**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN 2**

Logo

Description automatically generated

**BÁO CÁO ĐỒ ÁN**

XÂY DỰNG CÁC HỆ THỐNG NHÚNG

**Đề tài:**   
**HỆ THỐNG NHẬN DIỆN GIỌNG NÓI**

|  |  |
| --- | --- |
| **Sinh viên thực hiện:** | **NGUYỄN NHẬT MINH**  **NGUYỄN TẤN LỘC**  **HUỲNH NGỌC TÂN**  **TRẦN VŨ PHƯƠNG NAM** |
| **Lớp:** | **D21CQCNPM01-N** |
| **Mã số sinh viên:** | **N21DCCN053**  **N21DCCN049**  **N21DCCN074**  **N21DCCN141** |
| **Nhóm:** | **3** |
| **Giảng viên hướng dẫn:** | **NGUYỄN TRỌNG KIÊN** |

*TP Hồ Chí Minh, ngày 04 tháng 06 năm 202**5*

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN 2**

**NHẬN XÉT VÀ CHẤM ĐIỂM CỦA GIẢNG VIÊN**

Họ và tên giảng viên: **Nguyễn Trọng Kiên.**

Tên đề tài: **Hệ thống nhận diện giọng nói.**

Nội dung nhận xét:

**Điểm:**

Bằng số:

Bằng chữ:

|  |  |
| --- | --- |
|  | **GIẢNG VIÊN CHẤM**  (Ký, ghi rõ họ tên) |
|  |  |

LỜI MỞ ĐẦU

Trong thời đại công nghệ số, hệ thống nhúng đã trở thành một phần không thể thiếu trong nhiều lĩnh vực, từ công nghiệp sản xuất, tự động hóa, y tế, giao thông, đến đời sống hằng ngày. Những hệ thống này đóng vai trò trung tâm trong việc xử lý dữ liệu, điều khiển thiết bị và tối ưu hóa hoạt động của nhiều hệ thống phức tạp. Với khả năng tích hợp phần mềm và phần cứng để thực hiện những nhiệm vụ chuyên biệt, hệ thống nhúng giúp tăng cường hiệu suất hoạt động, giảm thiểu sai sót và mở ra nhiều cơ hội phát triển trong các ngành khác nhau.

Việc áp dụng hệ thống nhúng mang lại ý nghĩa to lớn trong sự phát triển khoa học công nghệ. Chúng không chỉ hỗ trợ con người trong việc điều khiển và giám sát từ xa mà còn góp phần nâng cao hiệu quả hoạt động của các hệ thống tự động. Hơn nữa, sự phát triển mạnh mẽ của trí tuệ nhân tạo và công nghệ kết nối đã mở ra nhiều tiềm năng mới trong việc tối ưu hóa hệ thống nhúng, giúp chúng trở nên thông minh và linh hoạt hơn.

Đề tài "Hệ thống nhận diện giọng nói" được chọn nhằm khai thác những lợi ích mà hệ thống nhúng có thể mang lại trong lĩnh vực nhận diện và xử lý tín hiệu âm thanh. Việc ứng dụng công nghệ nhận diện giọng nói vào các hệ thống nhúng không chỉ giúp tăng khả năng tương tác giữa con người và máy móc, mà còn hỗ trợ nhiều lĩnh vực như tự động hóa, trợ lý ảo, bảo mật, và nhiều ứng dụng khác. Mục tiêu của đề tài là nghiên cứu và phát triển một hệ thống có khả năng nhận diện giọng nói một cách hiệu quả và chính xác.

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến thầy Nguyễn Trọng Kiên đã tận tình giảng dạy và truyền đạt cho chúng em những kiến thức quý báu trong suốt quá trình học tập tại trường. Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến thầy Nguyễn Trọng Kiên đã luôn đồng hành, hướng dẫn và hỗ trợ chúng em trong suốt quá trình thực hiện bài tập nhóm này.

Chúng em cũng xin cảm ơn bạn bè và các thành viên trong nhóm đã luôn sẵn lòng giúp đỡ, chia sẻ kinh nghiệm và động viên tinh thần, giúp chúng em vượt qua những khó khăn trong quá trình thực hiện bài tập nhóm.

Cuối cùng, chúng em muốn gửi lời cảm ơn đến gia đình, những người luôn là chỗ dựa vững chắc, tiếp thêm động lực để chúng em hoàn thành tốt nhiệm vụ học tập cũng như bài tập này.

Mặc dù đã cố gắng hết sức, nhưng với kiến thức và thời gian còn hạn chế, bài báo cáo không tránh khỏi những thiếu sót. Chúng em rất mong nhận được ý kiến đóng góp từ thầy và các bạn để hoàn thiện hơn trong tương lai.

Trân trọng cảm ơn!

MỤC LỤC

[I. Giới thiệu 6](#_Toc199912840)

[1. Tên đồ án 6](#_Toc199912841)

[2. Thành viên và trách nhiệm trong đồ án 6](#_Toc199912842)

[3. Mục đích của đồ án 6](#_Toc199912843)

[4. Mục tiêu 6](#_Toc199912844)

[5. Phạm vi thực hiện 7](#_Toc199912845)

[II. Cơ sở lí thuyết 8](#_Toc199912846)

[1. Khái niệm và định nghĩa 8](#_Toc199912847)

[2. Các nghiên cứu liên quan 11](#_Toc199912848)

[3. Các lý thuyết và mô hình áp dụng 12](#_Toc199912849)

[III. Xây dựng hệ thống 16](#_Toc199912850)

[1. Kiến trúc hệ thống 16](#_Toc199912851)

[2. Chuẩn bị cho hệ thống 22](#_Toc199912852)

[3. Xây dựng thành phần hệ thống 31](#_Toc199912853)

[IV. Thực nghiệm và đánh giá 36](#_Toc199912854)

[1. Triển khai thực tế 36](#_Toc199912855)

[2. Kết quả và đánh giá 40](#_Toc199912856)

DANH MỤC HÌNH VẼ

[Hình 1: Sơ đồ khối tổng thể của hệ thống 16](#_Toc199912857)

[Hình 2: Khối thu âm của hệ thống 18](#_Toc199912858)

[Hình 3: Khối truyền dữ liệu của hệ thống 19](#_Toc199912859)

[Hình 4: Khối xử lí đặc trưng của hệ thống 20](#_Toc199912860)

[Hình 5: Khối dự đoán của hệ thống 21](#_Toc199912861)

[Hình 6: Kiến trúc mô hình CNN 1D 27](#_Toc199912862)

[Hình 7: Biểu đồ hiển thị quá trình huấn luyện mô hình 30](#_Toc199912863)

[Hình 8: Mạch ESP32 sau khi kết nối với mic và kết nối với máy tính 36](#_Toc199912864)

[Hình 9: Quá trình nạp code vào cho ESP32 37](#_Toc199912865)

[Hình 10: Tổng quan code server thu âm thanh 37](#_Toc199912866)

[Hình 11: Tổng quan code Extracting Feature server 38](#_Toc199912867)

[Hình 12: Sử dụng mic để thu âm thanh podcast từ loa máy tính để ESP32 nhận diện 39](#_Toc199912868)

[Hình 13: Dữ liệu log từ 2 server thu âm và server lấy đặc trưng 40](#_Toc199912869)

# Giới thiệu

## Tên đồ án

* Hệ thống nhận diện giọng nói.

## Thành viên và trách nhiệm trong đồ án

|  |  |  |
| --- | --- | --- |
| **Mã số sinh viên** | **Họ và tên** | **Trách nhiệm trong đồ án** |
| N21DCCN053  (Nhóm trưởng) | Nguyễn Nhật Minh | - Tìm hiểu xây dựng CNN model.  - Nhúng model. |
| N21DCCN049 | Nguyễn Tấn Lộc | - Xây dựng JS server để nhận, xử lí dữ liệu.  - Làm báo cáo. |
| N21DCCN074 | Huỳnh Ngọc Tân | - Viết ESP32 code gửi data lên server.  - Nhúng model. |
| N21DCCN151 | Trần Vũ Phương Nam | - Thu thập data, tìm hiểu human speech features.  - Xây dựng server trích xuất human speech feature. |

## Mục đích của đồ án

* Vận dụng các kiến thức đã học về hệ thống nhúng, xử lý tín hiệu số, lập trình nhúng và trí tuệ nhân tạo để xây dựng một hệ thống tập trung vào việc xác định người nói dựa trên đặc trưng giọng nói, có khả năng phân biệt và xác định danh tính của từng người thông qua phân tích đặc điểm âm thanh riêng biệt.
* Nắm bắt quy trình thu âm, xử lý tín hiệu âm thanh, trích xuất đặc trưng và nhận diện giọng nói thông qua tích hợp mô hình AI (CNN 1D) lên vi điều khiển, tối ưu hóa tài nguyên bộ nhớ và tốc độ xử lý.

## Mục tiêu

* Thu âm và xử lý tín hiệu với tần số 16kHz, độ phân giải 16-bit.
* Đạt độ chính xác nhận diện ≥ 90% với 2 người dùng trong môi trường yên tĩnh.
* Độ trễ toàn hệ thống < 500ms từ khi thu âm đến khi xuất kết quả.

## Phạm vi thực hiện

* Giới hạn phần cứng:
* Vi điều khiển: ESP32 Dev Module.
* Microphone: INMP441 (giao tiếp I2S).
* Giới hạn phần mềm:
* Số lượng người dùng nhận diện: 2 người.
* Môi trường thử nghiệm: Sử dụng âm thanh podcast từ 2 người tham gia cùng với background noise để train mô hình trí tuệ nhân tạo nhận diện giọng nói. Sau đó sử dụng ESP32 để thu âm và từ âm thanh thu được sẽ sử dụng model đã huấn luyện trước đó để nhận diện giọng nói thu âm được là của ai hay là noise.

# Cơ sở lí thuyết

## Khái niệm và định nghĩa

1. **Nhận diện giọng nói**

* Nhận diện giọng nói (Speaker Recognition) là quá trình sử dụng công nghệ để xác định hoặc xác thực danh tính của người nói dựa trên đặc trưng giọng nói. Hệ thống này phân tích âm thanh, trích xuất các đặc trưng riêng biệt của giọng nói và sử dụng mô hình học máy để phân loại người nói.
* Nhận diện giọng nói có hai loại chính:
* Nhận dạng người nói (Speaker Identification): Xác định danh tính người nói từ một tập hợp đối tượng đã biết. Ví dụ, hệ thống có thể phân loại giọng nói là của "Anh bạn thân" hoặc "Giang ơi".
* Xác thực người nói (Speaker Verification): Kiểm tra xem giọng nói có khớp với danh tính được cung cấp hay không, thường dùng trong bảo mật sinh trắc học (Voice Authentication).
* Nhận diện giọng nói có nhiều ứng dụng thực tế, bao gồm:
* Bảo mật và xác thực: Dùng trong hệ thống mở khóa bằng giọng nói, truy cập tài khoản bằng sinh trắc học giọng nói.
* Hệ thống trợ lý ảo: Nhận diện giọng nói để tùy chỉnh phản hồi của trợ lý AI theo từng người dùng.
* Điều khiển thiết bị IoT: Cho phép bật/tắt thiết bị thông minh bằng giọng nói.
* Hệ thống kiểm soát truy cập: Dùng trong văn phòng, ngân hàng để xác thực danh tính người dùng.
* Dịch vụ chăm sóc khách hàng: Phân loại giọng nói của khách hàng để tối ưu hóa trải nghiệm người dùng.

1. **Giao tiếp I2S (Inter-IC Sound)**

* I2S (Inter-IC Sound) là giao thức truyền tín hiệu âm thanh số giữa các thiết bị điện tử, thường được sử dụng để kết nối microphone số với vi điều khiển như ESP32. Nó đảm bảo tín hiệu âm thanh được gửi đi chính xác và đồng bộ hóa mà không cần chuyển đổi sang dạng analog.
* Giao thức I2S gồm ba tín hiệu chính:
* Bit Clock (BCK - GPIO 26): Xác định thời điểm truyền từng bit dữ liệu.
* Word Select (WS - GPIO 25): Đồng bộ hóa kênh âm thanh (trái/phải).
* Data (SD - GPIO 32): Chứa dữ liệu âm thanh số từ microphone.

