BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**TÌM HIỂU CẤU TRÚC FILE ÂM THANH (WAV), THỂ HIỆN ĐỒ HỌA DẠNG SÓNG, PHỔ VÀ CÁC THÔNG TIN HEADER.**

**GVHD:** Nguyễn Thị Hương Lý

**Lớp:** 64.CNTT-1

**Sinh viên thực hiện:** Phạm Nhật Trường

**MSSV :** 64132785

**NHA TRANG – 2024**

# MỤC LỤC

[MỤC LỤC 2](#_Toc186924796)

[LỜI NÓI ĐẦU 3](#_Toc186924797)

[PHẦN MỞ ĐẦU. 4](#_Toc186924798)

[1. Lý do chọn đề tài. 4](#_Toc186924799)

[2. Mục tiêu của đề tài. 5](#_Toc186924800)

[3. Phạm vi nghiên cứu. 5](#_Toc186924801)

[4. Phương pháp nghiên cứu. 6](#_Toc186924802)

[CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI. 7](#_Toc186924803)

[1.1 CƠ SỞ LÝ THUYẾT. 7](#_Toc186924804)

[1.1.1 Tổng quan về file âm thanh (wav). 7](#_Toc186924805)

[1.1.1.1 Khái niệm định nghĩa file âm thanh (wav). 7](#_Toc186924806)

[1.1.1.2 Đặc điểm và vai trò của file âm thanh (wav). 7](#_Toc186924807)

[1.1.1.3 Ưu và nhược điểm của file âm thanh (wav). 8](#_Toc186924808)

[1.1.1.4 Header của file âm thanh WAV 9](#_Toc186924809)

[1.1.2 Đồ họa dạng sóng và phổ 11](#_Toc186924810)

[1.1.2.1 Dạng sóng (Waveform). 11](#_Toc186924811)

[1.1.2.2 Phổ (Spectrum). 13](#_Toc186924812)

[1.1.2.3 Biểu Đồ Phổ (Spectrograms). 14](#_Toc186924813)

[1.1.2.4 Biến đổi Fourier. 16](#_Toc186924814)

[1.1.2.5 Biến đổi Fourier rời rạc (DFT). 18](#_Toc186924815)

[1.1.2.6 Biến đổi Fourier rời rạc (FFT). 20](#_Toc186924816)

[1.1.2.7 Biến đổi Fourier trong phân tích phổ. 23](#_Toc186924817)

[1.2 CÔNG CỤ VÀ NGÔN NGỮ LẬP TRÌNH ĐƯỢC SỬ DỤNG. 24](#_Toc186924818)

[1.2.1 Visual Studio Code (VS Code). 24](#_Toc186924819)

[1.2.2 Python Jupyter Notebook. 25](#_Toc186924820)

[1.2.3 Ngôn Ngữ Lập Trình Python. 25](#_Toc186924821)

[CHƯƠNG 2. ĐẶC TẢ BÀI TOÁN. 26](#_Toc186924822)

[2.1 YÊU CẦU ĐỀ TÀI. 26](#_Toc186924823)

[2.2 HƯỚNG XỬ LÝ BÀI TOÁN 26](#_Toc186924824)

[2.2.1 Đọc và thể hiện các thông tin header của file âm thanh wav, hiển thị dạng sóng. 26](#_Toc186924825)

[2.2.1.1 Đọc và thể hiện các thông tin header của file âm thanh wav. 26](#_Toc186924826)

[2.2.1.2 Hiển thị đồ họa dạng sóng. 29](#_Toc186924827)

[2.2.1.3 Phân tích phổ và hiển thị. 30](#_Toc186924828)

[CHƯƠNG 3. CÀI ĐẶT CHƯƠNG TRÌNH. 33](#_Toc186924829)

[3.1 MÃ NGUỒN CHƯƠNG TRÌNH 33](#_Toc186924830)

[3.1.1 Mức 1 33](#_Toc186924831)

[3.1.1.1 Đọc thông tin header file WAV. 33](#_Toc186924832)

[3.1.1.2 Đồ họa dạng sóng của file âm thanh. 34](#_Toc186924833)

[3.1.1 Mức 2. 36](#_Toc186924834)

[3.1.2.1 Phân tích phổ, biểu đồ phổ và hiển thị. 36](#_Toc186924835)

[CHƯƠNG 4. KẾT LUẬN. 38](#_Toc186924836)

[4.1 KẾT QUẢ ĐẠT ĐƯỢC 38](#_Toc186924837)

[4.2 ƯU ĐIỂM 39](#_Toc186924838)

[4.3 NHƯỢC ĐIỂM 40](#_Toc186924839)

[4.4 HƯỚNG PHÁT TRIỂN. 40](#_Toc186924840)

[TÀI LIỆU THAM KHẢO 40](#_Toc186924841)

# LỜI NÓI ĐẦU

Trước hết, em xin gửi đến quý thầy, cô Khoa Công nghệ Thông tin - Trường Đại học Nha Trang lời cảm ơn chân thành nhất vì đã tạo điều kiện và môi trường học tập tuyệt vời để em có thể hoàn thành bài báo cáo thực tập cơ sở này.

Đặc biệt, em xin bày tỏ lòng biết ơn sâu sắc đến cô Nguyễn Thị Hương Lý, người đã tận tình hướng dẫn, chỉ bảo và luôn đồng hành cùng em trong suốt quá trình thực hiện đề tài. Chính sự tận tâm, kiên nhẫn và những góp ý quý báu của cô đã giúp em vượt qua những khó khăn và hoàn thiện bài báo cáo này. Em thực sự trân trọng sự tận tụy và tâm huyết của cô dành cho em, không chỉ trong đề tài này mà còn trong suốt quá trình học tập tại trường.

Trong quá trình thực hiện đề tài và bài báo cáo, với kiến thức và kinh nghiệm thực tiễn còn hạn chế, em hiểu rằng không thể tránh khỏi những thiếu sót. Vì vậy, em rất mong nhận được sự góp ý từ quý thầy, cô để có thể cải thiện và hoàn thiện hơn, đồng thời tích lũy thêm kinh nghiệm và kỹ năng cần thiết cho chặng đường phía trước.

Một lần nữa, em xin chân thành cảm ơn và kính chúc cô Nguyễn Thị Hương Lý cùng quý thầy, cô nhiều sức khỏe, thành công và hạnh phúc!

# PHẦN MỞ ĐẦU.

## 1. Lý do chọn đề tài.

Âm thanh đóng một vai trò quan trọng trong nhiều lĩnh vực của đời sống, từ truyền thông, giải trí đến học thuật. File WAV (Waveform Audio File Format) là một trong những định dạng âm thanh phổ biến nhất hiện nay nhờ khả năng lưu trữ dữ liệu âm thanh không nén, đảm bảo chất lượng cao. Việc nghiên cứu cấu trúc của file WAV không chỉ giúp hiểu rõ cách thức dữ liệu âm thanh được tổ chức và lưu trữ mà còn mở ra những hướng ứng dụng trong phân tích và xử lý tín hiệu âm thanh.

Đồ họa dạng sóng và phân tích phổ âm thanh là những công cụ trực quan và mạnh mẽ để tiếp cận và hiểu rõ hơn về bản chất của âm thanh. Hiển thị dạng sóng giúp chúng ta hình dung được sự thay đổi của tín hiệu âm thanh theo thời gian, trong khi phân tích phổ tần số cho phép khám phá thành phần cấu tạo của âm thanh. Những kỹ thuật này được ứng dụng rộng rãi trong các lĩnh vực như kỹ thuật âm thanh, nghiên cứu khoa học và phát triển sản phẩm công nghệ.

Đề tài này cũng mang lại cơ hội để rèn luyện và phát triển các kỹ năng quan trọng, như lập trình Python và làm việc với file nhị phân. Đây là những kỹ năng thiết yếu không chỉ trong xử lý tín hiệu âm thanh mà còn trong nhiều lĩnh vực liên quan đến công nghệ phần mềm. Hơn nữa, việc áp dụng thuật toán Biến đổi Fourier (FFT) để phân tích phổ là một bước tiến quan trọng trong việc làm quen với các kỹ thuật xử lý tín hiệu số hiện đại.

Với những lý do trên, đề tài không chỉ giúp mở rộng hiểu biết về âm thanh mà còn tạo nền tảng cho các ứng dụng thực tế, góp phần phát triển kỹ năng lập trình và khả năng nghiên cứu trong các lĩnh vực công nghệ và kỹ thuật.

## 2. Mục tiêu của đề tài.

Hiểu cấu trúc file WAV và cách truy xuất thông tin header: Mục tiêu đầu tiên là nắm vững cách tổ chức dữ liệu trong file WAV, bao gồm các thành phần cơ bản trong phần header như kích thước file, số lượng kênh, tần số mẫu (sample rate),… và các thông tin liên quan khác. Việc truy xuất và hiển thị các thông tin này sẽ giúp hiểu rõ cách file WAV lưu trữ dữ liệu âm thanh không nén, tạo nền tảng cho việc phân tích và xử lý âm thanh trong tương lai.

Hiển thị đồ họa dạng sóng của tín hiệu âm thanh: Tín hiệu âm thanh sẽ được hiển thị dưới dạng đồ họa dạng sóng (waveform), sử dụng các công cụ và thư viện lập trình.Điều này giúp trực quan hóa tín hiệu âm thanh, giúp người dùng dễ dàng nhận biết được sự biến đổi của tín hiệu theo thời gian, từ đó hỗ trợ cho việc phân tích và đánh giá tín hiệu.

