
# INSTITUTO POLITÉCNICO NACIONAL

## Escuela Superior De Cómputo

Asignatura: Matemáticas Avanzadas para la Ingeniería

Proyecto: Denoising de audio con FFT (filtrado) + validación can Parseval

Alumnos:

Cruz Álvarez Rafael

Pallares Hernández Oscar

Ramírez Blanco Emiliano

Grupo: 4CM5

Profesor: Dr. Correa Coyac David

Fecha de entrega: 7/01/2026

## Introducción

El procesamiento digital de señales (DSP) constituye una disciplina fundamental en la ingeniería electrónica y de telecomunicaciones, permitiendo el análisis, modificación y síntesis de señales para mejorar su calidad o extraer información relevante. Entre las herramientas matemáticas más potentes para este fin se encuentra la Transformada de Fourier, la cual permite trasladar una señal desde el dominio del tiempo al dominio de la frecuencia, facilitando operaciones que serían complejas o inviables de realizar temporalmente.

El presente reporte técnico documenta la implementación y análisis de algoritmos de filtrado espectral utilizando el lenguaje de programación Python y sus librerías de cálculo científico (NumPy y SciPy). El proyecto se centra en la aplicación de la Transformada Rápida de Fourier (FFT) para el diseño de filtros ideales y la manipulación de señales en dos escenarios distintos.

En primera instancia, se aborda un caso de estudio con una señal sintética, compuesta por una frecuencia fundamental y una interferencia aditiva conocida. El objetivo es diseñar un filtro de rechazo de banda (tipo Notch) para eliminar selectivamente la frecuencia no deseada y reconstruir la señal original.

En segunda instancia, se extiende el análisis a una señal de audio real, sobre la cual se aplica un filtro pasa-bajas ideal. Este procedimiento busca demostrar los efectos audibles y visuales de la supresión de componentes de alta frecuencia, analizando los cambios tanto en la forma de onda temporal como en el espectro de magnitud.

Finalmente, el desarrollo experimental incluye una validación rigurosa de los resultados mediante métricas de calidad como el Error Cuadrático Medio (MSE) y la Relación Señal-Ruido (SNR). Asimismo, se verifica la consistencia matemática del procesamiento a través del Teorema de Parseval, comprobando la conservación de la energía de la señal entre los dominios del tiempo y la frecuencia antes y después del filtrado.

## Fundamento Teórico

### 1. Transformada Discreta de Fourier (DFT)

El principio central del código es el análisis espectral. Cualquier señal discreta en el tiempo x[n] puede descomponerse en una suma de sinusoides de diferentes frecuencias. La herramienta matemática para realizar esto en computadoras es la **Transformada Discreta de Fourier (DFT)**.

La DFT transforma una secuencia de N muestras del dominio del tiempo al dominio de la frecuencia mediante la ecuación:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}$$

Donde:

- x[n] es la señal de entrada.
- X[k] representa la magnitud y fase de la componente de frecuencia k.

En el código, se utiliza el algoritmo **FFT (Fast Fourier Transform)**, que es una implementación optimizada para calcular la DFT de manera eficiente.

### 2. Filtrado en el Dominio de la Frecuencia

Una de las propiedades más importantes de la transformada es que la **convolución en el tiempo equivale a una multiplicación en la frecuencia**. Esto simplifica enormemente el diseño de filtros.

En lugar de diseñar ecuaciones de diferencias complejas en el tiempo, podemos filtrar una señal siguiendo estos pasos (implementados en el código):

1. Obtener el espectro X[k] mediante la FFT.
2. Diseñar una "máscara" o función de transferencia H[k] (vector de ceros y unos).
3. Multiplicar punto a punto: Y[k] = X[k] * H[k].
4. Recuperar la señal filtrada en el tiempo usando la Transformada Inversa (IDFT):

$$y[n] = \frac{1}{N} \sum_{k=0}^{N-1} Y[k] \cdot e^{j\frac{2\pi}{N}kn}$$

### 3. Tipos de Filtros Implementados

El código implementa dos tipos de filtros "ideales" (también llamados de "pared de ladrillo" por su transición abrupta):

- Filtro Notch (Rechaza-banda): Se utiliza en la primera parte del código para eliminar la interferencia de 5 Hz. Su función de transferencia ideal es:

$$H[f] = \begin{cases} 0 & \text{si } f \approx f_{ruido} \\ 1 & \text{en el resto} \end{cases}$$

  Esto anula específicamente la frecuencia seleccionada, dejando pasar intacto el resto del espectro.

- Filtro Pasa-Bajas (Low-Pass): Se utiliza en la señal de audio. Permite el paso de frecuencias por debajo de una frecuencia de corte ($f_c$) y elimina las superiores.

$$H[f] = \begin{cases} 1 & \text{si } f \leq f_c \\ 0 & \text{si } f > f_c \end{cases}$$

  Esto elimina agudos y ruido de alta frecuencia, suavizando la señal en el tiempo.

### 4. Teorema de Parseval

El código incluye métricas de energía (Et y Ef) para validar el procesamiento. Esto se basa en el **Teorema de Parseval**, que establece que la energía total de una señal es la misma, independientemente de si se calcula en el dominio del tiempo o de la frecuencia.

