**Integrantes: Ciccio Ernesto José**

**Geréz Jiménez Juan José Armando**

**Robles Héctor**

**Torres Juan Santiago Simón**

1. a) Para obtener los nuevos audios con la nueva frecuencia de muestreo de 328KHz escribimos el siguiente código utilizando la función “resample”:

[audio, Fs]= audioread('hector8k.wav');

Fs\_new=328000;

audio\_fsnew = resample(audio, Fs\_new, Fs);

audiowrite('hector328k.wav',audio\_fsnew, Fs\_new);

[audio, Fs]= audioread('santi8k.wav');

Fs\_new=328000;

audio\_fsnew = resample(audio, Fs\_new, Fs);

audiowrite('santi328k.wav',audio\_fsnew, Fs\_new);

[audio, Fs]= audioread('juanjo8k.wav');

Fs\_new=328000;

audio\_fsnew = resample(audio, Fs\_new, Fs);

audiowrite('juanjo328k.wav',audio\_fsnew, Fs\_new);

[audio, Fs]= audioread('ernesto8k.wav');

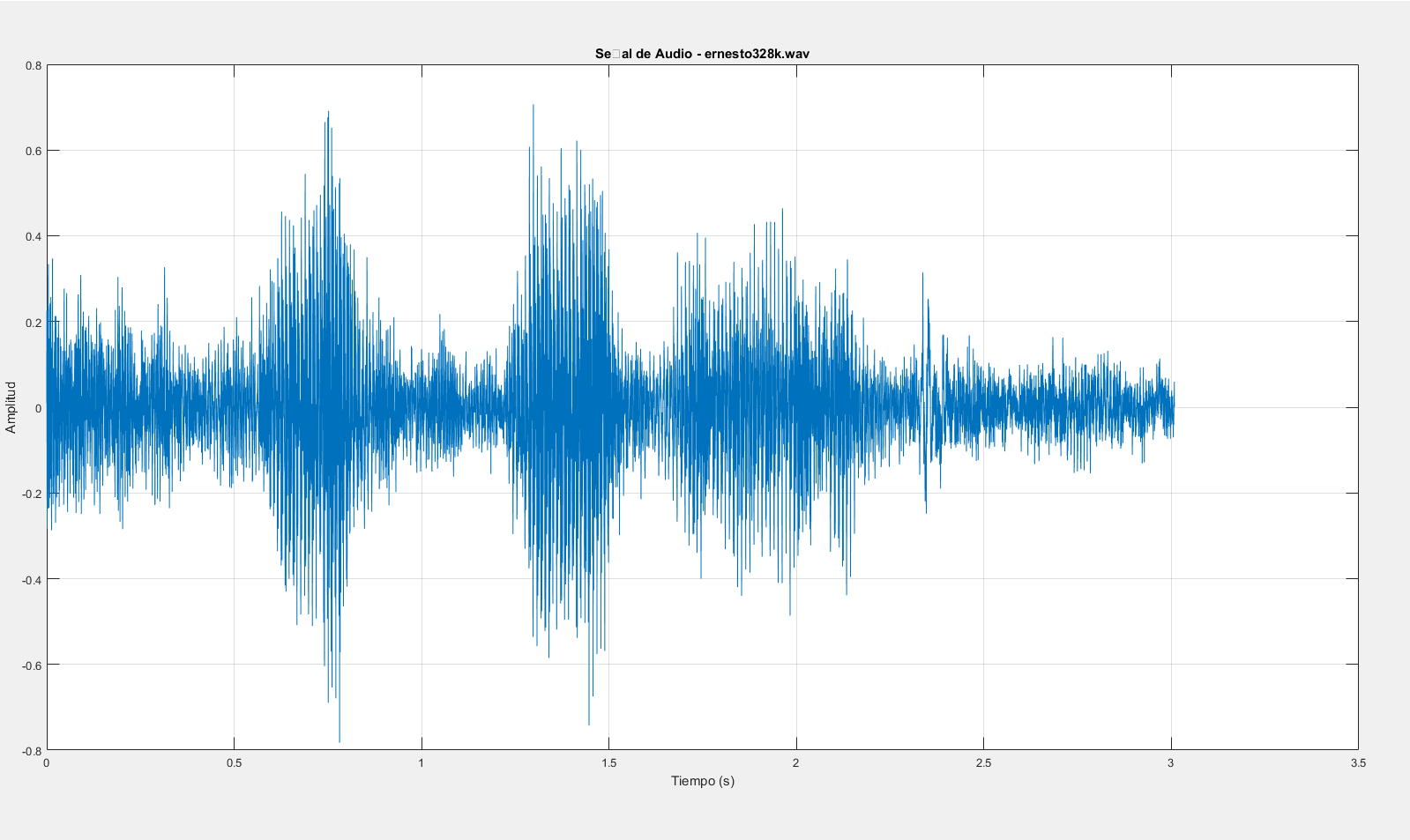
Fs\_new=328000;

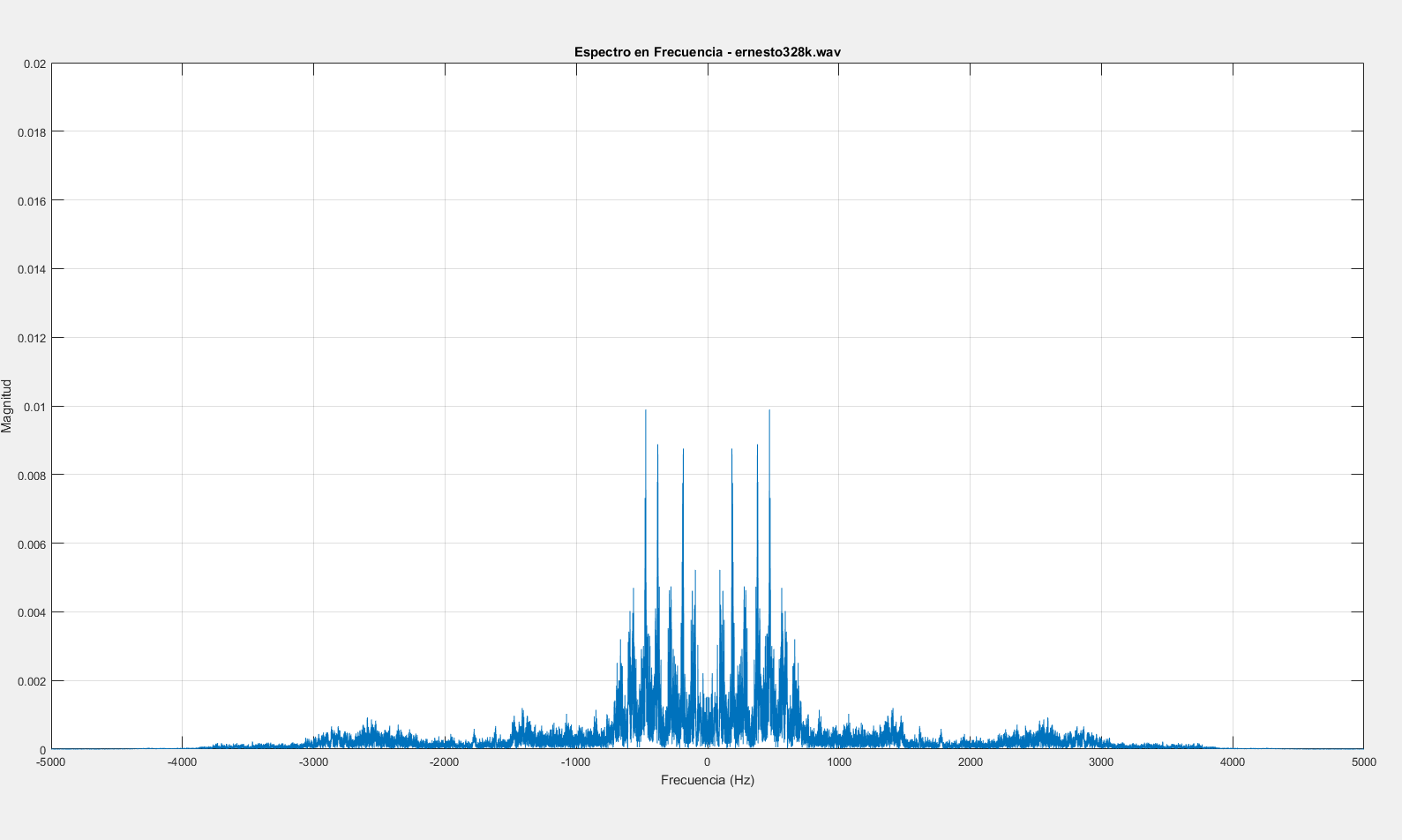
audio\_fsnew = resample(audio, Fs\_new, Fs);

audiowrite('ernesto328k.wav',audio\_fsnew, Fs\_new);

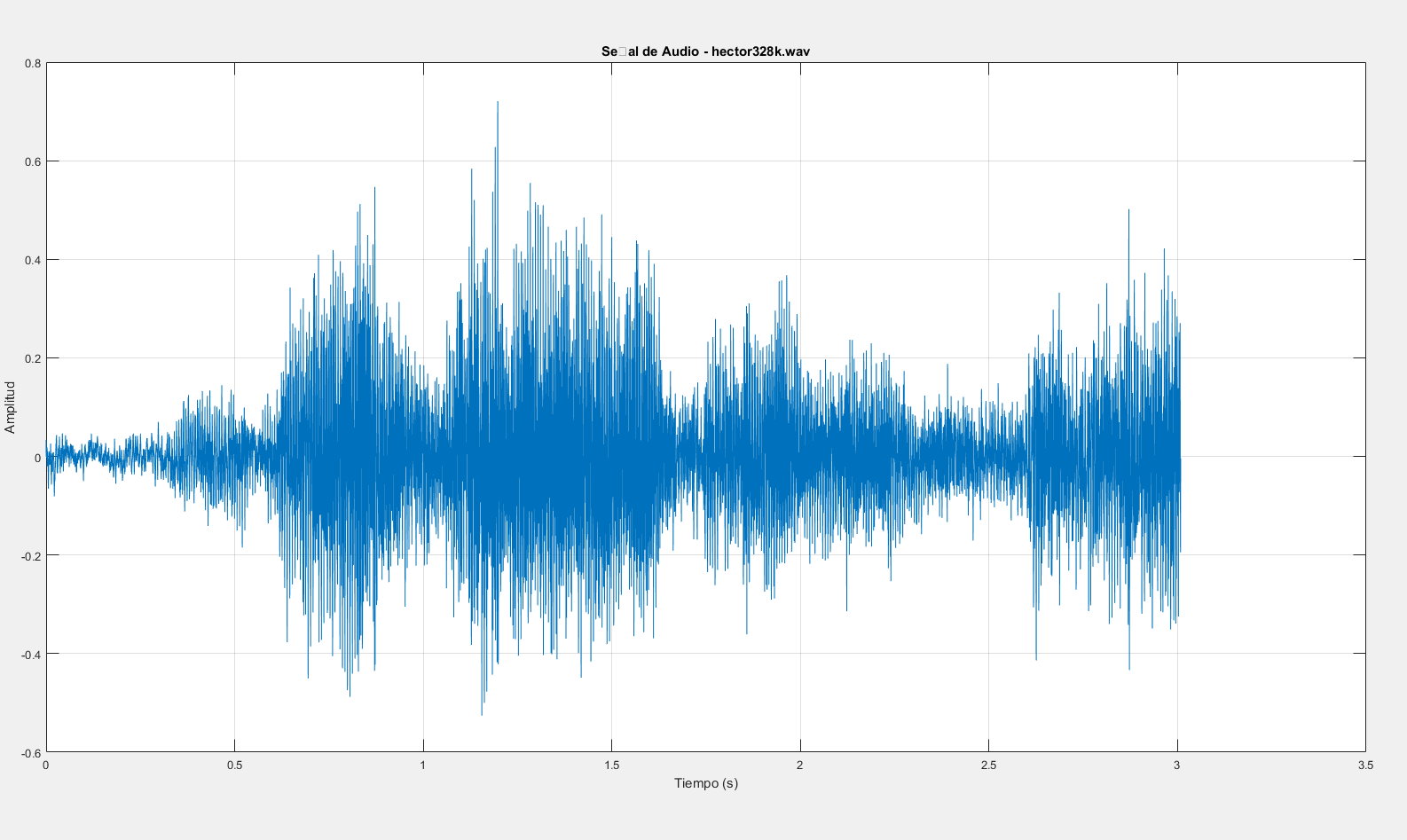
De esta forma obtenemos los archivos hector8k.wav, santi328k.wav, juanjo328k.wav y ernesto328k.wav.