ESP32 sử dụng I2S driver để xử lý tín hiệu từ microphone INMP441, chuyển đổi âm thanh thu được thành mảng số nguyên 16-bit.

* Ứng dụng:
* Kết nối microphone số với vi điều khiển, giúp ghi âm tín hiệu mà không cần bộ ADC.
* Nhận tín hiệu âm thanh số từ nhiều nguồn như microphone MEMS, DAC, bộ giải mã âm thanh.
* Ứng dụng trong hệ thống nhận diện giọng nói, cải thiện độ chính xác so với tín hiệu analog.
* Dùng trong thiết bị IoT âm thanh, giúp xử lý lệnh thoại, giao tiếp AI, điều khiển bằng giọng nói.

1. **Định dạng PCM (Pulse Code Modulation)**

* PCM (Pulse Code Modulation) là phương pháp số hóa tín hiệu âm thanh bằng cách lấy mẫu và lượng tử hóa tín hiệu analog thành dữ liệu số. Đây là một định dạng âm thanh không nén, được sử dụng phổ biến trong xử lý tín hiệu, ghi âm, và truyền dữ liệu âm thanh.
* Quá trình chuyển đổi tín hiệu analog sang PCM gồm ba bước chính:
* Lấy mẫu (Sampling) – Xác định giá trị tín hiệu tại các thời điểm cụ thể.
* Lượng tử hóa (Quantization) – Chuyển đổi giá trị tín hiệu thành số nguyên theo độ sâu bit.
* Mã hóa (Encoding) – Biểu diễn tín hiệu dưới dạng nhị phân để lưu trữ hoặc truyền đi.

ESP32 sử dụng PCM 16-bit, 16kHz, nghĩa là mỗi mẫu có độ sâu 16-bit và tần số lấy mẫu 16,000 lần/giây, đủ để nhận diện giọng nói chính xác.

* Ứng dụng:
* Ghi âm giọng nói, lưu dữ liệu thô để xử lý bằng MFCC.
* Truyền dữ liệu âm thanh số, giúp giao tiếp không mất dữ liệu giữa microphone và ESP32.
* Xử lý tín hiệu trong hệ thống nhận diện giọng nói, cho phép phân tích từng khung âm thanh.
* Ứng dụng trong hệ thống IoT, giúp thu và xử lý lệnh thoại với độ trễ thấp.

1. **Đặc trưng MFCC (Mel-Frequency Cepstrum Coefficient)**

* MFCC (Mel-Frequency Cepstrum Coefficient) là một nhóm đặc trưng dựa trên phổ tần số, giúp phân tích và nhận diện giọng nói. MFCC chuyển đổi tín hiệu âm thanh thành một dạng biểu diễn số gọn, cho phép mô hình học máy xác định người nói dựa trên đặc điểm giọng của họ.
* Vai trò của MFCC trong nhận diện giọng nói:
* Mô phỏng cách con người nghe âm thanh: Tai người không cảm nhận tần số tuyến tính, mà theo thang Mel, nơi tần số thấp quan trọng hơn.
* Tách thông tin quan trọng của giọng nói: Loại bỏ thông tin dư thừa, chỉ giữ lại đặc trưng giúp phân biệt người nói.
* Hiệu quả và gọn nhẹ: Thay vì phân tích toàn bộ phổ tần, MFCC chỉ sử dụng 12–13 hệ số, giúp mô hình học máy xử lý nhanh hơn.
* Quá trình trích xuất MFCC gồm các bước:
* Pre-emphasis – Tăng cường thành phần tần số cao để cân bằng năng lượng tín hiệu.
* Framing – Chia tín hiệu thành các khung nhỏ (~20–40 ms) để phân tích theo từng đoạn ngắn.
* Windowing – Dùng Hamming window để giảm nhiễu phổ.
* Fast Fourier Transform (FFT) – Chuyển đổi từ miền thời gian sang miền tần số.
* Mel Filter Bank – Dùng bộ lọc thang Mel để mô phỏng cảm nhận âm thanh của con người.
* Logarithm of Amplitude – Chuyển đổi biên độ sang thang log, như cách tai người cảm nhận âm lượng.
* Discrete Cosine Transform (DCT) – Giảm số chiều dữ liệu, giữ lại 12–13 hệ số MFCC quan trọng.
* Ứng dụng của MFCC
* Nhận diện giọng nói – Dùng để xác định danh tính người nói trong các hệ thống bảo mật sinh trắc học.
* Nhận dạng lời nói – Dùng trong Siri, Google Assistant để phân tích ngôn ngữ nói.
* Nhận diện cảm xúc từ giọng – Giúp phân tích cảm xúc của người nói thông qua âm sắc.
* Phân loại nhạc – Dùng để nhận diện thể loại nhạc dựa trên đặc trưng âm thanh.
* Cải thiện độ chính xác bằng MFCC
* Kết hợp MFCC với các đặc trưng bổ sung như chroma features, spectral contrast, để phân tích giọng nói tốt hơn.
* Tối ưu hóa MFCC cho ESP32 bằng cách giảm số hệ số nhưng vẫn giữ đủ thông tin nhận diện.
* Sử dụng MFCC trong mô hình học sâu như CNN, RNN, giúp tăng độ chính xác khi phân loại người nói.

## Các nghiên cứu liên quan

* Trong quá trình phát triển hệ thống nhận diện giọng nói, có nhiều nghiên cứu trước đây đã đề xuất các phương pháp bảo mật và tối ưu hóa thuật toán nhận diện giọng nói. Dưới đây là hai nghiên cứu tiêu biểu có liên quan chặt chẽ đến hệ thống đang xây dựng:
* **"Voice-Authentication Model Based on Deep Learning for Cloud Environment"**
* Tác giả: Ethar Abdul Wahhab Hachim, Methaq Talib Gaata, Thekra Abbas
* Nội dung chính:
* Nghiên cứu này đề xuất một mô hình xác thực giọng nói trong môi trường điện toán đám mây.
* Sử dụng mã hóa 3DES và RC4 để bảo vệ dữ liệu giọng nói trong quá trình truyền và lưu trữ, giúp hệ thống có độ tin cậy cao hơn.
* Hệ thống tập trung vào bảo mật giọng nói, đảm bảo dữ liệu không bị giả mạo khi truyền giữa các thiết bị và máy chủ đám mây.
* Ý nghĩa với hệ thống ESP32:
* Dù hệ thống của chúng ta không sử dụng điện toán đám mây, nhưng ý tưởng về bảo vệ dữ liệu giọng nói có thể được áp dụng vào ESP32, thông qua các cơ chế mã hóa dữ liệu trước khi gửi lên WebSocket Server.
* **"On-Device MFCC-CNN Voice Recognition System with ESP-32 and Web-Based** **Application"**
* Tác giả: Muhammad Ichsan Ramadani P; Iqbal Burhanul H; Hasbi N. P. Wisudawan; Suatmi Murnani; Hendra Setiawan
* Nội dung chính:
* Nghiên cứu này triển khai hệ thống nhận diện giọng nói trực tiếp trên ESP32, sử dụng thuật toán MFCC để trích xuất đặc trưng giọng nói.
* Dùng mạng CNN để học và phân loại giọng nói, giúp tăng độ chính xác so với các mô hình truyền thống.
* Hệ thống có hiệu suất tốt khi chạy trên ESP32, chứng minh rằng vi điều khiển này đủ mạnh để thực hiện nhận diện giọng nói mà không cần kết nối với máy chủ bên ngoài.
* Ý nghĩa với hệ thống ESP32:
* Nghiên cứu này có định hướng tương tự với hệ thống của chúng ta, đặc biệt ở việc tích hợp MFCC với CNN để nhận diện giọng nói.
* Nó giúp xác nhận rằng ESP32 có khả năng xử lý mô hình học sâu, không chỉ dừng lại ở việc thu âm và truyền dữ liệu.

## Các lý thuyết và mô hình áp dụng

1. **Đặc điểm giọng nói con người**

* Giọng nói của mỗi người là duy nhất, được quyết định bởi các yếu tố sinh học như cấu trúc dây thanh quản, tần số, độ cao (pitch), cường độ (amplitude) và cách phát âm. Hệ thống nhận diện giọng nói sử dụng các đặc điểm này để xác định danh tính người nói dựa trên tín hiệu âm thanh.
* Nhận diện giọng nói dựa vào những đặc trưng quan trọng, bao gồm:
* Tần số cơ bản (Fundamental Frequency - F0): Đặc điểm chính quyết định độ cao giọng nói của một người.
* Độ lệch chuẩn (Standard Deviation): Phản ánh sự biến thiên của tần số, giúp phân biệt âm sắc giữa người nói.
* Cường độ (Amplitude): Xác định mức năng lượng giọng nói, giúp nhận diện cách phát âm.
* Tỷ lệ zero-crossing (Zero-Crossing Rate): Số lần tín hiệu giọng nói chuyển từ âm sang dương, hỗ trợ phân biệt loại âm thanh.
* MFCC (Mel-Frequency Cepstrum Coefficients) là phương pháp biến đổi tín hiệu giọng nói thành một vector đặc trưng số. Hệ thống sử dụng 12 hệ số MFCC, kết hợp với 4 đặc trưng bổ sung (tần số trung bình, độ lệch chuẩn, biên độ và tỷ lệ zero-crossing) để xây dựng vector đặc trưng gồm 16 phần tử.
* Sau khi thu được vector đặc trưng 16 phần tử, hệ thống sử dụng mạng nơ-ron tích chập 1D (CNN 1D) để phân loại người nói. CNN học các mẫu giọng nói riêng biệt từ tập dữ liệu huấn luyện, và khi có dữ liệu mới, nó dự đoán danh tính bằng cách phân tích mẫu đặc trưng MFCC và các yếu tố âm thanh khác.

1. **Thuộc tính phổ âm thanh**

* Thuộc tính phổ âm thanh là cách giọng nói phân bố năng lượng trên các tần số, giúp xác định âm sắc, cao độ (pitch), độ lớn (amplitude), và đặc điểm ngữ âm của từng người. Đây là nền tảng để nhận diện giọng nói vì mỗi người có phổ âm riêng biệt.
* Các thuộc tính phổ quan trọng
* Thành phần tần số (Formants)
  + Là các đỉnh trong phổ âm, giúp phân biệt các nguyên âm.
  + Hai formant đầu tiên (F1, F2) đặc biệt quan trọng trong nhận diện phát âm.
* Cao độ giọng nói (Fundamental Frequency - F0)
  + F0 là tần số thấp nhất trong giọng nói, quyết định giọng cao hay trầm.
  + Giọng nam: 85-180 Hz, giọng nữ: 165-255 Hz, trẻ em >300 Hz.
* Phân bố năng lượng phổ
  + Năng lượng thấp (<1 kHz) → Chủ yếu là nguyên âm.
  + Năng lượng cao (>1 kHz) → Chủ yếu là phụ âm như "s", "f".
* Phổ âm tổng quan (Spectral Envelope - Timbre)
  + Định hình giọng nói riêng biệt của mỗi người dựa vào cấu trúc thanh quản và cách phát âm.
  + Hai người nói cùng một từ nhưng vẫn có sự khác nhau trong phổ âm.
* Tỷ lệ hài hòa với nhiễu (Harmonic-to-Noise Ratio - HNR)
  + HNR cao → Giọng trong trẻo, ít nhiễu (ca sĩ, diễn giả).
  + HNR thấp → Giọng khàn, nhiễu nhiều (giọng nói bị mất chất lượng).
* Tỷ lệ Zero-Crossing (Zero-Crossing Rate - ZCR)
  + ZCR cao → Âm thanh có tần số cao, thường là phụ âm xát như "s".
  + ZCR thấp → Chủ yếu là nguyên âm, giúp nhận diện âm hữu thanh.

