|  |  |
| --- | --- |
|  |  |
| NGUYỄN HỒNG SƠN | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
|  |
| **ĐỒ ÁN TỐT NGHIỆP** |
| NGÀNH CÔNG NGHỆ THÔNG TIN |
| **XÂY DỰNG PHẦN MỀM PHÂN TÍCH ÂM THANH SA** |
|  |
|  |
| **CBHD:Ths. Lê Anh Thắng** |
| NGÀNH CÔNG NGHỆ THÔNG TIN | **Sinh viên: Nguyễn Hồng Sơn** |
| **Mã số sinh viên: 2019607343**  **Lớp: CNTT02** |
|  |
|  |
|  |
| Hà Nội – Năm 2024 |
|  |

MỤC LỤC

[LỜI CẢM ƠN iii](#_Toc167276574)

[DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT iv](#_Toc167276575)

[MỞ ĐẦU 1](#_Toc167276576)

[1. Lý do chọn đề tài 1](#_Toc167276577)

[2. Mục đích nghiên cứu 2](#_Toc167276578)

[3. Đối tượng và phạm vi nghiên cứu 2](#_Toc167276579)

[4. Ý nghĩa khoa học và thực tiễn của đề tài 2](#_Toc167276580)

[CHƯƠNG 1: GIỚI THIỆU VỀ NGÔN NGỮ,CÁC ĐẶC TRƯNG CƠ BẢN VÀ PHÉP BIẾN ĐỔI TRONG ÂM THANH 4](#_Toc167276581)

[1.1 Giới thiệu về ngôn ngữ sử dụng 4](#_Toc167276582)

[1.1.1. Giới thiệu Python 4](#_Toc167276583)

[1.1.2. Lịch sử phát triển Python 5](#_Toc167276584)

[1.1.3. Ưu điểm của Python 5](#_Toc167276585)

[1.1.4. Nhược điểm của Python 6](#_Toc167276586)

[1.2. Các đặc trưng cơ bản của âm thanh 6](#_Toc167276587)

[1.2.1. Tần số(Frequence) 6](#_Toc167276588)

[1.2.2. Biên độ(Amplitude) 6](#_Toc167276589)

[1.2.3. Hình dạng biên độ(Amplitude Envelope) 7](#_Toc167276590)

[1.2.4. Độ kéo dài(Duration) 7](#_Toc167276591)

[1.2.5. Biên độ tín hiệu(Signal-to-Noise Ratio-SNR) 7](#_Toc167276592)

[1.2.6. Tần số cực đại 7](#_Toc167276593)

[1.2.7. Cường độ âm thanh 7](#_Toc167276594)

[1.2.8. Số lượng các thành phần tần số(Spectral Components) 7](#_Toc167276595)

[1.3. Phép biến đổi trong âm thanh 8](#_Toc167276596)

[1.3.1. Phép biến đổi Fourier 8](#_Toc167276597)

[1.3.2. Tại sao cần biến đổi Fourier 10](#_Toc167276598)

[1.3.3. Miền thời gian và tần số 10](#_Toc167276599)

[1.3.4. Biến đổi Fourier rời rạc(DFT) 12](#_Toc167276600)

[CHƯƠNG 2: XÂY DỰNG ỨNG DỤNG 21](#_Toc167276601)

[2.1. Các hàm chính 21](#_Toc167276602)

[2.1.1. Hàm ghi âm 21](#_Toc167276603)

[2.1.2. Hàm tính THD+N 23](#_Toc167276604)

[2.1.3. USB HID 28](#_Toc167276605)

[2.1.4. Phân tích FFT 31](#_Toc167276606)

[CHƯƠNG 3: CÀI ĐẶT VÀ TRIỂN KHAI 37](#_Toc167276607)

[3.1. Giới thiệu về công cụ và môi trường cài đặt 37](#_Toc167276608)

[3.1.1. Visual Studio Code 37](#_Toc167276609)

[3.1.2. Những tính năng vượt trội của Visual Studio Code 38](#_Toc167276610)

[3.1.3. Các bước cài đặt Visual Studio Code 38](#_Toc167276611)

[3.2. Giao diện phần mềm 38](#_Toc167276612)

[KẾT LUẬN 47](#_Toc167276613)

[TÀI LIỆU THAM KHẢO 53](#_Toc167276614)

**LỜI CẢM ƠN**

Trước tiên cho phép em được bày tỏ lòng biết ơn đến Ths. Lê Anh Thắng cùng các thầy cô trong khoa đã tạo điều kiện hỗ trợ, giúp đỡ em trong suốt quá trình học tập và nghiên cứu đề tài này. Trong suốt thời gian từ khi bắt đầu học tập tại trường đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý thầy cô và bạn bè.

Với lòng biết ơn sâu sắc nhất, em xin gửi đến quý thầy cô ở khoa công nghệ thông tin - người đã truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Nhờ có những lời hướng dẫn, dạy bảo của các thầy cô nên đề tài nghiên cứu của em mới có thể hoàn thiện tốt đẹp.

Đồ án được thực hiện trong khoảng thời gian 9 tuần. Bước đầu đi vào thực tế của em còn hạn chế và còn nhiều bỡ ngỡ nên không tránh khỏi những thiếu sót, em rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô để kiến thức của em trong lĩnh vực này được hoàn thiện hơn đồng thời có điều kiện bổ sung, nâng cao trình độ cũng như lối tư duy của bản thân.

# DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **STT** | **Từ viết tắt** | **Ý nghĩa** |
| 1 | FFT | Biến đổi Fourier nhanh(Fast Fourier Transform) |
| 2 | DFT | Biến đổi Fourier rời rạc(Discrete Fourier Transform) |
| 3 | THD | Tổng méo hài(Total Harmonic Distortion) |

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Âm thanh, một trong những yếu tố vô cùng quan trọng trong cuộc sống hàng ngày của chúng ta, không chỉ là một phương tiện truyền đạt thông tin mà còn là một phần không thể tách rời của trải nghiệm của con người. Từ tiếng hát trong buổi sáng đầu tiên của ngày mới đến những tiếng cười và cuộc trò chuyện vui vẻ cùng bạn bè, từ âm nhạc yêu thích trên đài phát thanh đến tiếng gió rì rào giữa cây cỏ, âm thanh bao quanh chúng ta, tạo nên một phần không gian sống đầy màu sắc và đa dạng.

Trong một thời đại kỹ thuật số như ngày nay, khả năng phân tích và hiểu biết về âm thanh không chỉ mở ra cánh cửa cho nhiều ứng dụng thú vị mà còn đem lại nhiều tiềm năng trong nhiều lĩnh vực khác nhau. Chính vì lẽ đó, việc chọn đề tài xây dựng ứng dụng phân tích âm thanh trở thành một lựa chọn hấp dẫn, đầy tiềm năng và ý nghĩa.

Cùng với sự phát triển của cuộc cách mạng khoa học và công nghệ đang diễn ra một cách sôi động, chúng ta đang tiến dần tới thế giới của sự số hóa. Với các ưu điểm của xử lý số, nhanh gọn, chính xác với chất lượng cao, mọi lĩnh vực thông tin liên lạc, phát thanh truyền hình… đều tiến tới việc áp dụng một cách đồng bộ và có hiệu quả các công cụ cũng như các phép xử lý số. Trong đó, âm thanh là một lĩnh vực đặc biệt quan trọng, đây là một phương thức dùng để trao đổi cũng như cảm nhận tin. Do đó lĩnh vực về âm thanh là một vấn đề rất có ý nghĩa trong tương lai.

Ngành sản xuất các thiết bị âm thanh như loa, tai nghe không dây, tai nghe có dây, micro…đang rất phát triển nhu cầu của xã hội rất cao. Vì vậy yêu cầu cần có hệ thống kiểm tra chất lượng của các thiết bị âm thanh tại các nhà máy sản xuất công nghiệp là quan trọng và cần thiết.

Vì vậy, em đã thực hiện đề tài “**XÂY DỰNG PHẦN MỀM PHÂN TÍCH ÂM THANH SA**”. Với phần mềm này, người sử dụng có thể đánh giá chất lượng của mic và loa của tai nghe bao gồm tai nghe không dây, có dây, hoặc loa nghe nhạc.

## 2. Mục đích nghiên cứu

Mục đích nghiên cứu của đề tài này là đánh giá chất lượng của mic và loa của tai nghe bao gồm tai nghe không dây, có dây hoặc loa nghe nhạc. Hoặc để đánh giá chất lượng của một bộ âm li hát karaoke.

Trong ngành công nghiệp sản xuất loa tai nghe, thì việc đánh giá chất lượng của chúng là quan trọng nếu không sẽ cho ra thị trường những chiếc loa, chiếc tai nghe có chất lượng không tốt.

## 3. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Xây dựng một phần mềm phân tích âm thanh có thể áp dụng vào trong 1 công ty.

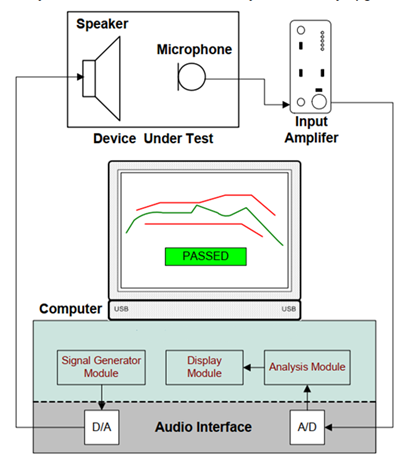
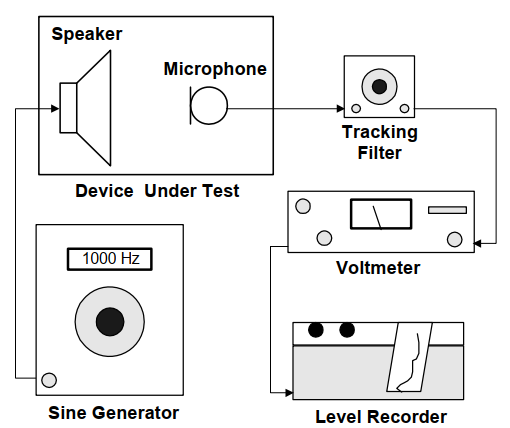
Phạm vi nghiên cứu:  
- Tìm hiểu kỹ thuật lập trình, cách thức hoạt động và các đối tượng trong Python.

- Hiểu được tính chất vật lí của âm thanh cũng như phép biến đổi âm thanh.

- Tham khảo các ứng dụng thương mại và tiêu chuẩn ngành để đảm bảo tính tin cậy và hiệu quả của ứng dụng.

## 4. Ý nghĩa khoa học và thực tiễn của đề tài

Phần mềm phân tích âm thanh giúp cho việc kiểm tra chất lượng âm thanh trở nên thuận tiện hơn từ đó đánh giá chất lượng của thiết bị âm thanh một cách chính xác và khoa học. Trên cơ sở giúp người sử dụng phần mềm kiểm tra chất lượng của một tai nghe không dây, có dây hoặc loa nghe nhạc từ đó tạo ra những sản phẩm với chất lượng tốt nhất.

Sound Analysis hoạt động trên các nguyên tắc giống như hệ thống đo lường độc lập, truyền thống bao gồm Bộ tạo tín hiệu, Đồng hồ vạn năng RMS, bộ lọc theo dõi và máy ghi mức. Với Sound Analysis , tất cả trong số các chức năng này được triển khai trong 1 phần mềm duy nhất giúp tiết kiệm chi phí đầu tư và triển khai 1 cách nhanh chóng

*Hệ đo lường âm thanh truyền thống*

*Hệ đo lường âm thanh sử dụng phần mềm Sound Analysis*

**CHƯƠNG 1: GIỚI THIỆU VỀ NGÔN NGỮ,CÁC ĐẶC TRƯNG CƠ BẢN VÀ PHÉP BIẾN ĐỔI TRONG ÂM THANH**

**1.1 Giới thiệu về ngôn ngữ sử dụng**

***1.1.1. Giới thiệu Python***

Python là một ngôn ngữ lập trình được sử dụng rộng rãi trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML). Các nhà phát triển sử dụng Python vì nó hiệu quả, dễ học và có thể chạy trên nhiều nền tảng khác nhau. Phần mềm Python được tải xuống miễn phí, tích hợp tốt với tất cả các loại hệ thống và tăng tốc độ phát triển.

Các nhà phát triển có thể dễ dàng đọc và hiểu một chương trình Python vì ngôn ngữ này có cú pháp cơ bản giống với tiếng Anh.

Python giúp cải thiện năng suất làm việc của các nhà phát triển vì so với những ngôn ngữ khác, họ có thể sử dụng ít dòng mã hơn để viết một chương trình Python.

Python có một thư viện tiêu chuẩn lớn, chứa nhiều dòng mã có thể tái sử dụng cho hầu hết mọi tác vụ. Nhờ đó, các nhà phát triển sẽ không cần phải viết mã từ đầu.

Cộng đồng Python tích cực hoạt động bao gồm hàng triệu nhà phát triển nhiệt tình hỗ trợ trên toàn thế giới. Nếu gặp phải vấn đề, bạn sẽ có thể nhận được sự hỗ trợ nhanh chóng từ cộng đồng.

Các nhà phát triển có thể dễ dàng sử dụng Python với các ngôn ngữ lập trình phổ biến khác như Java, C và C++.

Python có thể được sử dụng trên nhiều hệ điều hành máy tính khác nhau, chẳng hạn như Windows, macOS, Linux và Unix.

Trên Internet có rất nhiều tài nguyên hữu ích nếu bạn muốn học Python. Ví dụ: bạn có thể dễ dàng tìm thấy video, chỉ dẫn, tài liệu và hướng dẫn dành cho nhà phát triển.

***1.1.2.******Lịch sử phát triển Python***

Guido Van Rossum cho ra mắt phiên bản đầu tiên của ngôn ngữ Python (phiên bản 0.9.0) vào năm 1991. Ngôn ngữ này đã bao gồm các tính năng hữu ích như một số kiểu dữ liệu và hàm để xử lý lỗi.

**Python 1.0:** Đã được ra mắt vào năm 1994 với các hàm mới để dễ dàng xử lý danh sách dữ liệu, chẳng hạn như ánh xạ, lọc và lược bỏ.

**Python 2.0**: Đã được ra mắt vào ngày 16 tháng 10 năm 2000, với các tính năng hữu ích mới cho lập trình viên, chẳng hạn như hỗ trợ ký tự Unicode và cách xử lý chi tiết một danh sách nhanh chóng hơn.

**Python 3.0**: Được ra mắt vào ngày mùng 3 tháng 12 năm 2008. Đây là một phiên bản lớn của Python không tương thích ngược hoàn toàn.Nhiều tính năng lớn của nó đã được chuyển mã ngược (backport) về loạt phiên bản Python 2.6.x và 2.7.x. Các bản phát hành của Python 3 có đi kèm với công cụ 2to3, có tác dụng tự động hoá việc dịch mã Python 2 sang Python 3.

**Python 3.9.2 và 3.8.8**: Được xúc tiến vì tất cả các phiên bản trước của Python (bao gồm cả 2.7 gặp một số vấn đề bảo mật, có thể dẫn đến thực thị mã từ xa và "đầu độc" bộ nhớ đệm.