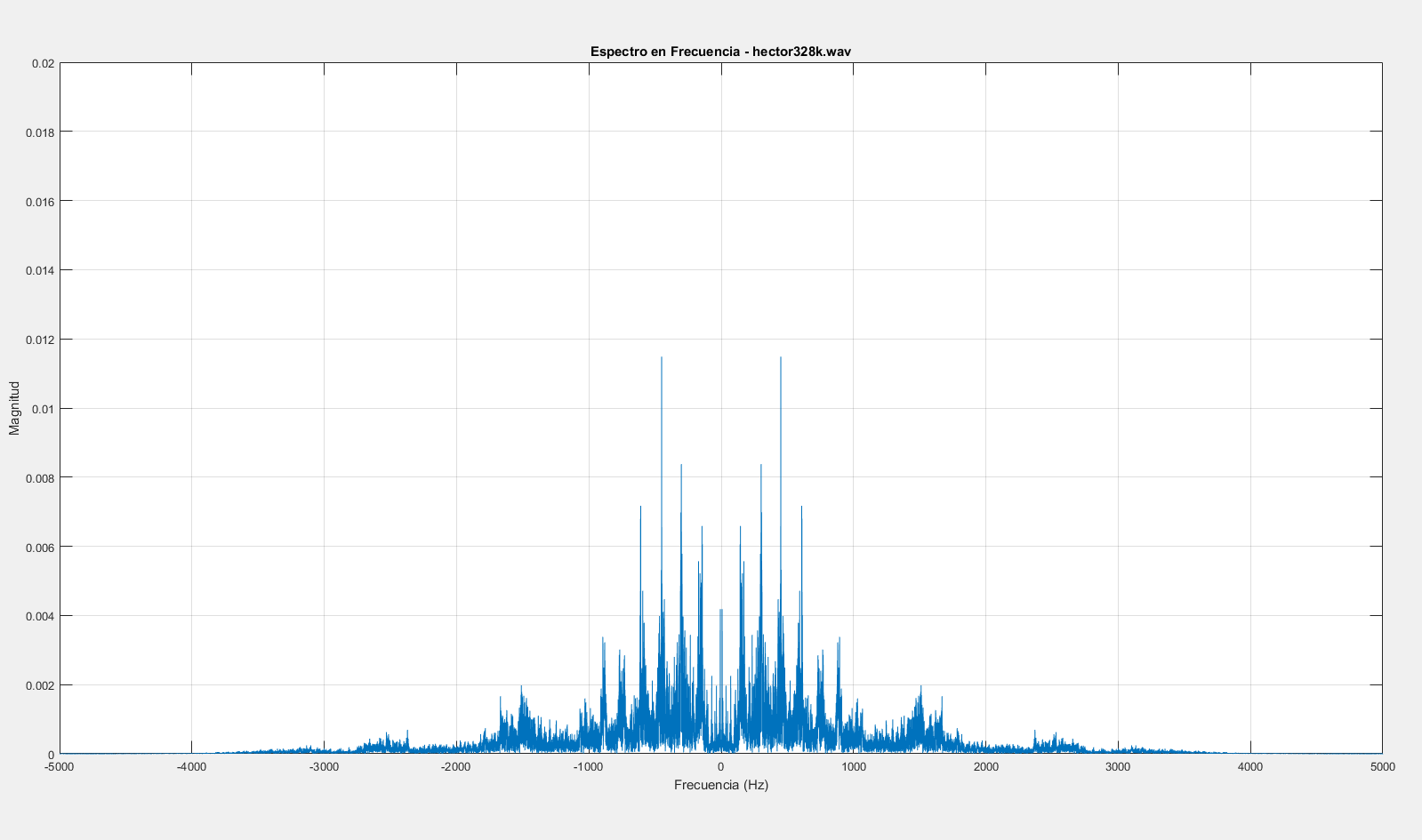
b)**ernesto328k.wav:**



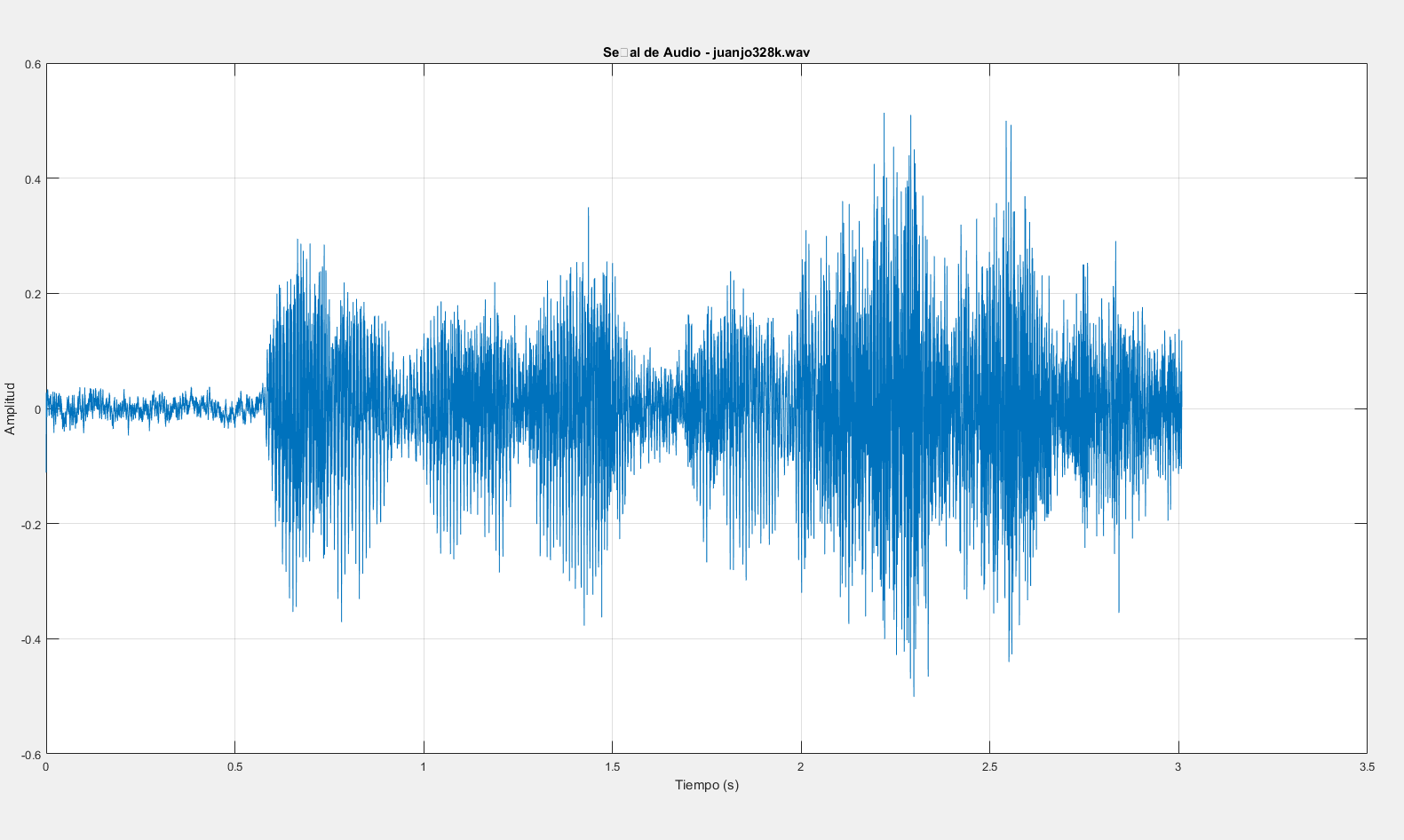


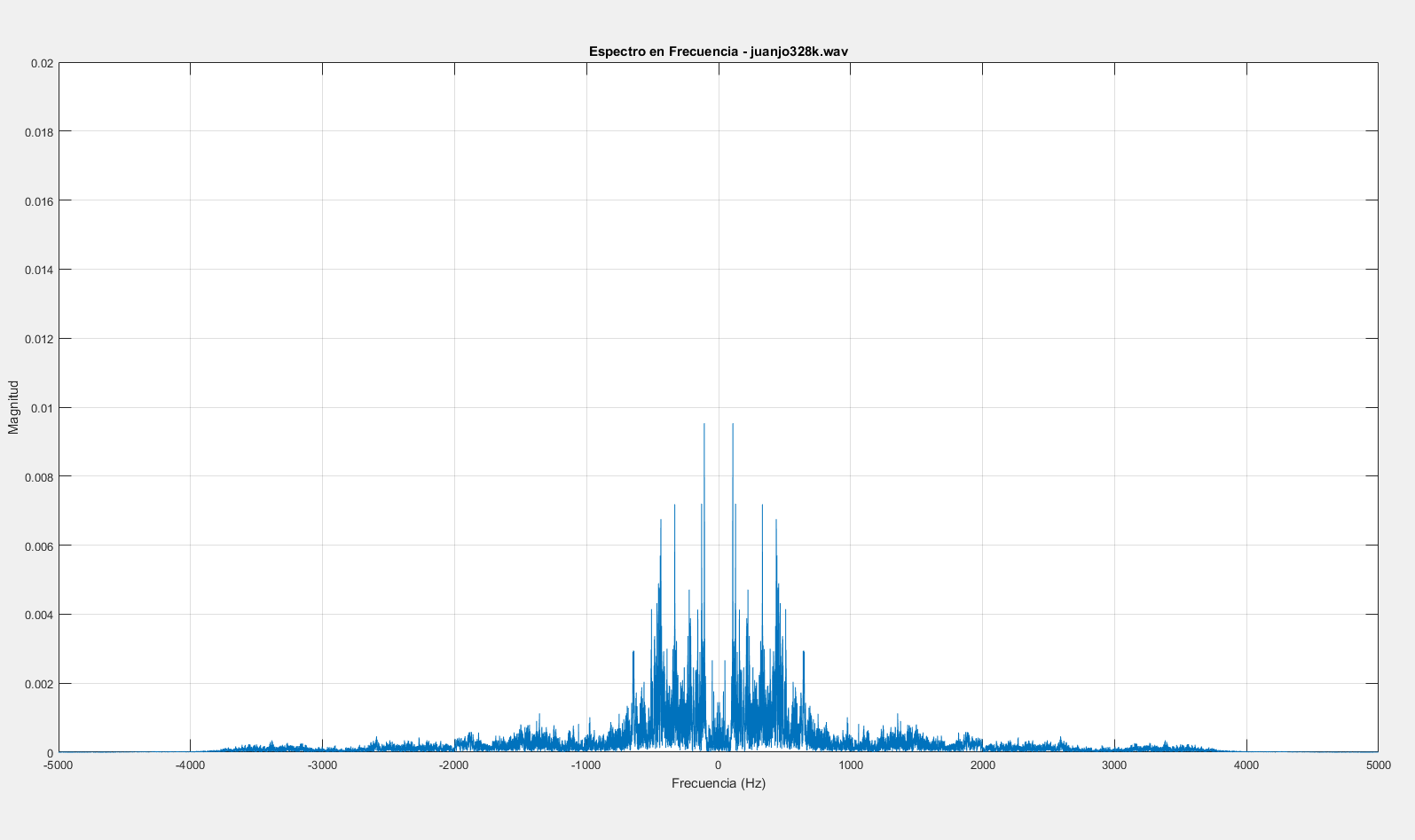
**hector328k.wav:**



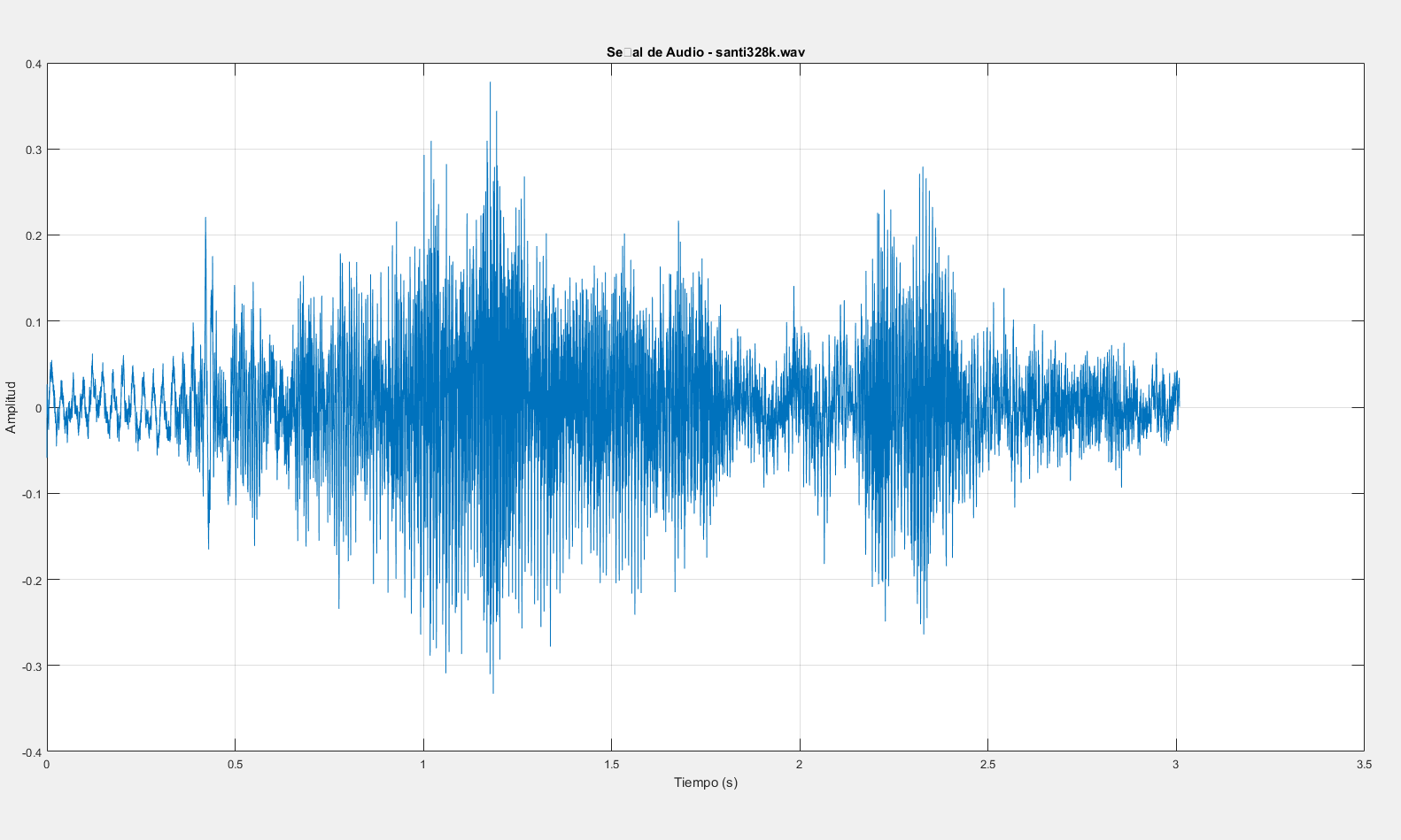


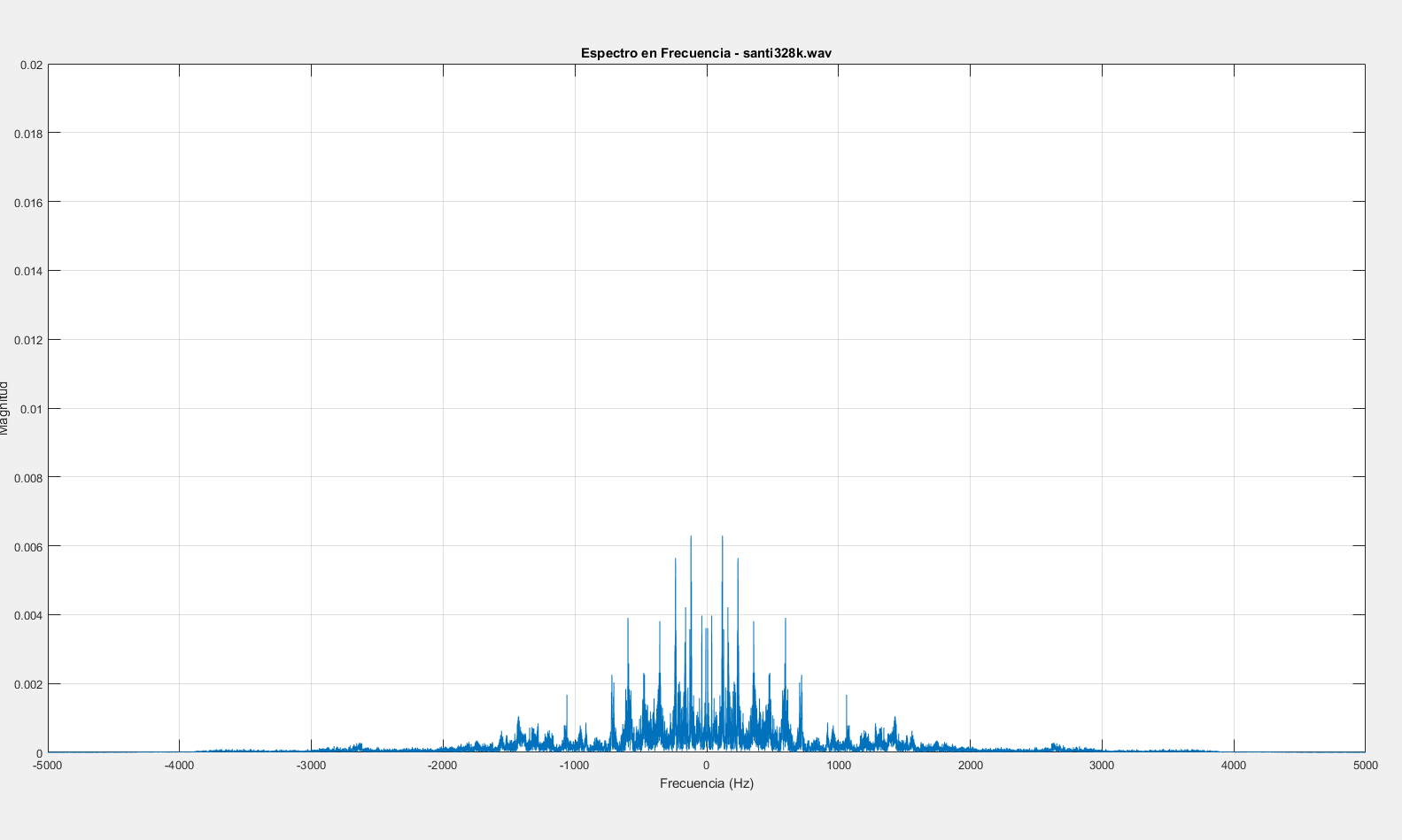
**juanjo328k.wav:**





**santi328k.wav**



  
  
*Código utilizado para hacer las graficas:*

*Gráficas en el tiempo:*

% Lista de archivos de audio

audioFiles = {'ernesto328k.wav', 'hector328k.wav', 'juanjo328k.wav', 'santi328k.wav'};

% N�mero de archivos

numAudios = length(audioFiles);

for i = 1:numAudios

% Crear una nueva figura para cada archivo de audio

figure;

% Cargar cada archivo de audio

[audioData, fs] = audioread(audioFiles{i});

% Crear el vector de tiempo

t = (0:length(audioData)-1) / fs;

% Graficar la se�al de audio

plot(t, audioData);

xlabel('Tiempo (s)');

ylabel('Amplitud');

% A�adir el t�tulo con el nombre del archivo de audio

title(['Se�al de Audio - ', audioFiles{i}]);

grid on;

end

*Gráficas de espectro de frecuencia:*

% Cerrar todas las figuras abiertas

close all;

% Lista de archivos de audio

audioFiles = {'ernesto328k.wav', 'hector328k.wav', 'juanjo328k.wav', 'santi328k.wav'}; % Agrega los nombres de tus archivos aqu�

% Especifica los rangos de los ejes

% Para frecuencia en Hz

xMin = -5000; % Rango m�nimo de frecuencia

xMax = 5000; % Rango m�ximo de frecuencia

% Para magnitud

yMin = 0; % Rango m�nimo de magnitud

yMax = 0.02; % Rango m�ximo de magnitud

% N�mero de archivos

numAudios = length(audioFiles);

for i = 1:numAudios

% Cargar cada archivo de audio

[audioData, fs] = audioread(audioFiles{i});

% Realizar la Transformada de Fourier

L = length(audioData); % Longitud de la se�al

Y = fftshift(fft(audioData)); % FFT y cambio para centrar el espectro

f = (-L/2:L/2-1)\*(fs/L); % Vector de frecuencia que incluye negativas

% Calcular la magnitud del espectro

P = abs(Y)/L;

% Crear una nueva figura para el espectro de frecuencia

figure;

% Graficar el espectro en frecuencia (incluyendo frecuencias negativas)

plot(f, P);

xlabel('Frecuencia (Hz)');

ylabel('Magnitud');

% A�adir el t�tulo con el nombre del archivo de audio

title(['Espectro en Frecuencia - ', audioFiles{i}]);

grid on;

% Ajustar los rangos de los ejes

xlim([xMin, xMax]); % Rango del eje x (frecuencia)