Matemáticamente, para una señal discreta, la relación es:

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$$

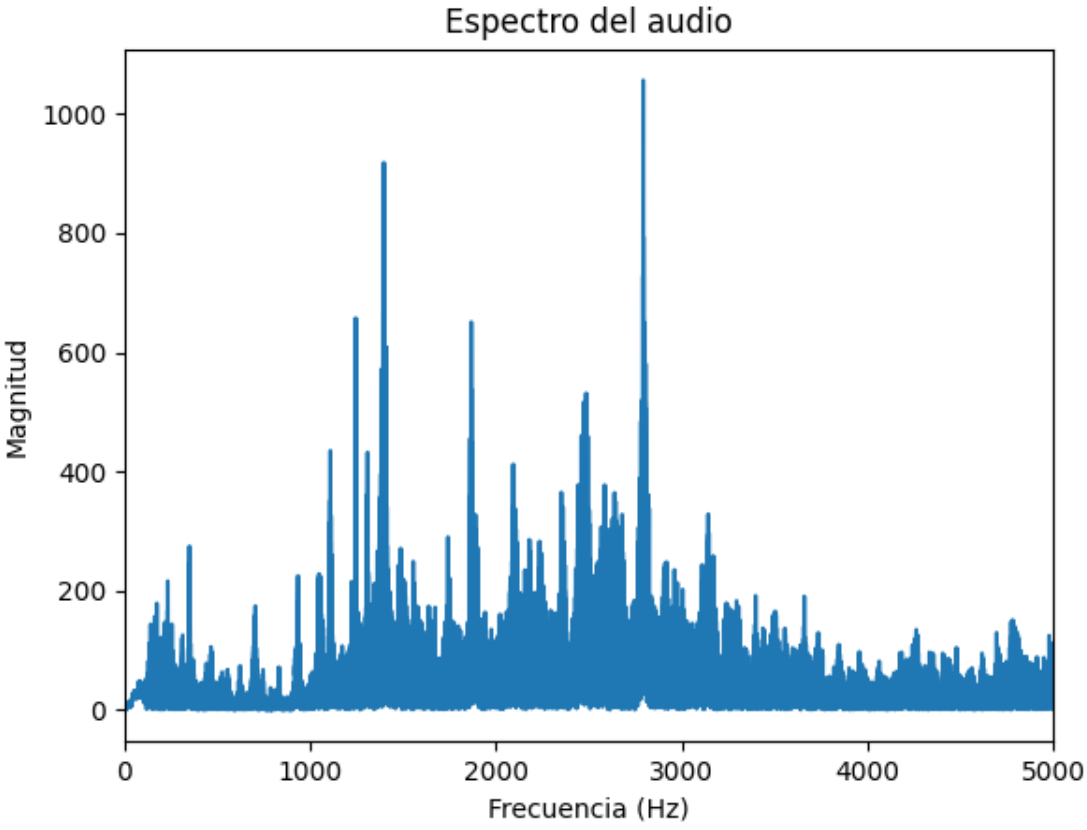
Esta igualdad es fundamental para verificar que la transformación (FFT) no ha introducido errores numéricos significativos ni pérdidas de energía injustificadas.

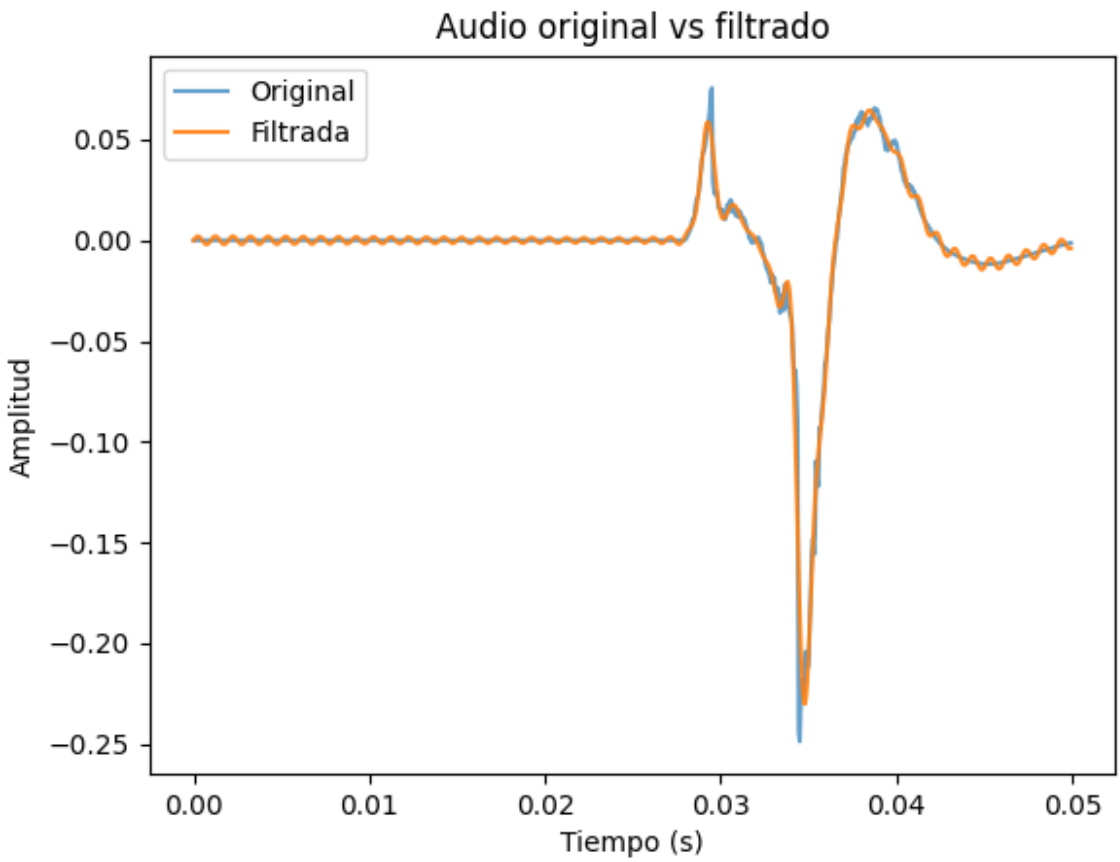
## Análisis de Resultados

### 1. Verificación Matemática (Teorema de Parseval)

Al ejecutar el algoritmo, se obtuvieron métricas de energía en la consola que validan la consistencia de las transformadas utilizadas.