1. **Mô hình CNN cho nhận diện giọng nói**

* CNN (Convolutional Neural Network) là mô hình học sâu có khả năng tự động trích xuất đặc trưng từ dữ liệu đầu vào, giúp nhận diện danh tính người nói dựa trên đặc điểm giọng nói.
* Mô hình CNN được xây dựng theo kiến trúc 1D, xử lý dữ liệu giọng nói bằng cách phân tích các mẫu đặc trưng thông qua các lớp:
* Lớp tích chập (Convolutional Layers)
  + Dùng kernel để phát hiện mẫu trong dữ liệu giọng nói.
  + Sử dụng hàm kích hoạt LeakyReLU (alpha = 0.3) để tăng tính phi tuyến tính.
* Lớp pooling (Pooling Layers)
  + Giảm kích thước dữ liệu bằng cách lấy đặc trưng quan trọng nhất.
  + Giúp mô hình học hiệu quả hơn với ít thông tin dư thừa.
* Lớp Flatten
  + Chuyển dữ liệu từ dạng ma trận sang vector, chuẩn bị cho lớp fully connected.
* Lớp Fully Connected (Dense Layers)
  + Kết hợp toàn bộ thông tin từ các lớp trước để đưa ra dự đoán danh tính.
* Lớp đầu ra (Output Layer)
  + Số neuron đầu ra tương ứng với số lượng người cần nhận diện.
  + Sử dụng hàm SoftMax để tính toán xác suất mỗi lớp người nói.
* Quá trình huấn luyện
* Dữ liệu đầu vào: Sử dụng vector 16 phần tử gồm 12 MFCC và 4 đặc trưng bổ sung (tần số trung bình, độ lệch chuẩn, biên độ, tỷ lệ zero-crossing).
* Huấn luyện mô hình: Sử dụng tập dữ liệu được gán nhãn, giúp mô hình học cách phân loại người nói theo giọng của họ.
* Thông số quan trọng: Hệ số learning rate = 0.001, đảm bảo tốc độ cập nhật trọng số hợp lý.
* Kiểm thử và xác thực
* Kiểm thử trên tập dữ liệu mới, đo độ chính xác khi nhận diện người nói.
* Xác thực danh tính bằng giọng: Khi có đầu vào, mô hình tính xác suất dự đoán và chỉ chấp nhận nếu vượt ngưỡng xác định.

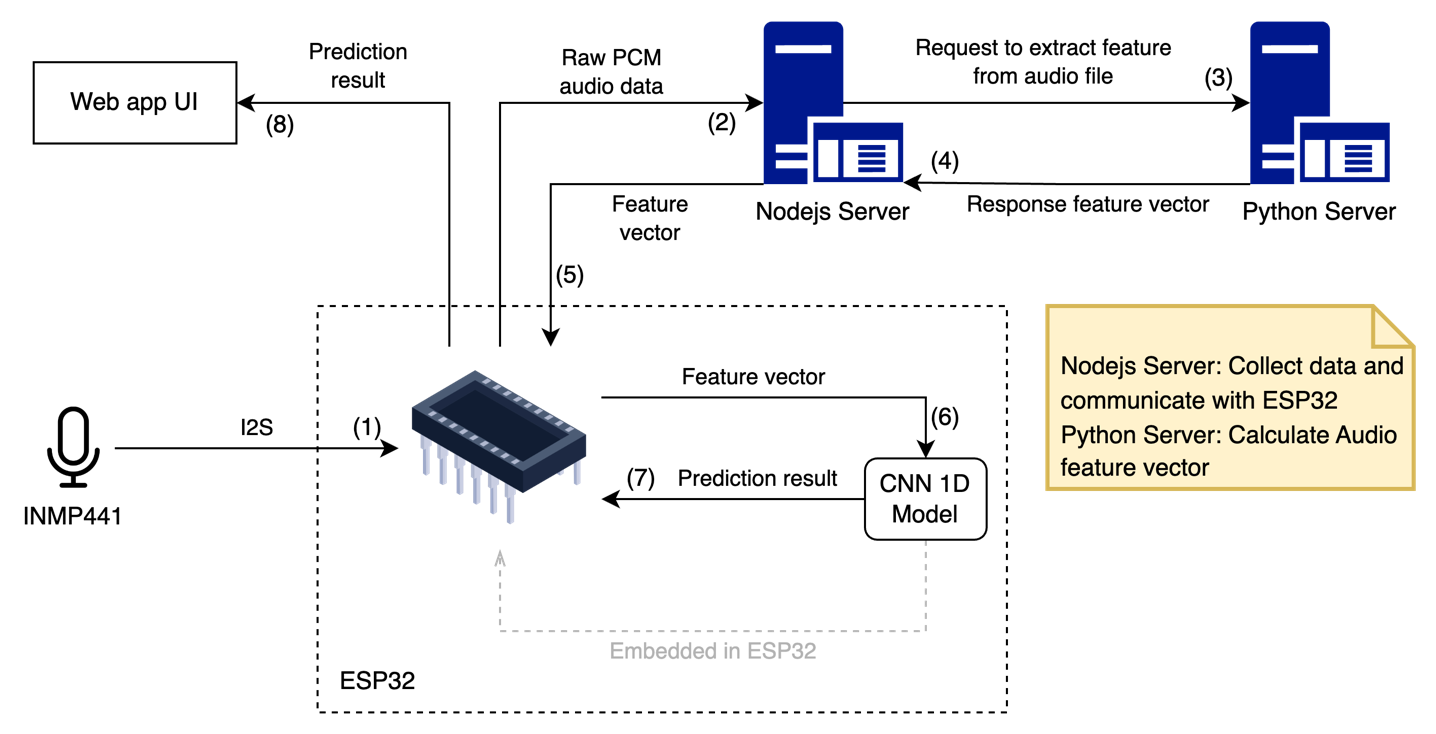
# Xây dựng hệ thống

## Kiến trúc hệ thống

1. **Giới thiệu tổng quan**

* Hệ thống nhận diện giọng nói dựa trên ESP32, sử dụng INMP441 thu âm, truyền dữ liệu qua WebSocket đến server trích xuất đặc trưng MFCC, và dự đoán bằng mô hình CNN nhúng trên ESP32 và xuất kết quả.

1. **Sơ đồ khối tổng thể**



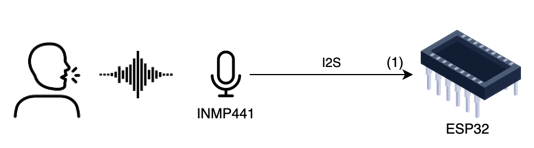
#### Hình 1: Sơ đồ khối tổng thể của hệ thống

1. **Thiết bị và công nghệ sử dụng**

* **INMP441 (I2S) Microphone**
* Là microphone MEMS (Micro-Electro-Mechanical Systems) hỗ trợ giao tiếp I2S (Inter-IC Sound), cho chất lượng âm thanh số với độ nhiễu thấp.
* Vai trò trong hệ thống: Thu âm thanh môi trường, chuyển đổi tín hiệu analog sang digital trực tiếp mà không cần ADC ngoài.
* **ESP32 Microcontroller**
* Vi điều khiển WiFi/Bluetooth dual-core, kiến trúc Xtensa LX6, tốc độ xử lý 160-240MHz. Hỗ trợ I2S tích hợp, phù hợp kết nối với INMP441. Bộ nhớ: 520KB SRAM, 4MB Flash.
* Vai trò trong hệ thống: Đọc dữ liệu từ INMP441 qua I2S, chạy mô hình để dự đoán, giao tiếp với server qua WiFi.
* **JavaScript WebSocket Server**
* Server sử dụng giao thức WebSocket để truyền dữ liệu real-time giữa ESP32 và Python Server. Duy trì kết nối hai chiều, độ trễ thấp hơn HTTP. Hỗ trợ truyền binary data (PCM audio).
* Vai trong hệ thống: Nhận dữ liệu âm thanh từ ESP32, gửi yêu cầu phân tích đặc trưng đến Python Server.
* **Python HTTP Server**
* Server xử lý backend, viết bằng Python (dùng thư viện Flask), tiếp nhận yêu cầu từ WebSocket.
* Vai trong hệ thống: Trích xuất đặc trưng MFCC sử dụng Librosa/TensorFlow.
* **TensorFlow/Keras library**
* Thư viện mã nguồn mở cho machine learning, hỗ trợ xây dựng mô hình CNN.
* Vai trong hệ thống: Huấn luyện mô hình nhận diện giọng nói trên máy tính.
* **TensorFlow Lite Micro library**
* Phiên bản tối ưu của TensorFlow cho vi điều khiển, hỗ trợ quantization (8-bit). Tính năng phù hợp với ESP32, chiếm ít bộ nhớ, hỗ trợ operator CNN 1D.
* Vai trong hệ thống: Convert model TensorFlow → TensorFlow Lite (.tflite). Nhúng vào firmware ESP32 thông qua thư viện ESP-IDF/Arduino.

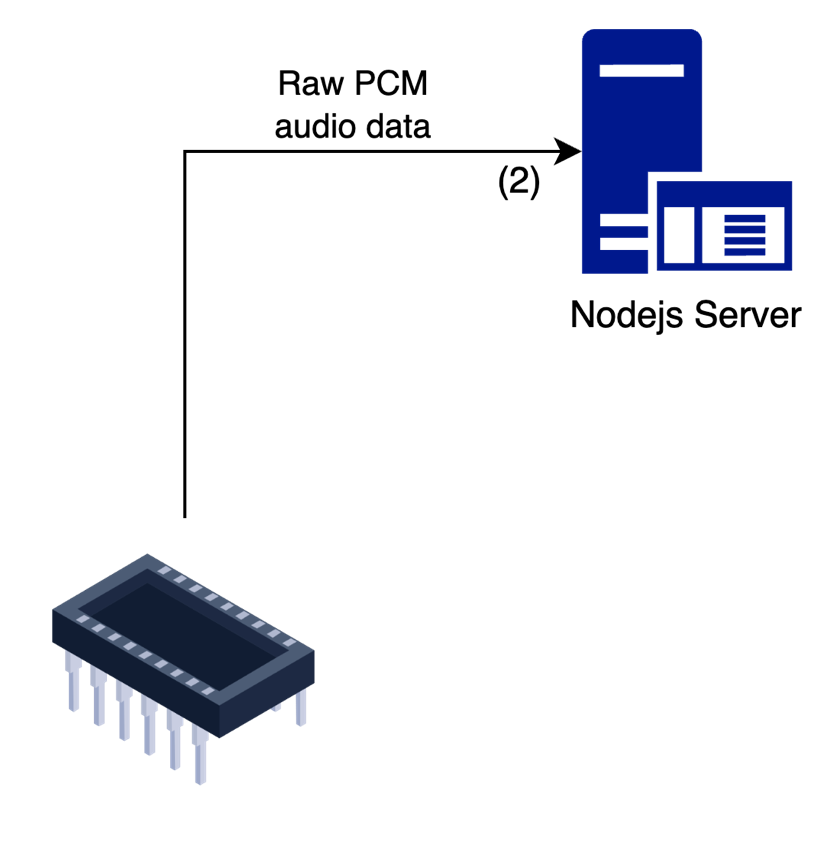
1. **Các khối chức năng**

* **Khối thu âm**
* Thành phần chính: ESP32, INMP441 Microphone, I2S.
* Chức năng:
* ESP32 giao tiếp với INMP441 qua I2S.
* Thu tín hiệu âm thanh ở tần số 16kHz.
* Chuẩn bị dữ liệu để truyền đến server.
* Kết quả: Dữ liệu âm thanh thô (raw PCM).