Phân tích phổ tần số của âm thanh: Thực hiện phân tích phổ tần số bằng thuật toán Biến đổi Fourier nhanh (FFT), qua đó hiển thị đồ họa phổ (spectrogram) của tín hiệu âm thanh. Mục tiêu này giúp xác định các thành phần tần số trong âm thanh, làm rõ hơn về cấu trúc tín hiệu và phục vụ cho các ứng dụng âm thanh như lọc nhiễu, nhận dạng giọng nói hay phân tích nhạc cụ.

Đồng thời, thông qua đề tài, em cũng hướng tới việc phát triển các kỹ năng lập trình và làm quen với những công cụ khoa học phổ biến như numpy, matplotlib, scipy... những thư viện mạnh mẽ trong phân tích và xử lý dữ liệu. Việc sử dụng các công cụ này không chỉ hỗ trợ thực hiện đề tài mà còn chuẩn bị nền tảng quan trọng cho các nghiên cứu và dự án lớn hơn trong tương lai, đặc biệt trong lĩnh vực xử lý tín hiệu và khoa học dữ liệu.

## 3. Phạm vi nghiên cứu.

Phạm vi nghiên cứu của đề tài tập trung vào việc phân tích các file âm thanh định dạng WAV, một trong những định dạng phổ biến và dễ tiếp cận nhất trong xử lý âm thanh số. Nội dung nghiên cứu bao gồm tìm hiểu cấu trúc file WAV, cách tổ chức dữ liệu, các thông tin trong phần header và các thành phần cơ bản khác. Bên cạnh đó, đề tài còn hướng đến việc trực quan hóa tín hiệu âm thanh thông qua đồ họa dạng sóng, giúp phân tích sự thay đổi tín hiệu theo thời gian, và thực hiện phân tích phổ âm thanh bằng thuật toán Biến đổi Fourier nhanh (FFT) để khám phá thành phần tần số, hiển thị dưới dạng phổ và biểu đổ phổ. Phạm vi đề tài được giới hạn ở việc xử lý và phân tích các file WAV cơ bản, không mở rộng sang các định dạng âm thanh khác hoặc các phương pháp xử lý âm thanh nâng cao.

Đặc điểm của file WAV được nghiên cứu.

* **Độ sâu bit (bit depth):** Đề tài sẽ nghiên cứu file WAV với độ sâu bit 16. Độ sâu bit 16 là một trong những lựa chọn phổ biến trong việc lưu trữ âm thanh không nén, đảm bảo chất lượng âm thanh cao mà không gây tốn nhiều dung lượng lưu trữ như các độ sâu bit cao hơn.
* **Tần số mẫu (sample rate):** Đề tài sẽ giới hạn nghiên cứu ở các file WAV có tần số mẫu 44.1 kHz. Đây là tần số mẫu tiêu chuẩn được sử dụng trong ngành công nghiệp âm nhạc và truyền thông, đặc biệt là trong các đĩa CD âm thanh, đảm bảo khả năng tái tạo âm thanh chính xác từ 20 Hz đến 20 kHz – dải tần mà tai người có thể nghe được.

## 4. Phương pháp nghiên cứu.

**Nghiên cứu lý thuyết:** Tìm hiểu tài liệu và các nguồn tham khảo về cấu trúc file WAV, các thuật toán xử lý tín hiệu âm thanh cơ bản như biến đổi Fourier(DFT, FFT...), khai thác các tài liệu liên quan đến cách tổ chức dữ liệu và thông tin header trong file âm thanh WAV.

**Thực hành và lập trình:** Sử dụng ngôn ngữ lập trình Python kết hợp với các thư viện như numpy, matplotlib, scipy, wave, để làm việc và đọc dữ liệu từ file WAV, truy xuất thông tin header và tổ chức dữ liệu của file âm thanh. Xây dựng các đoạn mã để biểu thị đồ họa dạng sóng, phổ và biểu đồ dạng phổ của file âm thanh WAV. Xây dựng thuật toán FFT để tiến hành phân tích thành phần tần số của tín hiệu âm thanh để từ đó tiến hành phân tích phổ của âm thanh.

**Kiểm thử và điều chỉnh:** Kiểm tra kết quả thực hành với các file Wav có các thông số khác nhau (tần số mẫu, số kênh, độ sâu bit, thời lượng...) để đảm bảo tính chính xác và khả năng ứng dụng của các phương pháp và tiến hành điều chỉnh lại nếu cần thiết.

# CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI.

## CƠ SỞ LÝ THUYẾT.

### 1.1.1 Tổng quan về file âm thanh (wav).

#### 1.1.1.1 Khái niệm định nghĩa file âm thanh (wav).

**WAV** (viết tắt của Waveform Audio File Format) là một định dạng file âm thanh tiêu chuẩn được phát triển bởi Microsoft và IBM vào năm 1991. Đây là định dạng file âm thanh không nén, thuộc loại RIFF (Resource Interchange File Format), có khả năng lưu trữ dữ liệu âm thanh với chất lượng cao mà không làm mất thông tin so với tín hiệu âm thanh gốc.

Cấu trúc của một file WAV bao gồm hai phần chính: header và data.

* **Header** là nơi lưu trữ thông tin về file, như định dạng âm thanh, số kênh, độ sâu bit (bit depth), tần số mẫu (sample rate), và tổng số khung hình (frames). Phần này giúp các phần mềm phát âm thanh hoặc xử lý tín hiệu hiểu được cách giải mã dữ liệu.
* **Data** là phần chứa các mẫu tín hiệu âm thanh (audio samples), được lưu trữ dưới dạng nhị phân.

#### 1.1.1.2 Đặc điểm và vai trò của file âm thanh (wav).

File WAV (Waveform Audio File Format) sở hữu nhiều đặc điểm nổi bật, khiến nó trở thành một trong những định dạng âm thanh phổ biến và được ưa chuộng trong ngành công nghiệp âm thanh. Đặc điểm chính của WAV là khả năng lưu trữ dữ liệu âm thanh ở dạng không nén (PCM - Pulse Code Modulation), đảm bảo chất lượng âm thanh nguyên bản mà không làm mất mát thông tin. Tuy nhiên, chính vì không áp dụng nén, file WAV thường có dung lượng lớn hơn so với các định dạng nén như MP3 hay AAC. Với cấu trúc đơn giản gồm hai phần chính (Header và Data), WAV rất dễ đọc, phân tích và xử lý bởi các phần mềm và thư viện âm thanh, đồng thời hỗ trợ lưu trữ âm thanh với nhiều kênh (mono, stereo, surround) và các mức chất lượng khác nhau (ví dụ: 16-bit/44.1kHz, 24-bit/48kHz). WAV cũng có tính tương thích cao, được hỗ trợ rộng rãi trên nhiều hệ điều hành và phần mềm.

Vai trò của file WAV rất quan trọng trong các lĩnh vực liên quan đến âm thanh. Trong ngành công nghiệp âm nhạc, WAV là tiêu chuẩn để lưu trữ và trao đổi dữ liệu âm thanh trước khi nén sang các định dạng khác như MP3 hoặc AAC. Với chất lượng nguyên bản, file WAV phù hợp cho các ứng dụng đòi hỏi chất lượng âm thanh cao, chẳng hạn như thu âm phòng thu, phát thanh, hoặc sản xuất âm nhạc và phim ảnh. Định dạng này cũng đóng vai trò lớn trong nghiên cứu và phân tích tín hiệu âm thanh, bởi dữ liệu không nén giúp thực hiện các thao tác như hiển thị đồ họa dạng sóng, phân tích phổ tần số, hoặc lọc nhiễu một cách chính xác. Ngoài ra, trong môi trường học thuật, file WAV được sử dụng rộng rãi làm dữ liệu đầu vào cho các thuật toán xử lý tín hiệu số, hỗ trợ sinh viên và nhà nghiên cứu kiểm tra và phát triển các phương pháp mới.

#### 1.1.1.3 Ưu và nhược điểm của file âm thanh (wav).

**Ưu điểm của file âm thanh định dạng WAV:**

**Chất lượng âm thanh vượt trội:** WAV lưu trữ dữ liệu âm thanh ở dạng không nén, giúp giữ nguyên chất lượng âm thanh gốc mà không bị mất mát thông tin. Nhờ đó, định dạng này đảm bảo mang đến trải nghiệm âm thanh tốt nhất, gần như giống hệt với bản ghi âm ban đầu, phù hợp cho các ứng dụng yêu cầu độ chính xác cao về âm thanh.

**Khả năng tương thích cao:** Định dạng WAV được hỗ trợ rộng rãi trên nhiều thiết bị và phần mềm xử lý âm thanh. Người dùng có thể phát file WAV trên hầu hết các trình phát đa phương tiện, ứng dụng ghi âm, và phần mềm chỉnh sửa âm thanh chuyên nghiệp mà không cần phải chuyển đổi định dạng.

**Hỗ trợ ghi âm chất lượng cao:** WAV cho phép ghi âm ở chế độ cao cấp, lưu giữ mọi chi tiết của tín hiệu âm thanh một cách chính xác. Điều này đặc biệt quan trọng trong các ứng dụng như thu âm phòng thu, lưu trữ dữ liệu nghiên cứu âm thanh, hoặc sản xuất nhạc và phim ảnh, nơi chất lượng âm thanh được đặt lên hàng đầu.

**Nhược điểm của file âm thanh định dạng WAV:**

**Kích thước tập tin lớn:** Do không nén dữ liệu, kích thước của file WAV lớn hơn rất nhiều so với các định dạng âm thanh khác như MP3 hay AAC. Điều này gây khó khăn trong việc lưu trữ hoặc chia sẻ file qua internet hoặc các thiết bị có dung lượng hạn chế.

**Không hỗ trợ metadata đầy đủ:** Không giống như MP3 hoặc FLAC, WAV không hỗ trợ metadata đầy đủ, chẳng hạn như thông tin về tác giả, album, năm phát hành, hoặc thể loại âm nhạc. Điều này hạn chế khả năng quản lý và tổ chức file âm thanh.