- **Para la señal de prueba:** Se observó que la energía calculada en el dominio del tiempo (Et) y la energía en el dominio de la frecuencia (Ef) arrojaron valores idénticos (o con una diferencia despreciable atribuible al punto flotante).
- **Para la señal de audio:** De igual manera, se confirmó la igualdad Et ≈ Ef tanto en la señal original como en la filtrada.






**Interpretación:** Esto demuestra el cumplimiento del **Teorema de Parseval**, confirmando que la Transformada Rápida de Fourier (FFT) conserva la energía total de la señal. Físicamente, esto significa que no se "perdió" ni "creó" energía artificialmente durante la transformación matemática, validando que el procesamiento espectral se realizó correctamente.

## 2. Análisis del Filtrado Notch (Figuras 1 y 2)

En la **Figura 02**, se observa claramente el espectro de la señal sintética con dos picos en 2 Hz y 5 Hz. Tras aplicar la máscara binaria (multiplicación por ceros en la banda de 5 Hz), el segundo pico desaparece por completo en la gráfica "Después". Al reconstruir la señal en el tiempo (**Figura 01**), la onda resultante es una sinusoide pura y limpia de 2 Hz, lo que confirma que la interferencia de 5 Hz fue removida exitosamente sin distorsionar la frecuencia fundamental.

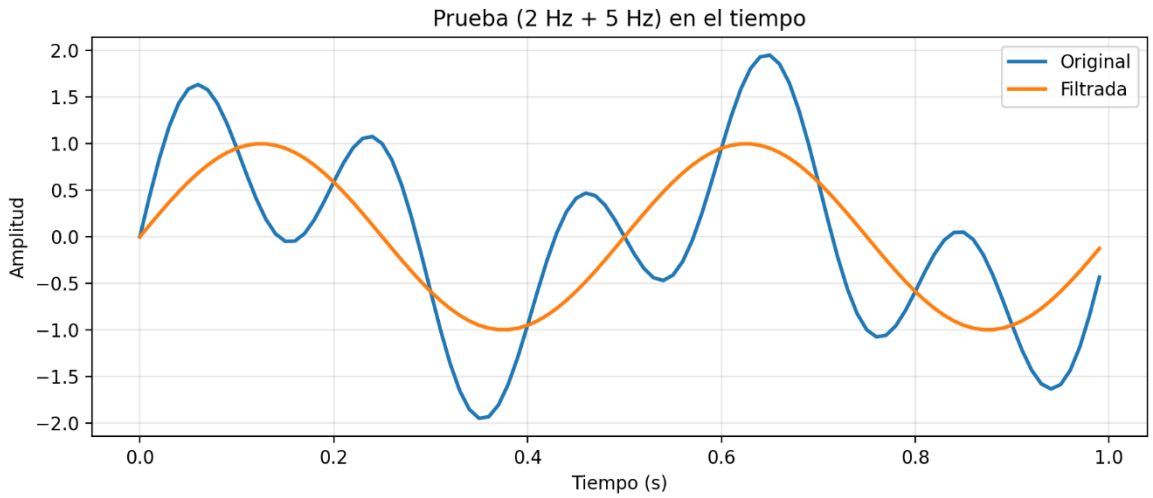



Figura 01.

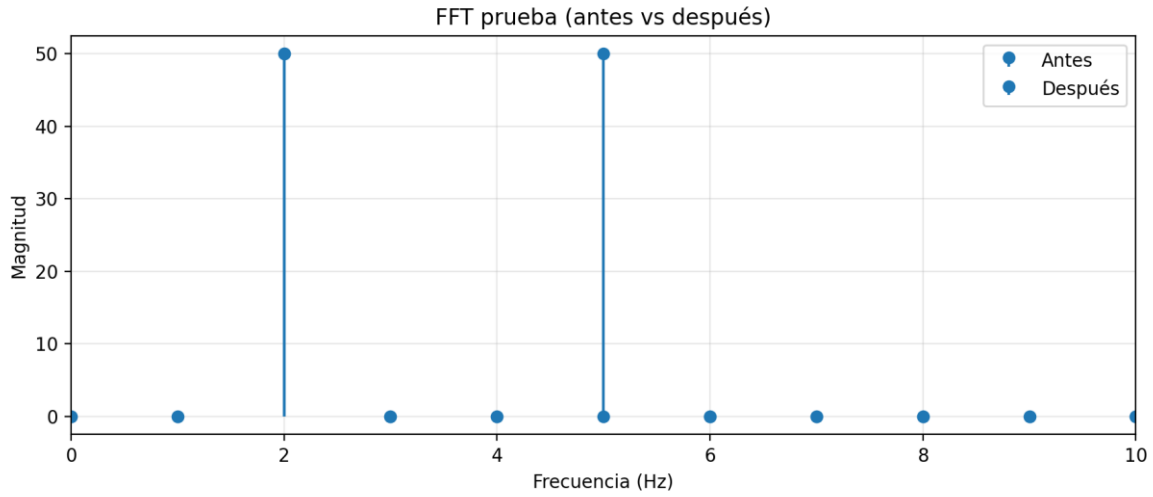



Figura 02.