Trong năm 2022, **Python 3.10.4 và 3.9.12** được xúc tiếncùng với 3.8.13 và 3.7.13, nguyên nhân là do một vài vấn đề về bảo mật. Khi Python 3.9.13 được phát hành vào tháng Năm năm 2022, loạt phiên bản 3.9 (cùng với loạt 3.8 và 3.7) được thông báo rằng sẽ chỉ nhận được các bản vá bảo mật trong tương lai. Vào ngày 7 tháng Chín năm 2022, bốn bản cập nhật mới được phát hành do có khả năng xảy ra một cuộc tấn công từ chối dịch vụ: 3.10.7, 3.9.14, 3.8.14 và 3.7.14.

***1.1.3. Ưu điểm của Python***

* Python là một ngôn ngữ thông dịch, điều này nghĩa là ngôn ngữ này trực tiếp chạy từng dòng mã. Nếu có lỗi trong mã chương trình, nó sẽ ngừng chạy. Do đó, lập trình viên có thể nhanh chóng tìm ra lỗi trong đoạn mã.
* Python sử dụng từ ngữ giống trong tiếng Anh. Không giống như các ngôn ngữ lập trình khác, Python không sử dụng dấu ngoặc ôm. Thay vào đó, ngôn ngữ này sử dụng thụt đầu dòng.
* Các lập trình viên không cần phải khai báo loại biến khi viết mã bởi vì Python sẽ xác định chúng vào thời điểm chạy. Vì vậy, bạn có thể viết các chương trình Python một cách nhanh chóng hơn.

- Python gần gũi với ngôn ngữ con người hơn các ngôn ngữ lập trình khác. Do đó, các lập trình viên không cần phải lo lắng về những chức năng cơ bản của nó như kiến trúc và quản lý bộ nhớ.

- Python coi mọi thứ đều là đối tượng, nhưng ngôn ngữ này cũng hỗ trợ các phương thức lập trình khác như lập trình hàm và lập trình cấu trúc.

### *1.1.4. Nhược điểm của Python*

* Python không có các thuộc tính như: protected, private hay public, không có vòng lặp do…while và switch….case.
* Python mặc dù nhanh hơn so với PHP, nhưng lại không nhanh hơn so với C++, Java.

**1.2. Các đặc trưng cơ bản của âm thanh**

Đặc trưng của âm thanh là các thuộc tính đặc biệt của tín hiệu âm thanh mà chúng ta có thể sử dụng để phân tích và hiểu biết về âm thanh. Dưới đây là một số đặc trưng cơ bản của âm thanh mà thường được sử dụng trong các ứng dụng phân tích và xử lý âm thanh

***1.2.1. Tần số(Frequence)***

Tần số âm thanh là mức độ âm thanh được phát ra ngoài con người nghe được. Dải tần số âm thanh đại lượng đặc trưng theo từng rung động tuần hoàn có tần số, được tính theo đơn vị đo tần số âm thanh là Hz. Khi tần số dao động càng lớn âm thanh phát ra càng bổng và tần số nhỏ âm thanh trầm.

***1.2.2. Biên độ(Amplitude)***

Biên độ đo lường độ lớn của âm thanh, tức là mức độ dao động của âm thanh. Nó ảnh hưởng đến âm lượng của âm thanh.

***1.2.3. Hình dạng biên độ(Amplitude Envelope)***

Hình dạng của biên độ biểu thị sự thay đổi của biên độ qua thời gian. Nó có thể phản ánh các thuộc tính như sự tăng giảm, đỉnh và đáy của âm thanh.

***1.2.4. Độ kéo dài(Duration)***

Độ kéo dài của âm thanh là thời gian mà âm thanh tồn tại trong một khoảng thời gian nhất định. Nó thường được đo bằng giây hoặc mili-giây.

***1.2.5. Biên độ tín hiệu(Signal-to-Noise Ratio-SNR)***

SNR là tỉ lệ giữa năng lượng của tín hiệu và năng lượng của nhiễu trong tín hiệu âm thanh. Nó thường được sử dụng để mô tả mức độ nhiễu trong tín hiệu so với tín hiệu chính.

***1.2.6. Tần số cực đại***

Tần số mà năng lượng của tín hiệu âm thanh đạt đến mức cực đại. Điều này thường liên quan đến các đặc trưng như âm giọng hoặc âm nhạc chính.

***1.2.7. Cường độ âm thanh***

SPL đo lường cường độ của âm thanh, thường được đo bằng đơn vị dB (decibel). Nó là một chỉ số quan trọng để đánh giá mức độ ồn trong môi trường.

***1.2.8. Số lượng các thành phần tần số(Spectral Components)***

Số lượng các thành phần tần số hoặc băng tần khác nhau trong một tín hiệu âm thanh. Điều này liên quan đến phân bố năng lượng của âm thanh theo các tần số khác nhau.

Các đặc trưng này cung cấp thông tin quan trọng để phân tích và hiểu biết về tín hiệu âm thanh trong các ứng dụng như xử lý ngôn ngữ tự nhiên, nhận dạng giọng nói, và xử lý âm thanh số.

**1.3. Phép biến đổi trong âm thanh**

***1.3.1. Phép biến đổi Fourier***

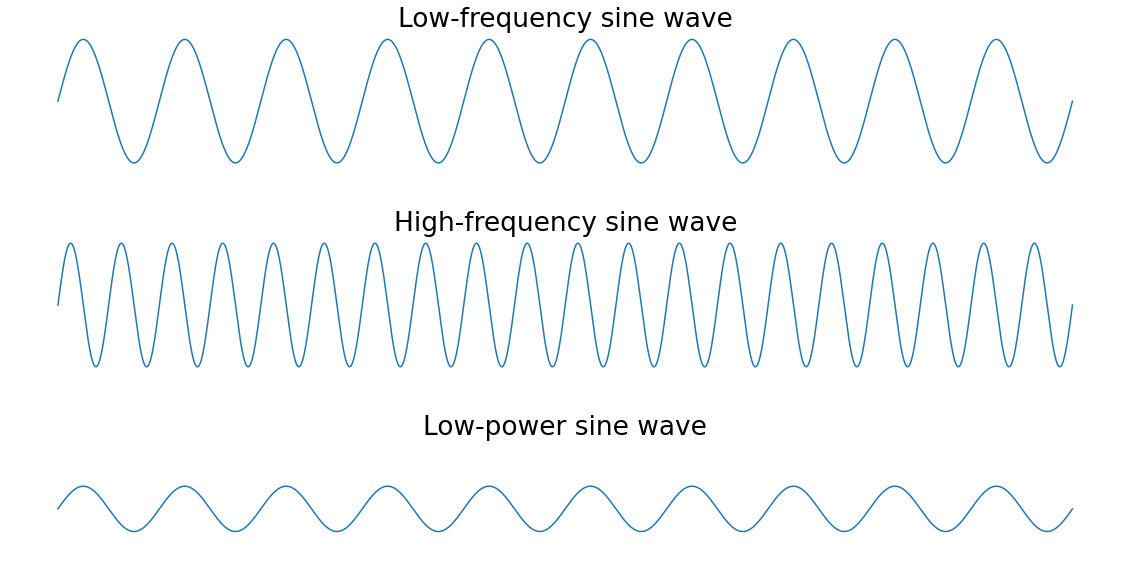
Phân tích Fourier là một lĩnh vực nghiên cứu cách một **hàm toán học** có thể được phân tách thành một chuỗi **các hàm lượng giác** đơn giản hơn . Biến đổi Fourier là một công cụ từ trường này để phân tách hàm thành các tần số thành phần của nó.

Có thể hiểu đơn giản biến đổi Fourier là một công cụ cho phép bạn lấy tín hiệu và xem sức mạnh của từng tần số trong đó. Sử dụng trong nhiều lĩnh vực khác nhau, từ toán học thuần túy đến kỹ thuật âm thanh và thậm chí cả tài chính.

Hãy xem các thuật ngữ quan trọng gồm:

* Tín hiệu là thông tin thay đổi theo thời gian. Ví dụ: dấu vết âm thanh, video và điện áp đều là ví dụ về tín hiệu.
* Tần số là tốc độ lặp lại của một cái gì đó. Ví dụ: đồng hồ tích tắc ở tần số một hertz (Hz) hoặc một lần lặp lại mỗi giây.
* Công suất , trong trường hợp này, chỉ có nghĩa là cường độ của từng tần số.

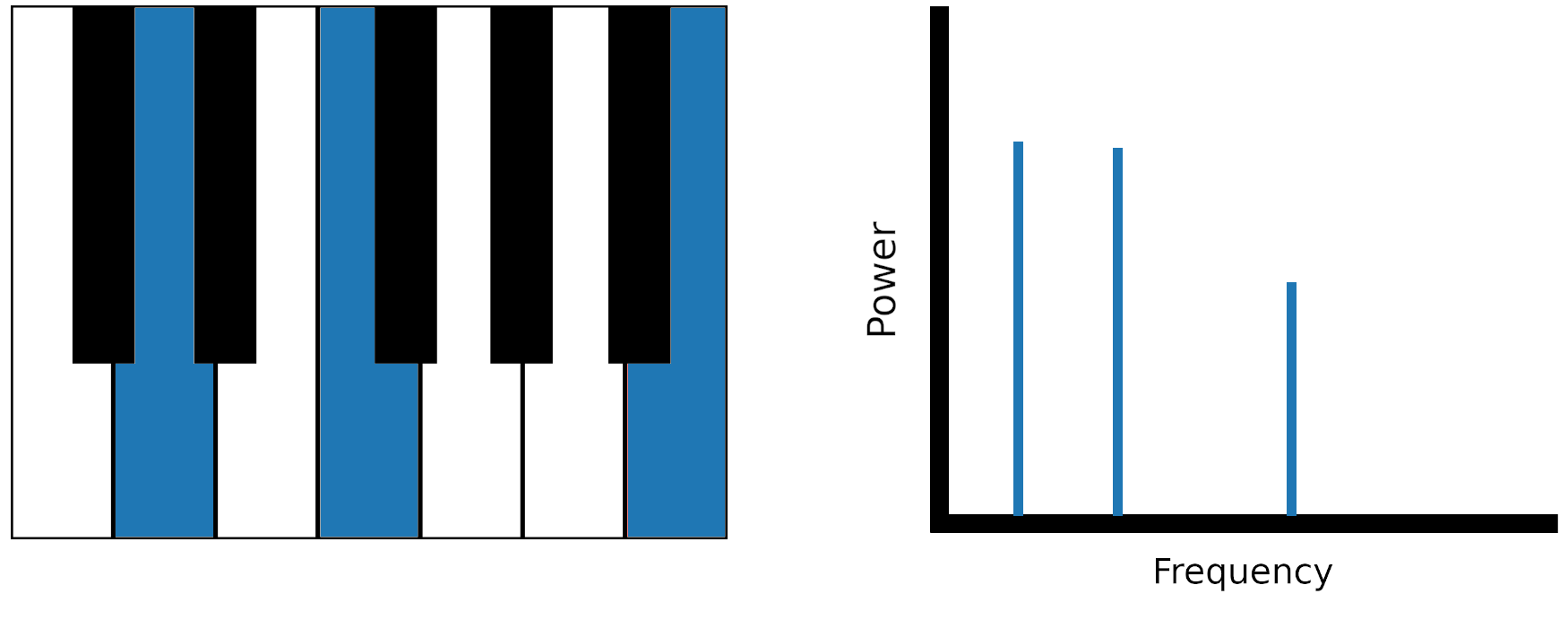
Hình ảnh sau đây là minh họa trực quan về tần số và công suất trên một số sóng hình sin :



Các đỉnh của sóng hình sin tần số cao gần nhau hơn các đỉnh của sóng hình sin tần số thấp vì chúng lặp lại thường xuyên hơn. Sóng hình sin công suất thấp có đỉnh nhỏ hơn hai sóng hình sin còn lại.

Để làm cho điều này cụ thể hơn, hãy tưởng tượng bạn đã sử dụng phép biến đổi Fourier trên bản ghi âm của một người nào đó chơi ba nốt trên đàn piano cùng một lúc. Phổ tần số thu được sẽ hiển thị ba đỉnh, một đỉnh cho mỗi nốt. Nếu người đó chơi một nốt nhạc nhẹ nhàng hơn những nốt còn lại thì cường độ tần số của nốt đó sẽ thấp hơn hai nốt còn lại.

Đây là hình ảnh trực quan của ví dụ về đàn piano đó:



Nốt cao nhất trên đàn piano được chơi êm hơn hai nốt còn lại, do đó phổ tần số thu được của nốt đó có đỉnh thấp hơn.

***1.3.2. Tại sao cần biến đổi Fourier***

Biến đổi Fourier rất hữu ích trong nhiều ứng dụng. Ví dụ: Shazam và các dịch vụ nhận dạng âm nhạc khác sử dụng phép biến đổi Fourier để xác định bài hát. Nén JPEG sử dụng một biến thể của biến đổi Fourier để loại bỏ các thành phần tần số cao của hình ảnh. Nhận dạng giọng nói sử dụng phép biến đổi Fourier và các phép biến đổi liên quan để khôi phục các từ được nói từ âm thanh thô.

Nói chung, bạn cần biến đổi Fourier nếu bạn cần xem tần số trong tín hiệu. Nếu việc xử lý tín hiệu trong miền thời gian gặp khó khăn thì việc sử dụng biến đổi Fourier để di chuyển tín hiệu đó sang miền tần số là điều đáng thử. Trong phần tiếp theo, bạn sẽ xem xét sự khác biệt giữa miền thời gian và miền tần số.

***1.3.3. Miền thời gian và tần số***

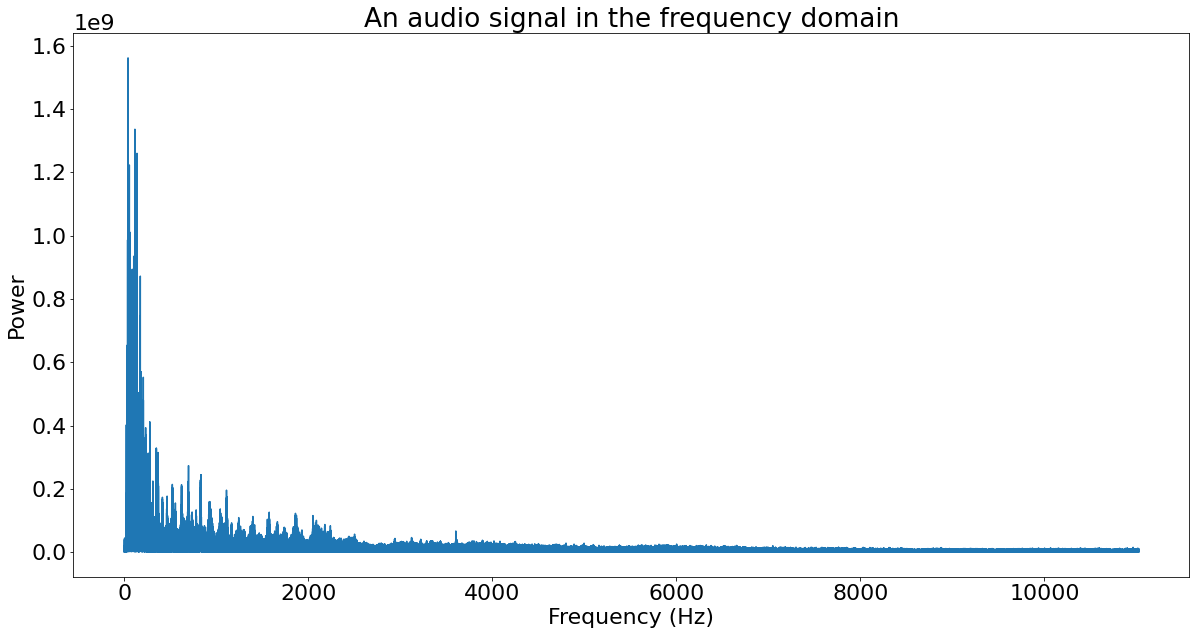
Phần tiếp theo là các thuật ngữ **miền thời gian** và **miền tần số**. Hai thuật ngữ này đề cập đến hai cách khác nhau để xem tín hiệu, dưới dạng tần số thành phần của nó hoặc dưới dạng thông tin thay đổi theo thời gian.

