CIAA

Dado el segmento de audio almacenado en el archivo clases/tp2/chapu_noise.npy con Fs=8000, mono de 16b y contaminado con ruido de alta frecuencia:

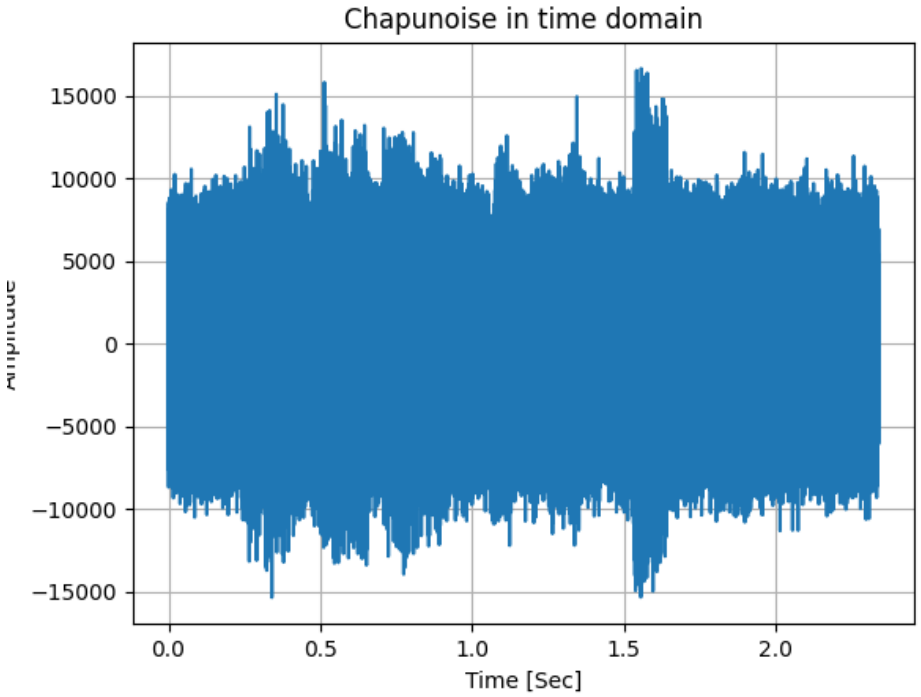
1) Diseñe un filtro que mitigue el efecto del ruido y permita percibir mejor la señal de interés.

Para diseñar el filtro inicialmente se realiza el análisis de la señal utilizando Python.

```
import numpy as np
import matplotlib.pyplot as plt

class return_values_FFT:
    def __init__(self, fft,fft_fs):
        self.fft = fft
        self.fft_fs = fft_fs

def FFT(x,fs):
    fft = np.abs(np.fft.fftshift(np.fft.fft(x)/len(x)))
    fft_fs = np.fft.fftshift(np.fft.fftfreq(len(x), 1/fs))
    y = return_values_FFT(fft,fft_fs)
    return y

fs = 8000
signal = np.load('chapu_noise.npy')
time = np.arange(len(signal)) * 1 / fs

plt.figure(1)
plt.plot(time, signal)
plt.grid()
plt.ylabel('Amplitude')
plt.xlabel('Time [Sec]')
plt.title('Chapunoise in time domain')

fft = FFT(signal, fs)
plt.figure(2)
plt.xlabel('Frequencies [Hz]')
plt.ylabel('Magnitude')
plt.plot(fft.fft_fs, fft.fft)

plt.figure(3)
plt.title('Chapunoise Spectrogram')
plt.specgram(signal, Fs=fs, cmap="rainbow")
plt.xlabel('Time [Sec]')
plt.ylabel('Frequency [Hz]')

## Validación del filtro relizado en

coe_filter = np.array(np.load('filtro.npy').astype(float))
coe_filter = np.ravel(coe_filter.T)
filter = np.convolve(coe_filter,signal)

plt.figure(4)
plt.plot(filter)
```