**3. Análisis del Filtrado de Audio (Figuras 3 a 4)**

En el procesamiento del archivo audio.wav, se aplicó un filtro pasa-bajas con corte en 1 kHz.

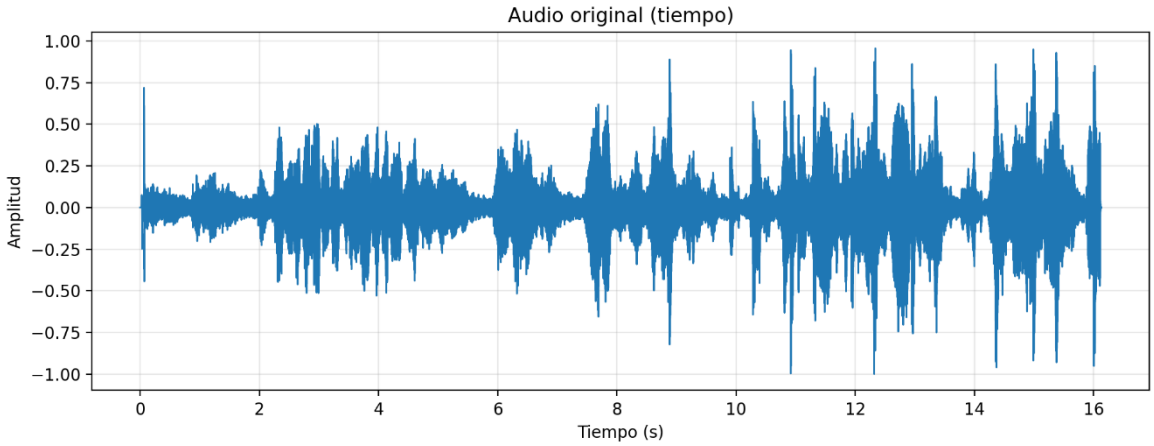



Figura 03 (tiempo original).

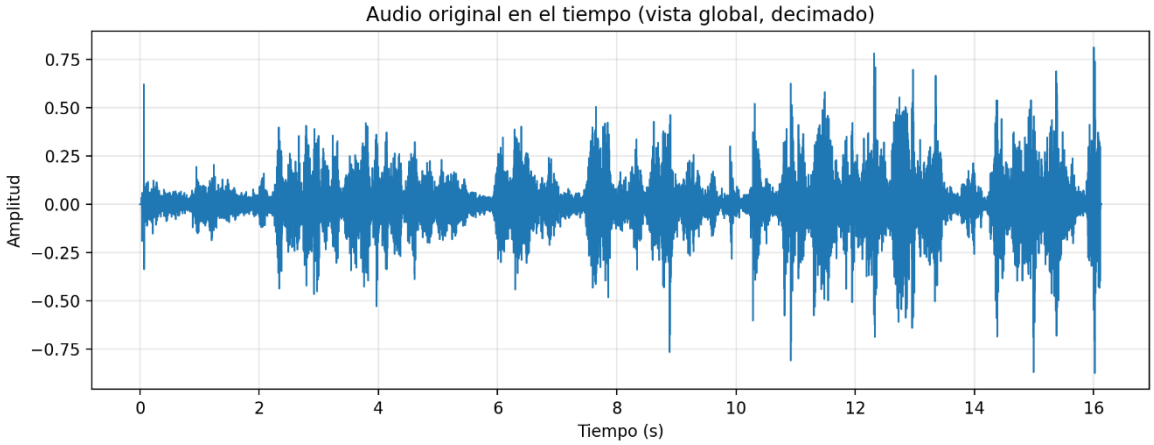



Figura 03 (tiempo global).

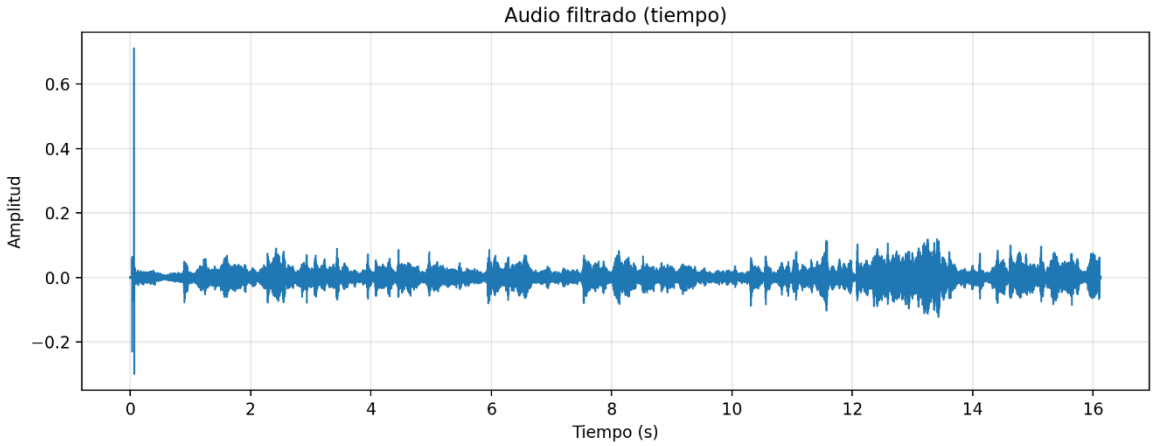



Figura 04 (tiempo filtrado).