Trong miền thời gian, tín hiệu là sóng có biên độ thay đổi (trục y) theo thời gian (trục x). Biểu đồ trong miền thời gian:



Đây là hình ảnh của một số âm thanh, là tín hiệu **miền thời gian** . Trục hoành biểu thị thời gian và trục tung biểu thị biên độ.

Trong miền tần số, tín hiệu được biểu diễn dưới dạng một chuỗi tần số (trục x) mà mỗi tần số có công suất liên quan (trục y). Hình ảnh sau đây là tín hiệu âm thanh trên sau khi được biến đổi Fourier:



Ở đây, tín hiệu âm thanh trước đó được biểu thị bằng các tần số cấu thành của nó. Mỗi tần số ở phía dưới có một công suất liên quan, tạo ra quang phổ mà bạn nhìn thấy.

***1.3.4. Biến đổi Fourier rời rạc(DFT)***

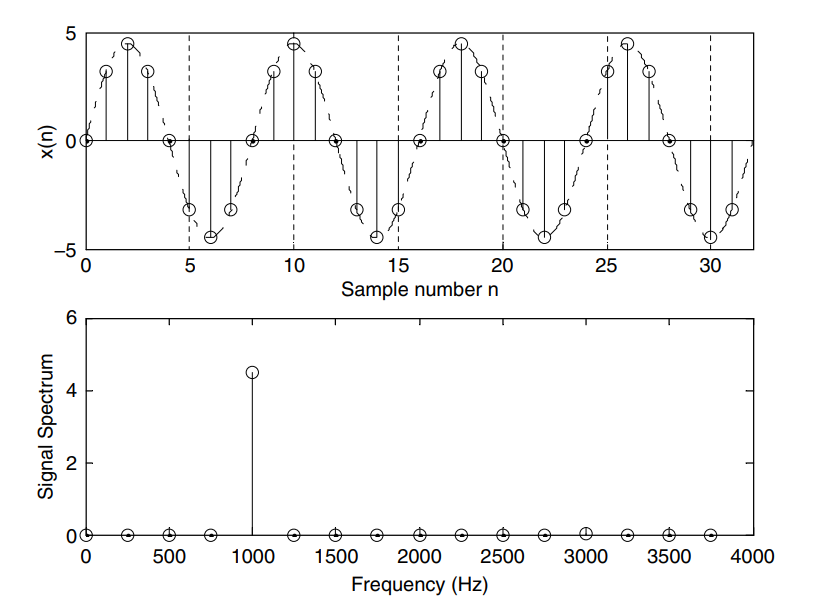
Biến đổi Fourier có thể được chia thành các loại biến đổi khác nhau. Việc phân chia cơ bản nhất dựa trên loại dữ liệu mà phép biến đổi hoạt động: hàm liên tục hoặc hàm rời rạc. Phần này sẽ chỉ đề cập đến **biến đổi Fourier rời rạc (DFT).**Bạn sẽ thường thấy các thuật ngữ DFT và FFT được sử dụng thay thế cho nhau, ngay cả trong hướng dẫn này. Tuy nhiên, chúng không hoàn toàn giống nhau. Biến **đổi Fourier nhanh (FFT)** là một thuật toán để tính toán biến đổi Fourier rời rạc (DFT), trong khi DFT chính là biến đổi.

Một điểm khác biệt khác mà bạn sẽ thấy trong scipy.fft thư viện là giữa các loại đầu vào khác nhau. FFT()chấp nhận đầu vào có giá trị phức tạp và RFFT()chấp nhận đầu vào có giá trị thực. Chuyển tới phần Sử dụng Biến đổi Fourier nhanh (FFT) để biết giải thích về số phức và số thực.

Hai phép biến đổi khác có liên quan chặt chẽ với DFT: **biến đổi cosin rời rạc (DCT)**và**biến đổi sin rời rạc (DST)**. Bạn sẽ tìm hiểu về những điều đó trong phần Biến đổi sin và cos rời rạc.

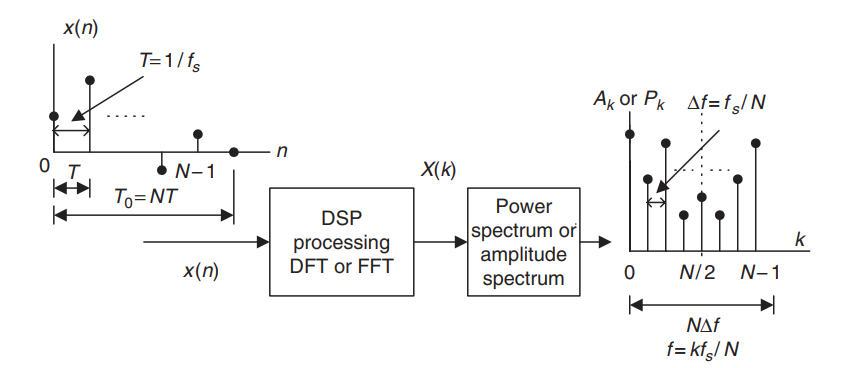
Thuật toán chuyển đổi các mẫu tín hiệu miền thời gian sang các thành phần miền tần số được gọi là biến đổi Fourier rời rạc hay DFT. DFT cũng thiết lập mối quan hệ giữa biểu diễn miền thời gian và biểu diễn miền tần số. Do đó, chúng ta có thể áp dụng DFT để thực hiện phân tích tần số của chuỗi miền thời gian. Ngoài ra, DFT còn được sử dụng rộng rãi trong nhiều lĩnh vực khác, bao gồm phân tích quang phổ, âm học, hình ảnh/video, âm thanh, thiết bị đo đạc và hệ thống truyền thông.

DFT ( Discrete Fourier Transform):

Trong miền thời gian, biểu diễn tín hiệu số mô tả biên độ tín hiệu so với thời điểm lấy mẫu hoặc số mẫu. Tuy nhiên, trong một số ứng dụng, nội dung tần số tín hiệu rất hữu ích hơn là các mẫu tín hiệu số. Việc biểu diễn tín hiệu số theo thành phần tần số của nó trong miền tần số, tức là phổ tín hiệu, cần được phát triển. Ví dụ, Hình 4.1 minh họa biểu diễn miền thời gian của hình sin 1.000 Hz với 32 mẫu ở tốc độ lấy mẫu 8.000 Hz; các biểu đồ phía dưới hiển thị phổ tín hiệu (miền tần số biểu diễn), trong đó chúng ta có thể quan sát rõ ràng rằng đỉnh biên độ nằm ở tần số 1.000 Hz trong phổ tính toán. Do đó, đồ thị phổ hiển thị tốt hơn thông tin tần số của tín hiệu số.

Dưới đây là hình ảnh phổ biên độ của tín hiệu số:

Phổ biên độ:

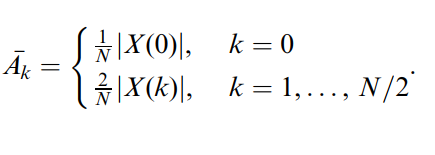
Một trong những ứng dụng của DFT là chuyển đổi tín hiệu số có độ dài hữu hạn x(n) thành phổ trong miền tần số. Hình 4.7 minh họa một ứng dụng như vậy, trong đó Ak và Pk lần lượt là phổ biên độ được tính toán và phổ công suất, sử dụng hệ số DFT X(k). Đầu tiên, chúng ta đạt được chuỗi kỹ thuật số x(n) bằng cách lấy mẫu tín hiệu tương tự x(t) và cắt tín hiệu được lấy mẫu bằng cửa sổ dữ liệu có độ dài To= NT, trong đó T là khoảng thời gian lấy mẫu và N là số lượng điểm dữ liệu. Thời gian cho cửa sổ dữ liệu là. To=NT.

Đối với chuỗi bị cắt ngắn x(n) có phạm vi n=0, 1, 2, …., N-1. Ta nhân được x(0), x(1), x(2), …., x(N-1)

Tiếp theo, áp dụng DFT cho chuỗi thu được x(n) để có được hệ số N DFT X(k) = trong đó k=0, 1, 2, 3, …., N-1.

Vì mỗi hệ số DFT được tính toán là một số phức nên việc vẽ đồ thị nó theo chỉ số tần số của nó là không thuận tiện. Do đó, sau khi đánh giá phương trình (4.18), biên độ và pha của từng hệ số DFT (chúng tôi gọi chúng lần lượt là phổ biên độ và phổ pha) có thể được xác định và vẽ đồ thị theo chỉ số tần số của nó. Chúng tôi định nghĩa phổ biên độ là

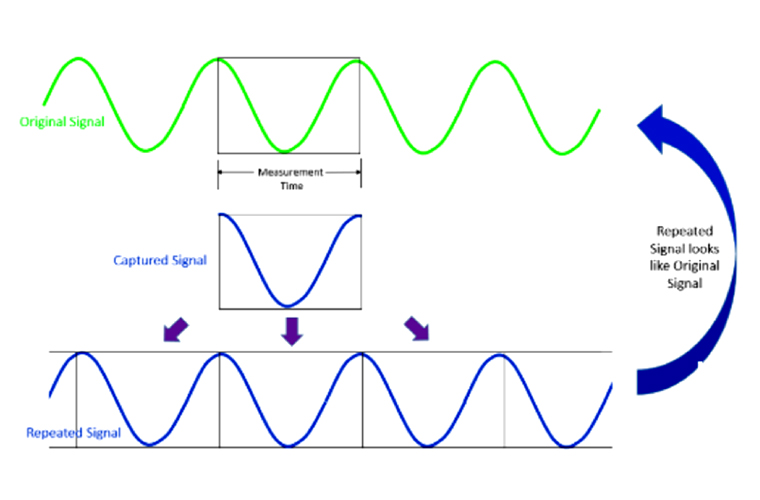
= tron đó k=0, 1, 2, 3,…., N-1

Chúng ta có thể sửa đổi phổ biên độ thành phổ biên độ một phía bằng cách nhân đôi biên độ trong phương trình (4.19), giữ số hạng DC ban đầu ở k = 0. Do đó, chúng ta có,

Tần số của tín hiệu thành phần được tính theo công thức. f= trong đó là tần số lấy mẫu.

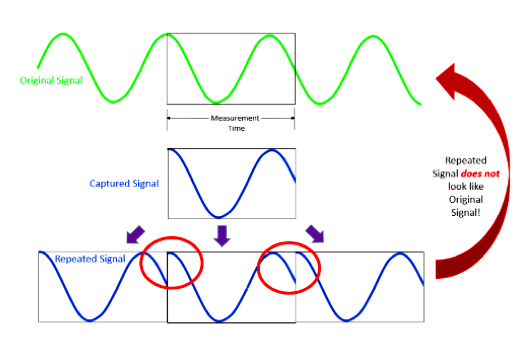
Cửa sổ dữ liệu window:

Biến đổi Fourier nhanh(FFT) có thể cực kỳ hữu ích trong việc xác định các tần số chính của tín hiệu. Nếu tín hiệu là định kỳ và dữ liệu được lấy mẫu trong một khoảng thời gian hoàn chỉnh hoặc nhiều khoảng thời gian hoàn chỉnh như trong Hình 1 thì thuật toán FFT sẽ cung cấp kết quả rõ ràng.



*Hình 1: Tín hiệu thu được có tính tuần hoàn và tín hiệu được tạo lại khớp với tín hiệu gốc.*

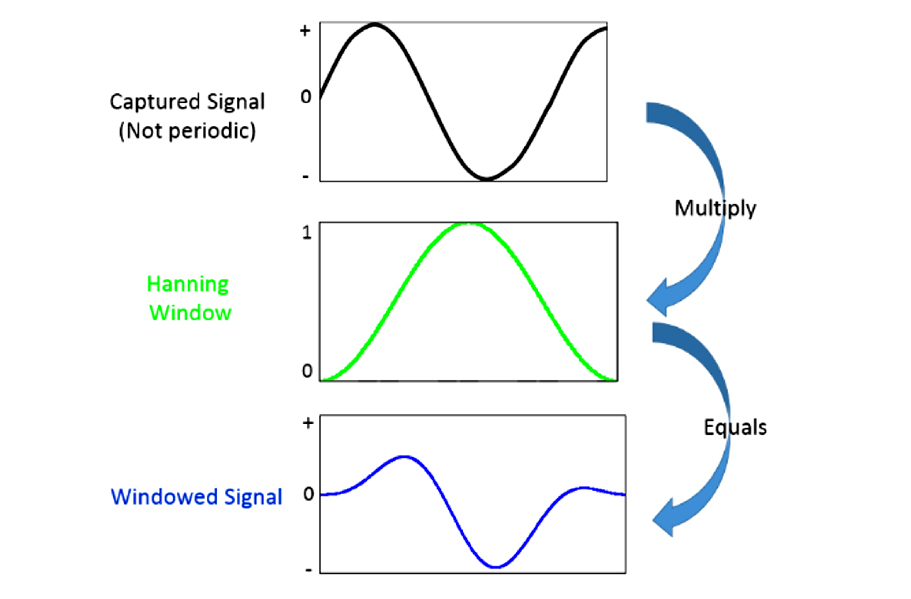
Trong thực tế, điều này khó có thể xảy ra nên tín hiệu tái tạo sẽ có những điểm gián đoạn:



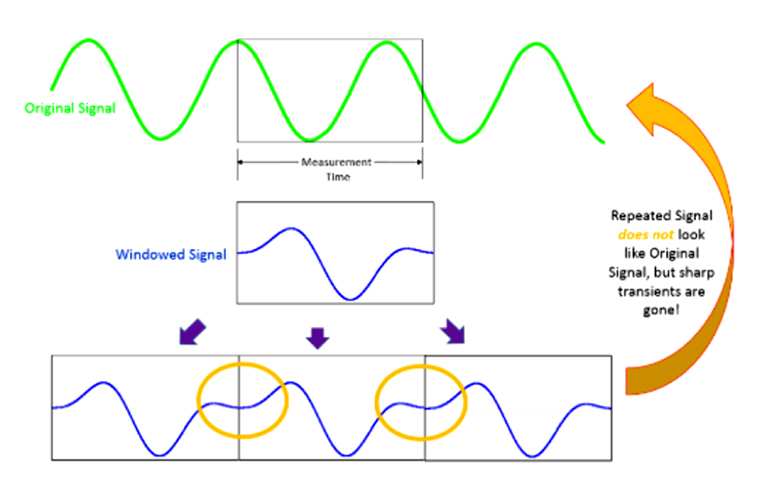
*Hình 2: Tín hiệu thu được không tuần hoàn, gây ra sự gián đoạn trong tín hiệu được tạo lại.*

Sự gián đoạn gây ra hiện tượng được gọi là rò rỉ hoặc “làm nhòe” kết quả, khiến cho có vẻ như năng lượng tử một dải tần số đang rò rỉ sang các giải tần số khác. Rò rỉ có thể hiển thị dưới dạng các thành phần tần số cao không thực sự có trong tín hiệu ban đầu.