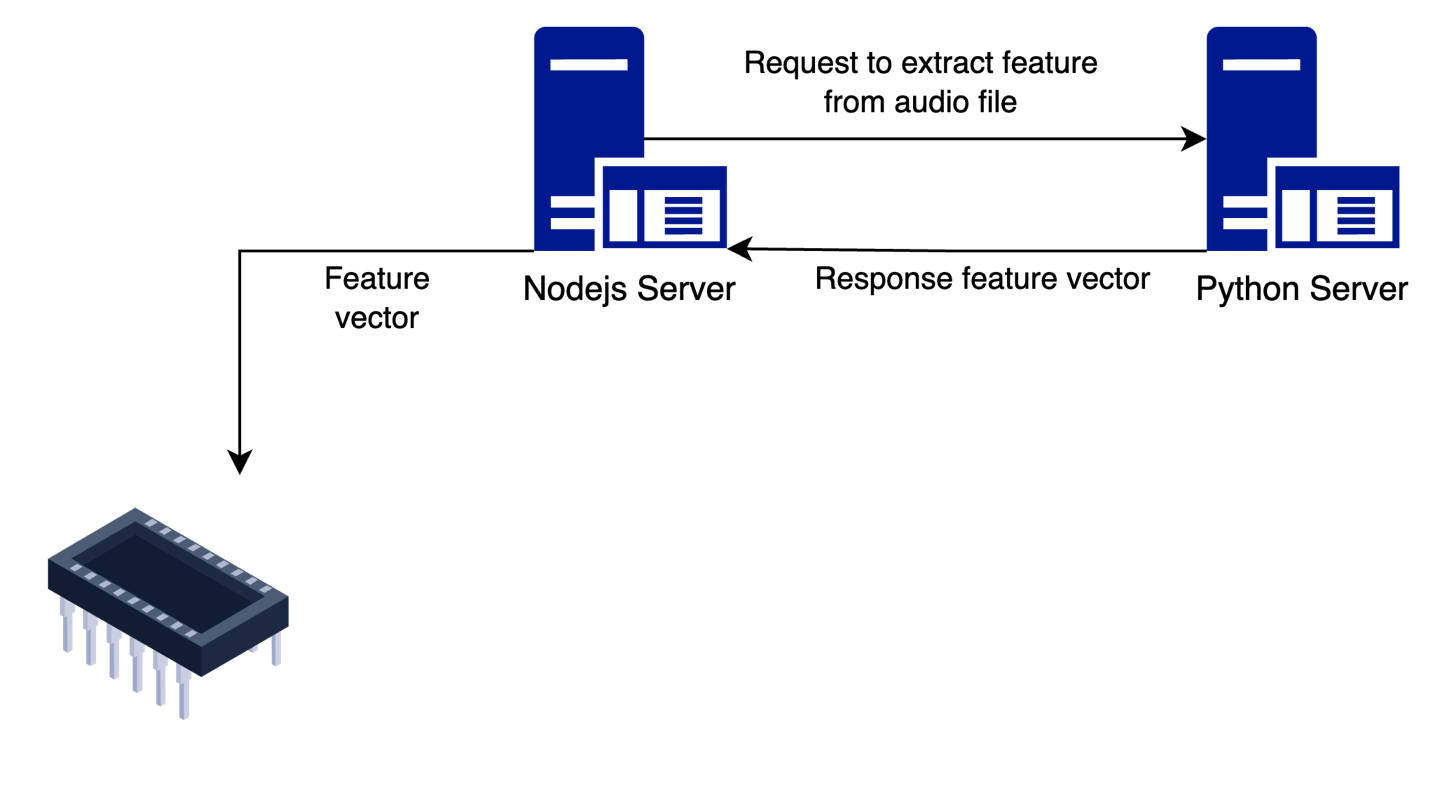
#### Hình 2: Khối thu âm của hệ thống

* **Khối truyền dữ liệu**
* Thành phần chính: WebSocket Client (ESP32), WiFi.
* Chức năng:
* Gửi dữ liệu dạng raw PCM đến server qua WebSocket.
* Gửi liên tục với kích thước mỗi buffer là 1024 byte.
* Tốc độ thu mẫu là 16kHz, 16-bit.



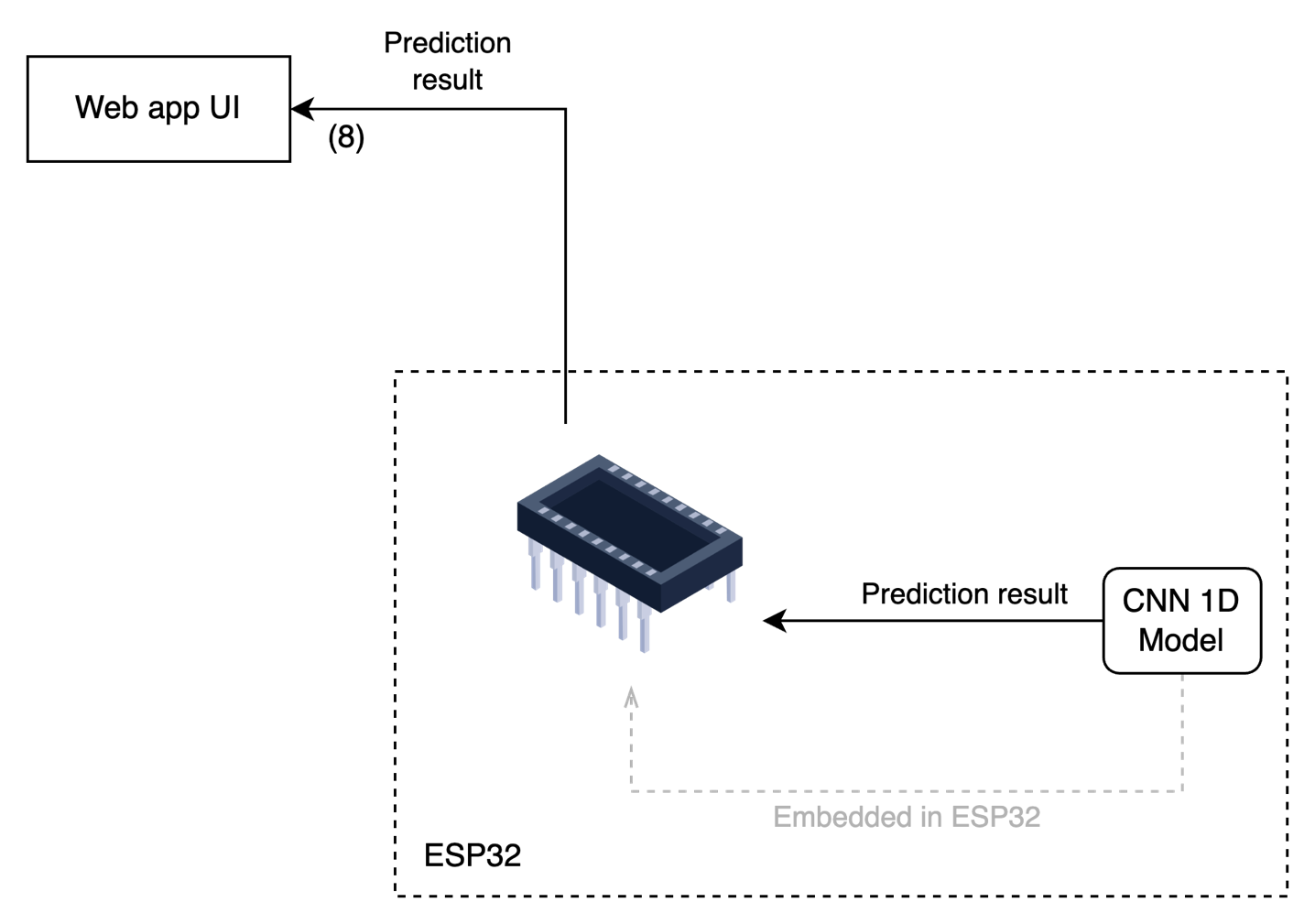
#### Hình 3: Khối truyền dữ liệu của hệ thống

* **Khối xử lí đặc trưng**
* Thành phần chính: JavaScript WebSocket Server, Feature extraction Pyhon Server
* Chức năng:
* Nhận dữ liệu từ ESP32.
* Lưu thành file nhị phân (10s/lần).
* Trích xuất đặc trưng MFCC.
* Gửi vector đặc trưng đến ESP32.
* Kết quả: Vector đặc trưng kích thước nhỏ.



#### Hình 4: Khối xử lí đặc trưng của hệ thống

* **Khối dự đoán**
* Thành phần chính: ESP32, model đã nhúng
* Chức năng:
* Nhận vector đặc trưng từ server.
* Dự đoán giọng nói (người A/B).
* Xuất kết quả ra Serial.



#### Hình 5: Khối dự đoán của hệ thống

## Chuẩn bị cho hệ thống

1. **Chuẩn bị thiết bị và môi trường phát triển**

* **Mục tiêu**
* Thiết lập đầy đủ phần cứng, phần mềm và dữ liệu để triển khai toàn bộ hệ thống nhận diện giọng nói, từ thu âm, truyền nhận dữ liệu, huấn luyện mô hình cho đến triển khai nhúng trên ESP32.
* **Phần cứng**
* Máy tính cá nhân:
* CPU: Apple M1
* RAM: 8GB
* Hệ điều hành: MacOS 15
* Thiết bị:
* ESP32
* INMP441
* Dây nối, cáp USB
* **Phần mềm và công cụ**
* Python: Huấn luyện mô hình, xử lí các đặc trưng âm thanh
* Các thư viện cần thiết:

|  |  |  |
| --- | --- | --- |
| absl-py==2.1.0  astunparse==1.6.3  audioread==3.0.1  bidict==0.23.1  blinker==1.9.0  certifi==2025.1.31  cffi==1.17.1  charset-normalizer==3.4.1  click==8.1.8  contourpy==1.3.1  cycler==0.12.1  decorator==5.2.1  Flask==3.1.0  flask-cors==5.0.1  Flask-SocketIO==5.5.1  flatbuffers==25.2.10  fonttools==4.56.0  gast==0.6.0  google-pasta==0.2.0  grpcio==1.71.0  h11==0.14.0  h5py==3.13.0  idna==3.10  itsdangerous==2.2.0  Jinja2==3.1.6  joblib==1.4.2  keras==3.9.0  kiwisolver==1.4.8 | lazy\_loader==0.4  libclang==18.1.1  librosa==0.11.0  llvmlite==0.44.0  Markdown==3.7  markdown-it-py==3.0.0  MarkupSafe==3.0.2  matplotlib==3.10.1  mdurl==0.1.2  ml\_dtypes==0.5.1  msgpack==1.1.0  namex==0.0.8  numba==0.61.0  numpy==2.1.3  opt\_einsum==3.4.0  optree==0.14.1  packaging==24.2  pandas==2.2.3  pillow==11.1.0  platformdirs==4.3.6  pooch==1.8.2  protobuf==5.29.3  PyAudio==0.2.14  pycparser==2.22  pydub==0.25.1  Pygments==2.19.1  pyparsing==3.2.1  python-dateutil==2.9.0.post0 | python-engineio==4.11.2  python-socketio==5.12.1  pytz==2025.1  PyYAML==6.0.2  requests==2.32.3  rich==13.9.4  scikit-learn==1.6.1  scipy==1.15.2  seaborn==0.13.2  simple-websocket==1.1.0  simpleaudio==1.0.4  six==1.17.0  sounddevice==0.5.1  soundfile==0.13.1  soxr==0.5.0.post1  tensorboard==2.19.0  tensorboard-data-server==0.7.2  tensorflow==2.19.0  tensorflow-io-gcs-filesystem==0.37.1  termcolor==2.5.0  threadpoolctl==3.6.0  typing\_extensions==4.12.2  tzdata==2025.1  urllib3==2.3.0  websockets==15.0.1  Werkzeug==3.1.3  wrapt==1.17.2  wsproto==1.2.0 |

* Node.js: Xây dựng WebSocket Server nhận dữ liệu từ ESP32.
* Các dependency cần thiết:

"dependencies": {

    "express": "^4.18.2",

    "ws": "^7.4.6"

  }

* Arduino IDE: Viết và nạp chương trình điều khiển cho ESP32.
* Các thư viện cần thiết:

#include <WiFi.h>

#include <driver/i2s.h>

#include <ArduinoJson.h>

#include <ArduinoWebsockets.h>

#include <TensorFlowLite\_ESP32.h>

1. **Thu thập dữ liệu âm thanh**

Hệ thống sử dụng microphone kỹ thuật số INMP441 kết nối với ESP32 qua giao thức I2S (Inter-IC Sound) để thu thập dữ liệu âm thanh. Dữ liệu sau đó được truyền thời gian thực đến máy tính thông qua WebSocket để lưu trữ và xử lý.