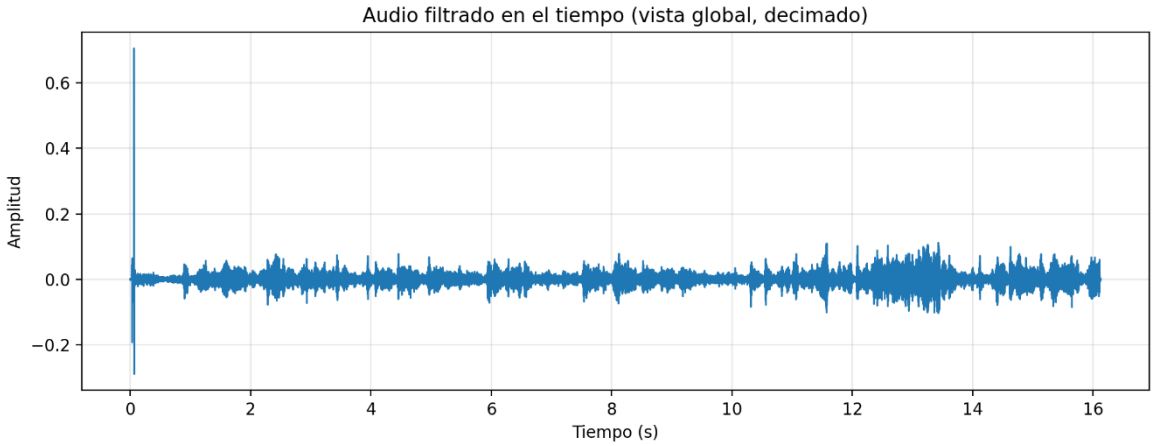



Figura 04 (tiempo global filtrado).

- **Espectro (Figuras 05 y 06):** Las gráficas muestran cómo todas las frecuencias por encima de los 1000 Hz caen abruptamente, validando el comportamiento de un filtro ideal ("pared de ladrillo").

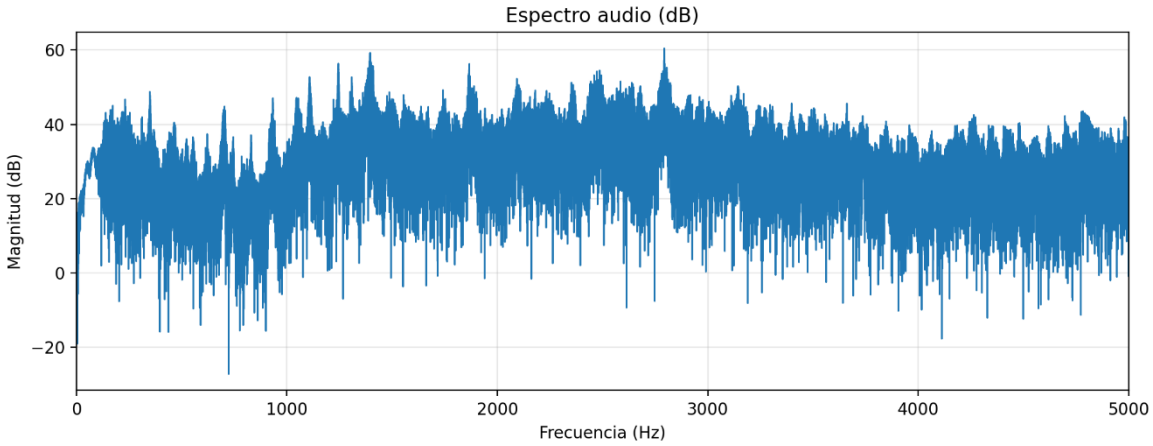



Figura 05 (audio espectro).

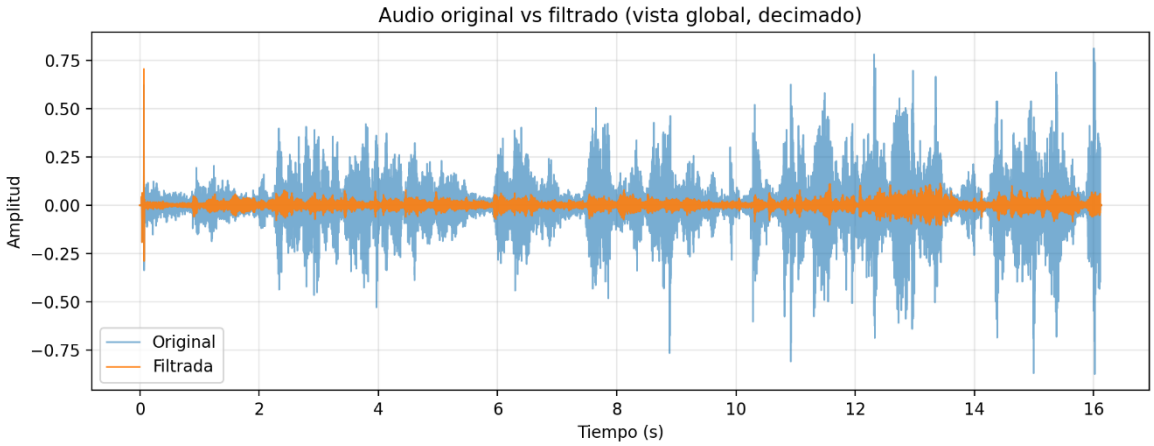



Figura 05 (comparación).