Để giảm mức độ nhất thời sắc nét trong tín hiệu được tạo lại càng nhiều càng tốt, hãy áp dụng hàm toán học được gọi là cửa sổ cho dữ liệu. Các hàm cửa sổ thường bắt đầu và kết thúc ở số 0, do đó loại bỏ được sự gián đoạn. Nhân tín hiệu với cửa sổ để có được tín hiệu không bị gián đoạn.



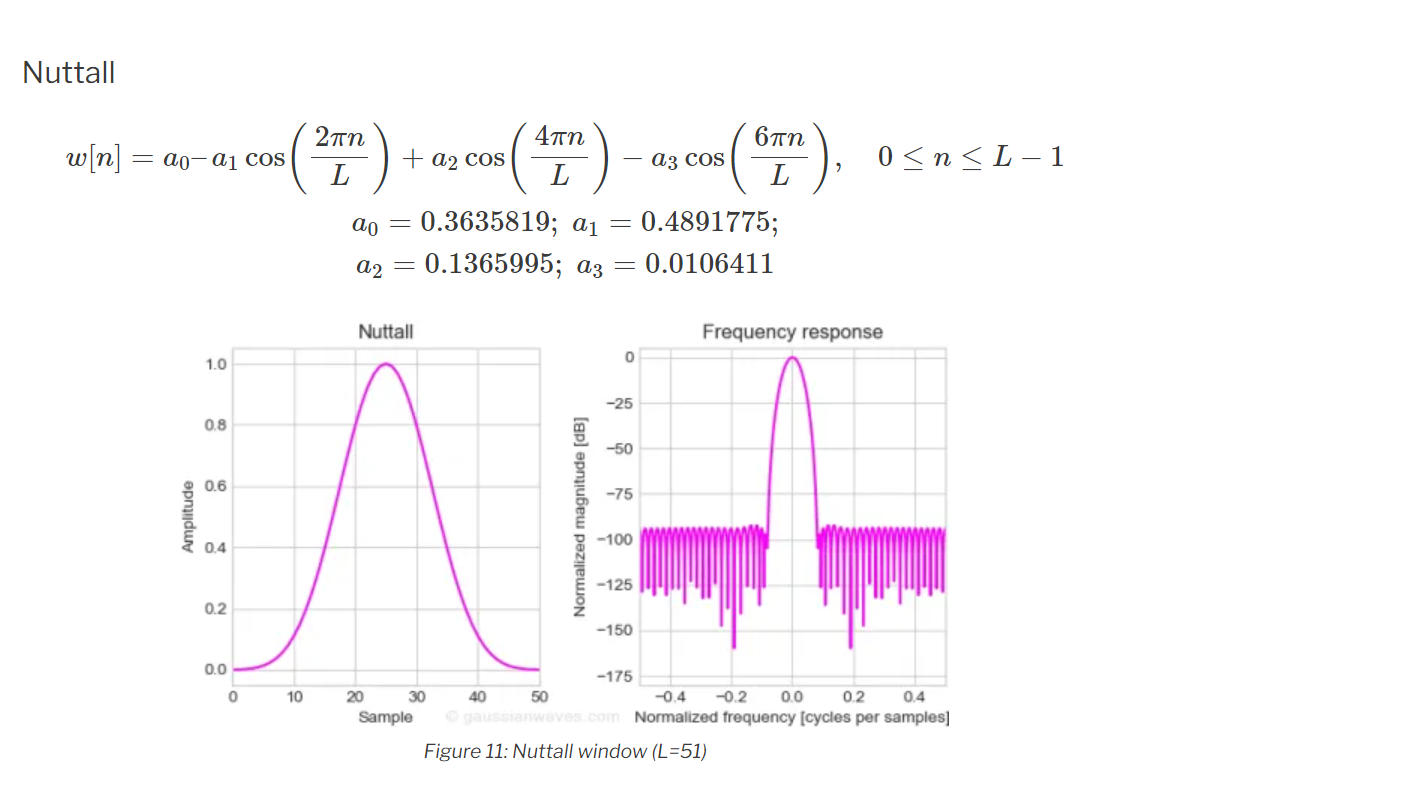
*Hình 3: Áp dụng của sổ cho tín hiệu*



*Hình 4. Kết quả của việc áp dụng cửa sổ cho tín hiệu không định kỳ.*

Tín hiệu thu được sẽ không được sao chép một cách hoàn hảo nhưng hiện tại, sự rò rỉ được giới hạn ở dài tần số nhỏ hơn. Vẫn có thể có sự rò rỉ từ ngăn dữ liệu chính sang các ngăn khác. Nói cách khác để làm cho thuật toán FFT hoạt động, các hàm cửa sổ tạo ra các tín hiệu được sửa đổi 1 chút. Những sửa đổi này có ý nghĩa là một số thao tác được thực hiện trên dữ liệu kết quả sẽ dẫn đến những lỗi nhỏ. Hậu quả của việc này là việc cộng năng lượng của tất cả các thùng không cho kết quả khớp với tổng mức năng lượng thực tế.

Lựa chọn cửa sổ dữ liệu Nuttall bởi vì hình dạng của nó phù hợp với tín hiệu hình sin sử dụng. Mất mát tín hiệu là ít nhất.



Nhân tín hiệu với cửa sổ dữ liệu sau đó thực hiện phép biến đổi DFT để hạn chế dò rỉ tín hiệu.

**CHƯƠNG 2: XÂY DỰNG ỨNG DỤNG**

## 2.1. Các hàm chính

### 2.1.1. Hàm ghi âm

Hàm này ghi âm từ thiết bị cụ thể, lưu trữ dữ liệu âm thanh vào một file WAV và cung cấp thông tin về quá trình ghi âm thông qua các ghi chú được chèn vào một TextView hoặc một giao diện người dùng khác.

def MIC\_record(self, duration):

self.tpath = self.desktopPath + "\\" + self.fileTime

        os.mkdir(self.desktopPath + "\\" + self.fileTime)

        wave\_Path = self.tpath + "\\MIC\_record.wav"

Ở đây file âm thanh sau khi được ghi sẽ được lưu với tên MIC\_record trong một thư mục tạo sẵn ở desktop.

self.textViewInsertData(False, wave\_Path)

        CHUNK = 1024

        FORMAT = pyaudio.paInt16

        CHANNELS = 1

        RATE = 44100

p = pyaudio.PyAudio()

Đây lần lượt là các thông số cho quá trình ghi âm:

* CHUNK: là kích thước của mỗi khối dữ liệu âm thanh được đọc và ghi vào file WAV. Ở đây, kích thước đang được để là 1024 bytes.
* FORMAT: là định dạng của dữ liệu âm thanh, định dạng là paInt16 có nghĩa là âm thanh được biểu diễn bằng số nguyên 16-bit. Đây là định dạng phổ biến cho các tệp âm thanh WAV.
* CHANNELS: là số kênh âm thanh được ghi âm, trong trường hợp này có một kênh (mono). Có nghĩa là âm thanh sẽ được ghi âm từ một micro hoặc một nguồn âm thanh duy nhất.
* RATE: là tần số mẫu tức là số mẫu âm thanh được lấy mỗi giây. Trong đoạn code này tần số mẫu là 44100mẫu/giây. Đây là một tần số mẫu tiêu chuẩn cho các file âm thanh chất lượng CD.

Sau đó tạo một đối tượng PyAudio mới. PyAudio là một thư viện Python cho phép thực hiện các hoạt động liên quan đến âm thanh như ghi âm, phát lại âm thanh, và xử lý âm thanh.

        try:

            stream = p.open(format=FORMAT, channels=CHANNELS, rate=RATE, input=True, input\_device\_index=self.inputID, frames\_per\_buffer=CHUNK)

        except:

            self.textViewInsertData(False, "\* recording error")

        wf = wave.open(wave\_Path, 'wb')

        wf.setnchannels(CHANNELS)

        wf.setsampwidth(p.get\_sample\_size(FORMAT))

        wf.setframerate(RATE)

        self.textViewInsertData(False, "\* recording")

        for i in tqdm(range(0, int(RATE / CHUNK \* duration))):

            data = stream.read(CHUNK)

            wf.writeframes(data)

        self.textViewInsertData(False, "\* done recording")

        stream.stop\_stream()

        stream.close()

        p.terminate()

        wf.close()

Trong đoạn code này sử dụng cấu trúc try-except để xử lý các ngoại lệ có thể xảy ra. Trong trường hợp có bất kì ngoại lệ nào một thông báo lỗi được chèn vào TextView cho biết có lỗi trong quá trình ghi âm.

Nếu không có vấn đề gì xảy ra sẽ hiện thị thông báo bắt đầu ghi âm và hoàn thành ghi âm sau khi ghi âm xong.

### 2.1.2. Hàm tính THD+N

THD(Total Harmonic Distortion) là một chỉ số đo lường mức độ méo hài tổng thể trong một tín hiệu âm thanh. THD được định nghĩa là tổng số lượng méo hài có trong tín hiệu âm thanh so với tín hiệu âm thanh gốc.

Khi tín hiệu âm thanh được truyền qua các thiết bị điện tử như ampli,loa,**cục đẩy công suất**... hay các thiết bị thu âm, tín hiệu sẽ bị méo hài do các yếu tố như méo hài bậc cao, méo hài bậc thấp, nhiễu và các tương tác giữa các thành phần khác nhau trong mạch.

THD thường được đo bằng phần trăm, với mức độ THD thấp hơn cho thấy tín hiệu âm thanh có chất lượng tốt hơn. Mức độ THD thấp cũng đảm bảo rằng tín hiệu âm thanh được tái tạo chính xác hơn, không bị méo hài và giúp truyền tải âm thanh một cách chính xác hơn.

THD là một thông số quan trọng trong ngành công nghiệp âm thanh, được sử dụng để đánh giá chất lượng âm thanh của các thiết bị điện tử. Các kỹ sư âm thanh và nhà sản xuất luôn cố gắng giảm thiểu mức độ THD của các thiết bị âm thanh để cải thiện chất lượng âm thanh và tạo ra trải nghiệm nghe nhạc tốt nhất cho người dùng.

def Get\_THD\_from\_FFT\_result(self, frequency, FFT\_result,Sample\_Rate):

        notch = 7

        FFT\_point = 32768

        position = int(frequency \* FFT\_point/Sample\_Rate)

        Fundamental=0

        Fundamental = max(FFT\_result[position - notch:position + notch + 1])

        Fundamental=pow(10,(Fundamental/20))

        sum\_noise = 0

        # for i in range(14, length):

        for i in range(14, 13654):

            if i < position - notch or i > position + notch:

                sum\_noise = sum\_noise + (10 \*\* (FFT\_result[i] / 20)) \*\* 2

            else:

                sum\_noise = sum\_noise + 0

        sum\_noise = sum\_noise \*\* 0.5

        THD\_value = 20 \* math.log10(sum\_noise / Fundamental)

        return THD\_value

Đoạn code này tính toán giá trị THD từ kết quả của biến đổi Fourier nhanh (FFT) của tín hiệu, dựa trên các thông tin về tần số, kích thước lấy FFT và tốc độ lấy mẫu. Nó tính toán tỷ lệ giữa nhiễu và tín hiệu cơ bản để đo lường mức độ biến dạng của tín hiệu.

notch = 7

    FFT\_point = 32768

      position = int(frequency \* FFT\_point/Sample\_Rate)

        Fundamental=0

        Fundamental = max(FFT\_result[position - notch:position + notch + 1])

        Fundamental=pow(10,(Fundamental/20))

Ở đây notch: là biến được sử dụng để xác định độ rộng của khe lọc khi tính toán THD. Khi tính toán, chỉ có các thành phần tần số trong khu vực gần tần số cơ bản được xem xét, và các tần số khác được bỏ qua. Điều này giúp loại bỏ nhiễu và các thành phần tần số không mong muốn khác.

FFT\_point: biến này cho biết số lượng điểm trong dãy dữ liệu khi thực hiện FFT. Trong trường hợp này, FFT được thực hiện trên một tín hiệu có 32768 điểm. Số điểm quá ít có thể dẫn đến mất mát thông tin, trong khi số điểm quá nhiều có thể làm tăng độ phức tạp tính toán mà không cần thiết.

position = int(frequency \* FFT\_point/Sample\_Rate)

Dòng này tính toán vị trí của tần số cơ bản trong mảng kết quả FFT. Điều này dựa trên tần số cần xem xét, kích thước mảng FFT và tốc độ lấy mẫu.

Tiếp theo đi tính Fundamental( biên độ tín hiệu âm thanh đầu vào). giá trị lớn nhất của biên độ trong một khoảng gần tần số cơ bản, được xác định bởi biến notch. Sau đó đi tính giá trị cực đại của tần số cơ bản được chuyển đổi từ đơn vị dB sang giá trị tuyệt đối.

Đi tính Fundamental áp dụng vào công thức:

THD(%) = 100 x

**THD+N** là một chỉ số đo lường mức độ méo hài và nhiễu tổng thể trong một tín hiệu âm thanh. THD+N thường được đo bằng phần trăm hoặc decibel(dB) và được tính bằng tổng số lượng méo hài và nhiễu trong tín hiệu âm thanh so với tín hiệu âm thanh gốc.

* THD+N(dB)= 10\*log(Cường độ của tổng tất cả các hài và nhiễu / Cường độ của tín hiệu âm thanh đầu vào).
* THD+N(%) = (Cường độ của tổng tất cả các hài và nhiễu / Cường độ của tín hiệu âm thanh đầu vào)\*100%.

Mức độ THD+N thấp hơn cho thấy tín hiệu âm thanh có chất lượng tốt hơn và không bị nhiễu. Các kỹ sư âm thanh và nhà sản xuất luôn cố gắng giảm thiểu mức độ THD+N của các thiết bị âm thanh để cải thiện chất lượng âm thanh và tạo ra trải nghiệm nghe nhạc tốt nhất cho người dùng.

sum\_noise = 0

# for i in range(14, length):

for i in range(14, 13654):

if i < position - notch or i > position + notch:

sum\_noise = sum\_noise + (10 \*\* (FFT\_result[i] / 20)) \*\* 2

else:

sum\_noise = sum\_noise + 0

sum\_noise = sum\_noise \*\* 0.5

THD\_value = 20 \* math.log10(sum\_noise / Fundamental)

return THD\_value

Khởi tạo biến sum\_noise là 0, sẽ được sử dụng để tính tổng của nhiễu từ các thành phần tần số không phải là tần số cơ bản.