**Tính linh hoạt hạn chế:** Mặc dù WAV có thể được nén bằng codec Audio Compression Manager (ACM), nó không phải là lựa chọn tối ưu cho việc chia sẻ hoặc lưu trữ, do thiếu khả năng tối ưu kích thước file.

#### 1.1.1.4 Header của file âm thanh WAV

Header của file WAV là phần dữ liệu đầu tiên trong cấu trúc file, đóng vai trò định nghĩa thông tin cơ bản về file âm thanh, giúp các ứng dụng hoặc trình phát hiểu cách xử lý dữ liệu âm thanh. Header bao gồm các trường thông tin cụ thể, được tổ chức theo định dạng RIFF (Resource Interchange File Format). Header của file WAV thường nằm trong 44 byte đầu tiên của file âm thanh nếu nó sử dụng định dạng PCM không nén (Linear PCM)

**Thông Tin Header của file WAV có thể trình bày như sau :**

**RIFF**

* **Vị trí:** Byte 1-4
* **Kích thước:** 4 byte
* **Nội dung:** Giá trị cố định "RIFF" (chuỗi ký tự) để xác định rằng file này là một file thuộc chuẩn RIFF.

**Chunk Size**

* **Vị trí:** Byte 5-8
* **Kích thước:** 4 byte (số nguyên 32-bit)
* **Nội dung:** Kích thước tổng thể của file WAV (trừ 8 byte đầu tiên cho "RIFF" và "Chunk Size").

**Format**

* **Vị trí:** Byte 9-12
* **Kích thước:** 4 byte
* **Nội dung:** Giá trị cố định "WAVE" (chuỗi ký tự) để xác định định dạng âm thanh này là file âm thanh dạng WAV

**fmt Chunk (Subchunk1)**

* **Vị trí:** Byte 13-36
* **Kích thước:** 24 byte (hoặc nhiều hơn, tùy định dạng)
* **Nội dung :**
* **"fmt " (4 byte):** Định danh "fmt chunk".
* **Subchunk1 Size (4 byte):** Kích thước dữ liệu của fmt chunk (thường là 16 byte).
* **Audio Format (2 byte):** Loại định dạng (1: PCM, âm thanh không nén).
* **Number of Channels (2 byte):** Số kênh âm thanh (1: mono, 2: stereo).
* **Sample Rate (4 byte):** Tần số mẫu (ví dụ: 44100 Hz). Là số lượng mẫu âm thanh được lấy trên mỗi giây để đại diện cho tín hiệu âm thanh liên tục.Ở đây giả sử là 44100Hz thì có nghĩa là âm thanh được lấy 44100 mẫu mỗi giây.Điều đó có nghĩa là tần số mẫu càng cao thì âm thanh thu được sẽ càng chi tiết hơn.
* **Byte Rate (4 byte):** Số byte mỗi giây = Sample Rate(Tần số mẫu) × Number of Channels(Số Kênh) × Bits Per Sample(Bit mỗi mẫu) / 8.
* **Block Align (2 byte):** Số byte cần thiết để lưu trữ một mẫu âm thanh cho tất cả các kênh.Nó xác định kích thước mỗi mẫu âm thanh khi lưu vào file bao gồm dữ liệu cho tất cả các kênh.
* **Bits Per Sample (2 byte):** Độ sâu bit (ví dụ: 16-bit).Cho biết số byte mà mỗi mẫu âm thanh (sample) sử dụng để lưu trữ dữ liệu.Giá trị này phản ánh độ chính xác qua độ phân giải của âm thanh.Mỗi mẫu âm thanh càng sử dụng nhiều byte thì độ chính xác của âm càng cao.

**data Chunk (Subchunk2)**

* **Vị trí:** Sau fmt chunk
* **Kích thước:** 8 byte cho header và X byte cho dữ liệu âm thanh
* **Nội dung:**
  + "data" (4 byte): Định danh "data chunk".
  + Subchunk2 Size (4 byte): Kích thước dữ liệu âm thanh (số byte thực tế).

Audio Data (X byte): Các mẫu tín hiệu âm thanh được lưu dưới dạng nhị phân.

**Bảng 1. Tóm tắt các thông tin Header của file âm thanh WAV**

|  |  |  |
| --- | --- | --- |
| **Vị trí** | **Giá trị mẫu** | **Mô Tả** |
| 1 - 4 | "RIFF" | Đánh dấu file là dạng RIFF. |
| 5 - 8 | File size | Kích thước toàn bộ file (trừ 8 byte đầu), lưu dưới dạng số nguyên 32 bit. |
| 9 - 12 | "WAVE" | Header định dạng file. Trong trường hợp này, luôn là "WAVE" |
| 13 - 16 | "fmt " | Đánh dấu phần định dạng của file. |
| 17 - 20 | 16 | Độ dài của dữ liệu định dạng, thường là 16 byte. |
| 21 - 22 | 1 | Loại định dạng (1 là PCM - âm thanh không nén).  **Các loại khác**  **1**: PCM (Pulse Code Modulation) - Âm thanh không nén.  **6:** A-law - Âm thanh nén theo chuẩn A-law (thường dùng trong viễn thông).  **7:** µ-law - Âm thanh nén theo chuẩn µ-law (cũng phổ biến trong viễn thông).  **17:** IMA ADPCM - Âm thanh nén theo Adaptive Differential Pulse Code Modulation.  **85:** MP3 - Mã hóa âm thanh bằng MPEG Layer III. |
| 23 - 24 | 2 | Số lượng kênh âm thanh (2 kênh = Stereo). |
| 25 - 28 | 44100 | Tần số mẫu, thường là 44,100 Hz (CD). Đây là số mẫu được thu mỗi giây. |
| 29 - 32 | 176400 | Tốc độ dữ liệu (Byte Rate): (Tần số mẫu × Bit mỗi mẫu × Số kênh) / 8. |
| 33 - 34 | 4 | Kích thước mỗi khối dữ liệu (Block Align): (Bit mỗi mẫu × Số kênh) / 8. |
| 35 - 36 | 16 | Số bit trên mỗi mẫu (16 bit). |
| 37 - 40 | "data" | Đánh dấu phần bắt đầu của dữ liệu âm thanh, |
| 41 - 44 | File size | Kích thước của phần dữ liệu âm thanh. |

### 1.1.2 Đồ họa dạng sóng và phổ

#### 1.1.2.1 Dạng sóng (Waveform).

Dạng sóng là biểu đồ biểu diễn tín hiệu âm thanh theo miền thời gian, giúp trực quan hóa sự biến thiên của tín hiệu âm thanh trong một khoảng thời gian. Dạng sóng cung cấp thông tin quan trọng về độ lớn (biên độ) của tín hiệu tại mỗi thời điểm.

**Trục X** (Thời gian - Time): Đại diện cho khoảng thời gian của tín hiệu âm thanh, giúp chúng ta phân tích tín hiệu trong miền thời gian (Time Domain).

**Trục Y** (Biên độ - Amplitude): Biểu diễn biên độ của tín hiệu tại từng thời điểm. Biên độ này thể hiện mức độ mạnh hay yếu (cường độ) của âm thanh tại thời điểm đó.

**Đặc điểm chính**

Dạng sóng cho phép chúng ta quan sát tín hiệu âm thanh thay đổi theo thời gian.

Đây là cách trực quan để nhận biết các đặc điểm như: nhịp điệu, độ lớn, hoặc các khoảng lặng trong âm thanh.

**Ứng dụng**:

Hiển thị dạng sóng giúp nhận biết ngay các phần âm thanh lớn, nhỏ, hoặc các đoạn cao trào trong một tệp âm thanh. Phân tích các đặc trưng như thời lượng tín hiệu, khoảng ngắt quãng, hoặc sự dao động bất thường trong tín hiệu.

A graph with blue lines

Description automatically generated

**Hình 1.Ví dụ về đồ họa dạng sóng của file âm thanh**

A blue sound wave with numbers

Description automatically generated

**Hình 2. Hình ảnh minh họa về đồ họa dạng sóng của file âm thanh.**

#### 1.1.2.2 Phổ (Spectrum).

Phổ (Spectrum) là tập hợp các tần số tạo nên tín hiệu âm thanh. Mỗi tín hiệu âm thanh phức tạp (như giọng nói, nhạc cụ) thực chất là sự kết hợp của nhiều tần số thành phần với các biên độ khác nhau.Ở phổ chúng ta có thể thấy được các tần số có trong tín hiệu, cường độ hoặc năng lượng của từng tần số

Ví dụ : Một đoạn nhạc có thể chứa các tần số cao (âm bổng), tần số thấp (âm trầm), hình ảnh phổ dưới đây cho thấy spectrum của một đoạn nhạc. Phổ (Spectrum) hiển thị tất cả các tần số có trong tín hiệu cùng với cường độ của mỗi tần số.

A graph of a number

Description automatically generated

**Hình 3. Phổ của một đoạn nhạc được phân tích và hiển thị.**

Phổ hiểu đơn giản là ví dụ như bạn đang nghe một đoạn nhạc. Âm thanh mà chúng ta nghe được thực chất là sự kết hợp của nhiều tần số khác nhac. Mỗi tần số này đóng góp một phần vào âm thanh tổng thể. Và Phổ (Spectrum) sẽ cho bạn biết những tần số nào đang có mặt trong âm thanh đó và cường độ của từng tần số đang có mặt trong đoạn nhạc này.

Như đã được nhắc ở trên đồ họa dạng sóng(waveform) hiển thị biên độ(độ lớn ) của âm thanh theo thời gian, trục x hiển thị khoảng thời gian của tín hiểu, vì thế chúng ta đang xem tín hiệu trong Miền Thời Gian(Time Domain).