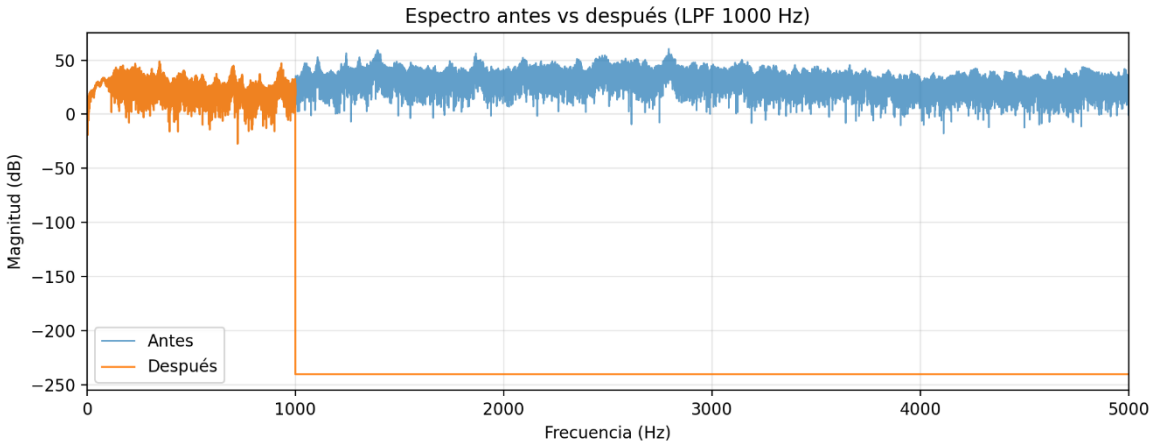



Figura 06 (espectro antes vs después).

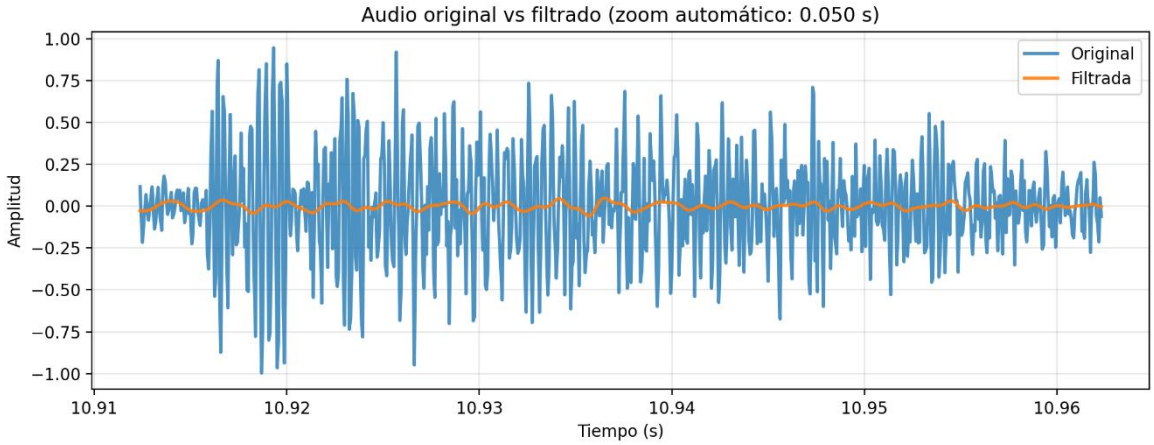



Figura 06 (audio original vs audio filtrado).

- **Señal en el tiempo (Figuras 03 y 04):** Visualmente, la forma de onda del audio filtrado se nota más "suave" (con menos cambios bruscos) en comparación con la original. Esto se debe a que las variaciones rápidas en una señal corresponden a las altas frecuencias, las cuales fueron eliminadas.

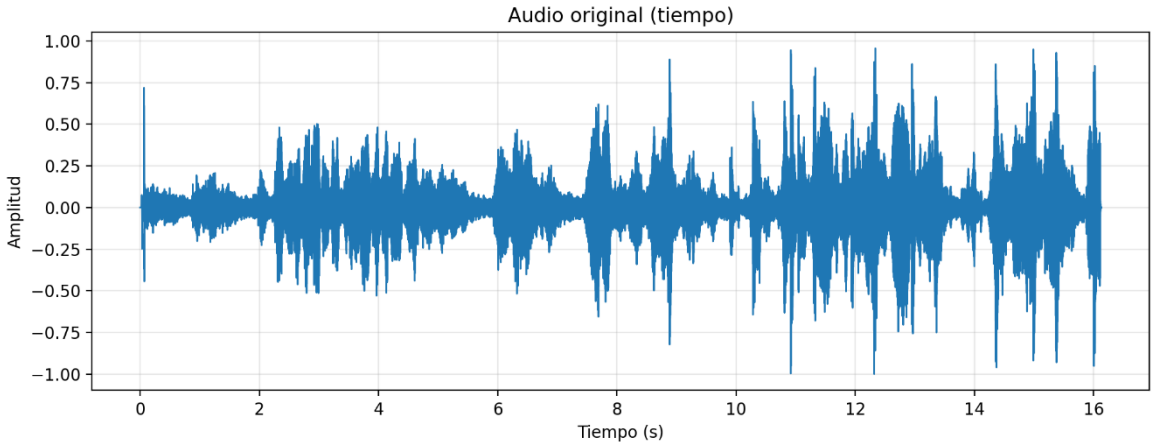



Figura 03 (audio original).