ylim([yMin, yMax]); % Rango del eje y (magnitud)

end

*Observaciones:*

A partir de las gráficas obtenidas, más concretamente desde los espectros de frecuencias de los audios, vemos como efectivamente nuestras voces tienen un rango que va desde un poco más de los 0 Hz hasta los 3,5 KHz aproximadamente, el cual nos de un ancho de banda similar al que esperábamos de la voz humana, con más presencia en los graves (frecuencias bajas) que en los agudos (frecuencias altas).

*Nota:* Podemos ver un reflejo del espectro de nuestra voz en las frecuencias negativas ya que estamos observando dicho espectro en banda base.

c)

Para realizar el desplazamiento de las señales desde banda base a frecuencias más altas dadas por los “osciladores” (Frecuencias portadoras), escribimos un código que realiza la operación , Donde representa a cada una de las señales moduladas, a las señales antes de modular y Wk es la frecuencia en radianes de la relación *frec. Portadora / frec. muestreo.*

*Código para hacer la modulación de cada señal:*

% Lista de archivos de audio resampleados

archivosAudio = {'ernesto328k.wav', 'hector328k.wav', 'juanjo328k.wav', 'santi328k.wav'};

% Frecuencia de muestreo

Fs = 328000; % 328 kHz

% Frecuencias portadoras para cada canal (en Hz)

frecuenciasPortadoras = [60000, 64000, 68000, 72000]; % 60 kHz, 64 kHz, 68 kHz, 72 kHz

% Inicializar celda para almacenar las se?ales moduladas

senalesModuladas = cell(1, length(archivosAudio));

for k = 1:length(archivosAudio)

% Leer la se?al de audio resampleada

[Xk\_n, ~] = audioread(archivosAudio{k});

% Crear vector de muestras n

n = (0:length(Xk\_n)-1)'; % Vector columna de enteros

% Calcular la frecuencia angular omega\_k

fk = frecuenciasPortadoras(k); % Frecuencia portadora en Hz

omega\_k = 2 \* pi \* fk / Fs; % Frecuencia angular en radianes

% Calcular cos(omega\_k \* n)

cos\_omega\_n = cos(omega\_k \* n);

% Realizar el desplazamiento en frecuencia utilizando la ecuaci?n X'\_k[n] = X\_k[n] \* cos(omega\_k \* n)

Xk\_modulada = Xk\_n .\* cos\_omega\_n;

% Guardar la se?al modulada

senalesModuladas{k} = Xk\_modulada;

%Guardar la se?al modulada en un archivo

nombreArchivoModulado = ['modulada\_' archivosAudio{k}];

audiowrite(nombreArchivoModulado, Xk\_modulada, Fs);