Phổ (Spectrum) là một cách khác để biễu diễn cùng một tín hiệu đó. Thay vì hiển thị biên độ theo thời gian, phổ hiển thì biên độ theo tần số. Trong biểu đồ spectrum , trục x thể hiện khoảng giá trị tần số của tín hiệu tại một thời điểm cụ thể. Vì thế chúng ta đang xem tín hiệu trong miền tần số(Frequency Domauni).

Nếu như bạn đang nghe một đoạn ghi âm giọng nói. Khi nhìn vào biểu đồ trong miền thời gian ở đây là đồ họa dạng sóng của âm thanh sau khi được phân tích bạn sẽ chỉ thấy được biên độ tín hiệu âm thanh thay đổi theo thời gian, bạn có thể thấy được mức độ to nhỏ của âm thanh tại mỗi thời điểm và bạn sẽ chẳng biết được tần số nào đang có mặt trong đó có.

Nhưng nếu bạn nhìn vào phổ (miền tần số), chúng ta sẽ thấy một biểu đồ hiển thị biên độ của các tần số khác nhau có trong tín hiệu tại một thời điểm cụ thể. Điều này có thể giúp chúng ta biết được tần số nào đang có mặt trong âm thanh này và độ lớn, cường độ nó ra sao.Và chắc chắn điều này rất quan trọng trong nhiều ứng dụng chẳng hạn như xử lý âm thanh, nhận diện giọng nói…

#### **1.1.2.3 Biểu Đồ Phổ (Spectrograms**).

Khi một tín hiệu thay đổi theo thời gian, các tần số cấu tạo tạo nên nó cũng sẽ thay đổi theo thời gian. Có thể nói cách khác là , phổ(Spectrum) của tín hiệu cũng có thể thay đổi theo thời gian.Trước đó chúng ta đã đề cập tới phổ là kết quả của việc áp dụng biến đổi **Fourier** (Fourier Transform) cho một tín hiệu trong miền thời gian. Nó cung cấp thông tin về các thành phần tần số có trong tín hiệu tại một thời điểm cụ thể với các tần số và biên độ của chúng.

**Biểu đồ phổ**  là một biểu đồ động thể hiện sự thay đổi của phổ theo thời gian. Nó cho thấy các tần số thành phần của tín hiệu ở mỗi thời điểm trong suốt quá trình tín hiệu thay đổi. Biểu đồ phổ Spectrogram được tạo ra bằng cách chia tín hiệu thành các cửa sổ thời gian nhỏ và thực hiện Biến đổi Fourier cho mỗi cửa sổ đó. Kết quả là một chuỗi phổ tại mỗi thời điểm. Nó giống như chúng ta chụp phổ nhiều lần tại các thời điểm khác nhau, sau đó ghép chúng lại thành một biểu đồ duy nhất.

A close-up of a graph

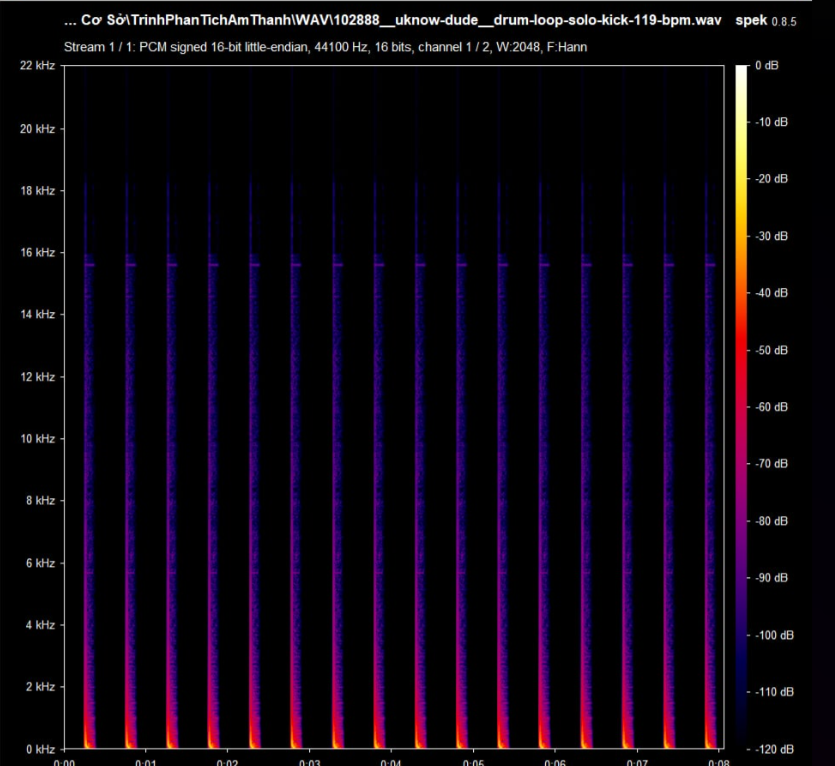
Description automatically generated

**Hình 4. Biểu đồ phổ và đồ họa dạng sóng của một đoạn âm thanh bất kỳ.**

Biểu đồ phổ sử dụng các màu sắc khác nhau để biểu diễn cường độ, độ lớn tần số. Màu càng sáng năng lượng của tín hiệu càng cao. Mỗi lát cắt dọc trong biểu đổ phổ thực chất là phổ của tín hiệu tại thời điểm đó và cho thấy cách phân bố cường độ tín hiệu trong mỗi tần số tại thời điểm đó.

Chúng ta có thể nhìn vào hình ảnh ở trên, giả sử chúng ta có 1 đoạn ghi âm. Hình ảnh đầu tiên hiển thị tín hiệu trong miền thời gian (đồ họa dạng sóng), tức là biên độ của âm thanh theo thời gian nó cho chúng ta biết mức độ âm thanh to hay nhỏ tại bất kỳ thời điềm nào. Hình ảnh thứ hai là biểu đồ phổ và hiển thị tín hiệu trong miền tần số. Nó cho thấy cách các tần số trong tín hiệu thay đổi theo thời gian và cường độ của từng tần số.

Qua đó cho thấy biểu đồ phổ là một công cụ mạnh mẽ giúp chúng ta hiểu rõ hơn về tín hiệu âm thanh. Bằng cách xem xét biểu đồ phổ chúng ta có thể thấy cách các tần số trong tín hiệu thay đồi theo thời gian và cường độ, độ lớn của từng tần số. Nó có thể giúp ích rất nhiều trong các lĩnh vực liên quan tới âm thanh dữ liệu âm thanh như phân tích âm thanh, nhận diện giọng nói.



**Hình 5. Biểu đồ phổ do phần mềm Spek vẽ cho một file âm thanh WAV.**

#### **1.1.2.4 Biến đổi Fourier.**

Biến đổi Fourier, hay còn gọi là chuyển hóa Fourier, là một phép toán quan trọng trong toán học và xử lý tín hiệu, được đặt theo tên nhà toán học người Pháp Jean-Baptiste Joseph Fourier. Phép biến đổi này cho phép chuyển đổi một hàm số hoặc tín hiệu từ miền thời gian sang miền tần số, qua đó giúp phân tích cấu trúc tần số của tín hiệu đó.

Biến đổi Fourier là một công cụ mạnh mẽ với rất nhiều ứng dụng trong các lĩnh vực khoa học và kỹ thuật, từ vật lý, toán học, xử lý tín hiệu, cho đến quang học, âm học, hải dương học, và nhiều lĩnh vực khác. Nó giúp chuyển đổi một tín hiệu từ miền thời gian sang miền tần số, qua đó phân tích các thành phần tần số của tín hiệu. Dưới đây là một số ứng dụng cụ thể và những tính chất lý thuyết quan trọng của Biến đổi Fourier:

* **Xử lý tín hiệu:** Biến đổi Fourier rất quan trọng trong các hệ thống xử lý tín hiệu, đặc biệt là trong việc phân tích và lọc tín hiệu. Ví dụ, trong điện tử, Viễn thông, và âm thanh, Biến đổi Fourier giúp phân tích các tín hiệu âm thanh và điện từ, lọc bỏ nhiễu hoặc cải thiện chất lượng tín hiệu.
* **Xử lý hình ảnh:** Biến đổi Fourier cũng được ứng dụng để xử lý và phân tích hình ảnh, chẳng hạn như trong việc làm mịn hoặc làm sắc nét hình ảnh, nén ảnh, hoặc giảm nhiễu trong ảnh vệ tinh và hình ảnh y khoa.
* **Vật lý và Toán học**: Biến đổi Fourier giúp giải các phương trình vi phân tuyến tính trong vật lý, đặc biệt là trong cơ học sóng, điện từ học và các hiện tượng dao động, với ứng dụng trong phân tích các sóng cơ học và sóng điện từ.
* **Phân tích quang học và hải dương học**: Biến đổi Fourier có thể được dùng để phân tích các tín hiệu ánh sáng hoặc sóng trong các thí nghiệm quang học. Trong hải dương học, nó được dùng để phân tích và xử lý dữ liệu sóng, giúp nhận diện các đặc trưng của sóng biển.

Tính chất của Biến đổi Fourier:

* **Tính tuyến tính**: Một trong những tính chất quan trọng của Biến đổi Fourier là tính tuyến tính, nghĩa là phép biến đổi có thể được áp dụng cho các tín hiệu kết hợp với trọng số. Cụ thể, nếu có hai tín hiệu f(t) và g(t), thì:

**A black and white image of a square with a black line

Description automatically generated**

Điều này có nghĩa là Biến đổi Fourier phân tách tín hiệu phức tạp thành các tín hiệu đơn giản hơn, giúp dễ dàng xử lý và phân tích các thành phần riêng biệt của tín hiệu.