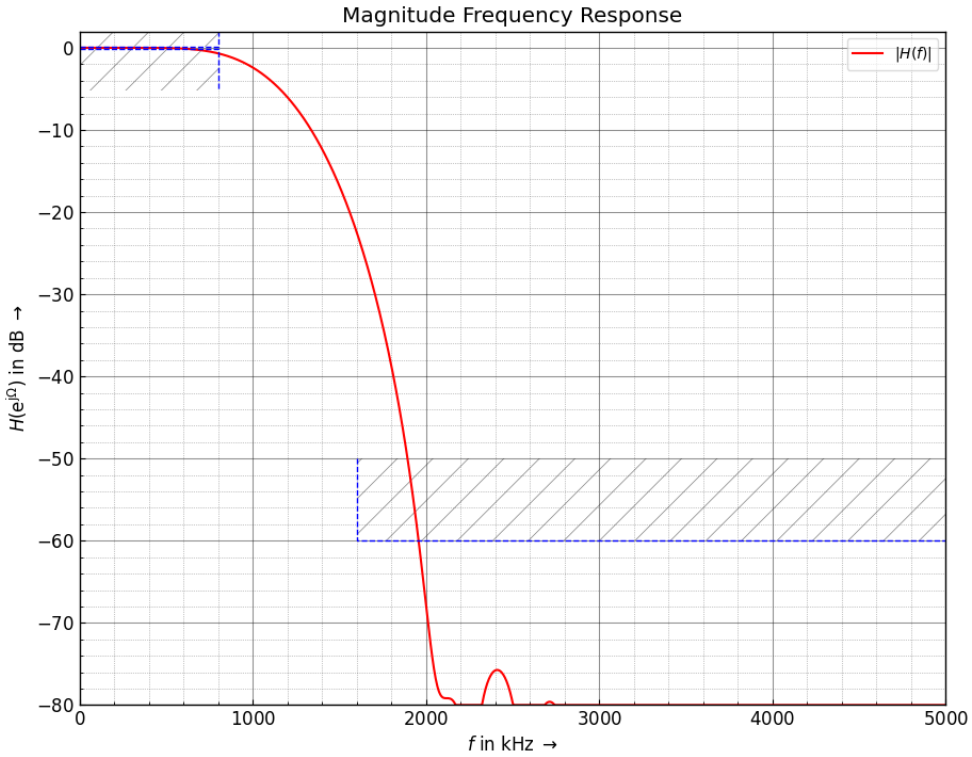
```
plt.grid()
plt.ylabel('Amplitude')
plt.xlabel('Time [Sec]')
plt.title('Chapunoise in time domain')

fft = FFT(filter, fs)
plt.figure(5)
plt.xlabel('Frequencies [Hz]')
plt.ylabel('Magnitude')
plt.plot(fft.fft_fs, fft.fft)

plt.figure(6)
plt.title('Chapunoise Spectrogram')
plt.specgram(filter, Fs=fs, cmap="rainbow")
plt.xlabel('Time [Sec]')
plt.ylabel('Frequency [Hz]')
plt.show()
```

Señal de entrada.



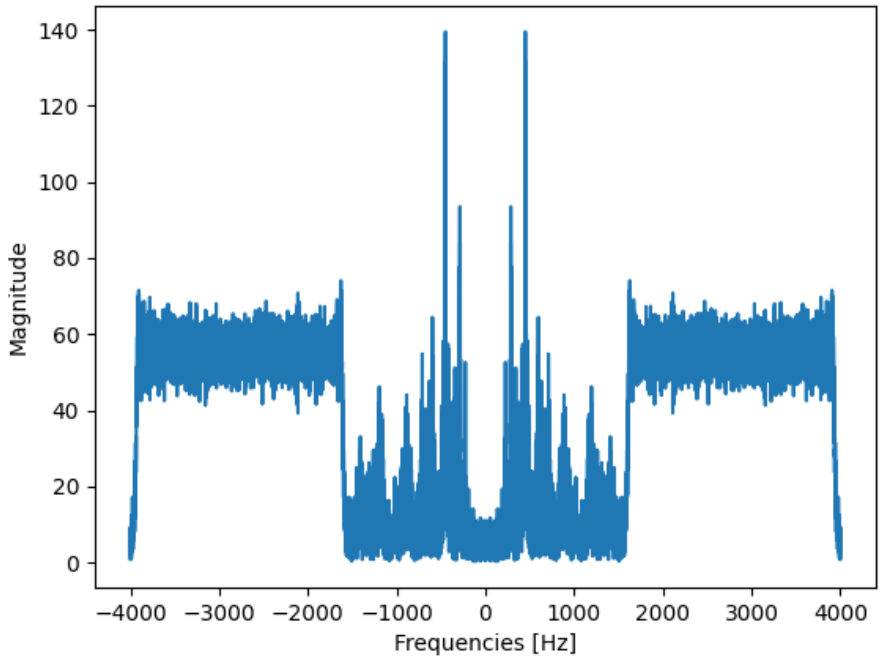
2) Filtre con la CIAA utilizando alguna de las técnicas vistas

Diseño del filtro.