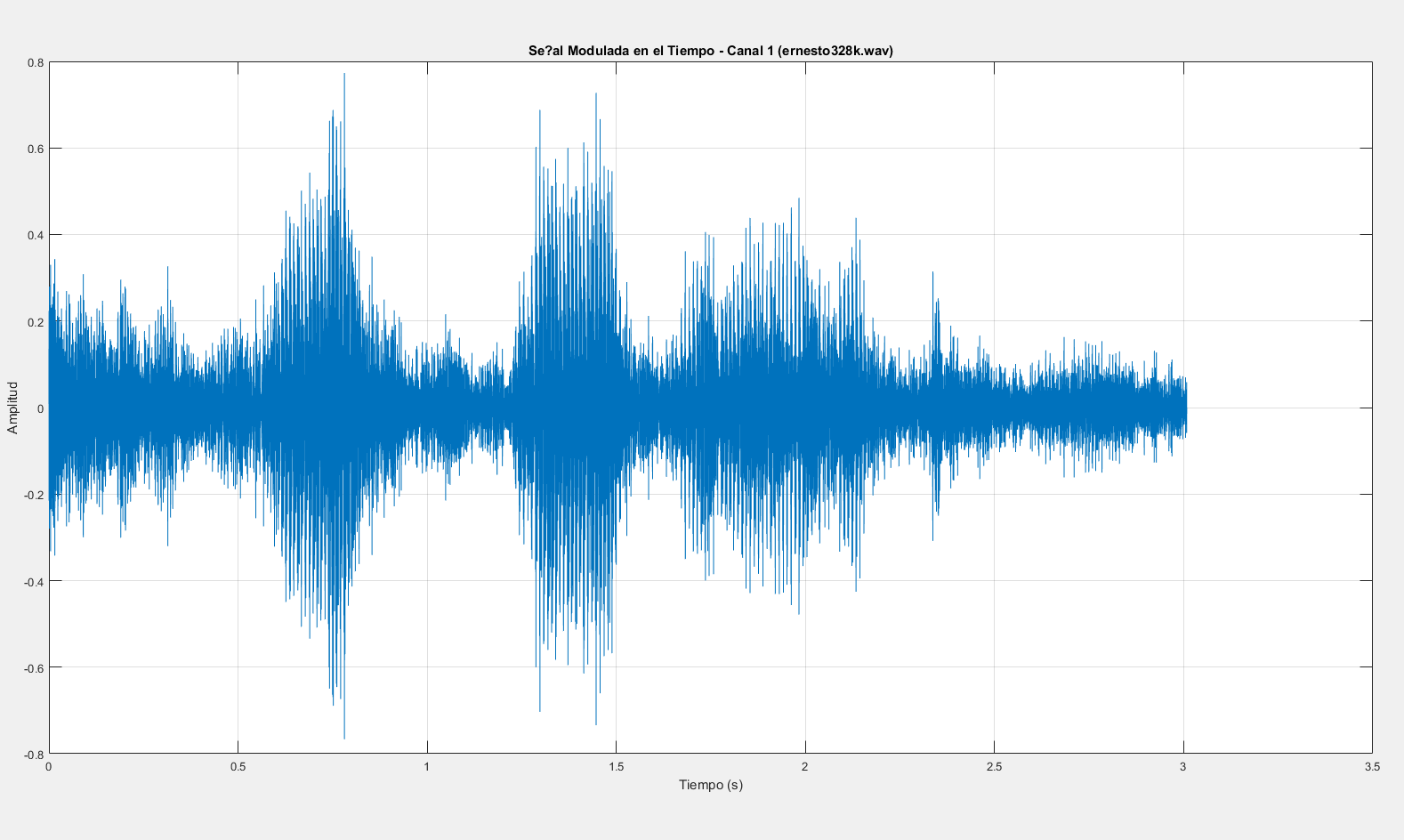
end

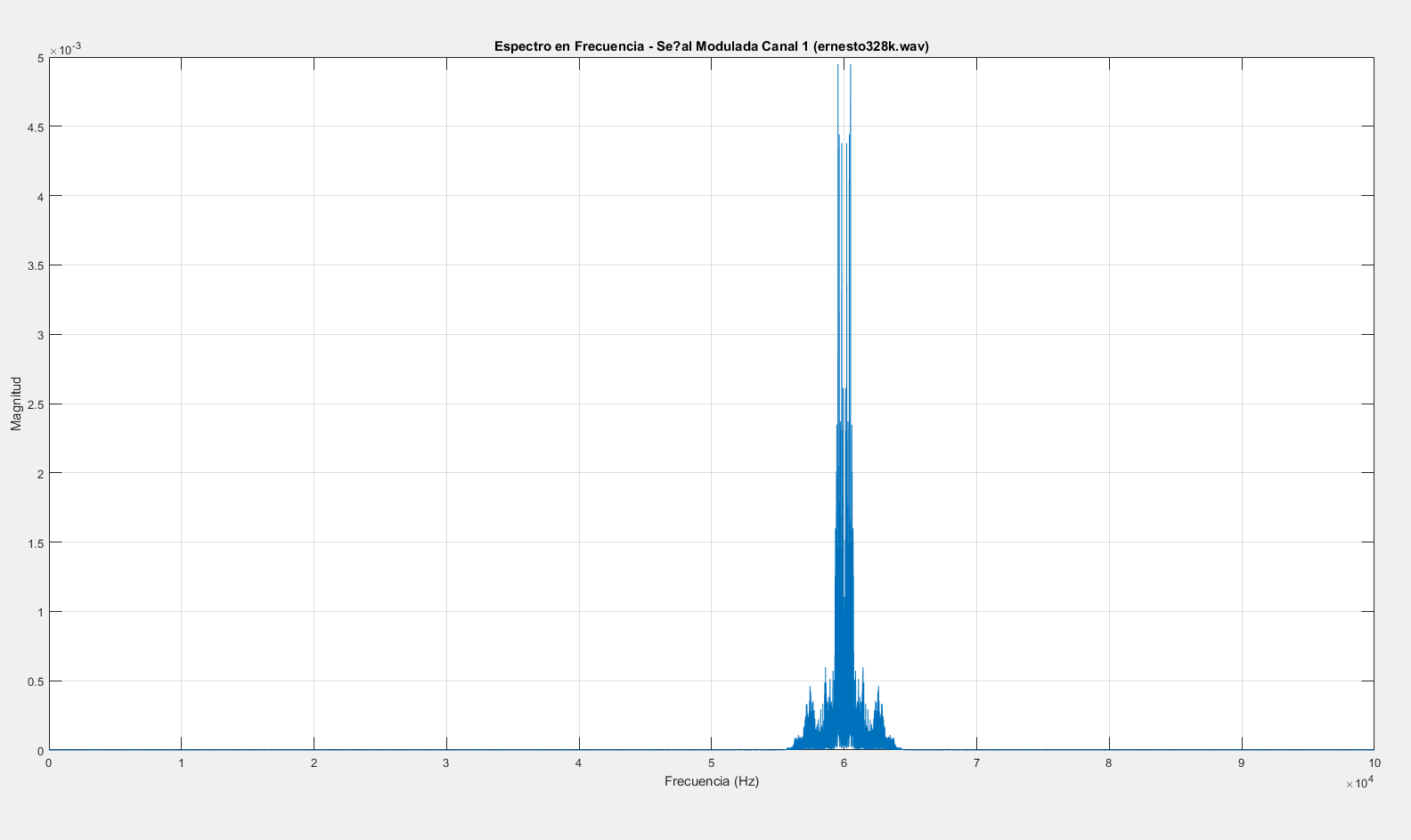
Con este código logramos multiplicar a cada una de las señales por logrando así desplazarlas en frecuencia.

d)

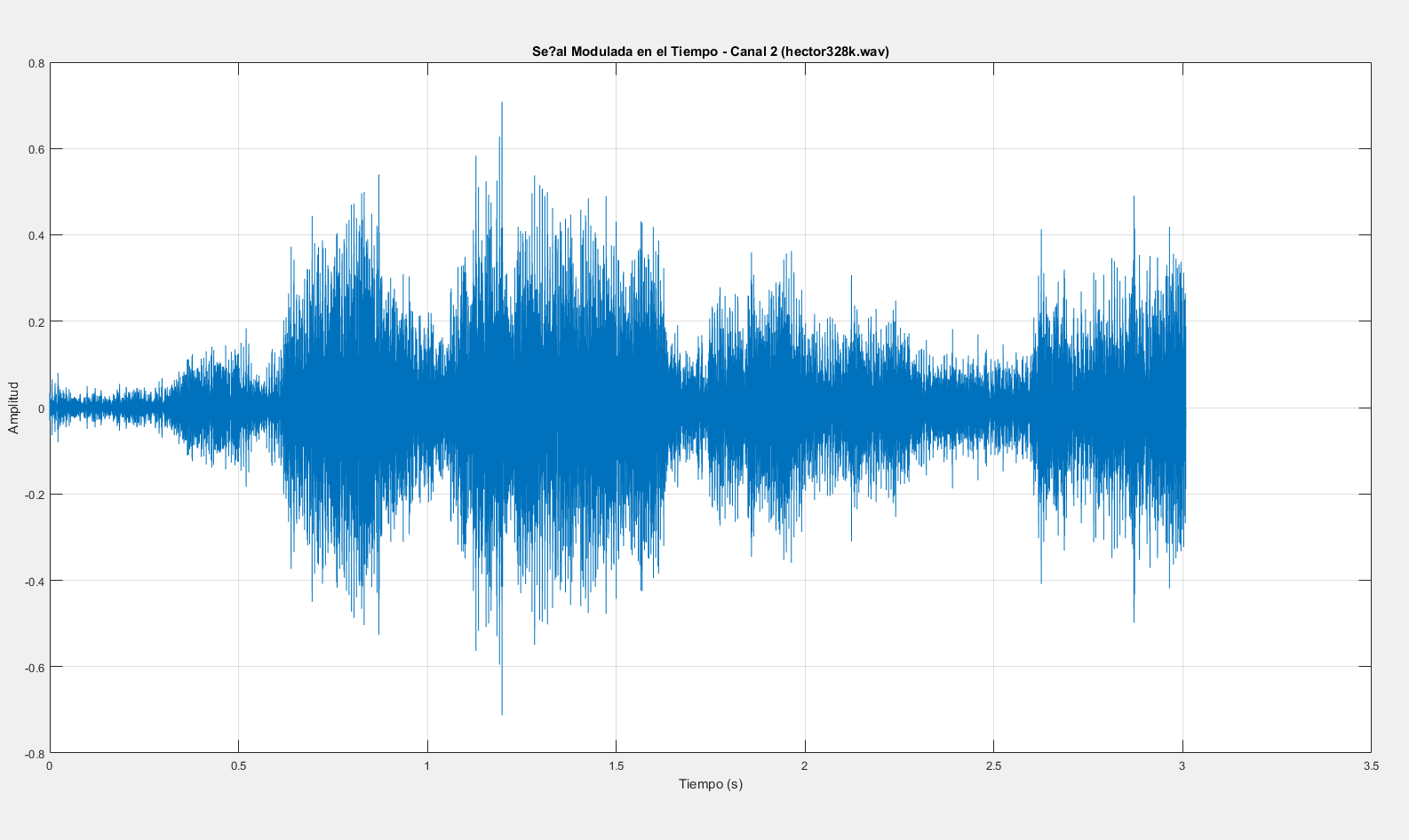
Las gráficas en tiempo y frecuencia de las señales moduladas en el apartado anterior serian:

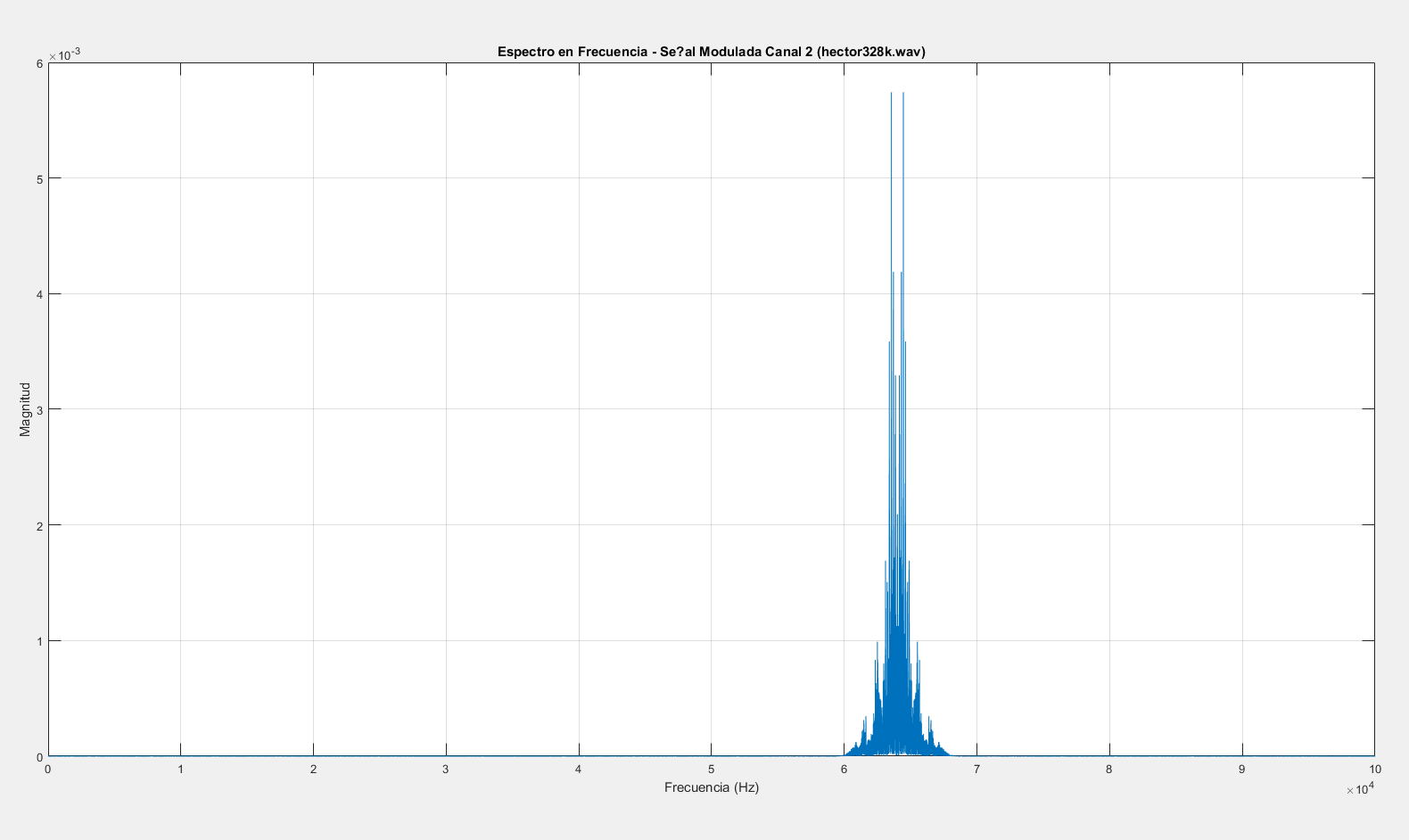
**ernesto328K.wav modulado a 60 KHz:**



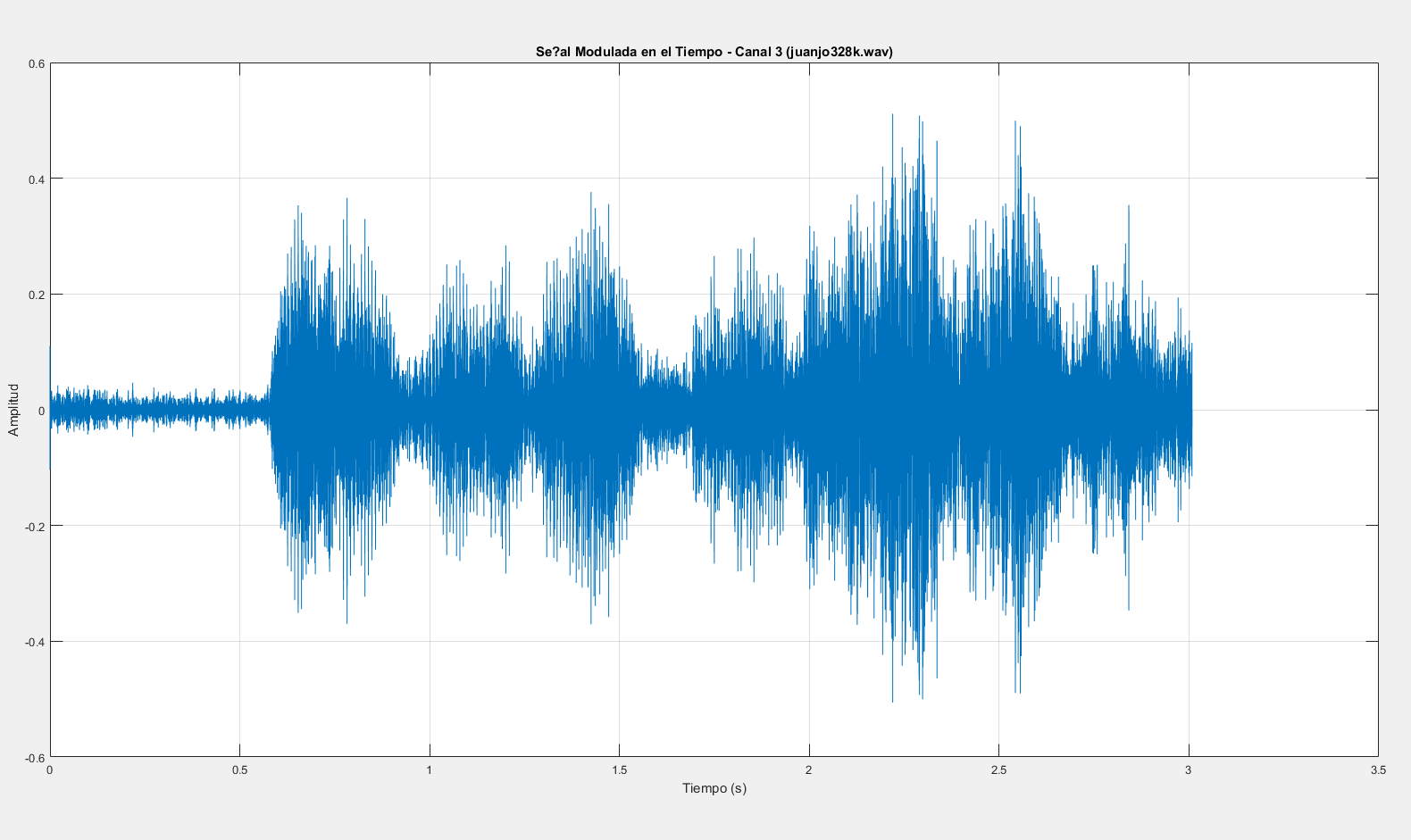


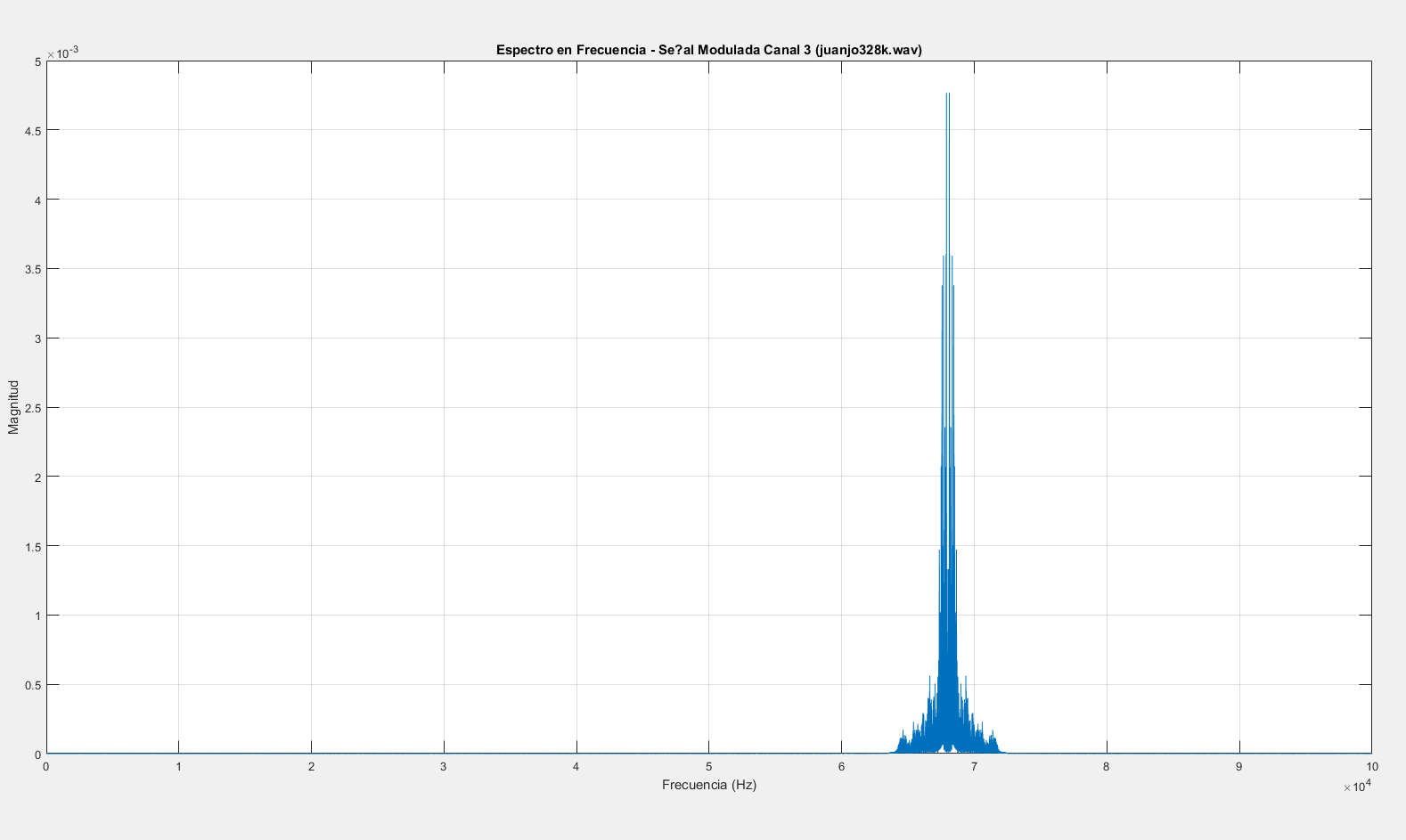
**hector328k.wav modulado a 64 KHz:**



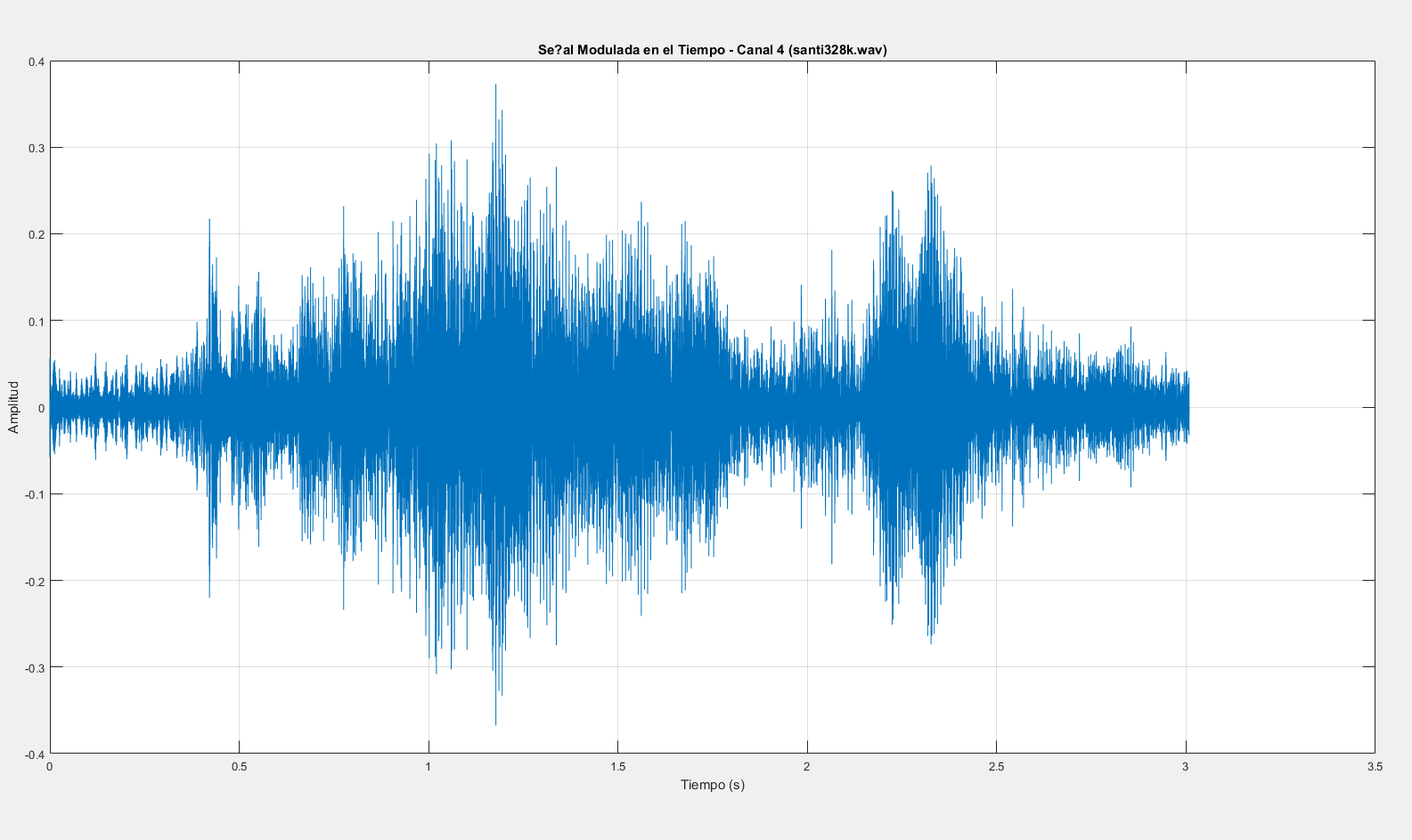


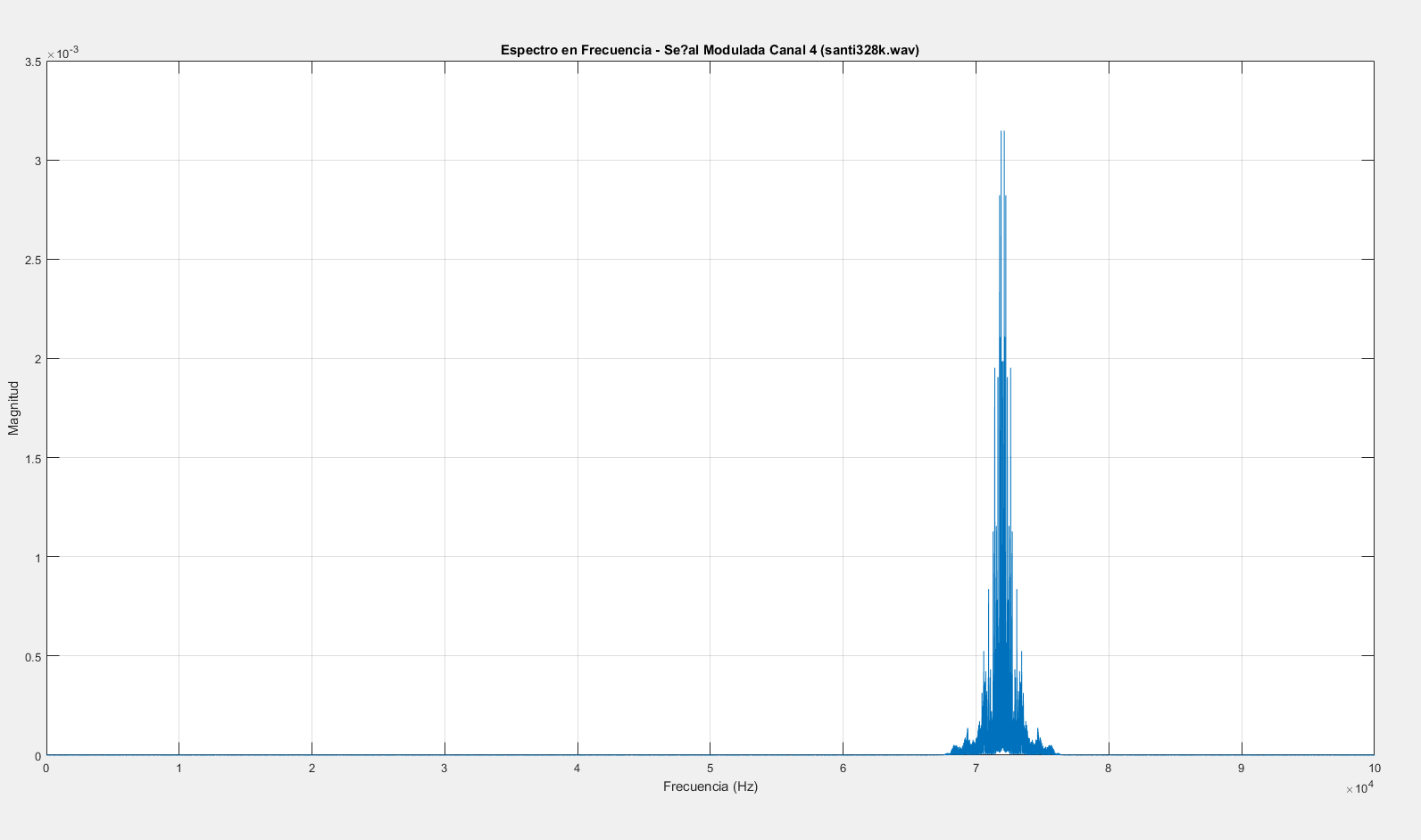
**juanjo328K.wav modulado a 68 KHz:**





**santi328K.wav modulado a 72 KHz:**





*Código utilizado para hacer las gráficas:*

% Cerrar todas las figuras abiertas

close all;

% Seleccionar el canal a graficar (del 1 al 4)

k = 1;

% Obtener la se?al modulada

Xk\_modulada = senalesModuladas{k};

% Crear vector de tiempo 'n' en segundos

t = n / Fs;

% Graficar la se?al modulada en el dominio del tiempo

figure;

plot(t, Xk\_modulada);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title(['Se?al Modulada en el Tiempo - Canal ', num2str(k), ' (', archivosAudio{k}, ')']);

grid on;

% Calcular y graficar el espectro de frecuencia

L = length(Xk\_modulada); % Longitud de la se?al

Y = fftshift(fft(Xk\_modulada)); % FFT y centrar el espectro

f = (-L/2:L/2-1)\*(Fs/L); % Vector de frecuencia

% Calcular la magnitud del espectro

P = abs(Y)/L;

% Graficar el espectro en frecuencia

figure;

plot(f, P);

xlabel('Frecuencia (Hz)');

ylabel('Magnitud');

title(['Espectro en Frecuencia - Se?al Modulada Canal ', num2str(k), ' (', archivosAudio{k}, ')']);

grid on;

% Ajustar los l?mites del eje x para enfocarse en la regi?n de inter?s

xlim([0, 100000]); % Solo frecuencias positivas

