Tarea 1:

¿Cada cuánto se procesa en término medio un bloque de muestras cuando se solicita una frecuencia de muestreo de 8000 Hz? ¿Cuál es el tiempo teórico que debería tardar?

-1

* El tiempo teórico que debería tardar es el siguiente: cada bloque que leemos tiene 2048 muestras, si la frecuencia de muestreo son 8000 quiere decir que lee 8000 muestras por cada segundo, esto implica que 2048/8000 = 0.256s , es decir 256.000 µs en promedio.
* El tiempo real:



el tiempo real promedio en procesar cada bloque fue de 256.020,6 µs, extremadamente similar que el tiempo teórico promedio.

- 2

Para 16khz de sample\_rate:

T\_teórico = 2048/16000 = 0,128s / 128.000 µs

T\_real = 

Tiempo real: 127.992,2 µs

Para 48khz de sample\_rate:

T\_teórico = 2048 / 48000 = 0,04266667s / 42.666,67 µs

T\_real = 

Tiempo real: 42.660,96 µs.

Evidentemente a mayores muestras leídas por segundos, menos será el tiempo de procesado de cada bloque, porque tardará menos en leerse cada muestra de los bloques.

Tarea 2:

**-1. Captura a 8000 Hz 2 tonos, uno con frecuencia 700 Hz y otro con frecuencia 5500 Hz. Escucha el fichero generado por capture. ¿Se oyen los dos tonos?:**

Se escuchan bajo, pero se escuchan perfectamente, de hecho, si vas a Audacity, analizar y trazas el espectro ves como hay enormes picos en los hz de esas señales.

**-2. Abre el fichero resultante en Audacity y muestra el espectro (Analyze → Plot Spectrum…). Configura el eje x en modo lineal en el desplegable Axis → Linear frequency. ¿Qué se observa? ¿Por qué ocurre esto?**

**A screen shot of a graph

AI-generated content may be incorrect.**

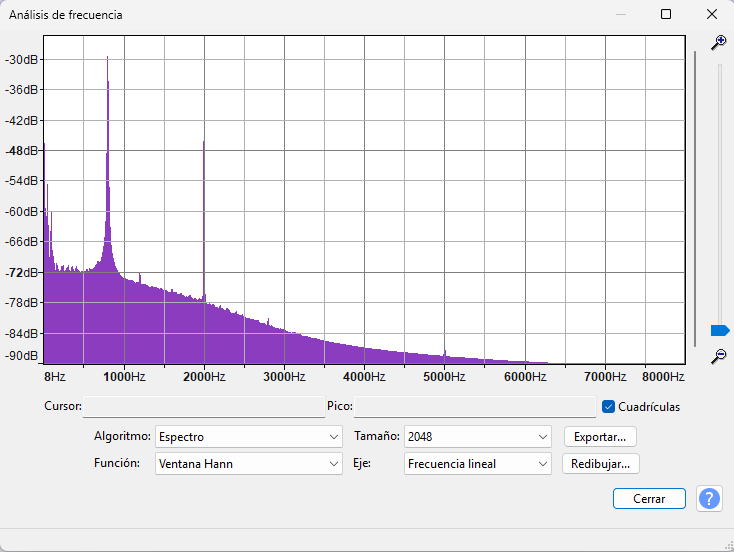
Pues como se dijo antes, se observan dos picos en las frecuencias que creamos desde

<https://onlinetonegenerator.com/multiple-tone-generator.html>. El gráfico anterior es bidimensional donde el eje de las abscisas corresponde a las frecuencias y el eje de las ordenadas a los dB, según chatgpt, los dB son negativos porque Audacity usa una escala logarítmica relativa donde 0 es el nivel máximo que puede tener un archivo digital.