* **Cấu hình phần cứng và giao tiếp I2S**
* INMP441 là microphone MEMS kỹ thuật số, đầu ra là tín hiệu I2S.
* Các chân kết nối:
* I2S\_SD (GPIO 32): dữ liệu âm thanh.
* I2S\_WS (GPIO 25): đồng hồ chọn kênh (left/right).
* I2S\_SCK (GPIO 26): đồng hồ bit (bit clock).
* ESP32 được cấu hình ở chế độ I2S Master RX, lấy mẫu âm thanh với:
* Tần số lấy mẫu: 16 kHz
* Độ phân giải mẫu: 16-bit (2 byte)
* Kênh âm thanh: mono (kênh trái)
* **Quá trình thu dữ liệu**
* Hàm micTask() thực thi liên tục, đọc dữ liệu từ microphone thông qua i2s\_read(), đưa vào buffer sBuffer có độ dài 1024 mẫu (tương đương 64 ms âm thanh).
* Mỗi khi buffer đầy, dữ liệu sẽ được gửi qua WebSocket đến server máy tính.
* Mỗi mẫu âm thanh là số nguyên 16-bit có giá trị trong khoảng [-32768, 32767].
* **Tính toán kích thước dữ liệu**

Với cấu hình hiện tại:

* Số mẫu mỗi giây: 16000 mẫu/giây
* Kích thước mỗi mẫu: 2 byte
* Kích thước dữ liệu mỗi giây: 16000×2= 32000 byte =32KB
* **Ghi chú kỹ thuật**
* Bộ đệm DMA có 10 buffer × 1024 mẫu = 10,240 mẫu (~0.64 giây).
* Nếu muốn thu liên tục 1 giây, cần xử lý nhanh hoặc tăng buffer.
* Do bộ nhớ RAM giới hạn, ESP32 chỉ xử lý buffer ngắn, dữ liệu được truyền về máy tính thay vì lưu cục bộ.

1. **Cấu trúc thư mục lưu trữ dữ liệu và đầu ra**

Dữ liệu âm thanh thu được từ ESP32 sẽ được lưu dưới dạng file nhị phân (.bin) theo từng người dùng, nhằm phục vụ cho huấn luyện mô hình nhận diện giọng nói. Ngoài ra, các đặc trưng và mô hình được sinh ra trong quá trình huấn luyện cũng được tổ chức thành thư mục rõ ràng, thuận tiện cho tích hợp và triển khai.

* **Cấu trúc thư mục dataset/**
* Thư mục dataset là nơi lưu trữ dữ liệu âm thanh gốc của từng người dùng.
* Mỗi người dùng có một thư mục con tương ứng với tên.
* Dữ liệu âm thanh được lưu dưới dạng nhị phân (.bin), mỗi file tương ứng với một đoạn ghi âm.
* File config.yml trong thư mục người dùng chứa thông tin cấu hình (metadata) của dữ liệu người dùng, dùng để quản lý và đọc dữ liệu.
* **Cấu trúc thư mục output/**

Thư mục output/ bao gồm:

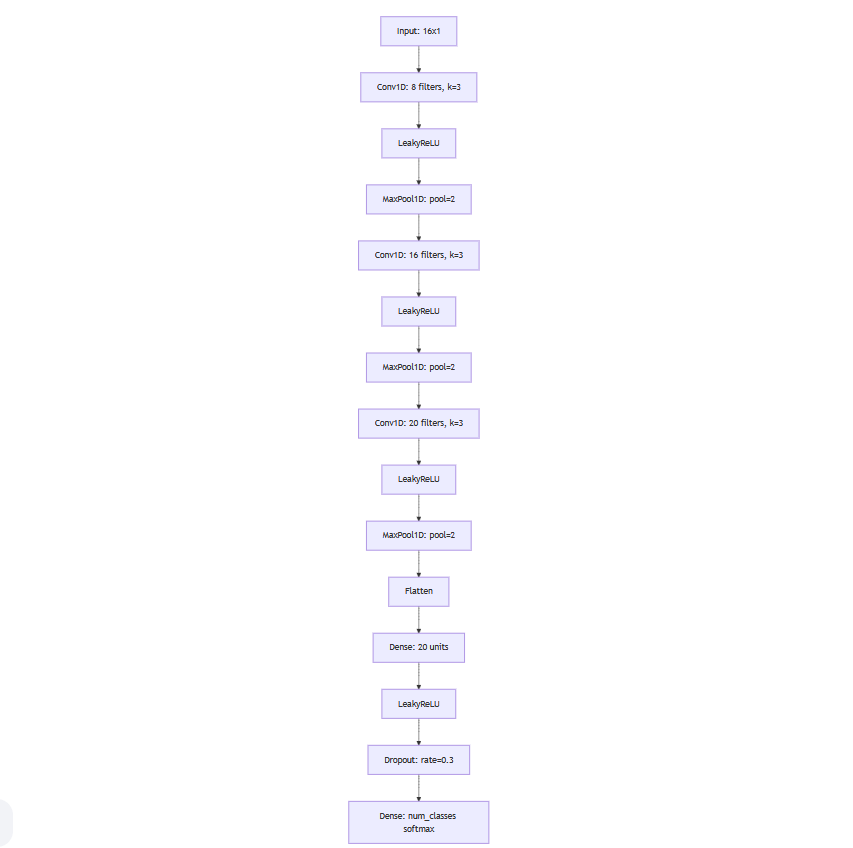
* Đặc trưng dữ liệu (features) đã được trích xuất từ tập dữ liệu giọng nói.
* Tập huấn luyện, kiểm tra và xác thực cho quá trình huấn luyện mô hình.
* Các tệp mô hình để tích hợp vào hệ thống.
* Hình ảnh trực quan hóa quá trình huấn luyện mô hình.
* Bản đồ nhãn (label map) dùng để chuyển đổi dự đoán của mô hình thành tên người dùng.
* Cấu trúc output/dataset là bắt buộc để đảm bảo tương thích với các thành phần xử lý và ứng dụng.
* Tên mô hình có thể thay đổi thông qua file application\_config.yml.
* Tên file biểu đồ (train-and-validation-accuracy.png) là cố định để đảm bảo hiển thị tự động trong giao diện.

1. **Huấn luyện mô hình**

* Hệ thống nhận diện giọng nói sử dụng mô hình học sâu CNN (Convolutional Neural Network) để tự động trích xuất đặc trưng và phân loại danh tính người nói dựa trên giọng. CNN có khả năng tự học các mẫu âm thanh, giúp nhận diện nhanh chóng và chính xác.
* **Chuẩn bị dữ liệu huấn luyện**
* Tiền xử lý dữ liệu âm thanh
* Đọc dữ liệu PCM từ file, sử dụng np.fromfile() hoặc np.frombuffer().
* Chuẩn hóa dữ liệu về phạm vi [-1, 1] để tăng độ chính xác.
* Trích xuất đặc trưng giọng nói

Từ mỗi mẫu giọng nói, hệ thống trích xuất vector 16 phần tử, gồm:

* 12 hệ số MFCC (Mel-Frequency Cepstral Coefficients)
* 4 đặc trưng bổ sung:
* Tần số trung bình (Mean Frequency)
* Độ lệch chuẩn (Standard Deviation)
* Cường độ âm thanh (Amplitude)
* Tỷ lệ zero-crossing (Zero-Crossing Rate)
* Xây dựng tập huấn luyện và kiểm thử
* Tách dữ liệu thành 80% huấn luyện, 20% kiểm thử để đảm bảo tổng quát hóa tốt.
* **Xây dựng mô hình CNN**
* Mô hình CNN có kiến trúc 1D, gồm các lớp chính như sau:
* Input Layer (16x1) – Nhận vector đặc trưng từ giọng nói.
* Conv1D (8 filters, k=3) – Trích xuất mẫu giọng nói.
* LeakyReLU Activation – Giữ thông tin trong gradient giúp mô hình học tốt hơn.
* MaxPooling1D (pool=2) – Giảm kích thước dữ liệu, giữ đặc trưng quan trọng.
* Conv1D (16 filters, k=3) – Lớp tích chập sâu hơn, học thêm đặc trưng.
* MaxPooling1D (pool=2) – Giảm dữ liệu, tránh dư thừa thông tin.
* Conv1D (20 filters, k=3) – Lớp tích chập cuối cùng, trích xuất mẫu giọng sâu hơn.
* Flatten Layer – Chuyển dữ liệu về dạng vector để đưa vào lớp fully connected.
* Dense Layer (20 units) + LeakyReLU – Xử lý đặc trưng quan trọng.
* Dropout (rate=0.3) – Giảm quá khớp, giúp mô hình hoạt động ổn định.
* Output Layer (softmax) – Dự đoán danh tính người nói theo xác suất.



#### Hình 6: Kiến trúc mô hình CNN 1D

* Cài đặt mô hình bằng TensorFlow

import tensorflow as tf

from tensorflow.keras import layers, models

# Define the CNN model

def build\_voice\_auth\_cnn(input\_shape=(16, 1), num\_classes=10):

model = models.Sequential([

layers.Conv1D(filters=32, kernel\_size=3, activation='leaky\_relu', input\_shape=input\_shape),

layers.MaxPooling1D(pool\_size=2),

layers.Conv1D(filters=64, kernel\_size=3, activation='leaky\_relu'),

layers.MaxPooling1D(pool\_size=2),

layers.Conv1D(filters=128, kernel\_size=3, activation='leaky\_relu'),

layers.MaxPooling1D(pool\_size=2),

layers.Flatten(),

layers.Dense(128, activation='relu'),

layers.Dense(num\_classes, activation='softmax') # Softmax for multi-user classification

])

return model

# Build the model

cnn\_model = build\_voice\_auth\_cnn()

cnn\_model.summary()

* **Huấn luyện mô hình**
* Cấu hình mô hình CNN
* Hàm mất mát: categorical\_crossentropy, dùng cho phân loại nhiều người nói.
* Bộ tối ưu hóa: Adam, với learning rate = 0.001.
* Đánh giá hiệu suất: accuracy.

cnn\_model.compile(

loss='categorical\_crossentropy',

optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001),

metrics=['accuracy']

)

* Huấn luyện trên tập dữ liệu giọng nói

history = cnn\_model.fit(

train\_features, train\_labels, # Training data

epochs=50,

batch\_size=32,

validation\_data=(test\_features, test\_labels) # Validation set

)

* **Kiểm thử và đánh giá**
* Kiểm tra độ chính xác

test\_loss, test\_acc = cnn\_model.evaluate(test\_features, test\_labels)

print(f"Test Accuracy: {test\_acc:.2f}")

* Độ chính xác > 95% → Mô hình hoạt động tốt.
* Nếu độ chính xác thấp, cần tinh chỉnh tham số hoặc tối ưu bộ trích xuất đặc trưng.
* Phân tích kết quả

import matplotlib.pyplot as plt

# Plot training and validation accuracy

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

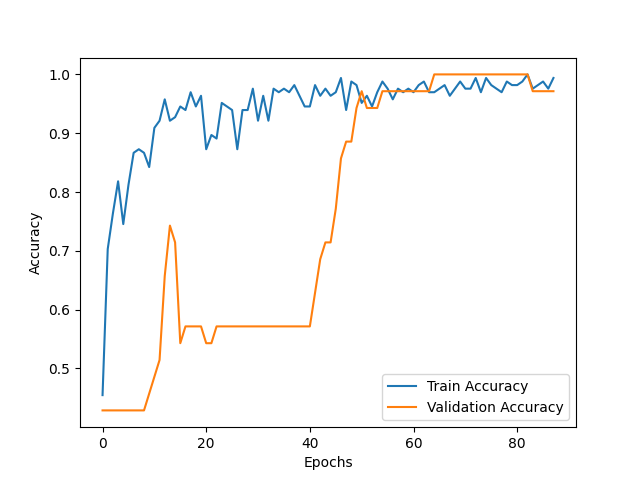
plt.xlabel("Epochs")

plt.ylabel("Accuracy")

plt.legend()

plt.show()

* Nếu độ chính xác huấn luyện cao nhưng kiểm thử thấp, mô hình có thể bị quá khớp.
* Cần điều chỉnh dropout, batch size, learning rate để tối ưu.