* **Biến đổi nghịch đảo:** Biến đổi Fourier không chỉ có thể chuyển tín hiệu từ miền thời gian sang miền tần số, mà còn có một biến đổi nghịch đảo, cho phép chuyển tín hiệu từ miền tần số về lại miền thời gian. Điều này có ý nghĩa rất quan trọng trong việc tái tạo tín hiệu sau khi đã phân tích hoặc xử lý trong miền tần số.
* **Các hàm sin cơ sở**: Các hàm sin cơ sở trong Biến đổi Fourier đóng vai trò là các hàm riêng của phép vi phân. Điều này có nghĩa là khi áp dụng Biến đổi Fourier vào các phương trình vi phân tuyến tính, chúng chuyển thành các phương trình đại số đơn giản, giúp giải quyết các vấn đề vật lý một cách dễ dàng và hiệu quả.
* **Định lý tích tổng chập:** Biến đổi Fourier giúp chuyển một tích tổng chập phức tạp thành một tích đại số đơn giản. Điều này rất hữu ích trong việc giải quyết các phương trình tích phân phức tạp, đặc biệt là trong vật lý và các hệ thống tuyến tính.
* **Biến đổi Fourier rời rạc và thuật toán FFT:** Biến đổi Fourier rời rạc (DFT) là một phiên bản của Biến đổi Fourier được áp dụng cho tín hiệu số. Nhờ vào thuật toán FFT (Fast Fourier Transform), Biến đổi Fourier có thể tính toán nhanh chóng và hiệu quả trên máy tính, giúp xử lý tín hiệu trong thời gian thực, như trong các ứng dụng truyền thông và xử lý âm thanh.
* **Định lý Parseval-Plancherel:** Định lý này cho biết rằng tổng năng lượng của tín hiệu (tính bằng bình phương giá trị tuyệt đối của tín hiệu) không thay đổi khi tín hiệu được chuyển từ miền thời gian sang miền tần số. Nói cách khác, năng lượng của tín hiệu trong miền thời gian bằng năng lượng của tín hiệu trong miền tần số, giúp đảm bảo tính bảo toàn năng lượng khi xử lý tín hiệu.

#### **1.1.2.5 Biến đổi Fourier rời rạc (DFT).**

Discrete Fourier Transform (DFT)là một phép biến đổi Fourier được sử dụng để phân tích tín hiệu rời rạc (discrete-time signal) trong miền thời gian và chuyển chúng sang miền tần số rời rạc. DFT giúp chúng ta hiểu rõ hơn về các thành phần tần số của tín hiệu và rất hữu ích trong việc xử lý tín hiệu số, bao gồm âm thanh, hình ảnh, và các tín hiệu khác. Công thức tính DFT của một tín hiệu rời rạc x[n] với N mẫu (tín hiệu có độ dài N) là

A mathematical equation with numbers and symbols

Description automatically generated

Trong đó :

* X[k] là giá trị của tín hiệu trong miền tần số rời rạc k.
* x[n] là giá tín hiệu tại thời gian rời rạc n.
* N là tổng số điểm trong tín hiệu.
* k là chỉ số tần số trong miền tần số.
* e: Đây là cơ số của lôgarit tự nhiên (số Euler), khoảng 2.71828. Trong trường hợp này, e là cơ sở của hàm mũ phức.
* j: Trong lý thuyết số học và tín hiệu, j là ký hiệu cho đơn vị ảo, tức là j2 = -1. Đây là phần quan trọng trong việc làm việc với các tín hiệu và sóng dao động.
* 2π/N : Đây là bước tần số trong Biến đổi Fourier, đại diện cho chu kỳ của hàm sóng trong không gian tần số.

Sau khi sử dụng Biến đổi Fourier rời rạc (DFT) để chuyển dữ liệu từ miền thời gian sang miền tần số, thì chúng ta cũng có thể chuyển ngược lại dữ liệu từ miền tần số sang thời gian bằng cách sử dụng phép Biển đổi Fourier rời rạc nghịch (IDFT- Inverse Discrete Fourier Transform) với công thức như sau:

A math equations on a white background

Description automatically generated

Trong đó :

* X[k] là giá trị của tín hiệu trong miền tần số rời rạc k.
* x[n] là giá tín hiệu tại thời gian rời rạc n.
* N là tổng số điểm trong tín hiệu.
* k là chỉ số tần số trong miền tần số.
* e: Đây là cơ số của lôgarit tự nhiên (số Euler), khoảng 2.71828. Trong trường hợp này, e là cơ sở của hàm mũ phức.
* j: Trong lý thuyết số học và tín hiệu, j là ký hiệu cho đơn vị ảo, tức là j2 = -1. Đây là phần quan trọng trong việc làm việc với các tín hiệu và sóng dao động.
* 2π/N : Đây là bước tần số trong Biến đổi Fourier, đại diện cho chu kỳ của hàm sóng trong không gian tần số.

#### **1.1.2.6 Biến đổi Fourier rời rạc (FFT).**

FFT (Fast Fourier Transform) là một thuật toán tối ưu của Discrete Fourier Transform (DFT). Nó giúp tính toán DFT một cách nhanh chóng hơn nhiều so với phương pháp DFT truyền thống. Trong khi DFT có độ phức tạp tính toán là O(N²), thì FFT có độ phức tạp thấp hơn, chỉ O(N log N), nhờ vào việc tận dụng các tính chất đặc biệt của DFT. Vì vậy, FFT rất hữu ích trong các bài toán xử lý tín hiệu, nhất là khi xử lý các tín hiệu dài hoặc tín hiệu thời gian thực.

FFT giúp phân tách một bài toán tính DFT thành các phần nhỏ hơn và giải quyết chúng một cách hiệu quả hơn, thay vì tính toán trực tiếp DFT theo công thức.

Cụ thể, thuật toán Cooley-Tukey FFT là một trong những thuật toán FFT phổ biến nhất. Nó sử dụng phương pháp chia để trị (divide-and-conquer) để tính toán DFT nhanh chóng. Thuật toán này chia tín hiệu ban đầu thành các phần nhỏ hơn, tính DFT của từng phần và sau đó kết hợp lại để có được kết quả cuối cùng.

Thuật toán Cooley-Tukey: Đây là thuật toán FFT phổ biến nhất, được phát triển bởi James Cooley và John Tukey vào năm 1965. Thuật toán này đặc biệt hiệu quả khi số lượng mẫu N là một lũy thừa của 2 ví dụ như  . Tuy nhiên cũng sẽ có trường hợp số lượng mẫu N của dãy tính hiệu không phải là lũy thừa của 2 vậy có thể giải quyết như thế nào ?. Phương pháp phổ biến nhất để giải quyết vấn đề này là ta có thể thêm các mẫu 0 vào cuối dãy tín hiệu sao cho tổng số mẫu trở thành một lũy thừa của 2. Việc này làm tăng kích thước tín hiệu nhưng không thay đổi thông tin tần số quan trọng, vì các mẫu 0 không ảnh hưởng đến quá trình phân tích tần số của tín hiệu, sau đó ta có thể làm việc với nó dễ dàng.

**Quy trình hoạt động của FFT**

* **Chia bài toán:** Một tín hiệu dài với N mẫu sẽ được chia thành hai tín hiệu ngắn hơn có độ dài là N/2. Điều này giúp giảm bớt số phép tính cần thực hiện.
* **Tính DFT cho các tín hiệu nhỏ hơn:** Áp dụng DFT cho các phần nhỏ của tín hiệu. Thay vì tính DFT cho toàn bộ tín hiệu một cách trực tiếp, FFT sẽ tính DFT cho các phần nhỏ rồi ghép kết quả lại.
* **Kết hợp lại kết quả:** Sau khi tính toán DFT cho các phần nhỏ, kết quả sẽ được kết hợp lại để có được kết quả DFT cho toàn bộ tín hiệu.

Minh họa cách tính toán của Biến đổi FFT (Fast Fourier Transform):

Thuật toán Cooley-Tukey FFT chia bài toán DFT thành hai bài toán DFT của nửa đầu và nửa sau của tín hiệu. Cụ thể, thuật toán này sử dụng chia để trị (divide and conquer), chia tín hiệu thành các phần nhỏ hơn, và áp dụng DFT cho các phần nhỏ đó.

**Bước 1 :** Chia bài toán

Giả sử x[n] là một tín hiệu có độ dai N (với N là lũy thừa của 2). Chia tín hiệu thành 2 dãy con,:

Dãy chẵn (là các phần tử có chỉ số chẵn trong mảng dữ liệu) :



Dãy kẻ (là các phần tử có chỉ số lẻ trong mảng dữ liệu):



Công thức này giúp chia bài toán DFT của một tín hiệu có N mẫu thành hai bài toán DFT của các tín hiệu có N/2 mẫu.

**Bước 2:** Tính DFT của các dãy con.

Tiến hành tính DFT cho các dãy con. Sau khi tính xong DFT của các dãy con ta sẽ kết hợp các kết quả lại để có được DFT của tính hiệu ban đầu.