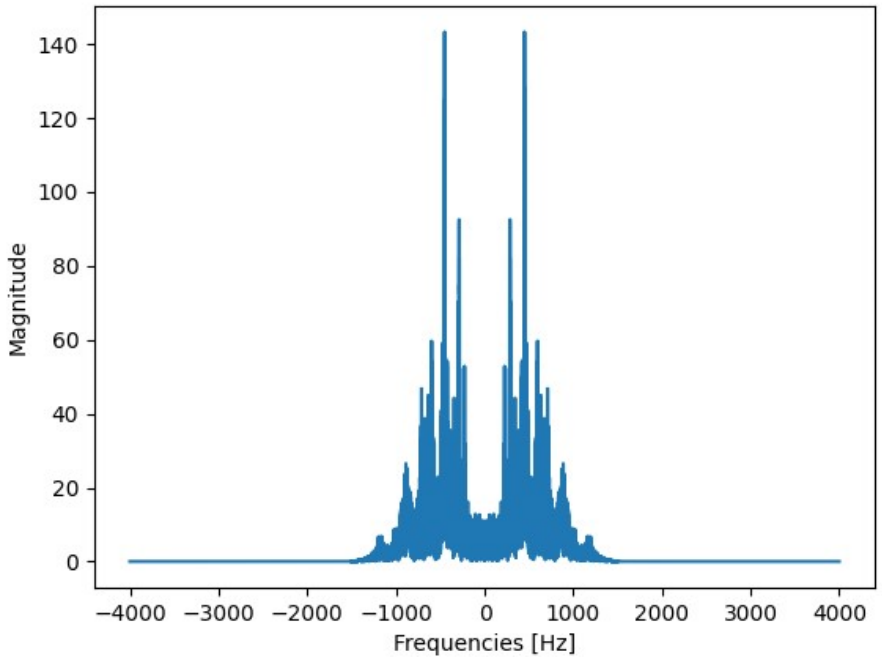


3) Grafique el espectro antes y después del filtro.

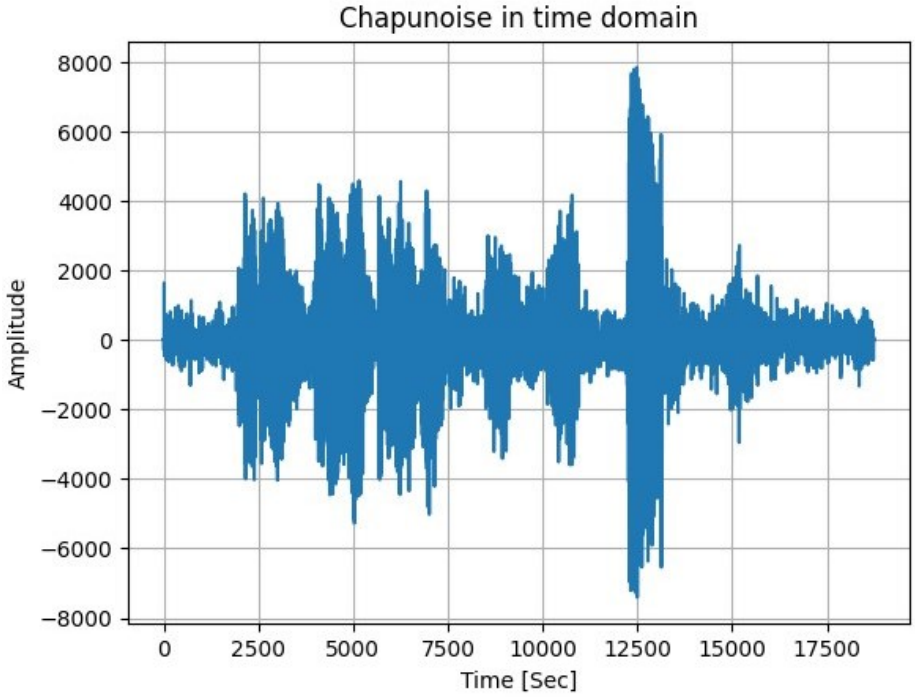
Espectro señal original.



Espectro señal filtrada.



Señal filtrada



4) Reproduzca el audio antes y después del filtro

```
import sounddevice as sd

sd.default.samplerate = 44100
wav_wave = np.load('chapu_noise.npy')

while True:
    sd.play(wav_wave, blocking=True)
    sd.play(filter, blocking=True)
```

5) Pegue el link a un .zip comentando los resultados y los criterios utilizados, la plantilla del filtro con capturas de la herramienta de diseño y un video mostrando la CIAA/HW en acción y la reproducción de audio antes y después del filtrado.

Funcionamiento en la NUCLEO-H7A3ZI-Q