Sau đó bắt đầu một vòng lặp từ chỉ số 14 đến 13654. Các giá trị cụ thể này có thể phụ thuộc vào kích thước thực tế của mảng FFT\_result và cần phải được điều chỉnh cho ứng dụng cụ thể. Kiểm tra xem chỉ số hiện tại có nằm ngoài phạm vi của tần số cơ bản không.

if i < position - notch or i > position + notch:

                sum\_noise = sum\_noise + (10 \*\* (FFT\_result[i] / 20)) \*\* 2

            else:

                sum\_noise = sum\_noise + 0

Nếu chỉ số không nằm trong phạm vi của tần số cơ bản, giá trị nhiễu tại chỉ số này được tính toán, được chuyển đổi từ dB sang giá trị tuyệt đối. Ta có A = 20 suy ra V = và độ lớn của mỗi harmonic được tính bằng bình phương của amplitudes của nó.

sum\_noise = sum\_noise \*\* 0.5

        THD\_value = 20 \* math.log10(sum\_noise / Fundamental)

        return THD\_value

Tính căn bậc hai của tổng bình phương của nhiễu, chuẩn hóa nó để tính toán THD. Sau đó tính giá trị THD bằng cách chia tổng nhiễu cho giá trị cơ bản, sau đó chuyển đổi kết quả sang đơn vị đo decibel (dB).

### 2.1.3. USB HID

Đoạn code này được sử dụng để giao tiếp với một thiết bị USB thông qua giao thức HID (Human Interface Device). Hình thức giao tiếp này thường được sử dụng cho các thiết bị như tai nghe, bàn phím, chuột, joystick, và các thiết bị người dùng khác.

def USBHID(self):

        isPressUp = False

        isPressDown = False

        isPressMid = False

Đầu tiên là các biến dùng để theo dõi trạng thái nhấn của các phím trên thiết bị HID. Ban đầu, chúng được đặt là False. Ở phần này có 3 nút chính(tăng âm lượng, giảm âm lượng, phát/tạm dừng). Vòng lặp While là một vòng lặp vô hạn được sử dụng để duy trì việc đọc dữ liệu từ thiết bị HID.

time.sleep(0.1)

            self.isPlugLabel.configure(text='Unplug')

            for hid\_interface in hid.enumerate(0x31B2, 0x0111):

                if hid\_interface['interface\_number'] != 3:

                    continue

                hid\_device = hid.device()

                hid\_device.open\_path(hid\_interface['path'])

                self.isPlugLabel.configure(text='Plugged')

Time.sleep(0.1) dùng để đợi 0.1 giây trước mỗi lần lặp để tránh quá tải hệ thống. Phần còn lại được sử dụng để cập nhật trạng thái hiển thị trên giao diện người dùng, thông báo về việc cắm hoặc rút thiết bị. Ở đây duyệt qua tất cả các thiết bị HID có Vendor ID (VID) là 0x31B2 và Product ID (PID) là 0x0111. Các VID và PID này có thể thay đổi tùy thuộc vào thiết bị HID cụ thể bạn đang sử dụng.

try:

     while self.hid\_flag == True:

      hid\_data = hid\_device.read(1024)

      if hid\_data:

      self.textViewInsertData(False, 'HID Data: "{}"'.format(hid\_data))

      if format(hid\_data) == "[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]":

        isPressUp = True

      self.volUpLabel.configure(background="green")

          self.textViewInsertData(False, "Vol + Pressed")

      if format(hid\_data) == "[1, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0]":

            isPressMid = True

            self.middleLabel.configure(background="green")

            self.textViewInsertData(False, "Middle Pressed")

      if format(hid\_data) == "[1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0]":

            isPressDown = True

            self.volDownLabel.configure(background="green")

            self.textViewInsertData(False, "Vol - Pressed")

      if format(hid\_data) == "[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]":

            if isPressUp == True:

               self.volUpLabel.configure(background="white")

               self.textViewInsertData(False, "Vol + Released")

               isPressUp = False

            if isPressMid == True:

               self.middleLabel.configure(background="white")

               self.textViewInsertData(False, "Middle Released")

          isPressMid = False

            if isPressDown == True:

               self.volDownLabel.configure(background="white")

               self.textViewInsertData(False, "Vol - Released")

               isPressDown = False

Dữ liệu được đọc từ thiết bị HID được kiểm tra để xác định xem các phím nào đang được nhấn hoặc thả. Cụ thể, các giá trị HID data tương ứng với các phím được kiểm tra và nếu phát hiện phím nào được nhấn hoặc thả, các hành động tương ứng được thực hiện, ví dụ như thay đổi màu nền của một label trên giao diện người dùng hoặc ghi log dữ liệu.

     except Exception as e:

                    hid\_device.close()

Được sử dụng để xử lý các ngoại lệ có thể xảy ra trong quá trình đọc dữ liệu từ thiết bị HID. Trong trường hợp này, thiết bị HID được đóng lại để chuẩn bị cho một vòng lặp mới.

### 2.1.4. Phân tích FFT

Hàm này được thiết kế để thực hiện phân tích FFT (Fast Fourier Transform) trên một tín hiệu âm thanh từ một tệp WAV.

def FFT\_Function(self,savePath,path\_file,wavName,flag):

        fft\_input\_raw\_data = self.Get\_wave\_data(path\_file, wavName,flag)

        a0 = 0.3635819

        a1 = 0.4891775

        a2 = 0.1365995

        a3 = 0.0106411

        FFT\_point = 32768

        plt.ioff()

        f1, test1 = plt.subplots()

        plt.plot(np.array(fft\_input\_raw\_data) / (FFT\_point-1))

        test1.set\_title("WaveForm", fontsize=10)

        test1.set\_ylabel('Amplitude(dBV)', rotation=90, labelpad=5)

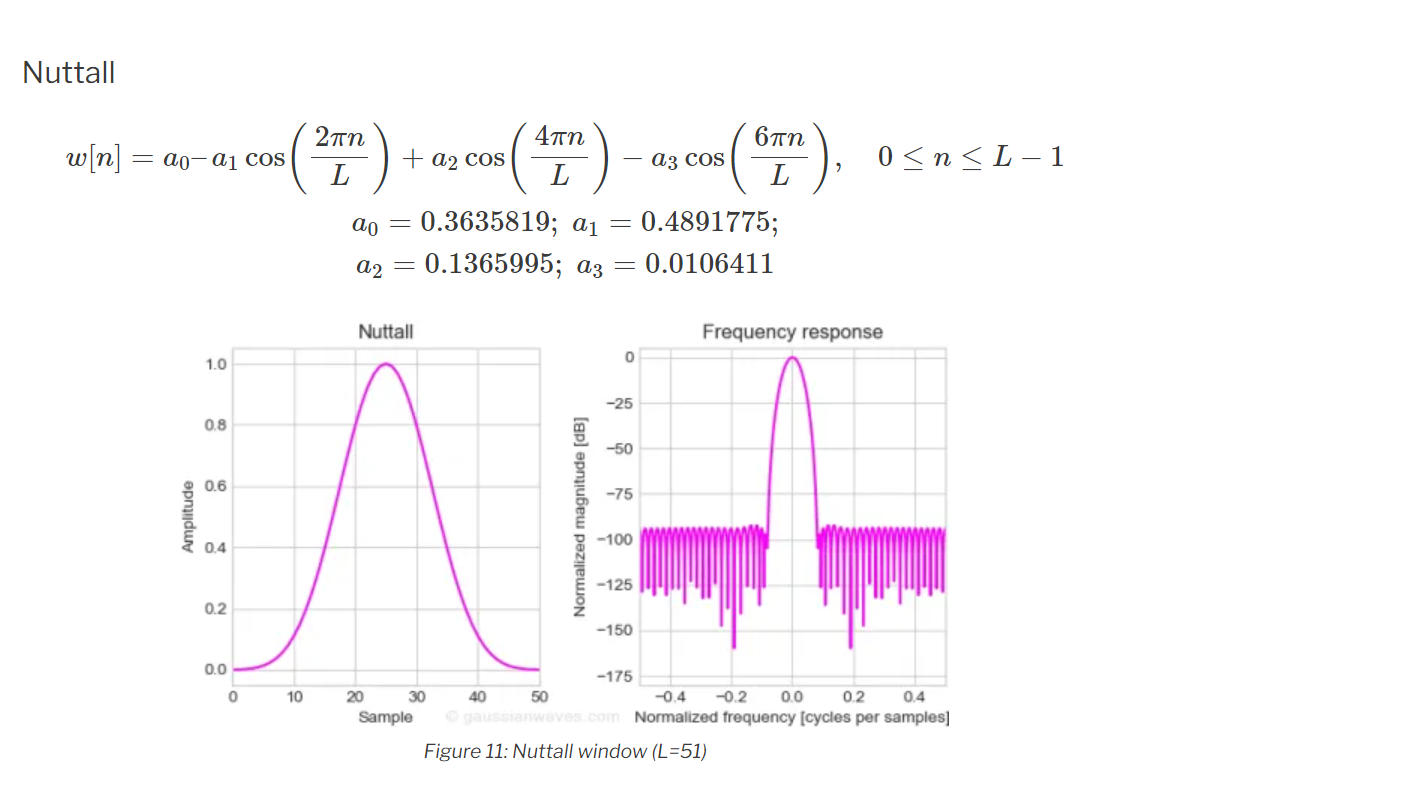
        test1.set\_xlabel('Time(s)', rotation=0, labelpad=5)

        plt.grid(True)

        wavePath = os.path.join(savePath, 'WaveForm.png')

        plt.savefig(wavePath)

Sau khi lấy được dữ liệu âm thanh từ tệp WAV được chỉ định sử dụng cửa sổ window để phân tích FFT. Như đã nói ở trên lựa chọn cửa sổ dữ liệu Nuttall bởi vì hình dạng của nó phù hợp với tín hiệu hình sin sử dụng. Mất mát tín hiệu là ít nhất.



Nhân tín hiệu với cửa sổ dữ liệu sau đó thực hiện phép biến đổi DFT để hạn chế dò rỉ tín hiệu.

Sau đó sử dụng thư viện matplotlib để vẽ đồ thị âm thanh gốc. Đặt tên và thư mục lưu file cho đồ thị sau khi vẽ.

fft\_input\_data = fft\_input\_raw\_data

        for i in range(0, FFT\_point):

            fft\_input\_data[i] = fft\_input\_data[i] \* (a0 - a1 \* np.cos(2 \* i \* np.pi / FFT\_point) + a2 \* np.cos(4 \* i \* np.pi / FFT\_point) - a3 \* np.cos(6 \* i \* np.pi / FFT\_point))

        fft\_output\_data = fft(fft\_input\_data, FFT\_point)

Tính toán dữ liệu đầu vào cho phân tích FFT bằng cách áp dụng cửa sổ dữ liệu Nuttall cho dữ liệu âm thanh gốc. Sau đó đi thực hiện phân tích FFT trên dữ liệu đầu vào.

fft\_output\_data\_amplitude = np.abs(fft\_output\_data)

        fft\_output\_data\_amplitude = 2\*fft\_output\_data\_amplitude / FFT\_point

        fft\_output\_data\_amplitude[0] = fft\_output\_data\_amplitude[0] / 2

        fft\_output\_data\_amplitude = fft\_output\_data\_amplitude / ((FFT\_point-1)\*a0)

        fft\_output\_data\_amplitude = 20 \* np.log10(fft\_output\_data\_amplitude)

Sau khi phân tích FFT sẽ đi tìm biên độ của tín hiệu. Để tính toán biên độ chúng ta cần tính module của số phức sử dụng hàm np.abs .

Khi lấy FFT biên độ sẽ có sự đối xứng và chia đôi độ lớn giữa hai miền âm và dương. Nhưng trong vật lý không tính toán phần âm nên khi tính toán cần gấp đôi biên độ để đảm bảo sự chính xác. Mặt khác tại vị trí [0] biên độ không có sự đối xứng nên tại vị trí này không cần nhân hai. Tiếp theo chuẩn hóa lại dữ liệu biên độ theo cửa sổ(Tín hiệu = tín hiệu gốc \* window = tín hiệu gốc \* (L-1)\*a0).

Chuyển đổi đơn vị sang dB bằng cách sử dụng hàm logarit. Điều này giúp hiển thị biên độ của tín hiệu một cách trực quan hơn trong phổ tần số.

n = np.linspace(0, int(FFT\_point / 2), int(FFT\_point / 2)+1 ,endpoint=False)

frequency = n \* 44100/ FFT\_point

Sau đó tiếp tục đi tìm tần số. Hàm np.linspace tạo ra một mảng các giá trị tuyến tính từ 0 đến FFT\_point / 2, bao gồm cả 0 và giá trị cuối cùng là FFT\_point / 2 và mỗi điểm cách đều nhau một khoảng FFT\_point / 2 + 1

Endpoint để loại bỏ giá trị cuối cùng của khoảng giá trị tạo ra, vì chúng ta không muốn bao gồm giá trị cuối cùng vì nó sẽ là một bước quá lớn khi chúng ta chia thành phần tần số.

Tần số của mỗi chỉ số được tính bằng công thức f= . Kết quả là một mảng các tần số tương ứng với các bước tần số

f2, test2 = plt.subplots()

        plt.semilogx(frequency, fft\_output\_data\_amplitude[0:int(FFT\_point / 2) + 1])

        test2.set\_title("FFT", fontsize=10)

        test2.set\_ylabel('Amplitude(dBV)', rotation=90, labelpad=5)

        test2.set\_xlabel('Frequency(Hz)', rotation=0, labelpad=5)

        test2.set\_xticks([10, 100, 1000, 10000,20000], ['10', '100', '1K', '10K','20k'])

        plt.grid(True)

        fftPath = savePath + "\\FFT.png"

        plt.savefig(fftPath)

        plt.close()

Đi vẽ đồ thị biên độ FFT với trục x là tần số trục y là biên độ. Truyền vào các tham số như tần số đã tính toán và biên độ FFT của các thành phần tần số tương ứng. Chỉ vẽ đồ thị cho nửa phổ(từ 0 đến FFT\_point / 2) vì phổ tần số của tín hiệu thường là đối xứng.

Thiết lập tiêu đề và nhãn trục, thiết lập điểm chia trên trục x, thêm lưới đồ thị để đọc các giá trị một cách dễ dàng hơn. Sau đó lưu hình ảnh dưới dạng png.

Sau khi phân tích FFT tính toán được biên độ và tần sổ xử dùng hàm kết quả phân tích FFT(Get\_FFT\_Result) để lấy ra biên độ và tần số.

for i in range(0, int(FFT\_point / 2), 1):

            file.write(str(frequency[i]) + "\t" + str(fft\_output\_data\_amplitude[i]) + "\n")

        file.close()

        return fft\_output\_data\_amplitude[0:int(FFT\_point / 2)]

Tạo một vòng lặp chạy từ 0 đến FFT\_point / 2 với bước nhảy là 1 sau đó ghi các giá trị tần số và biên độ tương ứng vào tệp tin. Sau đó chuyển đổi giá trị biên độ thành một chuỗi. Sau khi đóng tập tin trả về mảng con chứa các giá trị biên độ tương ứng.

**CHƯƠNG 3: CÀI ĐẶT VÀ TRIỂN KHAI**

**3.1. Giới thiệu về công cụ và môi trường cài đặt**

***3.1.1. Visual Studio Code***

Visual Studio Code là trình soạn thảo, biên tập lập trình mã nguồn miễn phí được sử dụng trên 3 nền tảng đó là: Windows, macOS và Linux được xây dựng, phát triển bởi Microsoft. Visual Studio Code được các chuyên gia công nghệ thông tin đánh giá cao, nó là sự kết hợp hoàn hảo giữa IDE và Code Editor.