**Bước 3:** Kết hợp kết quả

Kết hợp kết quả từ DFT của dãy chẵn và dãy lẻ bằng công thức sau:

A math equations with black text

Description automatically generated with medium confidence

Chứng minh:

DFT của một tín hiệu x[n] có N mẫu được định nghĩa như sau:

A math equations on a white background

Description automatically generated

Tách tín hiệu thành dãy chẵn và lẻ: Để giảm số phép tính, FFT chia tín hiệu x[n] thành hai phần:

* Dãy chẵn :



* Dãy lẻ:



* Do đó, tín hiệu ban đầu x[n] có thể được viết lại như:

A close up of a number

Description automatically generated with medium confidence

Thay vào công thưc DFT gốc:

Với X[k], thay x[n] bằng dãy chẵn và lẻ, ta có:

A math equations with numbers

Description automatically generated with medium confidence

Với phần chẵn :

A mathematical equation with black text

Description automatically generated

Với phần lẻ :

A math equations and formulas

Description automatically generated with medium confidence

A black text on a white background

Description automatically generated

Do đó ta có công thức cuối cùng:

A math equations on a white background

Description automatically generated

**Bước 4**: Tiến hành đệ quy

* Tiến hành đệ quy tính FFT cho các dãy con nhỏ hơn (có độ dài bằng 2) cho đến khi kích thước tín hiệu là 1 (tức là tín hiệu trở thành một số phức đơn lẻ). Khi kích thước tín hiệu là 1, DFT đã được tính toán trực tiếp.

#### **1.1.2.7 Biến đổi Fourier trong phân tích phổ.**

Biến Đổi Fourier (Fourier Transform) là một công cụ toán học mạnh mẽ dùng để chuyển đổi tín hiệu từ miền thời gian (time domain) sang miền tần số (frequency domain). Quá trình này giúp chúng ta phân tích được các thành phần tần số có trong tín hiệu, cung cấp thông tin về biên độ và pha của từng tần số.

Khi làm việc với tín hiệu âm thanh hoặc tín hiệu khác, dạng sóng trong miền thời gian thường không cung cấp đủ thông tin để hiểu rõ bản chất của tín hiệu. Để biết tín hiệu có những thành phần tần số nào (ví dụ: âm trầm, âm cao) và cường độ của từng tần số, chúng ta cần chuyển tín hiệu sang miền tần số thông qua Biến Đổi Fourier. Điều này đặc biệt hữu ích trong các ứng dụng như: Loại bỏ nhiễu, phân tích phổ, nhận dạng âm thanh.

Biến Đổi Fourier là cầu nối giữa miền thời gian và miền tần số. Nó không chỉ giúp chúng ta hiểu rõ hơn về tín hiệu mà còn hỗ trợ nhiều ứng dụng trong xử lý tín hiệu, truyền thông, và khoa học âm thanh. Sự thay đổi của tín hiệu trong một miền (thời gian hoặc tần số) sẽ ảnh hưởng trực tiếp đến tín hiệu trong miền còn lại.

A diagram of a frequency function

Description automatically generated

**Hình 6. Chuyển đổi từ miền thời gian sang miền tần số.**

## CÔNG CỤ VÀ NGÔN NGỮ LẬP TRÌNH ĐƯỢC SỬ DỤNG.

### 1.2.1 Visual Studio Code (VS Code).

Visual Studio Code là một mã nguồn mở và miễn phí, được phát triển bởi Microsoft. Đây là một trong những công cụ phổ biến nhất cho lập trình viên nhờ vào giao diện người dùng thân thiện và khả năng hỗ trợ nhiều ngôn ngữ lập trình.

**Tính năng nổi bật:**

**Hỗ trợ đa ngôn ngữ:** VS Code hỗ trợ nhiều ngôn ngữ lập trình như Python, JavaScript, C++, Java, và nhiều ngôn ngữ khác thông qua các extension.

Tính năng IntelliSense: Cung cấp các gợi ý thông minh về mã nguồn giúp lập trình viên viết mã nhanh chóng và chính xác hơn.

**Debugging:** VS Code cung cấp các công cụ debugging mạnh mẽ giúp kiểm tra và sửa lỗi trong mã nguồn.

**Tích hợp Git:** VS Code cho phép tích hợp với Git, giúp quản lý mã nguồn và phối hợp làm việc nhóm hiệu quả.

**Extension phong phú:** Cung cấp một hệ sinh thái phong phú các extension giúp mở rộng tính năng của công cụ, đặc biệt là hỗ trợ các công cụ phân tích, kiểm thử, và phát triển dự án.

### 1.2.2 Python Jupyter Notebook.

Jupyter Notebook là một ứng dụng web mã nguồn mở cho phép tạo và chia sẻ tài liệu với mã nguồn, tính toán, biểu đồ và văn bản mô tả. Đây là một công cụ lý tưởng trong việc phát triển các dự án về khoa học dữ liệu và học máy.

**Tính năng nổi bật**:

**Tích hợp mã và kết quả:** Jupyter cho phép viết mã Python, chạy mã ngay trong tài liệu và hiển thị kết quả ngay lập tức. Điều này giúp lập trình viên dễ dàng kiểm thử và phân tích dữ liệu.

**Hỗ trợ nhiều ngôn ngữ:** Ngoài Python, Jupyter còn hỗ trợ nhiều ngôn ngữ khác như R, Julia, và Scala thông qua các kernel.

**Chia sẻ dễ dàng:** Các notebook có thể được chia sẻ và tái sử dụng bởi người khác, điều này rất hữu ích trong các môi trường học tập hoặc nghiên cứu.

**Visualizations:** Jupyter hỗ trợ các thư viện trực quan hóa dữ liệu như Matplotlib, Seaborn, Plotly, giúp trình bày các kết quả phân tích một cách rõ ràng và sinh động.

**Tích hợp Markdown:** Jupyter cũng hỗ trợ cú pháp Markdown cho phép bạn thêm chú thích, giải thích hoặc ghi chú trong các tài liệu, giúp làm rõ các bước thực hiện trong quá trình lập trình.

### 1.2.3 Ngôn Ngữ Lập Trình Python.

Python là một ngôn ngữ lập trình cấp cao, thông dịch, và động, được phát triển bởi Guido van Rossum vào cuối những năm 1980. Python được thiết kế để dễ dàng học và sử dụng, đồng thời có cú pháp rõ ràng, dễ đọc, giúp lập trình viên tập trung vào giải quyết vấn đề thay vì phải lo lắng về cú pháp phức tạp.

**Cú pháp đơn giản và dễ đọc:** Python có cú pháp rõ ràng, dễ hiểu, giúp lập trình viên nhanh chóng phát triển phần mềm mà không cần lo lắng về cú pháp phức tạp.

**Mã nguồn mở và miễn phí:** Python là ngôn ngữ mã nguồn mở, được phát triển và hỗ trợ bởi cộng đồng, giúp người dùng dễ dàng tiếp cận và sử dụng.

**Thư viện mạnh mẽ:** Python hỗ trợ nhiều thư viện phong phú như NumPy, Pandas, Matplotlib, TensorFlow giúp giải quyết các bài toán về dữ liệu, học máy, và khoa học dữ liệu.

**Khả năng tương thích cao:** Python có thể chạy trên nhiều hệ điều hành khác nhau và tích hợp với các ngôn ngữ khác như C, C++, Java.

**Ứng dụng rộng rãi:** Python được sử dụng trong nhiều lĩnh vực như phát triển web, khoa học dữ liệu, học máy, tự động hóa, xử lý ảnh và video.

# CHƯƠNG 2. ĐẶC TẢ BÀI TOÁN.

## 2.1 YÊU CẦU ĐỀ TÀI.

Vận dụng kiến thức lập trình xử lý yêu cầu bài toán,biết làm việc với file nhị phân. Tìm hiểu biến đổi Fourier để phân tích phổ.

**Mức 1:** Đọc và thể hiện các thông tin header của file wav, hiển thị dạng sóng.

**Mức 2:** Phân tích phổ và hiển thị.

## 2.2 HƯỚNG XỬ LÝ BÀI TOÁN

### 2.2.1 Đọc và thể hiện các thông tin header của file âm thanh wav, hiển thị dạng sóng.

#### 2.2.1.1 Đọc và thể hiện các thông tin header của file âm thanh wav.

Mục tiêu là lấy thông từ header của file âm thanh WAV chứa các thông tin quan trọng như : Kích thước file, tần số mẫu, số kênh âm thanh, định dạng file, độ sâu bit, kích thước dữ liệu âm thanh….

Xây dựng chương trình thực hiện điểu trên :



Để phân tích chúng ta cần xây dựng một hàm để xử lý việc này hàm “read\_wav\_header” sẽ thực hiện điều đó

Đầu tiên hàm này nhận vào một biến được gọi là “file\_path” biến này sẽ chứa đường dẫn file đầu vào cho hàm này tiến hành phân tích.

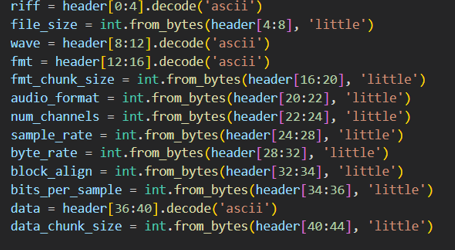
Các file âm thanh trong máy tính đều được lưu trữ dưới nhị phân , không phải dạng văn bản thông thường. Do đó chúng ta cần phải mở file ở chế độ đọc nhị phân “rb” bằng dòng lệnh sau :



Sau khi file âm thanh đã được đọc ở chế độ nhị phân ta sẽ tiến hành bước phân tích và lấy các thông tin header của file âm thanh WAV, theo tiêu chuẩn của RIFF header của file âm thanh WAV luôn có kích thước cố định là 44 byte đầu tiên. Vì vậy chúng ta sẽ thực hiện đọc 44 byte đầu tiên của file để tiến hành phân tích và lấy thông tin.



Sau khi tiến hành đọc 44 byte đầu tiên của file chúng ta sẽ tiến hành phân tích và giãi mã thông tin của file âm thanh wav để lấy thông tin. Từng phần trong header sẽ được trích xuất dựa trên chỉ số byte cố định các giá trị nhị phân sẽ được giải mã thành dạng có thể đọc được dễ dàng (chuỗi, số nguyên).



