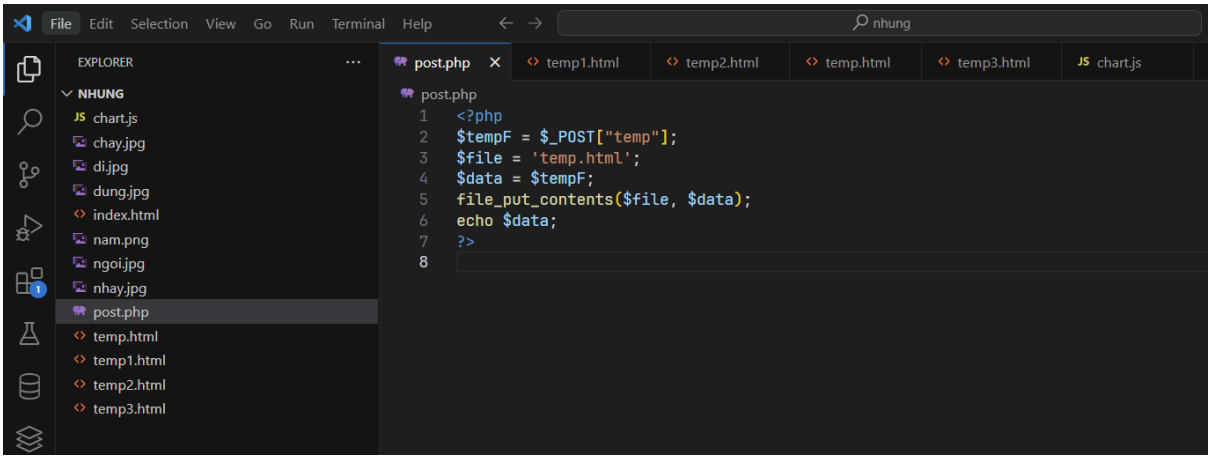
```
void taskProcessData(void *parameter) {
    while (1) {
        if (gotData) {
            for (int i = 0; i < num_timesteps * num_features; i++) {
                tflInputTensor->data.f[i] = input_data[i];
            }

            TfLiteStatus invokeStatus = tflInterpreter->Invoke();
            if (invokeStatus != kTfLiteOk) {
                Serial.println("Invoke failed!");
                return;
            }
            int max_index = 0;
            float max_value = tflOutputTensor->data.f[0];
            for (int i = 1; i < NUM_GESTURES; i++) {
                if (tflOutputTensor->data.f[i] > max_value) {
                    max_index = i;
                    max_value = tflOutputTensor->data.f[i];
                }
            }
            predicted_gesture = max_index;
            gotData = false;
        }
        vTaskDelay(20 / portTICK_PERIOD_MS);
        yield();
    }
}
```