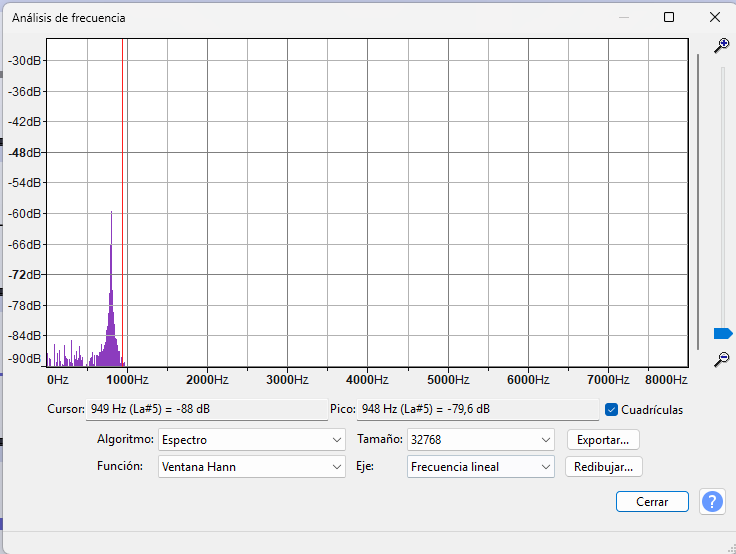
**Tarea 3**

**-1Diseña un filtro de L = 10 taps que elimine tonos por encima de 1000 Hz (filtro paso bajo) cuando son muestreados a 16000 Hz. Captura a esta frecuencia una señal compuesta por 3 tonos, uno de 800 Hz, otro de 2000 Hz y otro de 5000 Hz y pásala por el filtro. Muestra el espectro de la señal resultante en Audacity y explica el resultado.**

Con tamaño 2048 muestras:



Con tamaño 32768 muestras:



La FFT (Fast Fourier Transformation) permite pasar del dominio del tiempo al dominio de la frecuencia, este resultado es lo que vemos en pantalla. Pero a menor muestras menor precisión con la que se realiza este cálculo, por tanto, para evaluar la efectividad del filtro, debemos usar las mayores muestras que nos sea posible. Y como vemos, con un tamaño de 32768 muestras podemos confirmar que el filtro funciona, solo deja pasar frecuencias inferiores a 1000hz.

**-2. Añade medidas de tiempos: t2 tras el filtrado y t3 tras el envío, y pásalas junto a t0 y t1 también a print\_partial(t0, t1, t2, t3). Diseña filtros de tamaños cada vez mayores (L = 15, 20, 25, etc.). ¿Cómo cambian los tiempos? ¿Hasta qué tamaño de filtro le da tiempo a procesar los bloques sin perder muestras?**

L=15:



L = 20:



L = 25:



En general podemos observar que el tiempo promedio de procesado de cada bloque no cambia demasiado, sin embargo, lo que sí cambia con una tendencia ascendente directamente proporcional a L es el tiempo de procesado, tiene todo el sentido porque L son los coeficientes de la respuesta al impulso o del filtro. A mayores coeficientes tenga el filtro más tiempo tardará en desarrollarse la convolución resultando en un tiempo de procesado mayor.