Visual Studio Code còn có nhiệm vụ hỗ trợ các nền tảng như: JavaScript, TypeScript và Node.js. Ta có thể hiểu cụ thể công việc của nó là mang đến một hệ sinh thái mới vô cùng phong phú cho các ngôn ngữ lập trình.

### *3.1.2. Những tính năng vượt trội của Visual Studio Code*

* Hỗ trợ đa ngôn ngữ lập trình
* Hỗ trợ tính năng đa dạng trên các nền tảng
* Kho lưu trữ dữ liệu an toàn
* Cung cấp nơi lưu trữ tiện ích mở rộng
* Hỗ trợ web
* Lưu trữ dữ liệu dạng phân cấp
* Hỗ trợ viết code
* Hỗ trợ thiết bị đầu cuối
* Màn hình đa nhiệm
* Hỗ trợ Git
* Intellisense

### *3.1.3. Các bước cài đặt Visual Studio Code*

**Bước 1**. Tải file cài đặt Visual Studio Code cho Windows.

**Bước 2**. Sau khi tải xong, chạy file VSCodeUserSetup.exe.

**Bước 3**. Nhấp vào Next để cài đặt. Tiếp theo đồng ý điều khoản sử dụng.

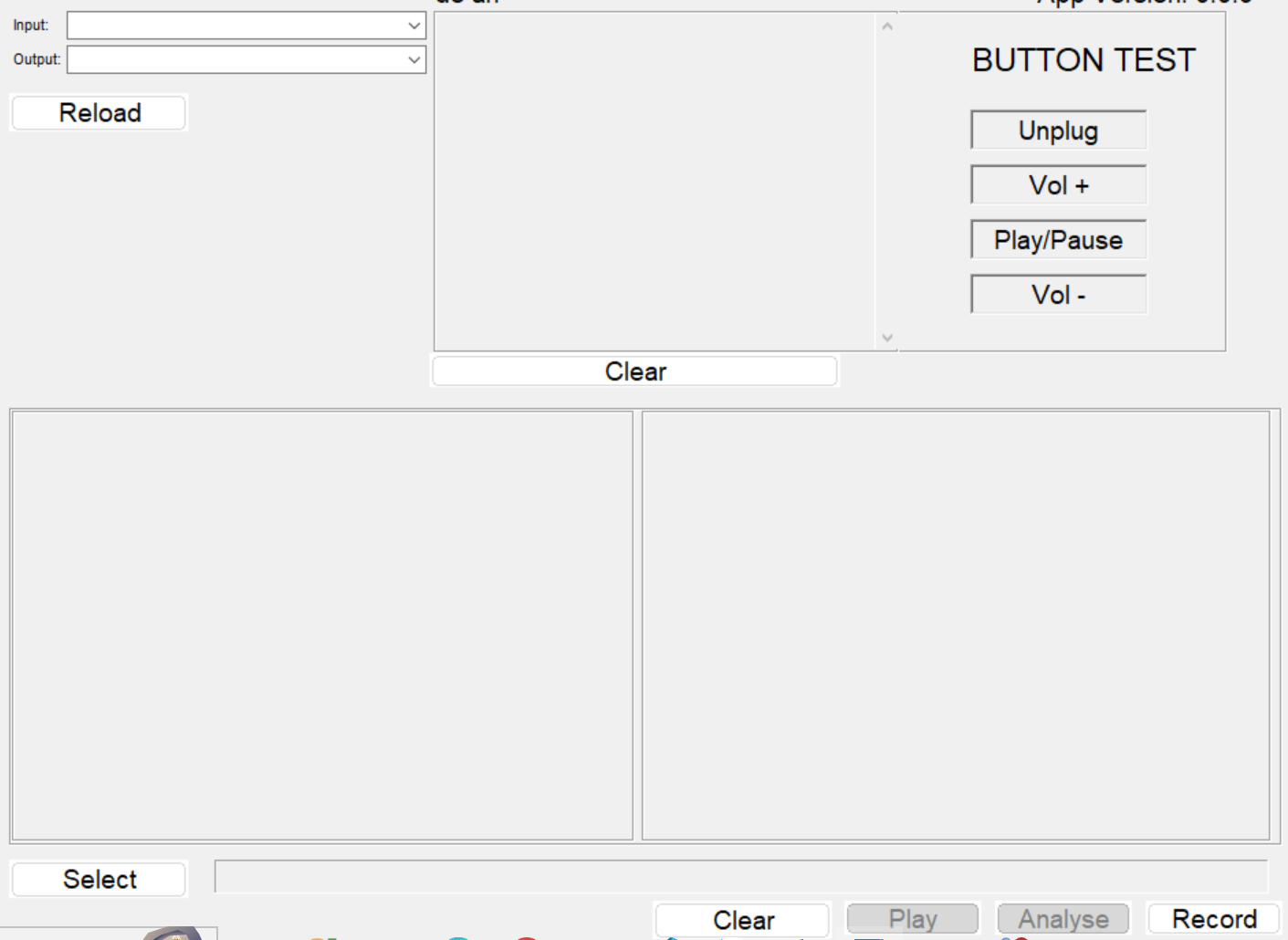
**Bước 4**: Lựa chọn nơi cài đặt (Nên để mặc định) sau đó nhấn Next.

**Bước 5**. Các bước tiếp theo tiếp tục nhấn Next cho tới khi hoàn tất. Trong quá trình này, mình khuyên nên tích chọn vào 2 chức năng: (1) Add “Open with Code” action to Windows Explorer file context menu và (2) Add “Open with Code” action to Windows Explorer directory context menu. Việc này giúp ta có thể click chuột phải vào thư mục sẽ có lựa chọn mở bằng VS Code.

**Bước 6**. Cài đặt hoàn tất, chúng ta có thể trải nghiệm.

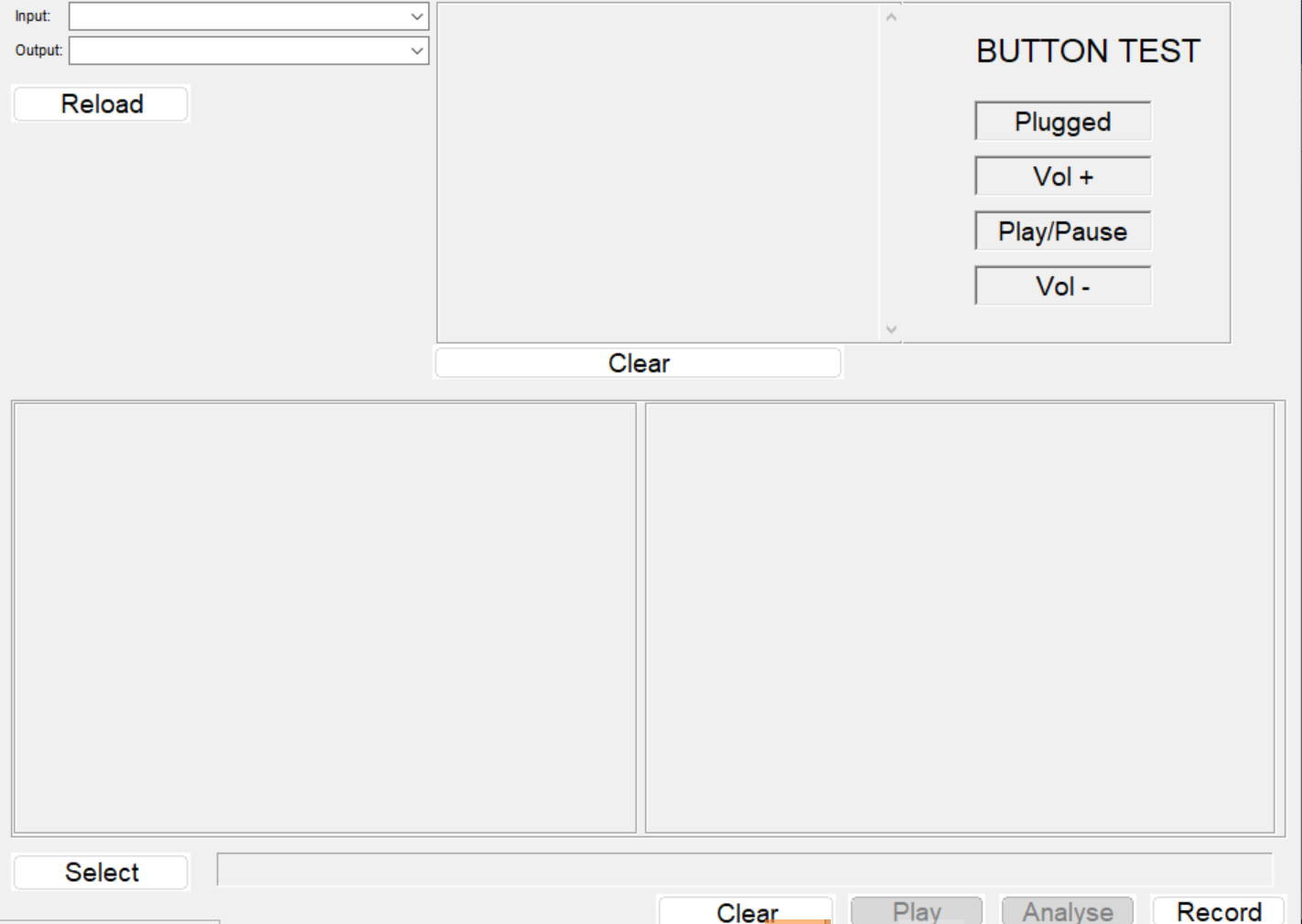
**3.2. Giao diện phần mềm**

Đây là giao diện của phần mềm sau khi chạy chương trình. Giao diện rất đơn giản và dễ dàng sử dụng.

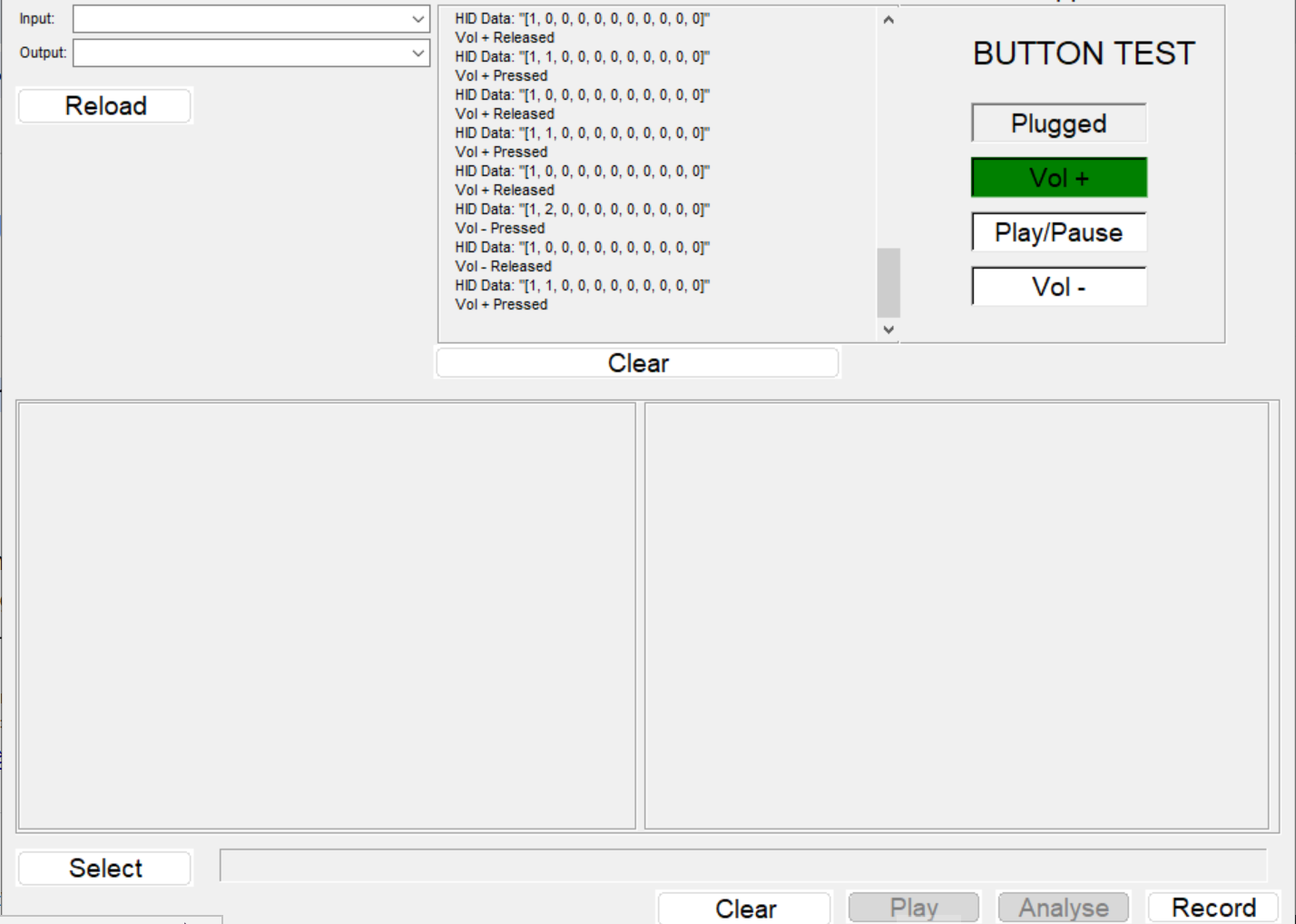


*Hình 3.1. Giao diện trang chủ*

Khi người dùng kết nối tai nghe thì bên phần BUTTON TEST sẽ chuyển sang chế độ plugged các nút tăng giảm âm lượng, chạy và dừng file âm thanh sẽ có màu xanh lá cây khi người dùng ấn những nút này trên tai nghe và nó sẽ hiện thị lịch sử thao tác ra bảng bên cạnh.