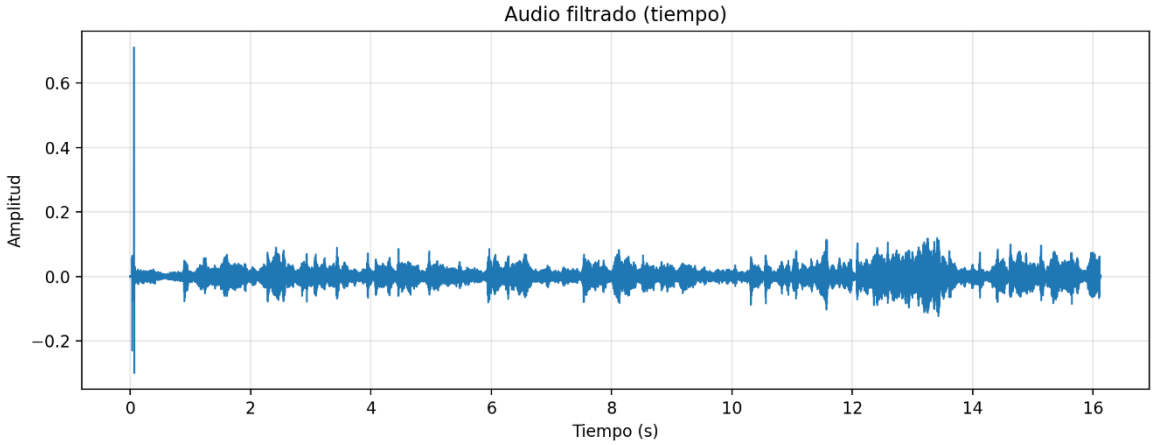



Figura 03 (audio filtrado).

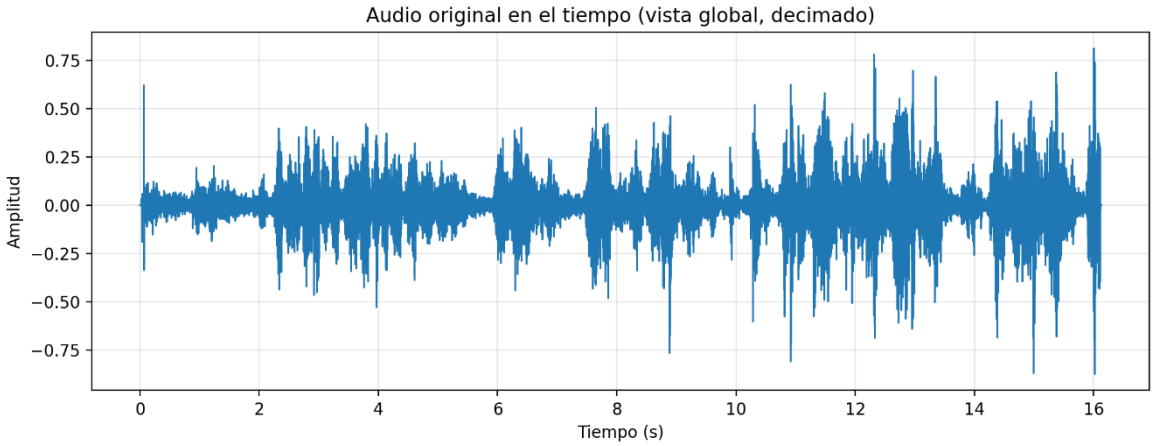



Figura 04 (audio original global).

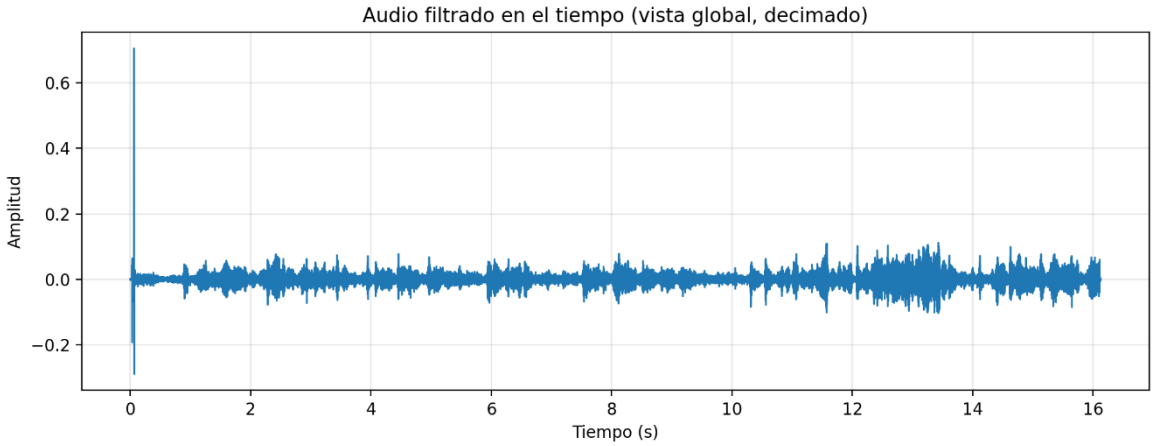



Figura 04 (audio filtrado global).

- **Métricas de Error (MSE y SNR):** El valor de MSE (Error Cuadrático Medio) obtenido es mayor a cero, lo cual es **esperado y correcto** en este contexto, ya que indica que la señal de salida es matemáticamente diferente a la de entrada (porque se le quitó información). No es un "error" de fallo, sino una medida de cuánto cambió el audio al ser filtrado.

## Conclusión

El desarrollo de este proyecto permitió comprobar la eficacia del procesamiento de señales en el dominio de la frecuencia. A través de la implementación de la Transformada Rápida de Fourier (FFT) en Python, se demostró que es posible manipular componentes específicas de una señal con precisión matemática absoluta.