Sau khi phân tich, các thông tin được lưu trữ trong một cấu trúc có tổ chức để dễ dàng truy cập và sử dụng.

A computer code with colorful text

Description automatically generated

* **from\_byte :** Chuyển đổi chuỗi byte thành số nguyên, ví dụ như “header[4:8]” lấy 4 byte từ chỉ số 4 đến 8 trong header do chuỗi byte này thể hiện tổng kích thước của file nên cần phải chuyển thành số nguyên để có thể đọc dễ dàng
* **decode(‘ascii’) :** Chuyển đổi chuỗi byte thành chuỗi ký tự dựa trên bảng mã ASCII.
* **little :** Là tham số trong from\_bytes để chỉ định kiểu lưu trữ "little-endian" của dữ liệu nhị phân. Ở các máy tính hiện đại sử dụng CPU x86/x64 thường sử dụng little-endian làm chuẩn để lưu trữ dữ liệu trong bộ nhớ .

#### 2.2.1.2 Hiển thị đồ họa dạng sóng.

Đồ họa dạng sóng được để hiện thông qua thư viện “matplotlib”. Đây là thư viện phổ biến dùng để vẽ các đồ thị và biểu đồ trong Python. Chúng ta sẽ dùng matplotlib.pyplot để vẽ đồ thị sóng âm thanh của một file WAV.

Sóng âm thanh được thể hiện qua đồ thì dạng sóng, trong đó trục x thể hiện thời gian (tính bằng giây) và trục tung y thể hiện biên độ. Thư viện “wave” trong Python sẽ giúp chúng ta mở và đọc dữ liệu âm thanh từ file WAV một cách dễ dàng hơn, trong khi thư viện numpy xử lý dữ liệu số học từ âm thanh tạo ra các mảng số liệu để vẽ đồ thị.

A screen shot of a computer program

Description automatically generated

Sử dụng “wave.open(file\_path, "rb")” để mở file WAV dưới dạng nhị phân ("rb").

Dùng các phương thức của đối tượng wav\_file.

* + - **getnchannels():** Trả về số kênh âm thanh (1 cho mono, 2 cho stereo).
    - **getframerate():** Trả về tần số mẫu (sample rate), tức là số mẫu âm thanh được ghi lại trong một giây
    - **getnframes()**: Trả về tổng số mẫu (frames) trong file âm thanh.

Tính toán thời lượng của file âm thanh bằng công thức : thời gian = tổng số mẫu/tần số mẫu.

Dữ liệu âm thanh được chuyển thành mảng số nguyên 16-bit (np.int16) từ dữ liệu nhị phân.

Nếu file có 2 kênh (stereo), dữ liệu sẽ được chia thành hai kênh (vì mỗi kênh có một giá trị biên độ riêng). Sau đó, chúng ta chỉ lấy kênh đầu tiên (bằng cách chọn cột đầu tiên trong mảng 2D).

Tạo mảng thời gian bằng cách sử dụng “np.linspace()”, giúp tạo ra một chuỗi số cách đều nhau từ 0 đến thời gian của file, với số phần tử bằng số mẫu âm thanh.

Sử dụng matplotlib.pyplot để vẽ đồ thị. Dữ liệu audio\_data sẽ được vẽ trên trục tung, và thời gian sẽ được vẽ trên trục hoành.

**plt.plot(time, audio\_data, color="blue")** vẽ đồ thị với màu xanh dương.

**plt.title(), plt.xlabel(), plt.ylabel()** để thêm tiêu đề và nhãn cho các trục.

**plt.grid()** thêm lưới vào đồ thị để dễ nhìn hơn.

**plt.tight\_layout()** giúp điều chỉnh các khoảng cách để đồ thị không bị chồng chéo.

#### 2.2.1.3 Phân tích phổ và hiển thị.

Phổ âm thanh và biểu đồ phổ được thể hiện thông qua thư viện “matplotlib”. Đây là thư viện phổ biến dùng để vẽ các đồ thị và biểu đồ trong Python. Chúng ta sẽ dùng matplotlib.pyplot để vẽ phổ tần số và biểu đồ phổ của một file WAV.

Phổ âm thanh được biểu diễn dưới dạng một đồ thị, trong đó trục x thể hiện tần số (Hz) và trục y thể hiện biên độ. Để tính toán phổ âm thanh, chúng ta sử dụng thuật toán FFT (Fast Fourier Transform) để chuyển đổi tín hiệu âm thanh từ miền thời gian sang miền tần số.

Biểu đồ phổ (Spectrogram) lại cung cấp một cái nhìn trực quan hơn về sự thay đổi tần số của tín hiệu âm thanh theo thời gian. Trong đó, trục x thể hiện thời gian (s), trục y thể hiện tần số (Hz), và màu sắc hiển thị biên độ của tín hiệu tại mỗi thời điểm. Thư viện “wave” trong Python được dùng để mở và đọc dữ liệu âm thanh từ file WAV, trong khi thư viện numpy hỗ trợ xử lý các phép toán số học để tính toán FFT và chuẩn bị dữ liệu cho các đồ thị.

A screen shot of a computer screen

Description automatically generated

**numpy (np):** Thư viện toán học mạnh mẽ trong Python, dùng để xử lý các phép toán số học và dữ liệu dạng mảng.

**matplotlib.pyplot (plt):** Dùng để vẽ đồ thị.

**wave:** Thư viện để mở và xử lý dữ liệu âm thanh từ file WAV.

**Hàm fft(x) :** Đây là hàm tự viết để tính toán Fast Fourier Transform (FFT), một thuật toán để chuyển đổi tín hiệu từ miền thời gian sang miền tần số với đầu vào là x một mảng tín hiệu.Kết quả đầu ra chúng ta sẽ được là mảng tín hiệu sau khi biến đổi Fourier.

**Hàm fft\_magnitude(x):** Hàm này gọi hàm fft ở trên để chuyển đổi dữ liệu và cũng đồng thời tính luôn biên độ phổ bằng cách lấy giá trị tuyệt đối của các kết quả vừa được biển đổi Fourier. Kết quả sẽ được một mảng biên độ phổ.

**Hàm fftfreq\_custom(N, d=1.0) :** Tạo mảng tần số tướng ứng với FFT, sử dụng công thức.

* Phần đầu (0 đến N/2): i / (N \* d), là các tần số dương.
* Phần sau (N/2 đến N): (i - N) / (N \* d), là các tần số âm.

Tiến hành đọc file WAV và lấy dữ liệu cần thiết để tính toán phổ và hiển thị như:

* **n\_channels:** Số kênh (mono hoặc stereo).
* **sample\_width:** Độ rộng mẫu (bytes/mẫu).
* **sample\_rate:** Tần số lấy mẫu (số mẫu/giây).
* **n\_frames:** Số lượng mẫu tổng.
* **duration:** Thời lượng âm thanh (giây).
* **audio\_data:** Mảng dữ liệu tín hiệu âm thanh (ở dạng int16).

**magnitude = fft\_magnitude(audio\_data):** Tính toán biên độ phổ.

**frequencies = fftfreq\_custom(len(audio\_data), 1/sample\_rate):** Tính toán tần số tương ứng

**min\_len = min(len(frequencies), len(magnitude)) :** Do frequencies và magnitude có thể có độ dài khác nhau, lấy độ dài nhỏ nhất (min\_len) để cắt đồng nhất.

Sau khi đã thực hiện tính toán để lấy dữ liệu thì ta tiến hành dùng thư viện “matplotlib

” để thể hiện đồ họa của chúng.