#### Hình 7: Biểu đồ hiển thị quá trình huấn luyện mô hình

* **Xác thực người dùng bằng giọng nói**
* Nhận diện giọng nói

def authenticate\_user(voice\_sample, cnn\_model, threshold=0.9):

# Extract 16 speech features from the sample

features = extract\_features(voice\_sample)

features = features.reshape(1, 16, 1) # Reshape for CNN

# Predict using trained CNN model

predictions = cnn\_model.predict(features)

predicted\_user = np.argmax(predictions)

confidence = np.max(predictions)

if confidence >= threshold:

print(f"Authenticated as User {predicted\_user} (Confidence: {confidence:.2f})")

return predicted\_user

else:

print("Authentication Failed: Unknown User")

return None

* Các bước xác thực
* Tiền xử lý giọng nói, trích xuất MFCC.
* Đưa vào CNN để dự đoán danh tính người nói.
* Kiểm tra ngưỡng xác suất, chỉ xác thực nếu dự đoán đủ tin cậy (>90%).

## Xây dựng thành phần hệ thống

1. **Triển khai Microphone Server**

* Microphone Server đóng vai trò là trung tâm giao tiếp giữa ESP32 và trình duyệt web, giúp truyền tải dữ liệu âm thanh **theo thời gian thực** mà không cần các yêu cầu HTTP liên tục. Nhờ WebSocket, hệ thống có thể xử lý dữ liệu nhanh chóng, đảm bảo tốc độ và tính ổn định khi nhận diện giọng nói.
* Chức năng:
* Thiết lập WebSocket Server để nhận dữ liệu âm thanh PCM từ ESP32.
* Ghi dữ liệu âm thanh vào tệp nhị phân mỗi 10 giây để lưu trữ và xử lý sau.
* Kết nối với trình duyệt giúp phát lại dữ liệu thu âm và hiển thị dạng sóng âm thanh.
* Gửi tệp âm thanh đến Python Server để trích xuất đặc trưng MFCC.
* Nhận vector đặc trưng MFCC từ Extracting Feature Server, gửi lại cho ESP32 để nhận diện giọng nói.
* Quy trình vận hành:
* Node Server tạo WebSocket để ESP32 có thể gửi dữ liệu âm thanh.
* ESP32 gửi dữ liệu PCM đến máy chủ WebSocket.
* Máy chủ Node.js lưu dữ liệu âm thanh và ghi thành tệp nhị phân.
* Tệp âm thanh được gửi đến Python Server để trích xuất đặc trưng MFCC.
* Vector đặc trưng MFCC được gửi lại cho ESP32, giúp nó nhận diện người nói.
* Các bước khởi chạy:
* Cài đặt dependencies cho máy chủ Node.js qua lệnh:

npm install

* Khởi động Node Server, giúp WebSocket Server lắng nghe kết nối từ ESP32.
* Cấu hình WebSocket IP trên trình duyệt (audio\_client.html) để trình duyệt kết nối được.
* Cấu hình WebSocket IP trên ESP32 để đảm bảo ESP32 gửi dữ liệu đúng máy chủ.
* Khởi chạy ESP32, bắt đầu gửi dữ liệu âm thanh đến WebSocket Server.

1. **Triển khai Extracting Feature Server**

* Extracting Feature Server là một máy chủ xử lý dữ liệu âm thanh được xây dựng bằng Flask. Máy chủ này nhận tệp nhị phân chứa tín hiệu giọng nói từ Microphone Server, trích xuất đặc trưng MFCC, và sử dụng mô hình học máy để xác định danh tính người nói trước khi gửi kết quả về lại ESP32.
* Chức năng:
* Nhận tệp âm thanh từ WebSocket Server dưới dạng PCM.
* Đọc và chuyển đổi dữ liệu âm thanh từ tệp nhị phân thành mảng số (NumPy).
* Trích xuất đặc trưng MFCC và các thông tin bổ sung như tần số trung bình, độ lệch chuẩn, biên độ tổng hợp, tỷ lệ zero-crossing.
* Dự đoán giọng nói bằng mô hình đã huấn luyện, xác định danh tính người nói.
* Trả về vector đặc trưng MFCC để ESP32 sử dụng cho quá trình nhận diện giọng nói.
* Quy trình vận hành:
* Xử lý dữ liệu âm thanh
* Dữ liệu âm thanh được đọc từ tệp nhị phân bằng read\_pcm\_binary(), chuyển đổi từ kiểu int16 thành số thực (float32) để có thể xử lý.
* Dữ liệu sau khi được đọc sẽ được chuẩn hóa (/ np.iinfo(dtype).max) để đảm bảo nằm trong phạm vi hợp lý cho phân tích.
* Trích xuất đặc trưng MFCC
* Sử dụng librosa.feature.mfcc() để tính toán 12 đặc trưng MFCC từ dữ liệu âm thanh.
* Tính giá trị trung bình của từng MFCC (np.mean(mfcc, axis=1)) để giảm nhiễu và tối ưu hóa đầu vào mô hình.
* Kết hợp với các đặc trưng bổ sung:
* Tần số trung bình (librosa.fft\_frequencies()).
* Độ lệch chuẩn tín hiệu (np.std(data)).
* Biên độ tổng hợp (np.sum(np.abs(data))).
* Tỷ lệ zero-crossing (librosa.feature.zero\_crossing\_rate()).
* Tất cả đặc trưng được hợp nhất vào một vector (np.hstack()) để làm đầu vào cho mô hình nhận diện giọng nói.
* Nhận diện giọng nói bằng mô hình đã huấn luyện
* Mô hình voice\_identification.keras được tải bằng load\_model().
* Vector đặc trưng MFCC được truyền vào mô hình (model.predict()), mô hình trả về kết quả phân loại.
* Chỉ số dự đoán lớn nhất (np.argmax(prediction)) xác định danh tính người nói.
* Nếu dự đoán thuộc nhóm 0, hệ thống xác định là "Anh bạn thân". Nếu thuộc nhóm 1, kết quả là "Giang ơi".
* Trả kết quả về ESP32
* Kết quả trích xuất đặc trưng được chuyển thành danh sách JSON (feature\_vector.tolist()).
* Flask trả phản hồi HTTP JSON chứa vector đặc trưng để WebSocket Server xử lý và gửi lại ESP32.
* Khởi chạy:
* Flask Server chạy trên cổng 8001, liên tục lắng nghe yêu cầu từ

WebSocket Server (app.run(host="0.0.0.0", port=8001, debug=True)).

1. **Lắp đặt thiết bị và nhúng mã vào ESP32**

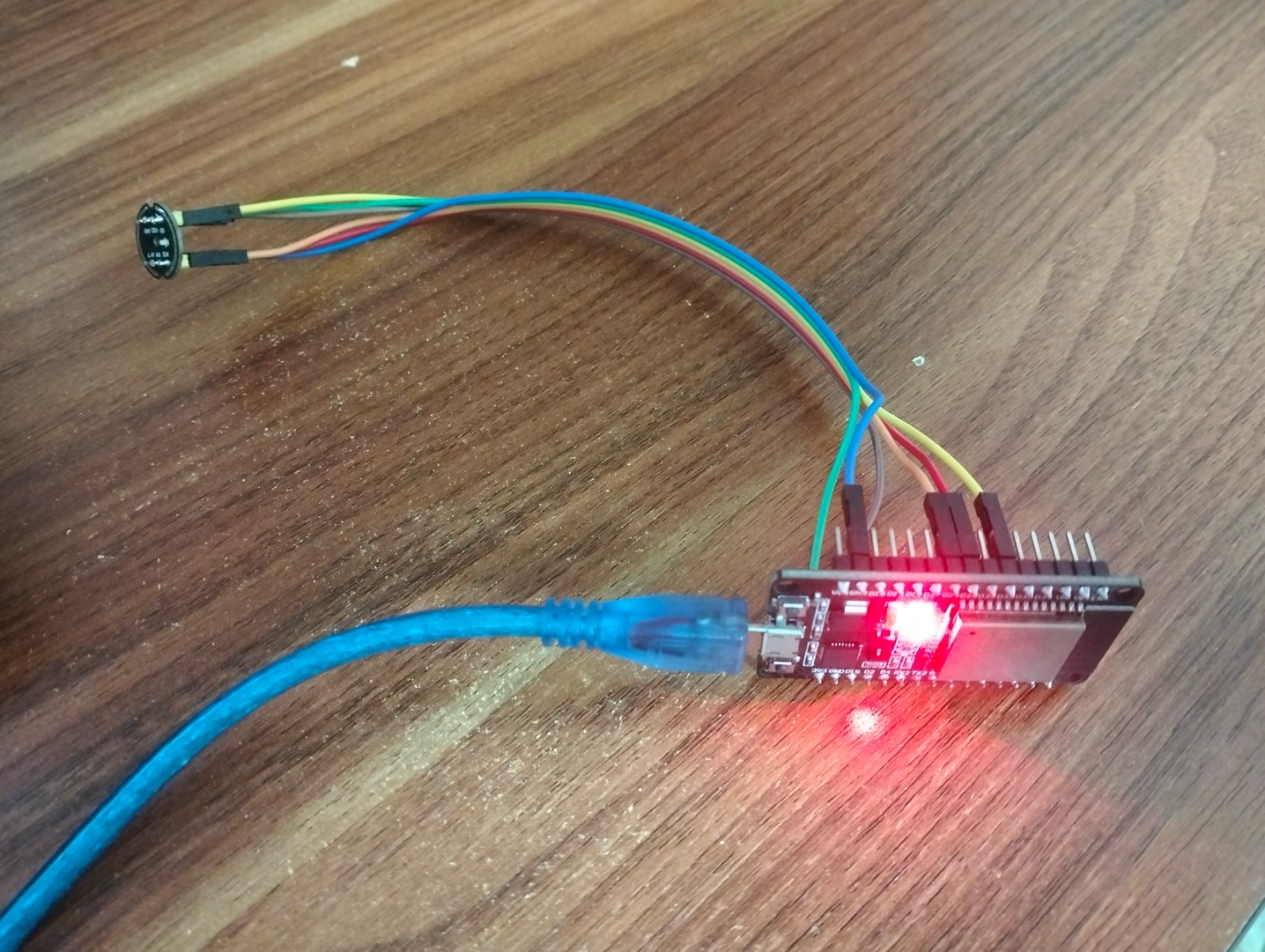
* ESP32 là thiết bị trung tâm trong hệ thống nhận diện giọng nói, có nhiệm vụ thu âm từ microphone INMP441, gửi dữ liệu PCM đến máy chủ WebSocket, nhận lại vector đặc trưng MFCC từ Python Server, và thực hiện suy luận bằng TensorFlow Lite Micro để xác định danh tính người nói.
* Chức năng:
* Kết nối ESP32 với microphone INMP441 bằng giao thức I2S, đảm bảo thu tín hiệu âm thanh số.
* Thiết lập kết nối WiFi, giúp ESP32 giao tiếp với WebSocket Server.
* Gửi dữ liệu PCM đến máy chủ WebSocket theo buffer 1024 byte, với tốc độ 16kHz, 16-bit.
* Nhận vector đặc trưng MFCC từ Python Server, sử dụng để nhận diện giọng nói.
* Thực hiện suy luận bằng TensorFlow Lite Micro, xác định danh tính người nói dựa trên mô hình đã huấn luyện.
* Xuất kết quả nhận diện ra cổng Serial, cho phép kiểm tra trực tiếp trên thiết bị.
* Quy trình vận hành:
* Kết nối phần cứng
* ESP32 giao tiếp với microphone INMP441 qua các chân I²S:
* I2S\_SD (GPIO 32): Chân nhận tín hiệu từ microphone.
* I2S\_WS (GPIO 25): Chân đồng bộ kênh âm thanh.
* I2S\_SCK (GPIO 26): Chân điều khiển bit clock.
* Cấu hình I²S bằng i2s\_install() và i2s\_setpin(), đảm bảo nhận đúng tín hiệu âm thanh.
* Khởi động bộ xử lý âm thanh I²S, bắt đầu thu tín hiệu từ microphone.
* Thiết lập kết nối WiFi
* Kết nối mạng WiFi bằng WiFi.begin(ssid, password).
* Chờ đến khi ESP32 nhận được kết nối, sau đó tiếp tục giao tiếp với WebSocket Server.
* Gửi dữ liệu PCM đến WebSocket Server
* Lắng nghe tín hiệu âm thanh bằng i2s\_read() từ microphone.
* Đưa dữ liệu PCM vào buffer (sBuffer[]), kiểm tra độ dài bytesIn.
* Gửi dữ liệu PCM đến WebSocket Server bằng client.sendBinary((const char\*)sBuffer, bytesIn).
* Chu kỳ gửi mỗi 5ms (vTaskDelay(5 / portTICK\_PERIOD\_MS)) để đảm bảo dữ liệu liên tục.
* Nhận vector đặc trưng MFCC từ Python Server
* ESP32 lắng nghe phản hồi từ WebSocket, xử lý dữ liệu JSON bằng onMessageCallback().
* Giải mã vector đặc trưng MFCC bằng deserializeJson(doc, message.data()).
* Lưu vector đặc trưng vào mảng featureVector[], chuẩn bị đưa vào mô hình.
* Nhận diện giọng nói bằng TensorFlow Lite Micro
* Tải mô hình nhúng từ bộ nhớ flash (tflite::GetModel(voice\_identification)).
* Khởi tạo bộ nhớ tensor (tensor\_arena với dung lượng 20KB).
* Sao chép vector MFCC vào tensor đầu vào, rồi thực hiện suy luận bằng interpreter->Invoke().
* Dự đoán xác suất của từng lớp, nếu xác suất class 0 lớn hơn class 1, ESP32 xác định A, ngược lại là B
* Xuất kết quả dự đoán ra Serial
* In kết quả dự đoán lên cổng Serial để kiểm tra (Serial.println()).
* Xuất xác suất nhận diện, cho phép kiểm tra độ tin cậy của dự đoán.
* Các bước khởi chạy:
* Lặp đặt ESP32 nối với INMP441 thông qua các dây nối.
* Thực hiện nhúng mã nguồn vào thiết bị ESP32 bằng Arduino IDE qua cáp USB.