*Observaciones:*

De las gráficas de los espectro de frecuencia, podemos ver que efectivamente el espectro de nuestras señales en banda base se desplazaron en frecuencia al centro del valor de la frecuencia de portadora con la que se módulo a cada una. Además podemos apreciar que se trasladó tanto el espectro positivo como el negativo que estaba reflejado, obteniendo así una banda lateral por arriba de la portadora y otra banda lateral por debajo de la portadora, en otras palabras, estamos viendo en espectro de una señal DBL (doble banda lateral).

*Nota:* cabe resaltar que ahora que tenemos dos bandas laterales, podemos ver como el ancho de banda de la señal en banda, una vez modulada, es el doble.

e)

Para hacer los filtros para cada canal, desarrollamos el siguiente código:

% Dise?o del filtro pasa-altos para eliminar la banda lateral inferior

Fc = 60000; % Frecuencia de corte (cambiar según corresponda)

Wn = Fc / (Fs / 2); % Normalizaci?n

% Incrementar el orden del filtro y usar un filtro Chebyshev

orden = 20; % Ajusta seg?n sea necesario

Rp = 1; % Ondulaci?n permitida en la banda de paso (en dB)

[b, a] = cheby1(orden, Rp, Wn, 'high'); % Filtro pasa-altos Chebyshev

% Verificar la respuesta del filtro

[H, f] = freqz(b, a, 1024, Fs); % Respuesta en frecuencia en Hz

figure;

plot(f, 20\*log10(abs(H))); % Graficar en dB

xlabel('Frecuencia (Hz)');

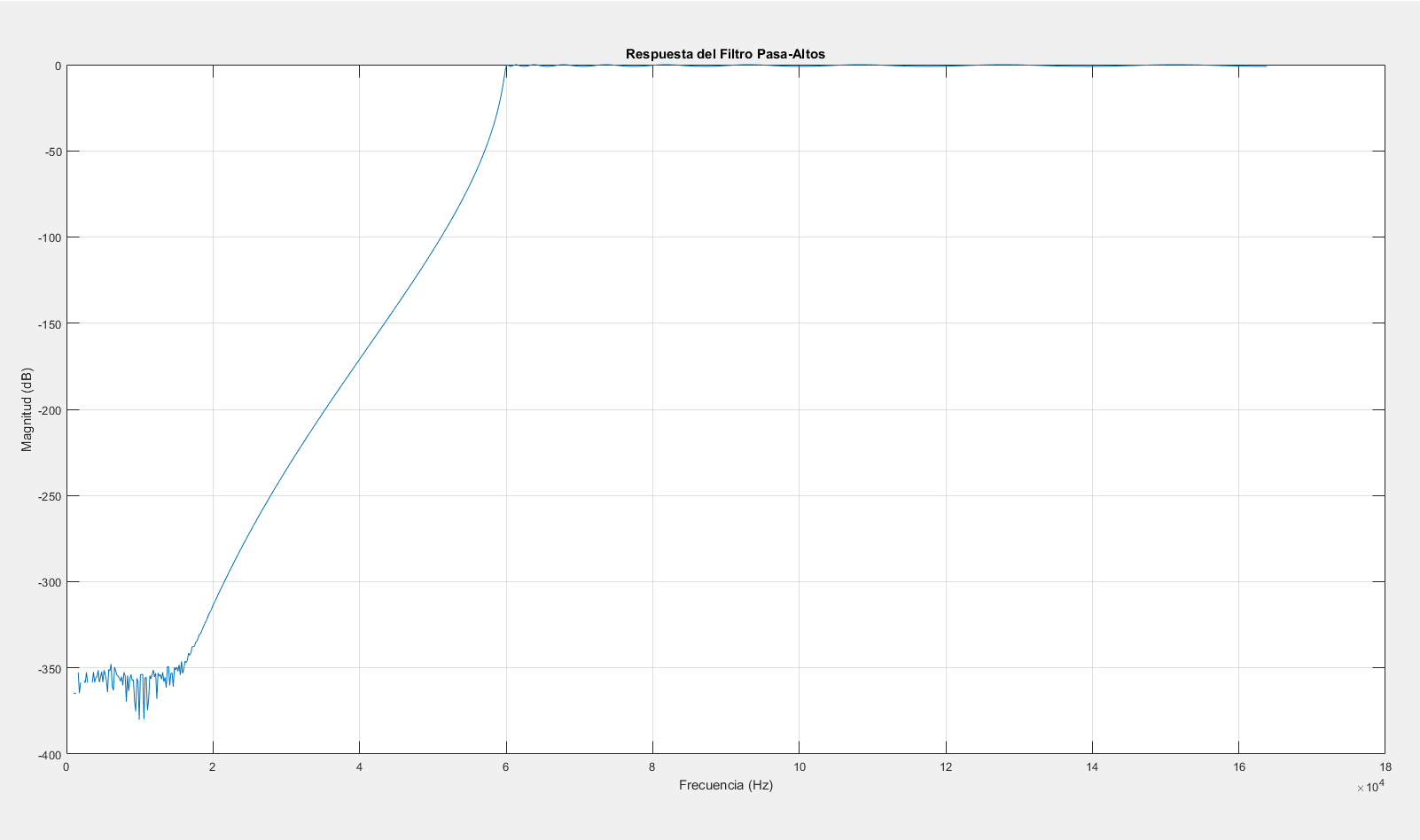
ylabel('Magnitud (dB)');

title('Respuesta del Filtro Pasa-Altos');