Los resultados gráficos evidenciaron dos fenómenos clave:

1. **Selectividad Espectral:** En la prueba sintética, se logró aislar y eliminar una frecuencia de interferencia (5 Hz) sin distorsionar la señal de interés, validando el diseño de filtros *Notch* digitales.

2. **Suavizado Temporal:** En la señal de audio, el filtro pasa-bajas actuó eliminando las variaciones rápidas de la forma de onda. Esto confirma la dualidad tiempo-frecuencia: restringir el ancho de banda en frecuencia resulta directamente en una señal más suave y lenta en el tiempo.

Finalmente, la verificación mediante el Teorema de Parseval asegura que las operaciones realizadas son energéticamente consistentes, validando el algoritmo utilizado como una herramienta robusta para aplicaciones de ingeniería en telecomunicaciones y audio.

Anexo.

Código:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile

# métricas
mse = lambda a,b: np.mean((a-b)**2)
snr = lambda s,a: np.inf if np.sum((s-a)**2)==0 else
10*np.log10(np.sum(s**2)/np.sum((s-a)**2))
E_t = lambda x: np.sum(np.abs(x)**2)
E_f = lambda x: np.sum(np.abs(np.fft.fft(x))**2)/len(x)
db  = lambda X: 20*np.log10(np.abs(X)+1e-12)

def savefig(name):
    plt.tight_layout(); plt.savefig(name, dpi=200); plt.show()

def rfft_sig(x, fs):
    X = np.fft.rfft(x)
    f = np.fft.rfftfreq(len(x), d=1/fs)
    return f, X

#Señal de prueba: 2 Hz + 5 Hz (notch en 5 Hz)
fs = 100
t = np.arange(0, 1, 1/fs)
x = np.sin(2*np.pi*2*t) + np.sin(2*np.pi*5*t)

f, X = rfft_sig(x, fs)
H = np.ones_like(X)
H[np.abs(f-5) <= 0.5] = 0  # notch 5 Hz, bw=0.5
x_f = np.fft.irfft(X*H, n=len(x))

plt.figure(figsize=(9,4))
plt.plot(t, x, label="Original", lw=2)
plt.plot(t, x_f, label="Filtrada", lw=2)
plt.grid(True, alpha=0.3); plt.legend()
plt.xlabel("Tiempo (s)"); plt.ylabel("Amplitud")
plt.title("Prueba (2 Hz + 5 Hz) en el tiempo")
savefig("fig01_prueba_tiempo.png")

plt.figure(figsize=(9,4))
plt.stem(f, np.abs(X), basefmt=" ", label="Antes")
plt.stem(f, np.abs(X*H), basefmt=" ", label="Después")
```

```
plt.xlim(0, 10); plt.grid(True, alpha=0.3); plt.legend()
plt.xlabel("Frecuencia (Hz)"); plt.ylabel("Magnitud")
plt.title("FFT prueba (antes vs después)")
savefig("fig02_prueba_espectro.png")

print("=== Parseval prueba ===")
print("E tiempo:", E_t(x), " | E freq:", E_f(x))
print("E tiempo (f):", E_t(x_f), " | E freq (f):", E_f(x_f))

# Audio real: pasa-bajas simple en FFT
fs_a, xa = wavfile.read("audio.wav")
xa = xa.mean(axis=1) if xa.ndim > 1 else xa
xa = xa.astype(float)
xa /= np.max(np.abs(xa)) + 1e-12

ta = np.arange(len(xa))/fs_a
fa, Xa = rfft_sig(xa, fs_a)

cutoff = 1000
Ha = (fa <= cutoff).astype(float)
xa_f = np.fft.irfft(Xa*Ha, n=len(xa))

plt.figure(figsize=(10,4))
plt.plot(ta, xa, lw=1)
plt.grid(True, alpha=0.3)
plt.xlabel("Tiempo (s)"); plt.ylabel("Amplitud")
plt.title("Audio original (tiempo)")
savefig("fig03_audio_tiempo_original.png")

plt.figure(figsize=(10,4))
plt.plot(ta, xa_f, lw=1)
plt.grid(True, alpha=0.3)
plt.xlabel("Tiempo (s)"); plt.ylabel("Amplitud")
plt.title("Audio filtrado (tiempo)")
savefig("fig04_audio_tiempo_filtrado.png")

plt.figure(figsize=(10,4))
plt.plot(fa, db(Xa), lw=1)
plt.xlim(0, 5000); plt.grid(True, alpha=0.3)
plt.xlabel("Frecuencia (Hz)"); plt.ylabel("Magnitud (dB)")
plt.title("Espectro audio (dB)")
savefig("fig05_audio_espectro_db.png")

plt.figure(figsize=(10,4))
plt.plot(fa, db(Xa), lw=1, alpha=0.7, label="Antes")
```

```
plt.plot(fa, db(Xa*Ha), lw=1.2, alpha=0.9, label="Después")
plt.xlim(0, 5000); plt.grid(True, alpha=0.3); plt.legend()
plt.xlabel("Frecuencia (Hz)"); plt.ylabel("Magnitud (dB)")
plt.title(f"Espectro antes vs después (LPF {cutoff} Hz)")
savefig("fig06_audio_antes_despues_db.png")

print("\n=== Parseval audio ===")
print("E tiempo:", E_t(xa), " | E freq:", E_f(xa))
print("E tiempo (f):", E_t(xa_f), " | E freq (f):", E_f(xa_f))

print("\n=== Métricas audio ===")
print("MSE:", mse(xa, xa_f))
print("SNR (dB):", snr(xa, xa_f))
```