# Thực nghiệm và đánh giá

## Triển khai thực tế

* Lắp đặt thiết bị

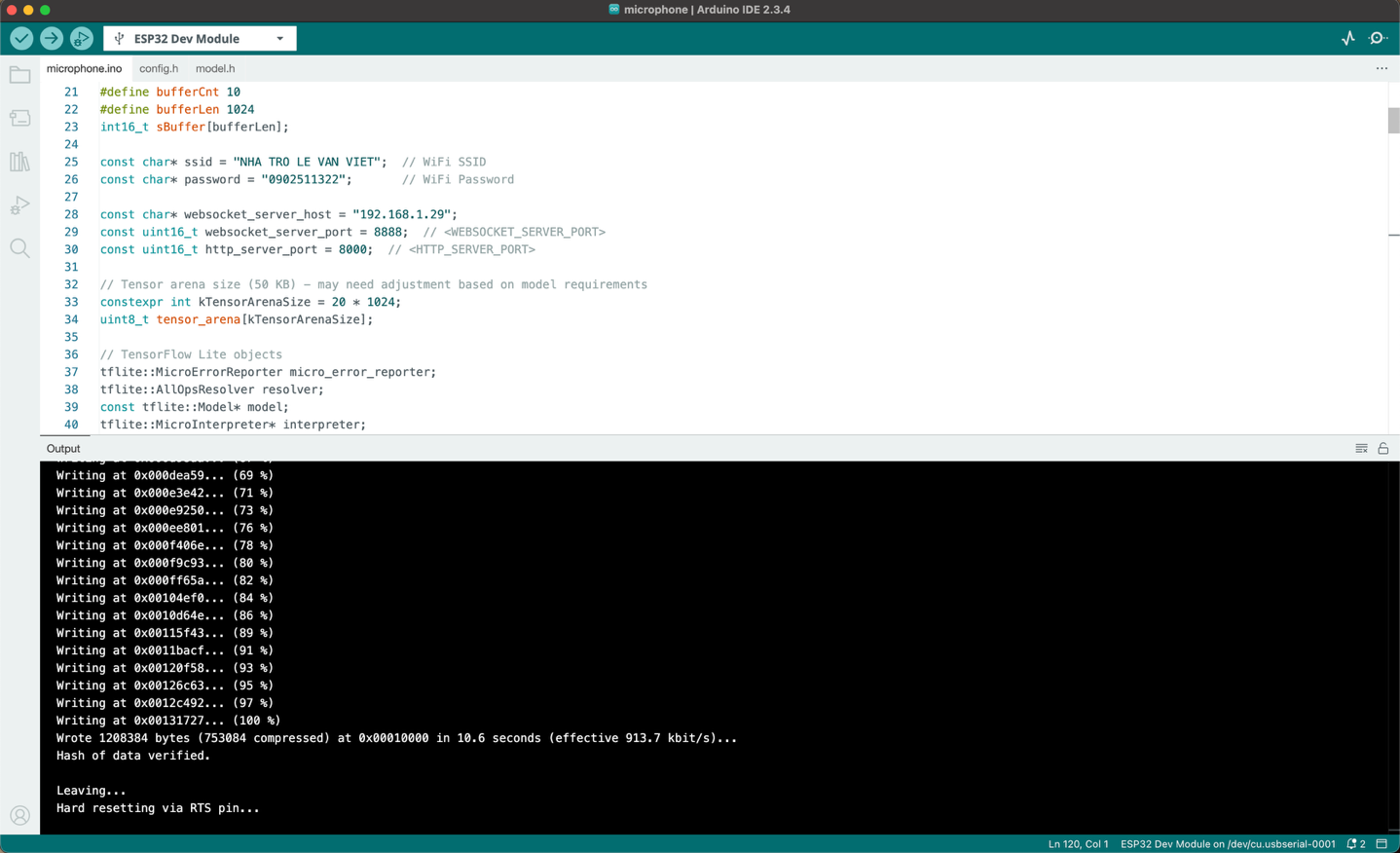
Kết nối ESP32 với microphone INMP441.



#### Hình 8: Mạch ESP32 sau khi kết nối với mic và kết nối với máy tính

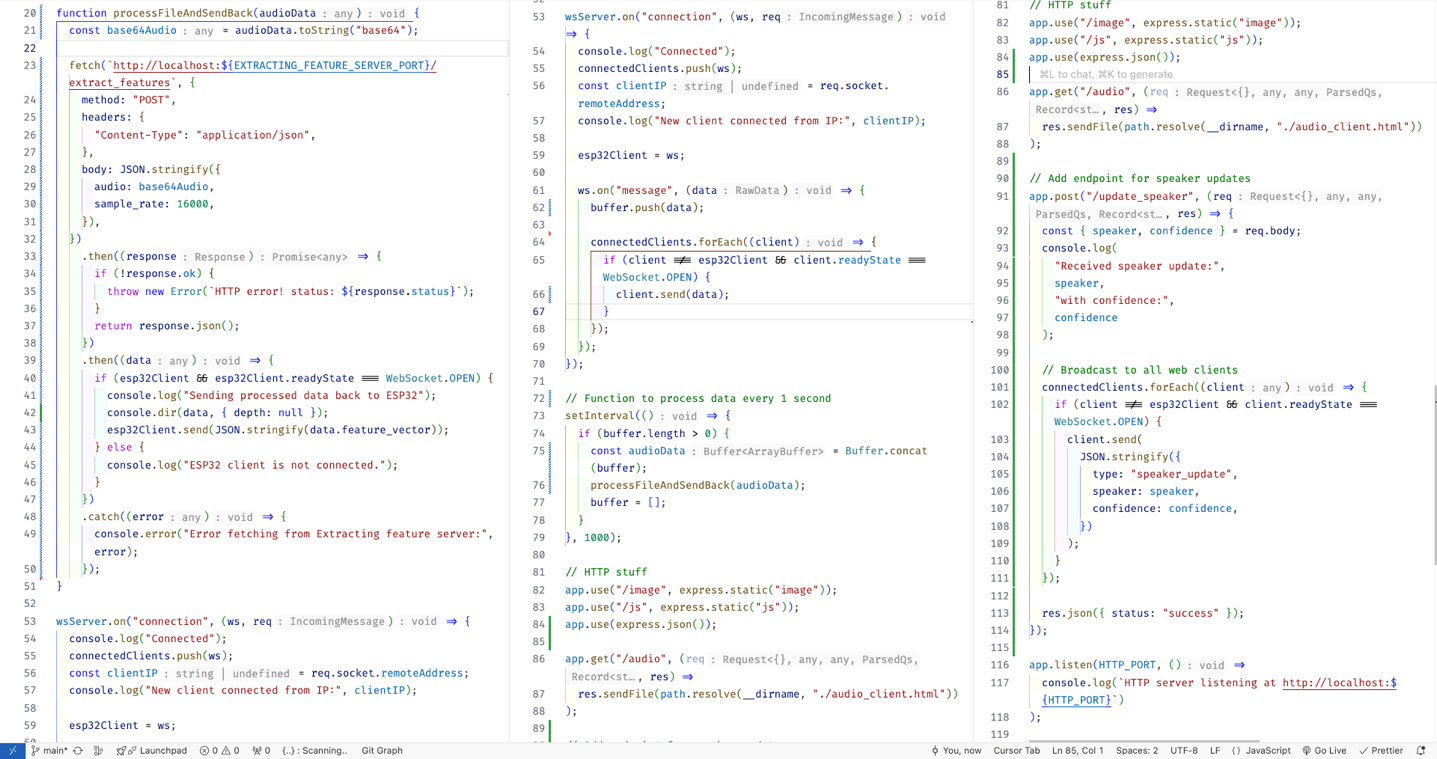
* Cấu hình phần mềm

Nạp mã vào ESP32



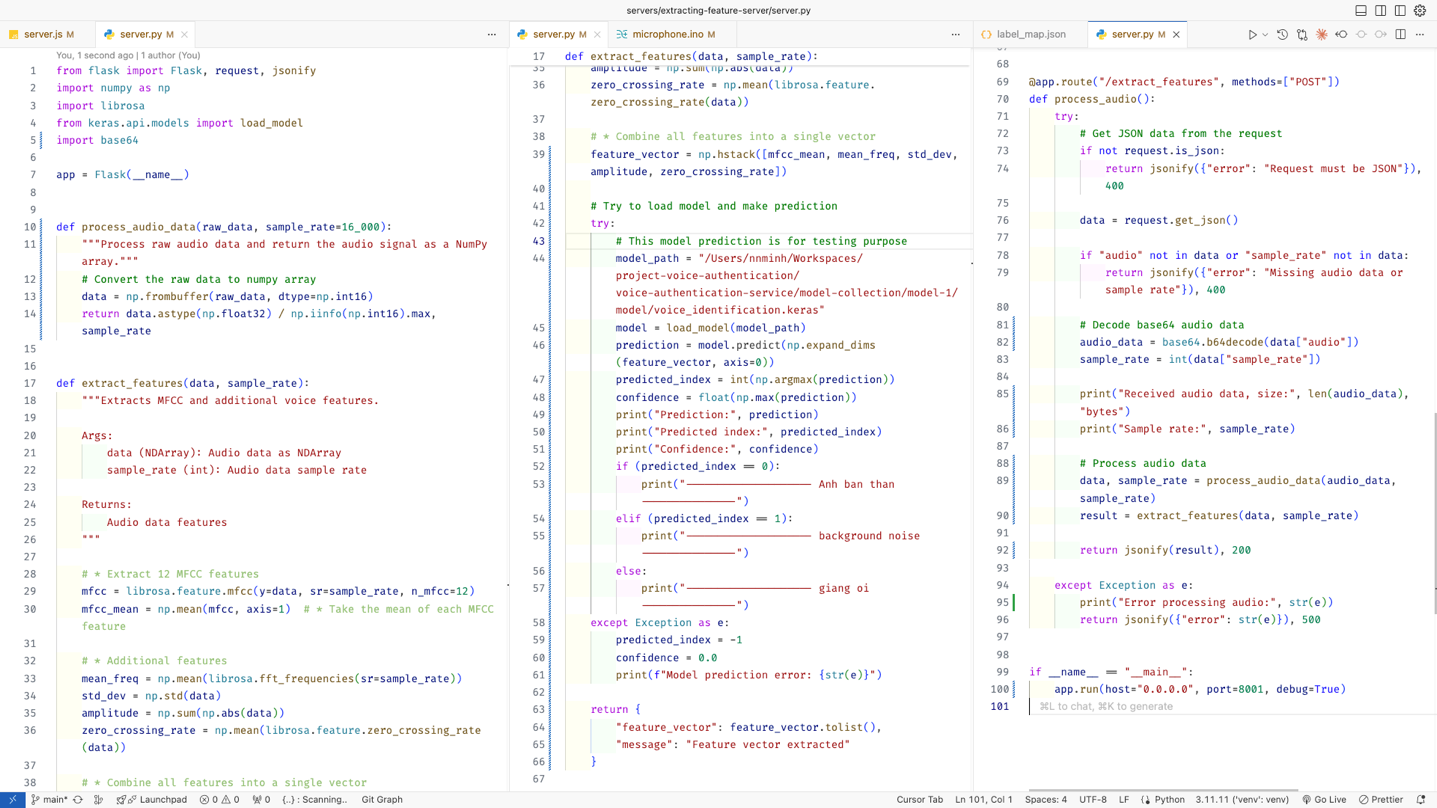
#### Hình 9: Quá trình nạp code vào cho ESP32