--- En16khz el tiempo teórico (aunque ya sabemos que se asemeja mucho al real) que demora en escribir un bloque entero en el buffer interno son 2048/16000 0,128s / 128.000 µs. Eso quiere decir que si nuestro tiempo de bloque (el tiempo de procesado que incluye la lectura/captura en al buffer de procesamiento, el de procesado que incluye el escalado y la convolución y el de envío) es mayor, estaremos perdiendo muestras, por pocas que sean, si es considerablemente mayor en el audio que grabamos incluso escucharemos sonidos entrecortados (para esta tarea se escuchan esos sonidos aunque no se pierdan muestras por este motivo a raíz de la convolución, el por qué se explicará en la tarea 4. Pero suponiendo que no perdemos muestras por convolución, escucharíamos sonidos entre cortados si el tiempo de procesado de bloque es considerablemente mayor que el de escritura del buffer interno). Como dice el PDF hay que buscar que el tiempo de bloque sea inferior que el de escritura teórico/real, de esta manera simplemente se quedará esperando a que esté listo el nuevo bloque en readinto antes de capturarlo en el otro buffer. Ahora bien, para la implementación desarrollada, tanto en L=15, como L=20, como L=25 se pierden muestras, por pocas que sean, para el momento en que se escribió esto, la terea 4 está finalizada, por lo que probé ejecutar con ese código y la frecuencia de muestreo a 16khz esos taps a ver si se escuchan sonidos entre cortados (grabando un sin de 800hz) y ver si en la práctica la pérdida es grave, en general, en ninguno de los tres se escucharon estos sonidos pero si usas L=50 o 40 la cosa cambia, ahí sí se escuchan. He probado con L=5 y devuelve 127878.1 µs. en promedio, así que en rigor se puede decir que con esta implementación L=5 es el máximo tamaño del filtro para no perder muestras a 16khz

**-3Muestra capturas del espectro de la señal filtrada con 10 taps, 20 taps y 30 taps. ¿Qué efecto tiene en la señal de salida aumentar el tamaño del filtro?**

A screenshot of a graph

AI-generated content may be incorrect.

A graph with purple lines

AI-generated content may be incorrect.

A screen shot of a graph

AI-generated content may be incorrect.

Tenemos que disminuir el tamaño para poder apreciar mejor los cambios. Como podemos ver al aumentar el número de taps o de coeficientes en el filtro podemos hacer que este sea más eficiente a costa de un mayor tiempo de procesado.

**4. ¿Puedes construir un filtro que deje pasar un tono de 800 Hz y elimine completamente uno de 1200 Hz? ¿Por qué?**

No acabo de entender la pregunta, es decir, un filtro paso bajo de 1000Hz deja pasar una frecuencia de 800hz y elimina una de 1200hz, pero eso ya está hecho.

Tras hacer un poco de research y probando, ni el filtro banda ancha ni rechazo banda sirven. Aun aumentando los coeficientes hasta el máximo que permite el código y en consecuencia teniendo pérdidas de muestras (a 63), no es posible eliminar por completo la frecuencia de 1200hz, a lo sumo atenuarla, pero eliminarla, al menos con un filtro FIR que usa convolución no. Esto es así porque este filtro elimina/atenúa intervalos de frecuencias, no frecuencias específicas, pero parece que para eliminar muestras específicas y dejar intactas las demás se puede usar scipy.signal.iirnotch

Su construcción sería de la siguiente manera:

b, a = iirnotch(1200, 30, 16000) # Coeficientes

filtered\_signal = lfilter(b, a, signal)

**Tarea 4**



**1. Genera un tono de 800 Hz y pásalo por el filtro paso bajo de 30 taps del apartado anterior, con la convolución sin corregir y corregida. ¿Escuchas alguna diferencia? Muestra una captura de ambas señales de salida en tiempo donde se vea el límite entre dos bloques y explica la diferencia.**



A screenshot of a computer

AI-generated content may be incorrect.



Desde luego se escucha una diferencia, en la primera se escucha un ruido entrecortado mientras que en la segunda se escucha la señal continua. Esto ocurre porque la convolución, por definición matemática de la operación, necesita los L-1 elementos anteriores donde L es el cardinal del conjunto de los coeficientes de la señal, sino la convolución para los valores anteriores los tomará como 0, entonces si taps es 30 por ej, el primer valor de la convolución para el nuevo bloque será 29 ceros + un valor distinto de cero, básicamente tiende a cero, para el siguiente serán 28 ceros, luego 27…. Y así, por eso vemos en la primera imagen “la no corregida” que aunque los valores tienden a cero, a medida que avanza el tiempo la amplitud sube muy muy poco, porque cada vez hay menos ceros hasta que volvemos a trabajar por completo con nuestra señal.

**Tarea 5**

