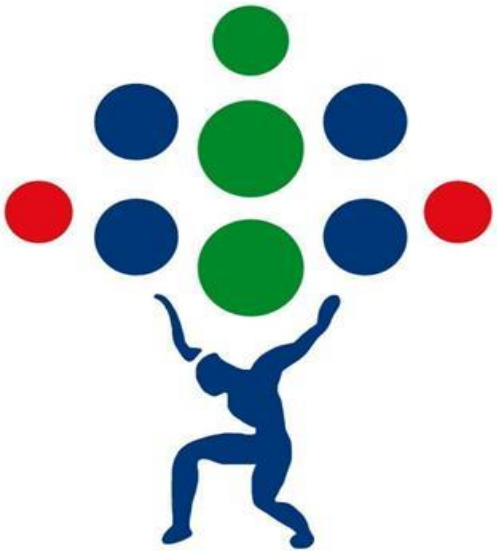
UNIVERSIDAD PRIVADA DOMINGO SAVIO



Separador de Voz y Ruido

Estudiante: Franz Denis Choque Villaca

Docente: Ing. Luis Miguel Flores Estrada

Carrera: Ingeniería de Sistemas

Gestión: 2024

Tarija-Bolivia Diciembre-2024

Índice

1. Objetivos del Proyecto

Objetivo General:

Desarrollar una aplicación en Python que permita separar la voz y el ruido de un archivo de audio o una grabación, utilizando técnicas de procesamiento de señales, específicamente la Transformada Rápida de Fourier (FFT).

Objetivos Específicos:

- Implementar una interfaz gráfica amigable que permita al usuario importar un archivo de audio o realizar una grabación en tiempo real.
- Analizar las frecuencias del audio utilizando la FFT y separar las componentes correspondientes a la voz y al ruido.
- Identificar e imprimir la frecuencia dominante de la voz en el rango de 85 Hz a 4000 Hz.
- Guardar los resultados procesados en archivos separados y reproducirlos automáticamente.
- Visualizar gráficamente las señales originales, de voz y de ruido.

2. Marco Referencial

El análisis y procesamiento de señales es fundamental en diversas áreas, como telecomunicaciones, acústica y control de calidad de audio. La Transformada Rápida de Fourier (FFT) es una herramienta matemática que permite transformar una señal del dominio del tiempo al dominio de la frecuencia, facilitando la identificación y manipulación de sus componentes frecuenciales.

En este proyecto, se define el rango de la voz humana como frecuencias entre 85 Hz y 4000 Hz, basado en estudios acústicos que identifican las frecuencias fundamentales y armónicos típicos de las voces humanas.

3. Materiales, Instrumentos y Herramientas

Software y librerías:

- Python 3.10 o superior: Lenguaje de programación para desarrollar el proyecto.
- Librerías utilizadas:
    - tkinter: Para la creación de la interfaz gráfica.
    - soundfile: Para la lectura y escritura de archivos de audio.
    - scipy.fft: Para realizar la Transformada Rápida de Fourier (FFT) e inversa (IFFT).
    - matplotlib: Para la visualización gráfica de las señales.
    - numpy: Para la manipulación y cálculo de datos numéricos.
    - pyaudio: Para la grabación de audio en tiempo real.
    - wave: Para el manejo básico de archivos WAV.

Hardware:

- Computadora con sistema operativo Windows o Linux.
- Micrófono para la grabación de audio (en caso de pruebas en tiempo real).

4. Ingeniería del Proyecto

Descripción del Sistema

El proyecto está compuesto por los siguientes módulos:

1. Importación de audio: Permite al usuario seleccionar un archivo de audio desde el sistema de archivos.
2. Grabación de audio: Habilita la captura de audio en tiempo real y su almacenamiento como archivo WAV.
3. Procesamiento del audio:

- Aplicación de la FFT para descomponer la señal en sus componentes frecuenciales.
- Separación de las frecuencias correspondientes a la voz y al ruido.
- Identificación de la frecuencia dominante de la voz dentro del rango establecido.

4. Generación de archivos: Almacena las señales procesadas (voz y ruido) en archivos WAV.
5. Visualización gráfica: Muestra las señales originales, de voz y de ruido mediante gráficos.
6. Reproducción de archivos: Reproduce automáticamente las señales procesadas.

Diagrama de Flujo

1. Importar o grabar audio.
2. Leer y preprocesar la señal.
3. Aplicar FFT para analizar las frecuencias.
4. Separar las frecuencias de voz y ruido.
5. Identificar e imprimir la frecuencia dominante de la voz.
6. Reconstruir las señales mediante IFFT.
7. Guardar y reproducir las señales separadas.
8. Visualizar los resultados.