Thiết lập Microphone Server



#### Hình 10: Tổng quan code server thu âm thanh

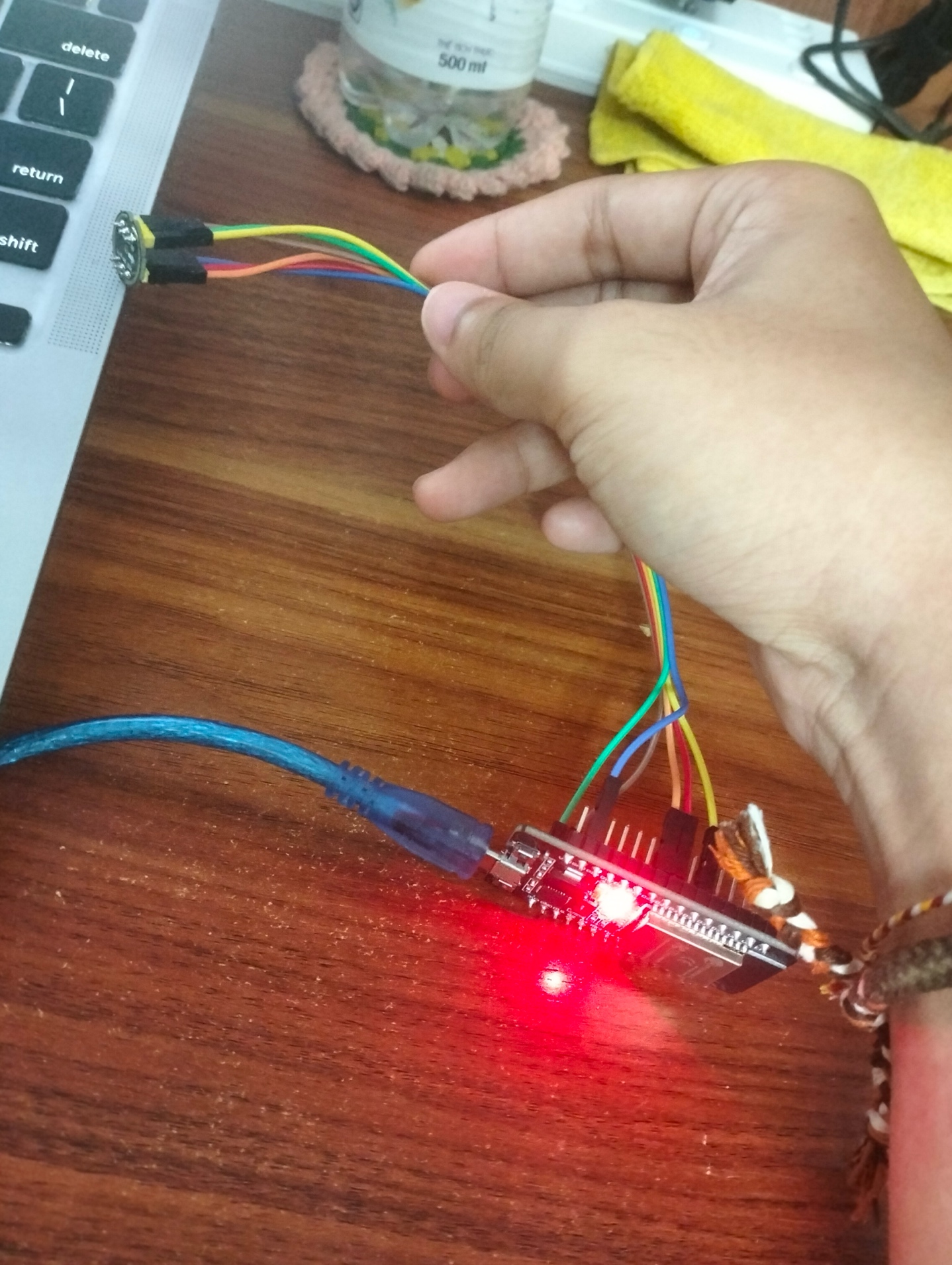
Thiết lập Extracting Feature Server.



#### Hình 11: Tổng quan code Extracting Feature server

* Ghi dữ liệu thực tế

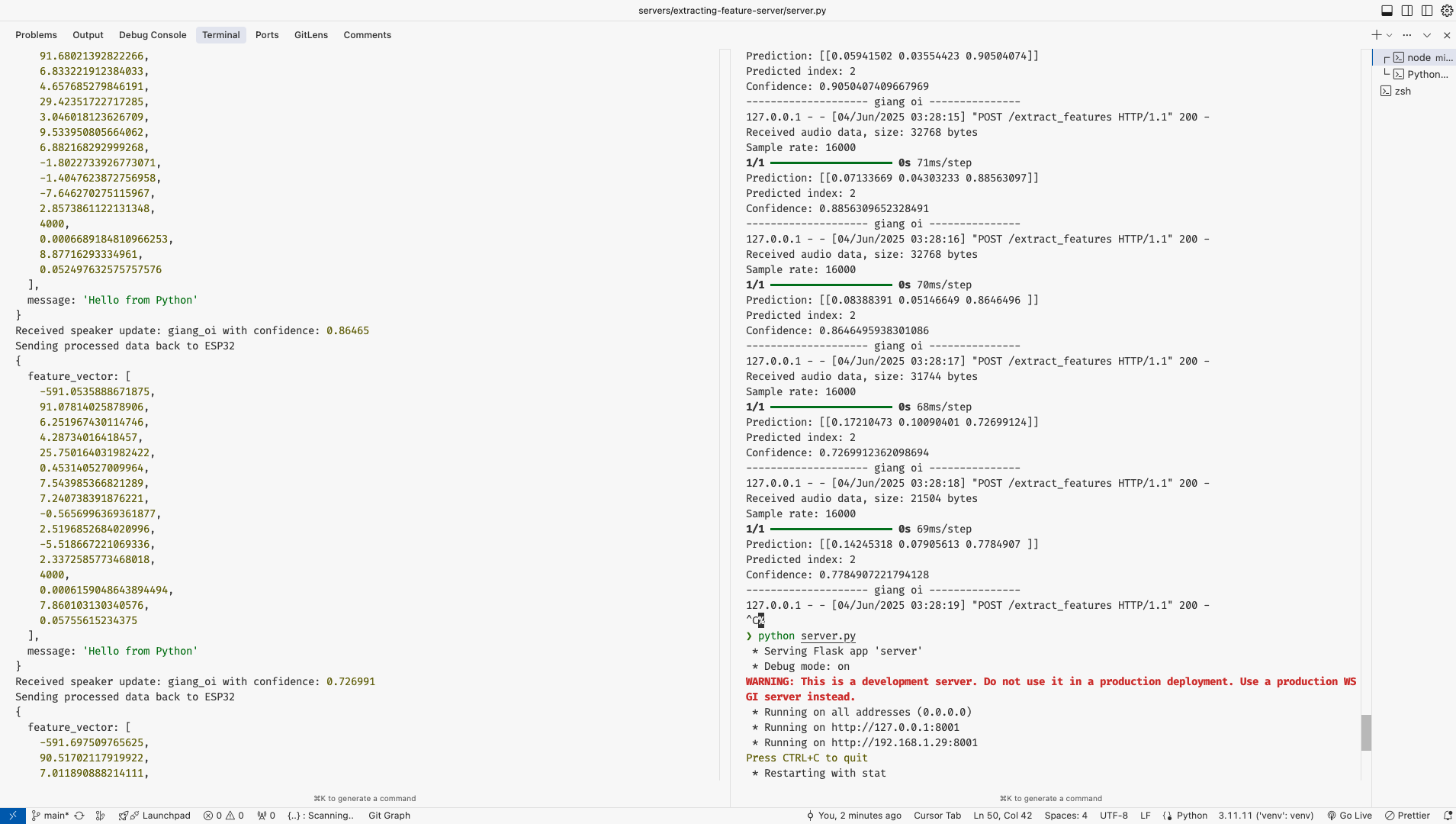
Thu âm giọng nói để kiểm tra độ chính xác nhận diện.



#### Hình 12: Sử dụng mic để thu âm thanh podcast từ loa máy tính để ESP32 nhận diện

* Nhận diện và đo lường

Kiểm tra kết quả dự đoán trên ESP32 và đánh giá hiệu suất.



Kết quả nhận diện từ ESP32

Dùng model để nhận diện đối chiếu kết quả

Server lấy đặc trưng

Server thu âm

#### Hình 13: Dữ liệu log từ 2 server thu âm và server lấy đặc trưng

## Kết quả và đánh giá

1. **Kết quả**

* ESP32 đã thu âm và gửi dữ liệu âm thanh về cho server thu âm.
* Server thu âm gửi tín hiệu đi đến server lấy đặc trưng âm thanh và gửi về lại cho server thu âm.
* Server thu âm gửi vector đặc trưng về cho ESP32 và ESP32 thực hiện dự đoán dựa trên dữ liệu đặc trưng âm thanh mà vector thu được.
* ESP32 gửi lại kết quả dự đoán cho server thu âm và server thu âm biểu diễn trên web app.

1. **Hạn chế**

* Kết quả nhận diện từ model chưa hoàn toàn chính xác vì nhiều lí do. Lí do chính có thể là do tập dữ liệu train chưa thật sự đủ vì trong audio podcast có thể có nhạc nền, âm thanh khác ngoài tiếng người. Do đó, model chưa thật sự được train với hoàn toàn âm thanh giọng nói của người.
* Quy trình hoạt động của hệ thống còn phức tạp, cần sử dụng 2 server để thu thập và xử lý dữ liệu.
* Chưa tích hợp được tính toán vector đặc trưng âm thanh ngay trên mạch ESP32.

1. **Hướng phát triển**

* Thiết bị tự vận hành: ESP32 có thể được thiết kế để hoạt động độc lập, tự thu âm, xử lý dữ liệu và đưa ra kết quả mà không cần kết nối liên tục với máy chủ. Điều này giúp tăng tính linh hoạt, nhất là trong các ứng dụng ngoại tuyến.
* Tăng cường khả năng khử nhiễu: Ứng dụng thuật toán noise reduction như spectral subtraction hoặc adaptive filtering giúp loại bỏ tạp âm và tăng chất lượng tín hiệu giọng nói trước khi trích xuất đặc trưng.
* Tối ưu mô hình để nhẹ hơn, nhanh hơn: Giảm tải tài nguyên bằng cách quantization để giảm kích thước mô hình, đồng thời thử nghiệm TinyML, giúp ESP32 xử lý suy luận với tốc độ cao hơn mà không ảnh hưởng nhiều đến độ chính xác.
* Ứng dụng thực tế: Hệ thống có thể được tích hợp vào thiết bị IoT, dùng trong điều khiển bằng giọng nói, hệ thống bảo mật, trợ lý ảo nhúng hoặc thiết bị nhận diện danh tính trong doanh nghiệp.

TÀI LIỆU THAM KHẢO

* Voice-Authentication Model Based on Deep Learning for Cloud Environment
* <https://www.analyticsvidhya.com/blog/2021/06/mfcc-technique-for-speech-recognition/>
* <https://jonathan-hui.medium.com/speech-recognition-feature-extraction-mfcc-plp-5455f5a69dd9>

PHỤ LỤC

Mã nguồn: [User voice authentication | GitHub Repo](https://github.com/nnminh-sam/user-voice-authentication-cnn).