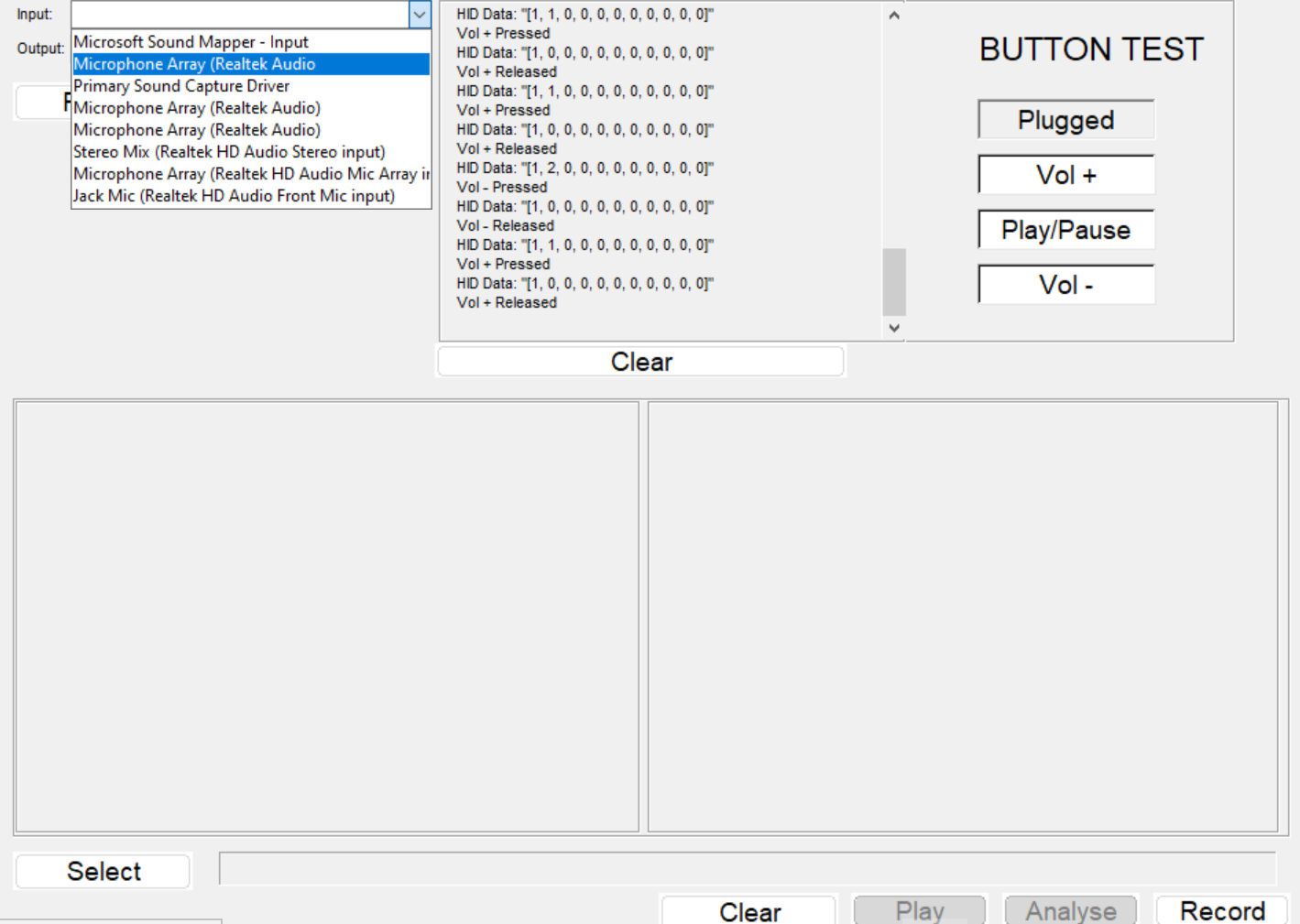


*Hình 3.2. Giao diện phần mềm khi kết nối tai nghe*

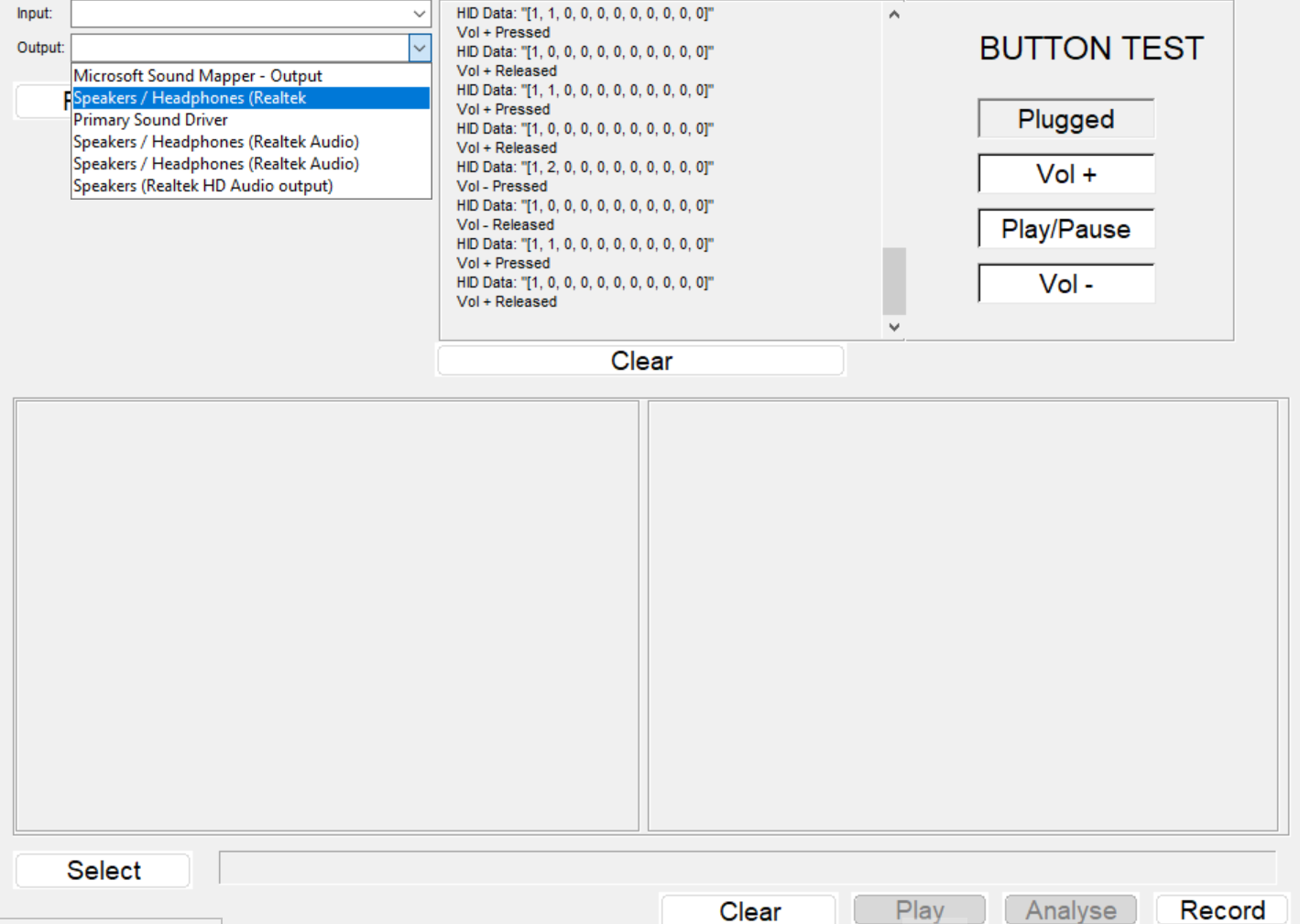


*Hình 3.3. Giao diện phần mềm khi ấn nút trên tai nghe*

Chúng ta cũng có thể lựa chọn thiết bị đầu vào và đầu ra của phần mềm.

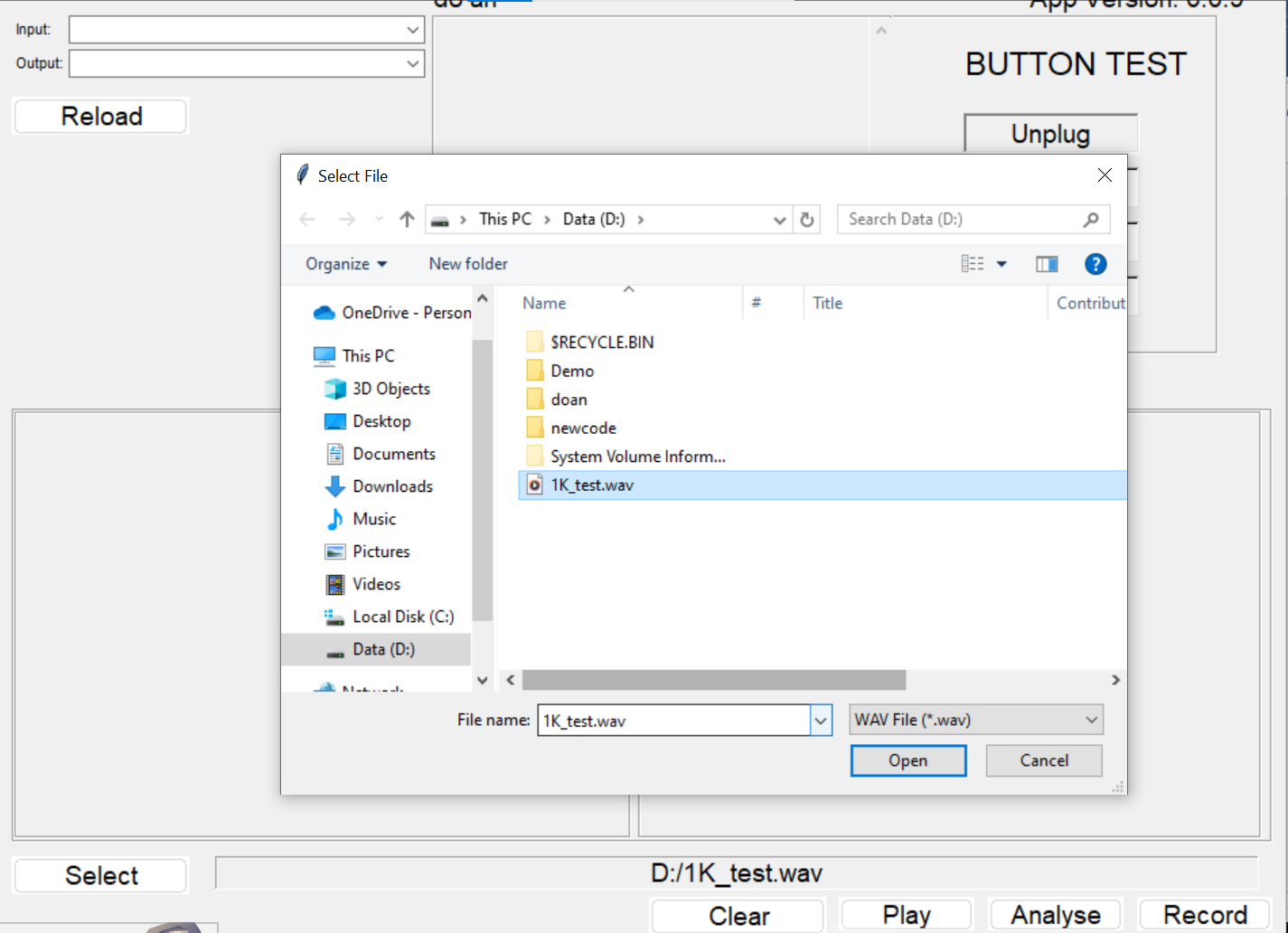


*Hình 3.4. Giao diện lựa chọn phần input*



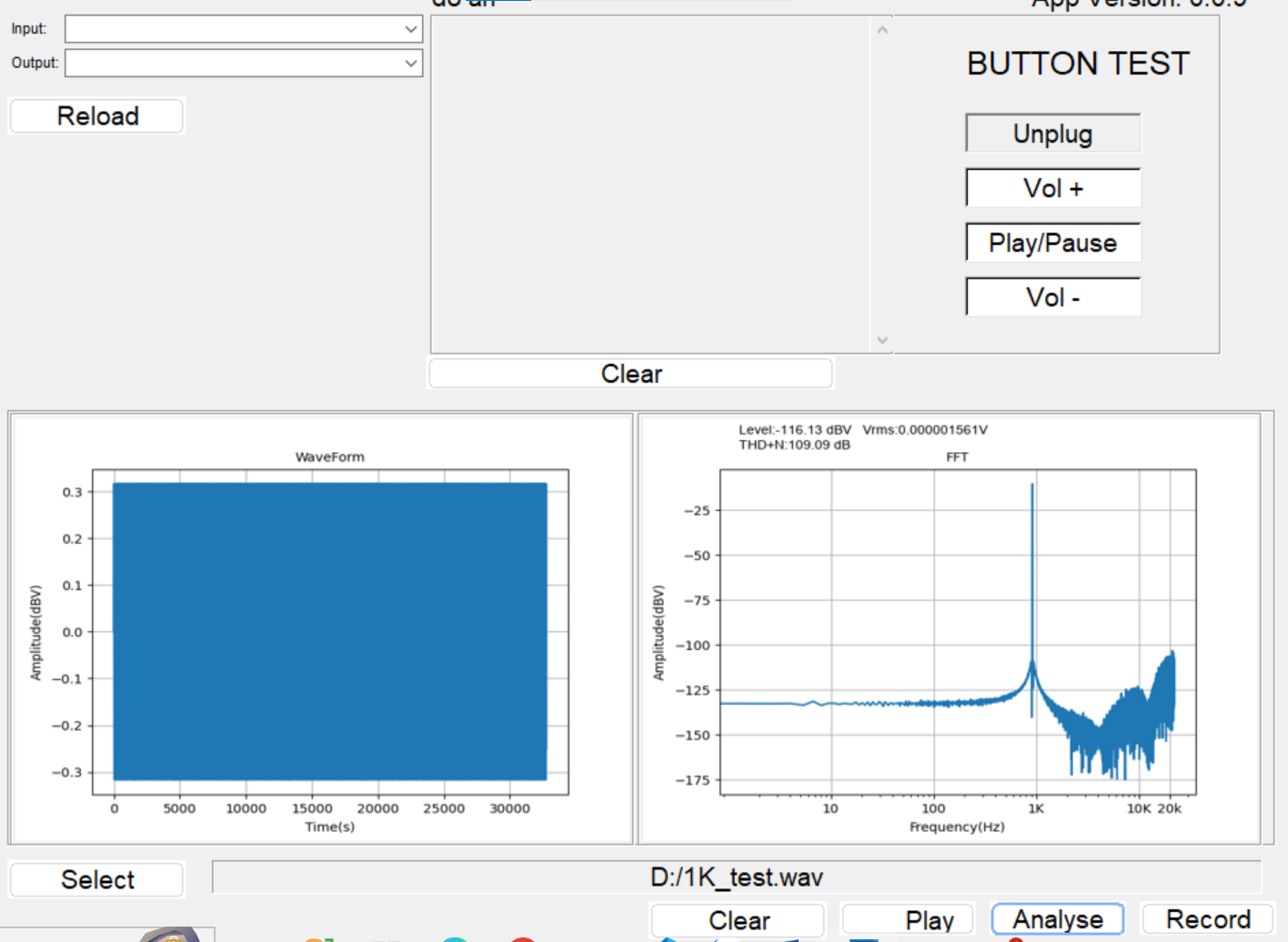
*Hình 3.5. Giao diện lựa chọn phần output*

Chúng ta cũng có thể lựa chọn những file âm thanh được lưu trong máy với đuôi .wav bằng cách ấn vào nút Select ở góc dưới bên tay trái.



*Hình 3.6. Giao diện lựa chọn file âm thanh được lưu trong máy*

Sau khi đã chọn được file âm thanh mong muốn có thể nghe file âm thanh bằng cách ấn vào nút play. Nếu muốn phân tích file âm thanh thì ấn nút Analyse bên cạnh.



*Hình 3.7. Giao diện đồ thị file âm thanh sau khi phân tích*

Ngoài ra người dùng cũng có thể trực tiếp ghi âm một file âm thanh bằng cách Record sau đó phân tích.