# CHƯƠNG 3. CÀI ĐẶT CHƯƠNG TRÌNH.

## 3.1 MÃ NGUỒN CHƯƠNG TRÌNH

### 3.1.1 Mức 1

#### 3.1.1.1 Đọc thông tin header file WAV.

**Mã nguồn:**

A screen shot of a computer program

Description automatically generated

**Hình 7. Mã nguồn phần đọc thông tin header file WAV.**

**Kết quả**

A computer screen shot of a black screen

Description automatically generated

**Hình 8 Kết quả của đọc thông tin header của file WAV(1).**

A computer screen shot of a black screen

Description automatically generated

**Hình 9. Kết quả đọc thông tin header của file WAV (2.)**

#### 3.1.1.2 Đồ họa dạng sóng của file âm thanh.

**Mã nguồn:**

A screen shot of a computer program

Description automatically generated

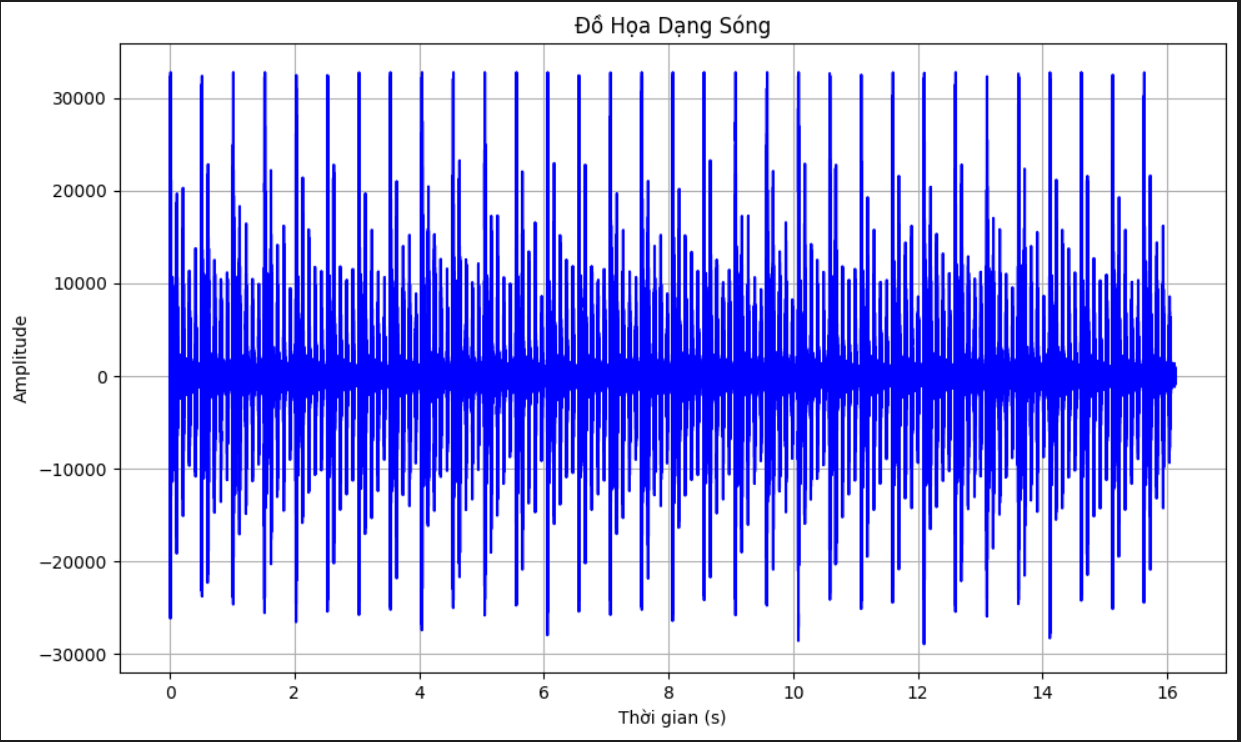
**Hình 10. Mã nguồn xử lý hiển thị đồ họa sóng**

**Kết quả:**

A graph with blue lines

Description automatically generated

**Hình 11. Kết quả khi chạy hiển thị đồ họa dạng sóng(1).**



**Hình 12. Kết quả khi chạy hiển thị đồ họa dạng sóng(2).**

### 3.1.1 Mức 2.

#### 3.1.2.1 Phân tích phổ, biểu đồ phổ và hiển thị.

**Mã nguồn:**

A screen shot of a computer screen

Description automatically generated

**Hình 13. Mã nguồn Phân tích phổ, biểu đồ phổ và hiển thị.**

**Kết quả:**

**A graph with red lines

Description automatically generated**

**Hình 14. Đồ họa thể hiện Phổ**

**A graph of orange lines

Description automatically generated with medium confidence**

**Hình 15. Đồ họa thể hiện Biểu đồ Phổ.**

# CHƯƠNG 4. KẾT LUẬN.

## 4.1 KẾT QUẢ ĐẠT ĐƯỢC

**Đọc và trích xuất thông tin từ file header WAV:** Thành công trong việc trích xuất thông tin quan trọng từ phần header của file âm thanh, bao gồm số kênh âm thanh (mono hoặc stereo), tỷ lệ mẫu (sample rate), độ rộng mẫu (sample width), và tổng số khung (frames)... Các thông tin này cung cấp một cái nhìn tổng quan về đặc điểm của file âm thanh, điều kiện cần thiết để tiếp tục xử lý và phân tích âm thanh.

**Vẽ đồ họa dạng sóng của tín hiệu âm thanh:** Đồ thị dạng sóng (waveform) đã được vẽ thành công, thể hiện sự thay đổi của biên độ theo thời gian. Đồ thị giúp dễ dàng nhận diện các biến đổi trong tín hiệu âm thanh, như các đỉnh, và vùng yên tĩnh, tạo cơ sở cho việc phân tích sâu hơn về đặc tính âm thanh.

**Phân tích phổ của âm thanh (Fourier Transform):** Đã triển khai và tính toán thành công FFT (Fast Fourier Transform) để chuyển đổi tín hiệu âm thanh từ miền thời gian sang miền tần số. Phổ tần số được vẽ rõ ràng, cho phép nhận diện các tần số chủ yếu trong tín hiệu âm thanh, từ đó có thể phân biệt các thành phần âm thanh (như âm trầm, âm bổng). Biểu đồ phổ giúp hiểu được sự phân bố tần số của tín hiệu âm thanh, phục vụ cho việc phân tích và xử lý các tín hiệu phức tạp hơn.

**Vẽ biểu đồ phổ (Spectrogram):** Đã thành công trong việc vẽ spectrogram, một biểu đồ thể hiện sự thay đổi tần số theo thời gian. Biểu đồ này cho phép chúng ta theo dõi sự biến đổi của các thành phần tần số trong suốt quá trình phát ra âm thanh, từ đó nhận diện được các sự kiện hoặc đặc điểm quan trọng của âm thanh.

## 4.2 ƯU ĐIỂM

**Trực quan hóa dữ liệu hiệu quả:** Các biểu đồ như đồ thị dạng sóng, phổ âm thanh (Spectrum), và biểu đồ phổ (Spectrogram) được hiển thị ngay trong notebook, giúp người dùng dễ dàng quan sát và phân tích tín hiệu.

**Linh hoạt và dễ thử nghiệm:** Nhờ môi trường tương tác của Jupyter Notebook, người dùng có thể thay đổi mã nguồn và xem kết quả ngay lập tức, giúp việc thử nghiệm và tinh chỉnh thuật toán trở nên đơn giản.

**Nền tảng cho bài toán phức tạp hơn:** Phân tích phổ và biểu đồ phổ trong bài toán này là bước cơ bản nhưng quan trọng, làm tiền đề để giải quyết các bài toán khó hơn, như nhận diện giọng nói hay phân loại tín hiệu âm thanh.

Ứng dụng các thư viện phổ biến:

**Chương trình tận dụng các thư viện mạnh mẽ :** Chương trình tận dụng các thư viện như numpy, matplotlib, và wave, giúp xử lý tín hiệu âm thanh và trực quan hóa dễ dàng, đồng thời tăng tính mở rộng cho các dự án khác.

**Khả năng chia sẻ và tái sử dụng:** Jupyter Notebook dễ dàng chia sẻ với người dùng khác qua file .ipynb, đồng thời có thể tái sử dụng cho các mục đích học thuật hoặc các dự án liên quan đến xử lý tín hiệu âm thanh.

## 4.3 NHƯỢC ĐIỂM

**Không có giao diện người dùng trực quan để tải file:** Chương trình hiện tại yêu cầu người dùng phải chỉnh sửa trực tiếp đường dẫn file âm thanh trong mã nguồn. Điều này có thể gây bất tiện cho người dùng không quen làm việc với mã code.

**Giới hạn trong việc xử lý định dạng file:** Chương trình chỉ hỗ trợ đọc và xử lý file âm thanh định dạng WAV, chưa hỗ trợ các định dạng phổ biến khác như MP3, FLAC hoặc AAC.

## 4.4 HƯỚNG PHÁT TRIỂN.

**Phát triển giao diện người dùng**: Xây dựng giao diện đồ họa (GUI) bằng các thư viện như Tkinter hoặc PyQt, giúp người dùng dễ dàng tải file âm thanh và xem kết quả trực quan mà không cần chỉnh sửa mã nguồn.

**Hỗ trợ nhiều định dạng file âm thanh**: Sử dụng thư viện như pydub hoặc librosa để mở rộng khả năng xử lý các định dạng file âm thanh khác như MP3, FLAC, AAC, v.v.

**Cải thiện hiệu suất:** Tối ưu hóa các thuật toán xử lý FFT và hiển thị phổ để giảm thời gian xử lý với các file âm thanh có kích thước lớn.

**Tích hợp phân tích chi tiết hơn:** Bổ sung các tính năng phân tích như lọc tần số, hiển thị phổ 3D, hoặc phân tích cường độ âm thanh trong từng khoảng thời gian.

**Ứng dụng thực tế:** Áp dụng chương trình để phân tích và so sánh đặc tính âm thanh trong các lĩnh vực như nhận diện giọng nói, xử lý nhạc, hoặc phát hiện âm thanh bất thường.

**Xây dựng công cụ trực tuyến:** Tích hợp chương trình thành một ứng dụng web, cho phép người dùng tải file âm thanh và phân tích trực tiếp thông qua trình duyệt mà không cần cài đặt phần mềm.

# TÀI LIỆU THAM KHẢO

1. Docs.FileFormat. (n.d.). *WAV*. Truy cập ngày 1 tháng 12 năm 2024, từ <https://docs.fileformat.com/audio/wav/#google_vignette>.
2. Nguyễn Anh Tú. (2019). *Xử lý dữ liệu âm thanh*. Viblo. Truy cập ngày 3 tháng 12 năm 2024, từ <https://viblo.asia/p/xu-ly-du-lieu-am-thanh-Qpmlezg95rd>.
3. Nguyễn Duy Cường. (2020). *Một số khái niệm cơ bản trong bài toán xử lý âm thanh sử dụng Deep Learning (Phần 1)*. Viblo. Truy cập ngày 5 tháng 12 năm 2024, từ <https://viblo.asia/p/mot-so-khai-niem-co-ban-trong-bai-toan-xu-ly-am-thanh-su-dung-deep-learning-phan-1-AZoJjrbAJY7>.
4. Nguyễn Đức Minh. (2021). *Một số kiến thức cơ bản về Text2Speech*. Viblo. Truy cập ngày 5 tháng 12 năm 2024, từ <https://viblo.asia/p/mot-so-kien-thuc-co-ban-ve-text2speech-63vKjWLNZ2R>.
5. All About Circuits. (n.d.). *An introduction to the discrete Fourier transform*. Truy cập ngày 5 tháng 12 năm 2024, từ <https://www.allaboutcircuits.com/technical-articles/an-introduction-to-the-discrete-fourier-transform/>.

.