5. Thiết lập Web Server và hiển thị kết quả dự đoán

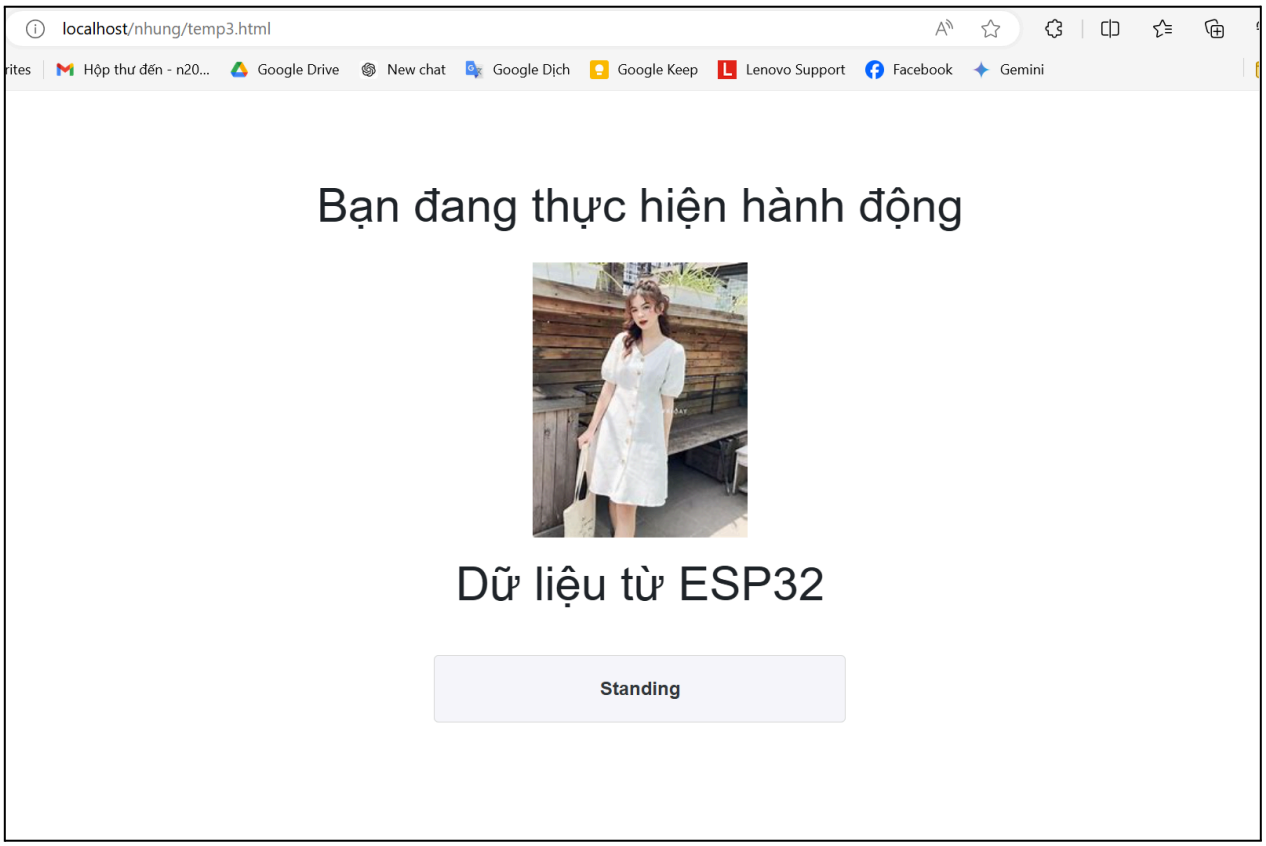
Bước 1: Cài đặt Xampp



Bước 2 : Viết server nhận dữ liệu từ arduino gửi lên



Trang kết quả



PHẦN IV : KẾT LUẬN

Bài báo cáo đã thực hiện tích hợp giữa cảm biến chuyển động MPU6050, kết nối WiFi, và mô hình nhận diện cử chỉ sử dụng TensorFlow Lite trên nền tảng ESP32. Bằng cách này, chúng ta có thể thu thập dữ liệu về các hoạt động vận động từ cảm biến, xử lý dữ liệu này và gửi tới một server từ xa để phân tích hoặc lưu trữ.

Trong quá trình thực hiện, nhóm đã thiết lập kết nối WiFi để truy cập mạng và gửi dữ liệu, cũng như khởi tạo cảm biến MPU6050 để thu thập thông tin về gia tốc và tốc độ góc của các hoạt động vận động. Dữ liệu này sau đó được chuẩn hóa và đưa vào một mô hình nhận diện cử chỉ TensorFlow Lite, để dự đoán hoạt động đang diễn ra.

Để thực hiện các công việc này, nhóm đã sử dụng các công cụ và thư viện như TensorFlow Lite, các thư viện WiFi và cảm biến, cùng với một số tính năng như FreeRTOS task để xử lý dữ liệu và tạo ra một quy trình làm việc song song.

Tích hợp này mở ra nhiều ứng dụng trong việc theo dõi và phân tích các hoạt động vận động, từ giám sát sức khỏe đến giáo dục và giải trí. Vẫn chưa đến mức độ chỉnh xác nhưng có thể đáp ứng về nhu cầu phát hiện chuyển động.