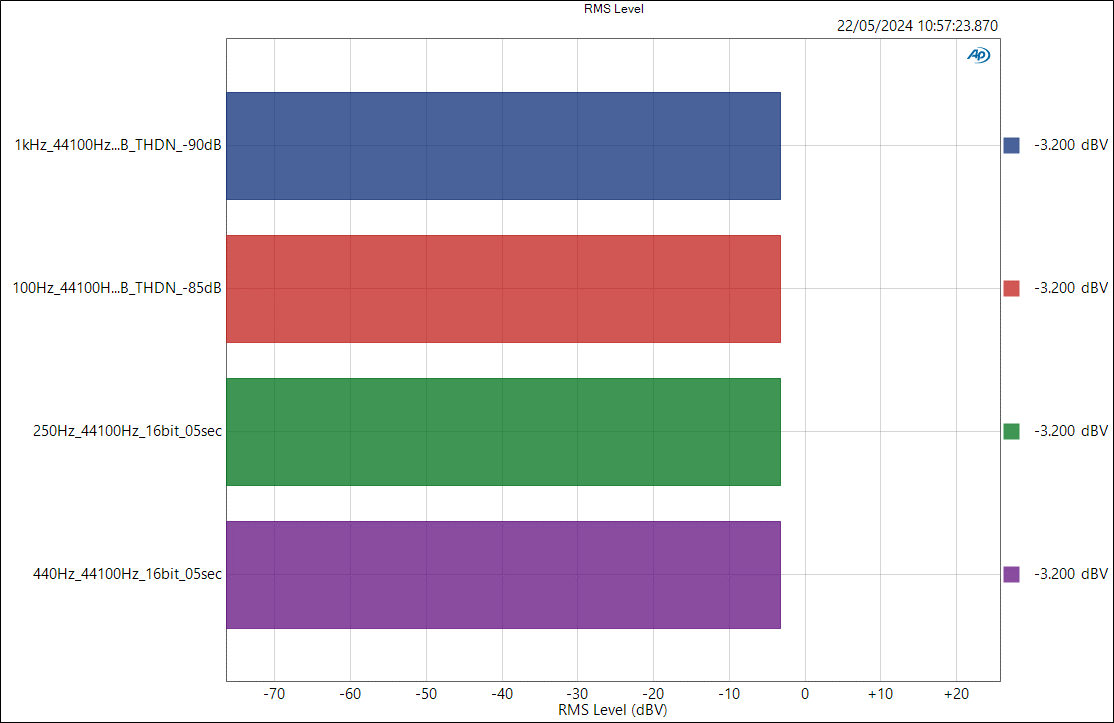
*Hình 3.8. Giao diện đồ thị file âm thanh phân tích sau khi record*

Khi được ghi âm ở ngoài môi trường file âm thanh sẽ có nhiễu dẫn đến đồ thị có xự rối loạn.

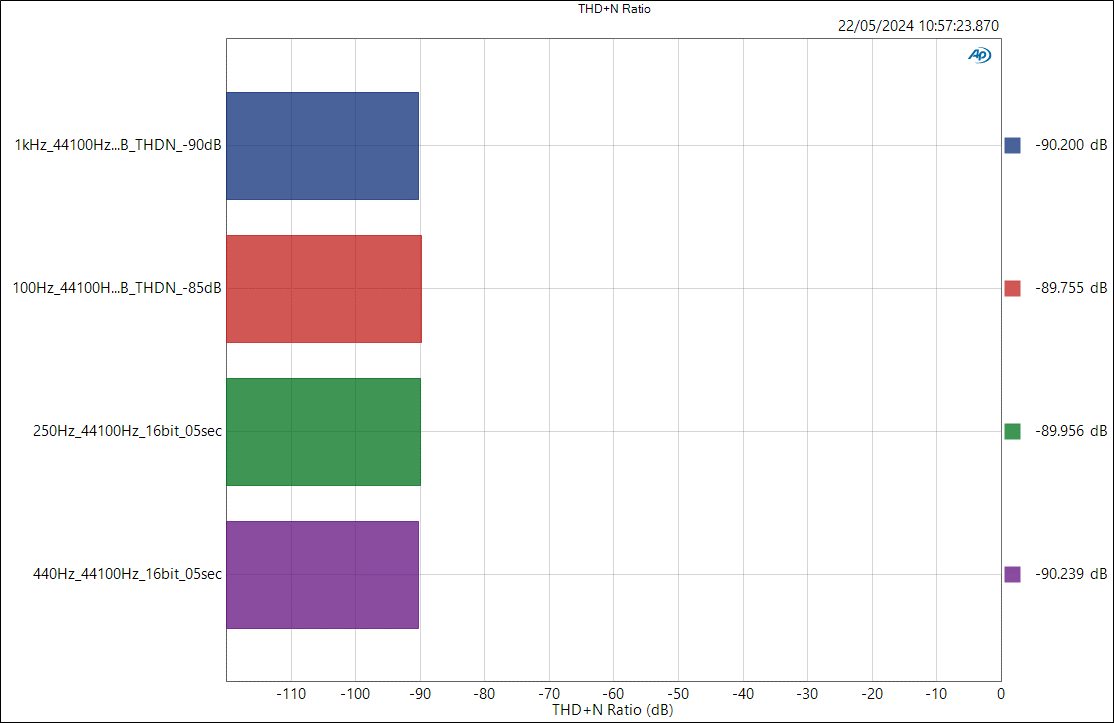
# KẾT LUẬN

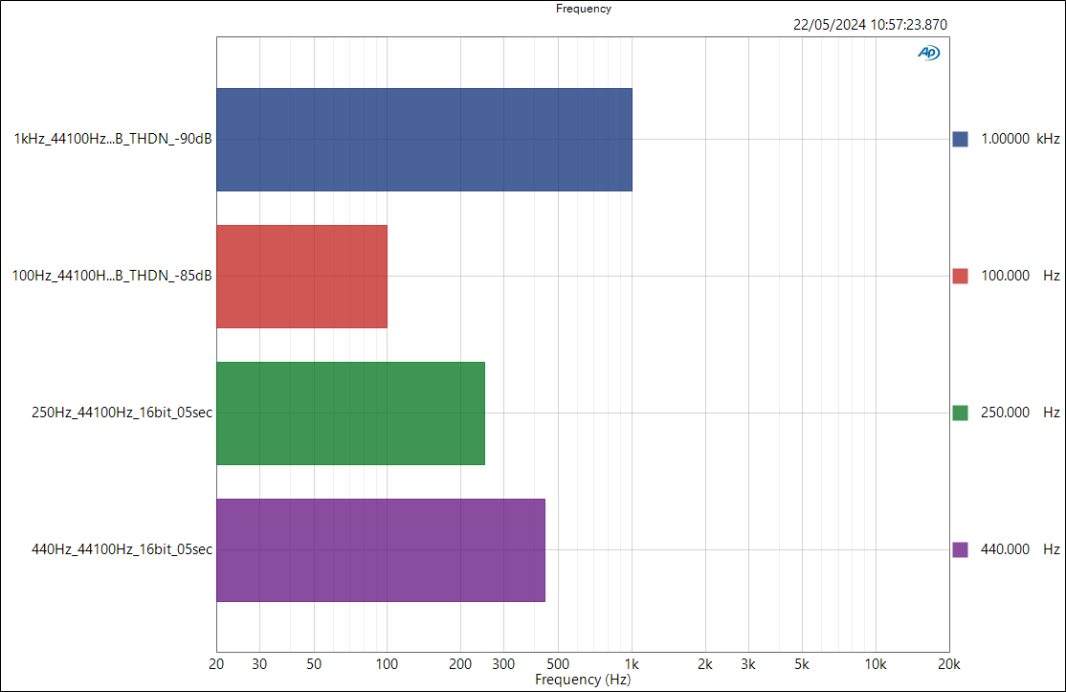
**So sánh kết quả đo tần số, biên độ, THD+N so với phần mềm thương mại APx500**.

Để so sánh kết quả đo ta sử dụng 4 file âm thanh mẫu tiêu chuẩn để kiểm tra

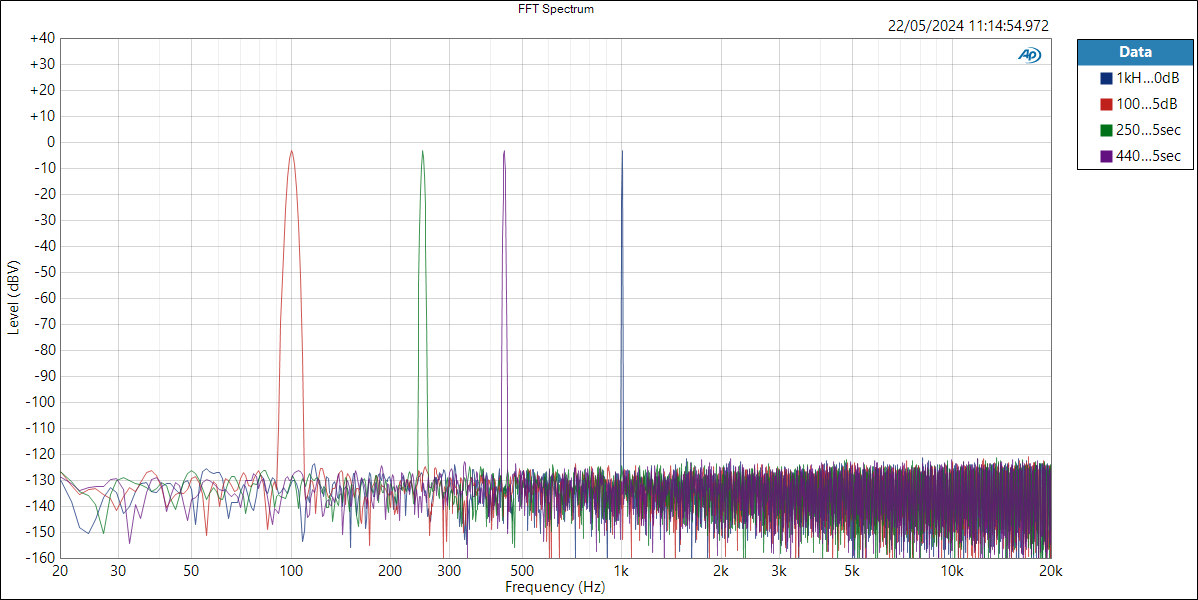
1. 100Hz\_Fs\_44100\_-3.2dB\_THDN\_-90dB.wav
2. 250Hz\_Fs\_44100\_-3.2dB\_THDN\_-90dB.wav
3. 440Hz\_Fs\_44100\_-3.2dB\_THDN\_-90dB.wav
4. 1000Hz\_Fs\_44100\_-3.2dB\_THDN\_-90dB.wav

*Hình 1. Kết quả đo biên độ của 4 file mẫu sử dụng phần mềm APx 500*

****

*Hình 2. Kết quả đo THD+N của 4 file mẫu sử dụng phần mềm APx500*

*Hình 3. Kết quả đo tần số của 4 file mẫu sử dụng phần mềm APx500*



*Hình 4. Tín hiệu trong miền tần số của 4 file mẫu*

***Bảng kết quả chạy phần mềm SA***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| File mẫu | Tần số (Hz) | Biên độ (dB) | THD+N(dB) | Tín hiệu trong miền tần số |
| 100Hz\_Fs\_44100\_-3.2dB\_THDN\_-90dB.wav | 100 | -3.51 | -83.46 |  |
| 250Hz\_Fs\_44100\_-3.2dB\_THDN\_-90dB.wav | 250 | -3.40 | -84.61 |  |
| 440Hz\_Fs\_44100\_-3.2dB\_THDN\_-90dB.wav | 440 | -3.21 | -89.95 |  |
| 1000Hz\_Fs\_44100\_-3.2dB\_THDN\_-90dB.wav | 1000 | -3.21 | -90.52 |  |

Như vậy thông qua kết quả so sánh giữa phần mềm tự phát triển và phần mềm thương mại APx ta thấy sai số nhỏ đủ đáp ứng được yêu cầu kỹ thuật

**Kết quả đạt được**

Trong thời gian nghiên cứu, xây dựng phần mềm mặc dù đã gặp nhiều khó khăn nhưng ứng dụng đã đạt được một số kết quả nhất định:

Xây dựng phần mềm phân tích âm thanh SA đã đáp ứng được yêu cầu thực tiễn.

* Phần mềm có giao diện đơn giản dễ sử dụng.
* Vẽ được biểu đồ tín hiệu trong miền thời gian và trong miền tần số
* Cho phép ghi âm một đoạn âm thanh và lưu lại dưới dạng .wav từ đó phân tích đoạn âm thanh mới ghi.
* Phân tích âm thanh một cách nhanh chóng, chính xác.
* Xuất ra file lưu thông tin của đoạn âm thanh vừa phân tích.
* Kiểm tra được các nút tính năng của tai nghe như tang giảm, âm lượng, play/pause

Phần mềm phân tích âm thanh SA là một đề tài hết sức thiết thực với nhu cầu của các công ty về thiết bị âm thanh như hiện nay.

Tuy nhiên với thời gian ngắn, kiến thức và kinh nghiệm còn hạn chế, ứng dụng phân tích âm thanh SA vẫn còn nhiều thiếu sót chưa thực hiện được. Rất mong thầy giúp đỡ để ứng dụng phân tích âm thanh SA ngày càng hoàn thiện hơn.

**Kỹ năng đạt được:**

* Củng cố và nắm vững ngôn ngữ lập trình Python.
* Củng cố lại được các kiến thức toán học cũng như các kiến thức cơ bản về âm thanh.

**Hạn chế:**

* Kiến thức về ngôn ngữ Python còn hạn chế nên trong quá xây dựng gặp đôi chút khó khăn.
* Thời gian thực hiện ứng dụng còn hạn chế.
* Chưa thể tự thích nghi với kích thước màn hình ngẫu nhiên.
* Chưa có chế độ phóng to, thu nhỏ.

**Hướng phát triển**

Hiện nay em đang cố gắng hoàn thiện phần để việc phân tích âm thanh đạt hiểu quả tốt hơn. Phần mềm sẽ nâng cấp, các tính năng sẽ được cải thiện để việc sử dụng được dễ dàng. Chỉnh sửa kích thước để phù hợp với giao diện của nhiều dòng máy.

Tiếp tục hoàn thiện phần mềm hi vọng có thể đáp ứng được nhu cầu của sử dụng một cách tốt nhất. Đó là gì mà những gì mà cá nhân em mong đợi cho đề tài này.

**TÀI LIỆU THAM KHẢO**

1. Phép biến đổi Fourier: Fourier Transforms With scipy.fft: Python Signal Processing – Real Python
2. Tạo cửa sổ cơ bản trong Python:

https://www.youtube.com/watch?v=R2hqIpvA7zY&list=PL14WBbXTfR-4X5K\_JqbLlO3txcmG8pZ4f

1. Ngôn ngữ Python:

https://aws.amazon.com/vi/what-is/python/.

https://vi.wikipedia.org/wiki/Python\_(ng%C3%B4n\_ng%E1%BB%AF\_l%E1%BA%ADp\_tr%C3%ACnh).

1. Window function: Window function - figure of merits – GaussianWaves

https://www-modalshop-com.translate.goog/rental/learn/basics/how-to-choose-fft-window?\_x\_tr\_sl=en&\_x\_tr\_tl=vi&\_x\_tr\_hl=vi&\_x\_tr\_pto=wapp

1. THD: https://baochauelec.com/thd-total-harmonic-distortion-la-gi