5. Cálculos y Resultados

Para realizar los cálculos aplique las funciones directa de FFT e IFFT que nos brinda la librería de Scipy aquí un ejemplo de que operaciones realizan estas funciones predertermiandas

Fórmulas utilizadas

- FFT: X(k)=∑n=0N−1x(n)e−j2πNknX(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}
- IFFT: x(n)=1N∑k=0N−1X(k)ej2πNknx(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn}

Resultados obtenidos

- La frecuencia dominante detectada en las pruebas realizadas fue de 158 Hz, indicando que la voz procesada pertenecía a un rango de voz humana normal.
- Los archivos generados (voz.wav y ruido.wav) reflejaron una separación clara entre la voz y el ruido.
- Los gráficos mostraron una representación precisa de la señal original y las componentes separadas.

6. Conclusiones y Recomendaciones

Conclusiones:

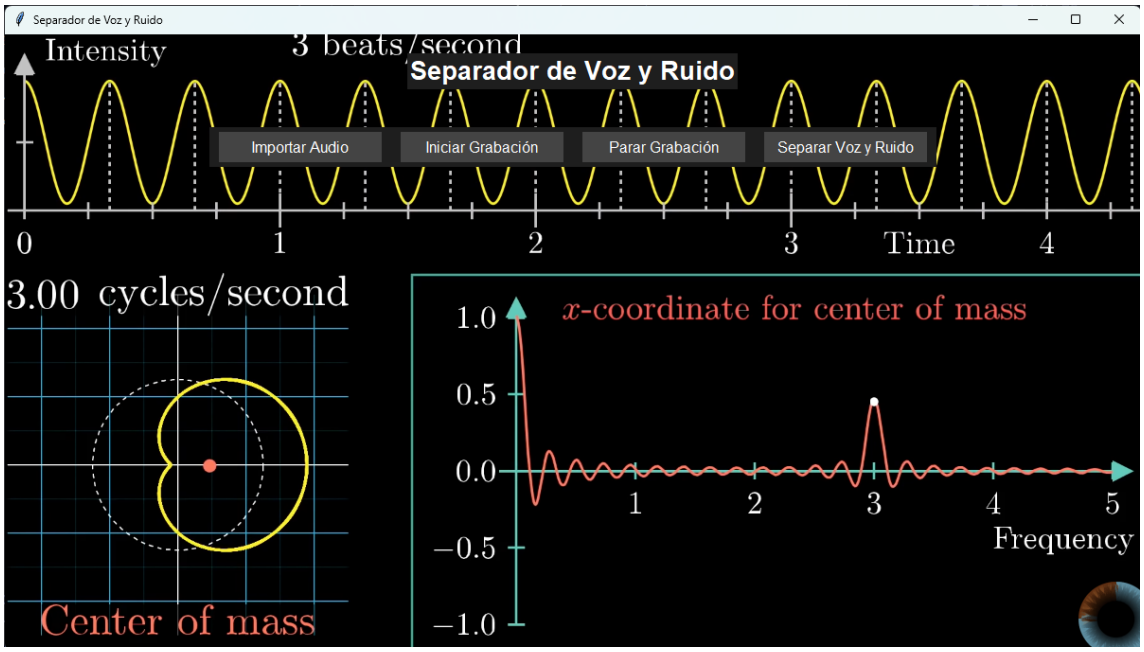
- El proyecto demuestra la efectividad de la FFT para analizar y separar señales de audio.
- La frecuencia dominante de la voz puede identificarse con precisión dentro del rango establecido.
- La interfaz gráfica facilita la interacción del usuario y la comprensión de los resultados.

Recomendaciones:

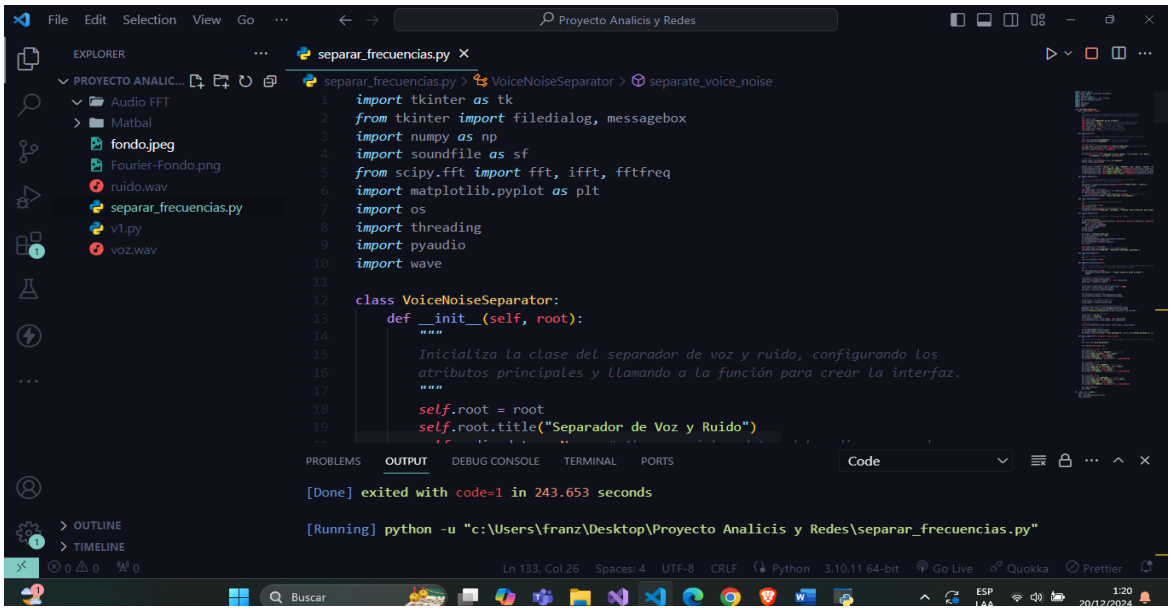
- Ampliar el rango de frecuencias si se desea procesar audios con voces más graves o agudas.
- Integrar filtros adicionales para mejorar la calidad de las señales separadas.
- Optimizar el procesamiento para manejar archivos de mayor duración o en formatos comprimidos.

7. Anexos

Interfaz de la aplicacion



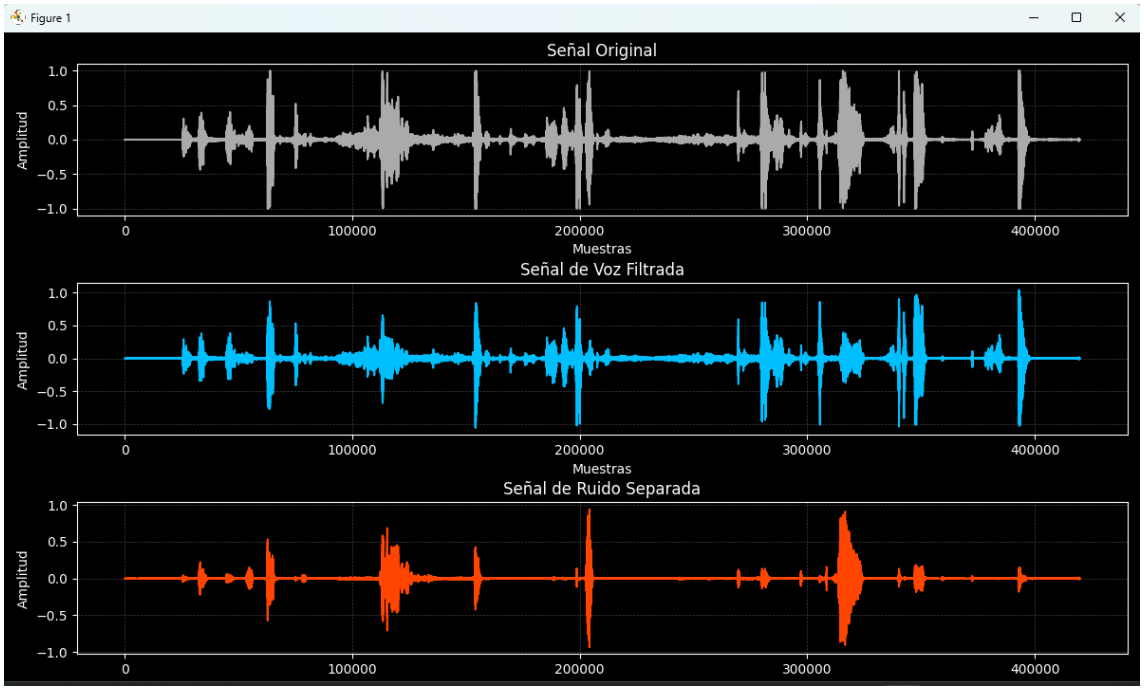
Código



Ejemplo de Resultados Visualizados

1. Gráfica de la señal original.
2. Gráfica de la señal de voz filtrada.
3. Gráfica del ruido separado.

(Capturas de pantalla de los gráficos generados por el programa).



8. Bibliografía

1. Oppenheim, A. V., & Schafer, R. W. (1999). "Discrete-Time Signal Processing". Prentice Hall.
2. Virtanen, P., et al. (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". Nature Methods.
3. Matplotlib Documentation. Recuperado de https://matplotlib.org/
4. Python Documentation. Recuperado de https://www.python.org/doc/