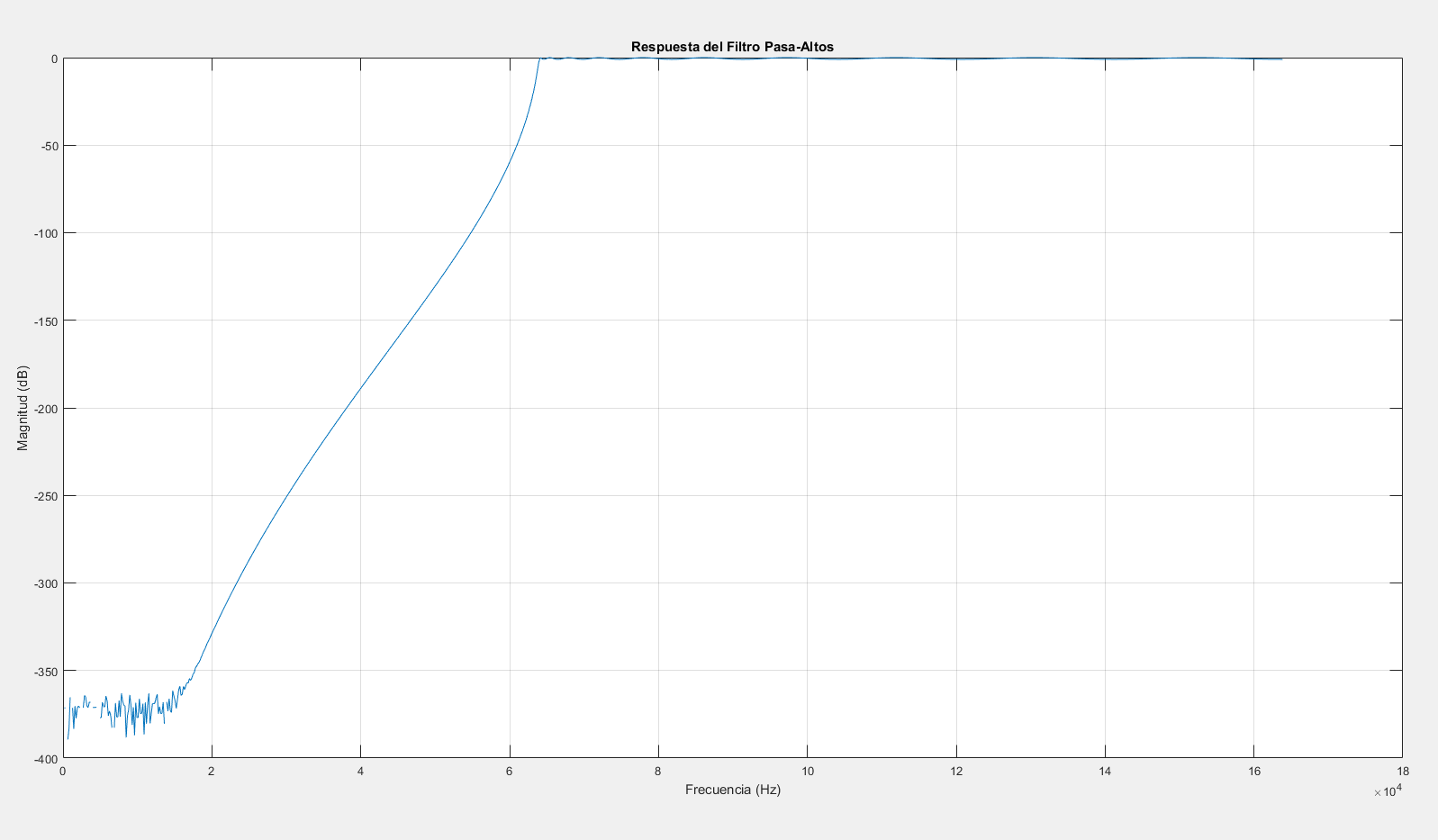
grid on;

Debido a que nuestro objetivo es suprimir la banda lateral inferior (BLI) de cada canal, en primera instancia optamos por trabajar con un filtro chebyshev, ya que utilizando un filtro butterworth no logramos suprimir de forma satisfactoria dicha banda, además realizando distintas simulaciones llegamos a la conclusión que un orden correcto para nuestra aplicación es el 20. De esta forma obtuvimos los siguientes filtros:

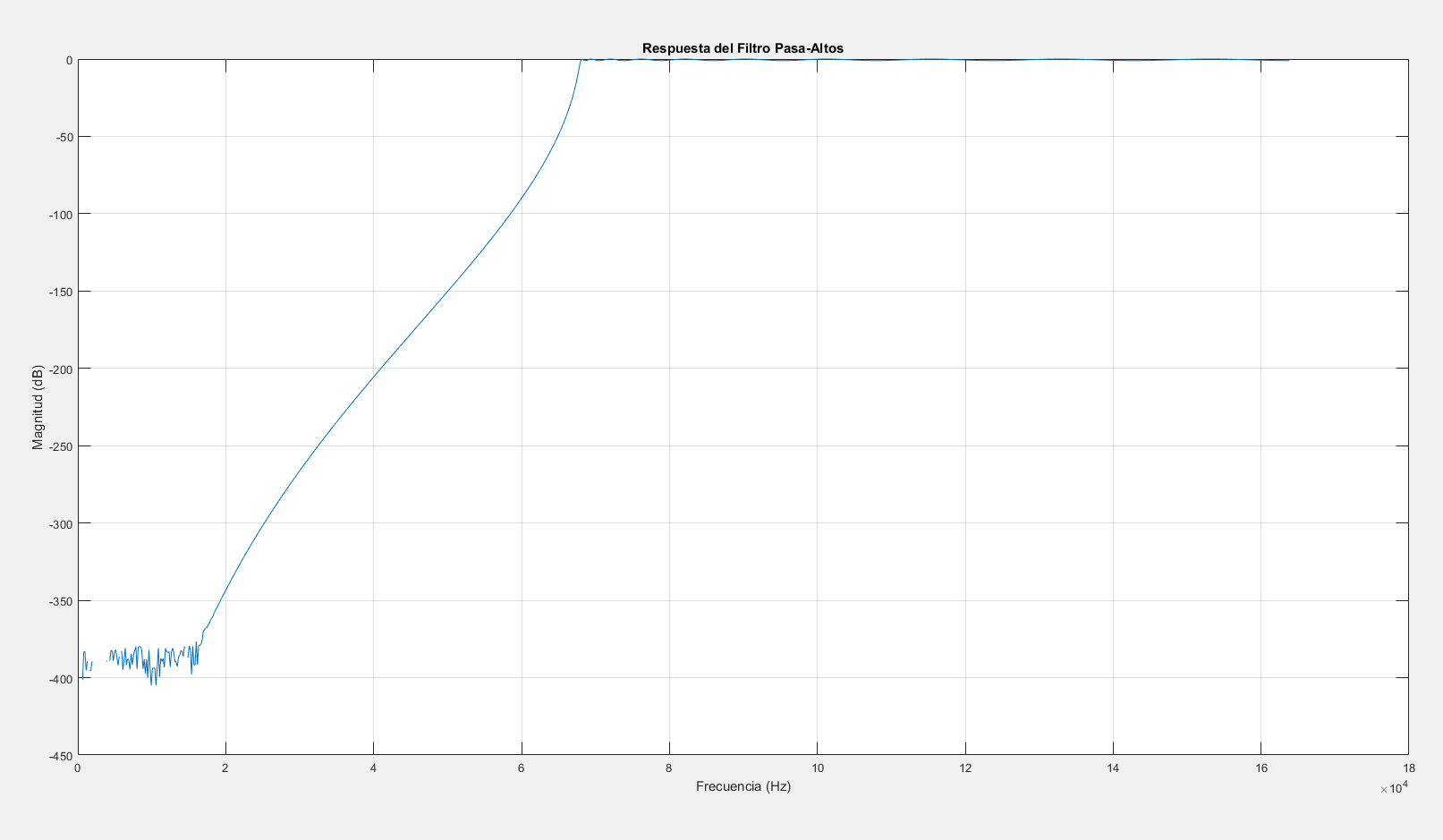
*Filtro para canal 1 (Fcorte = 60Khz):*



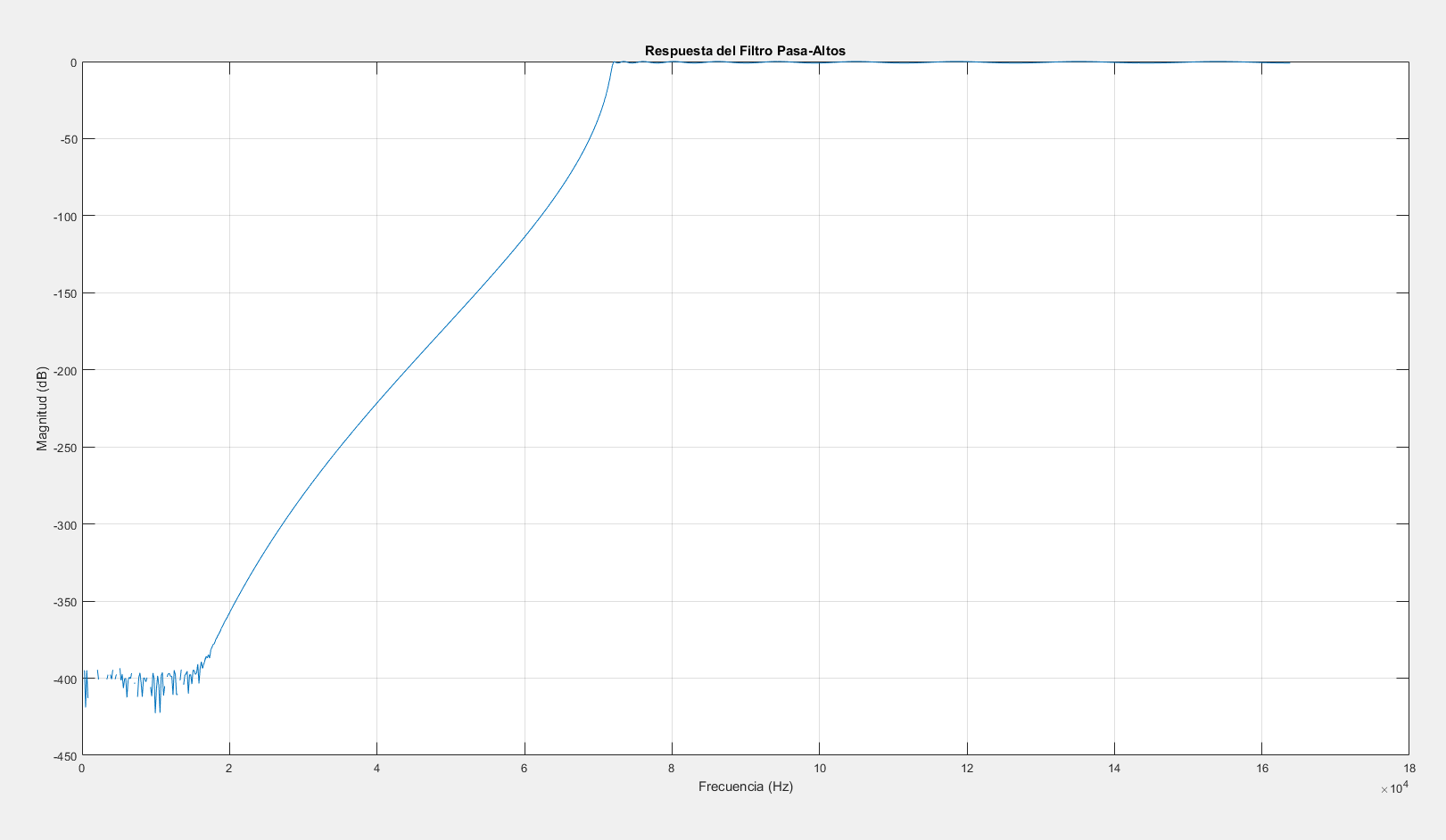
*Filtro para canal 2 (Fcorte = 64Khz):*

**

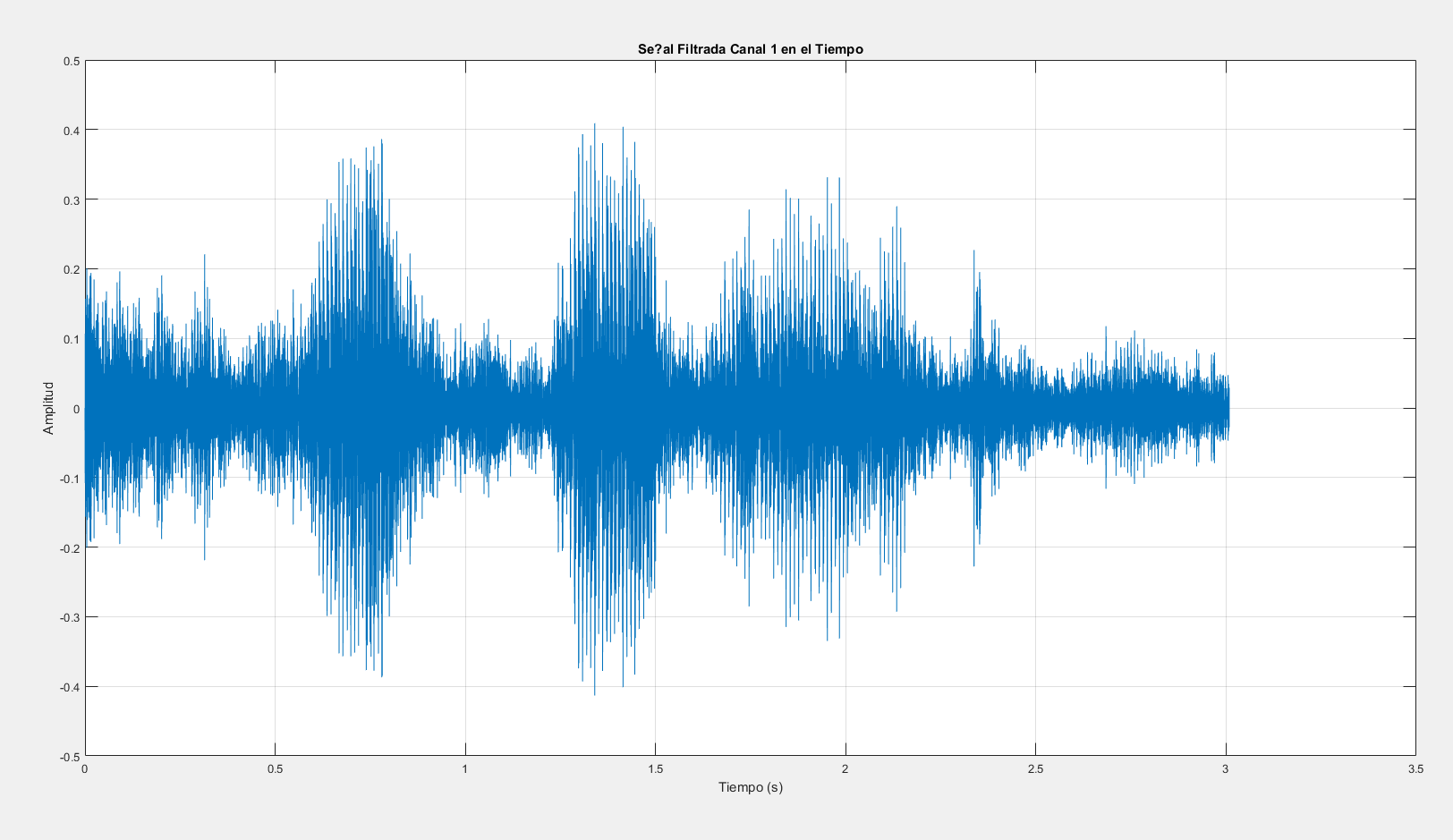
*Filtro para canal 3 (Fcorte = 68Khz):*

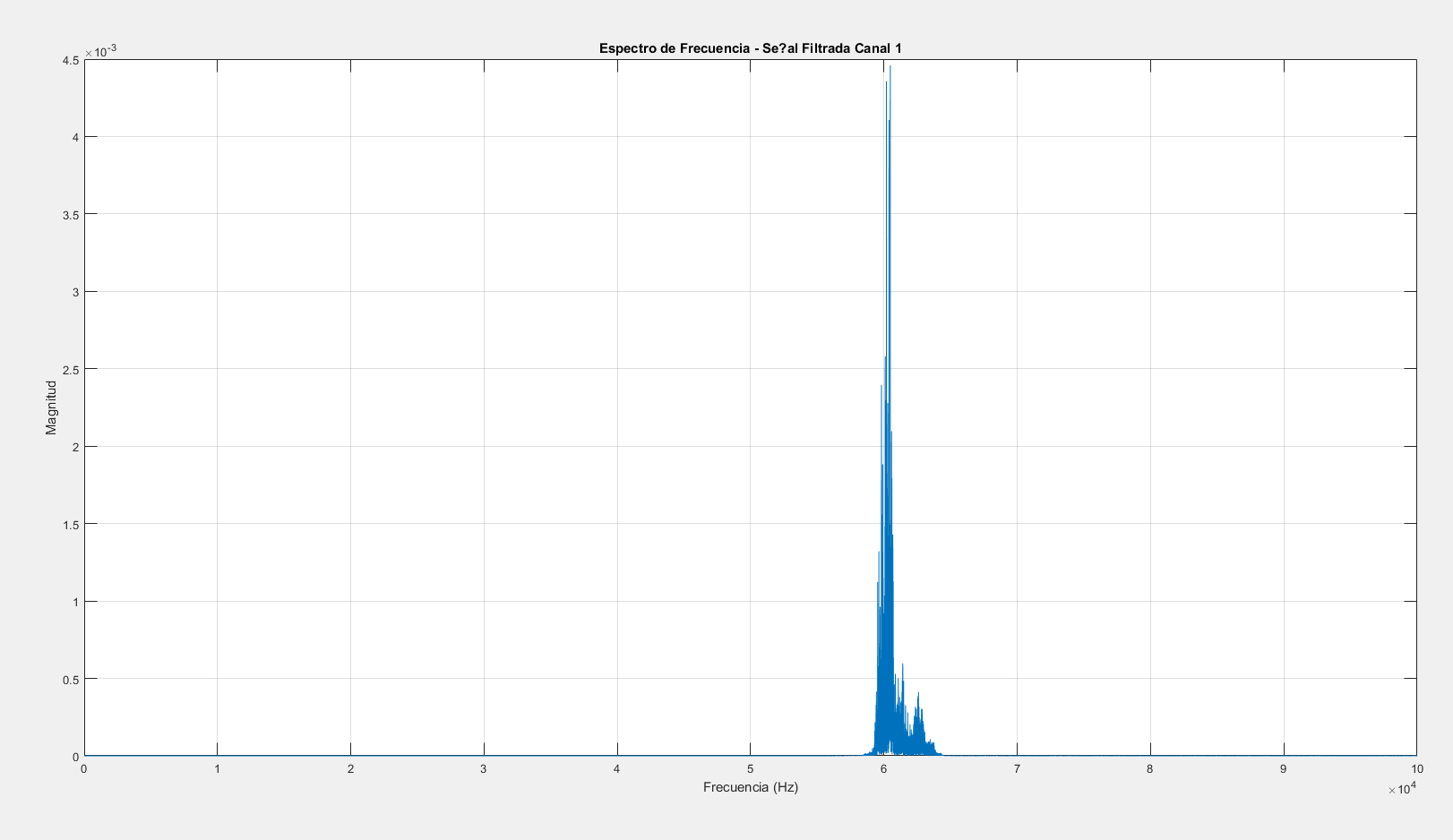
**

*Filtro para canal 4 (Fcorte = 72Khz):*

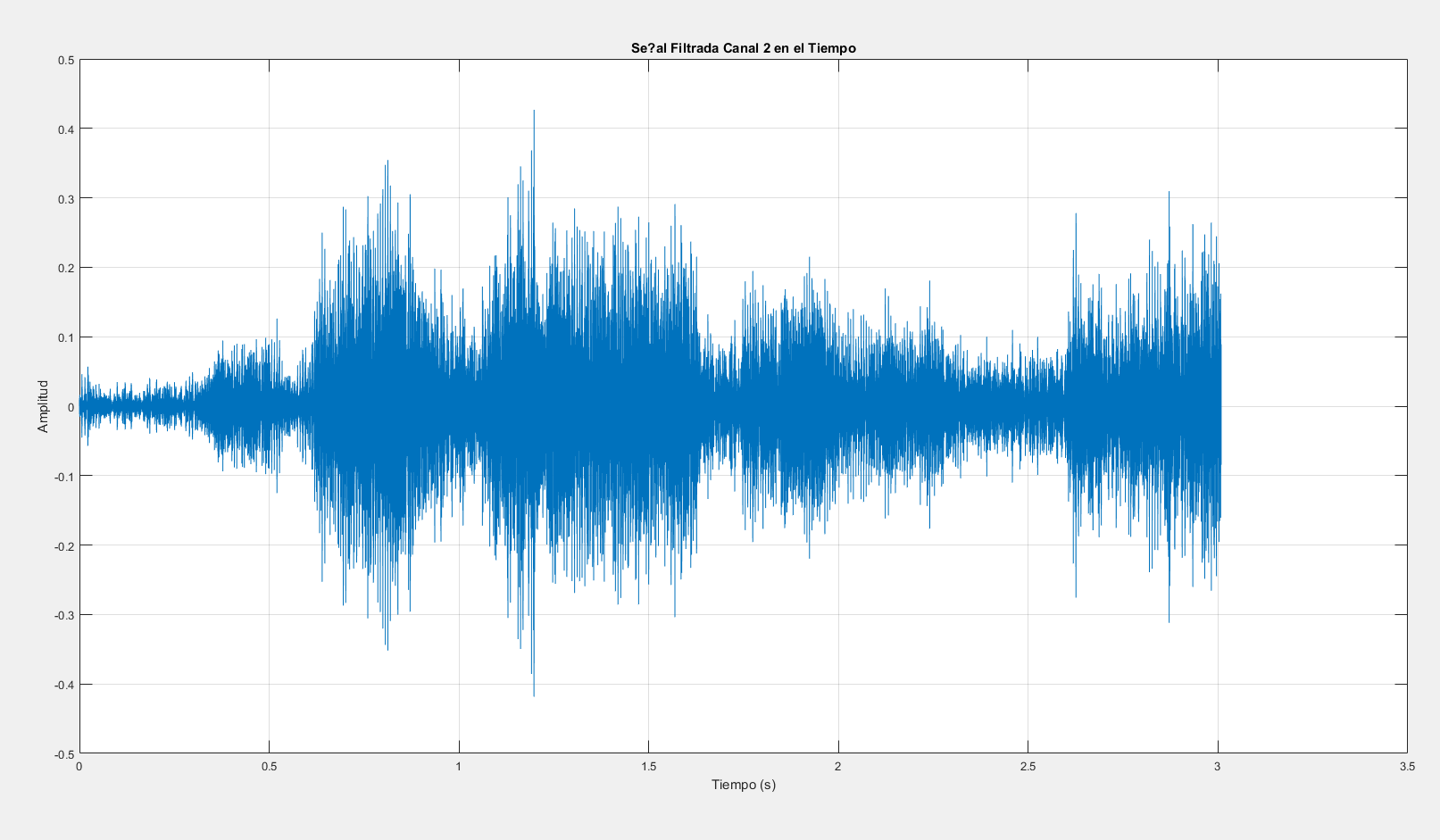


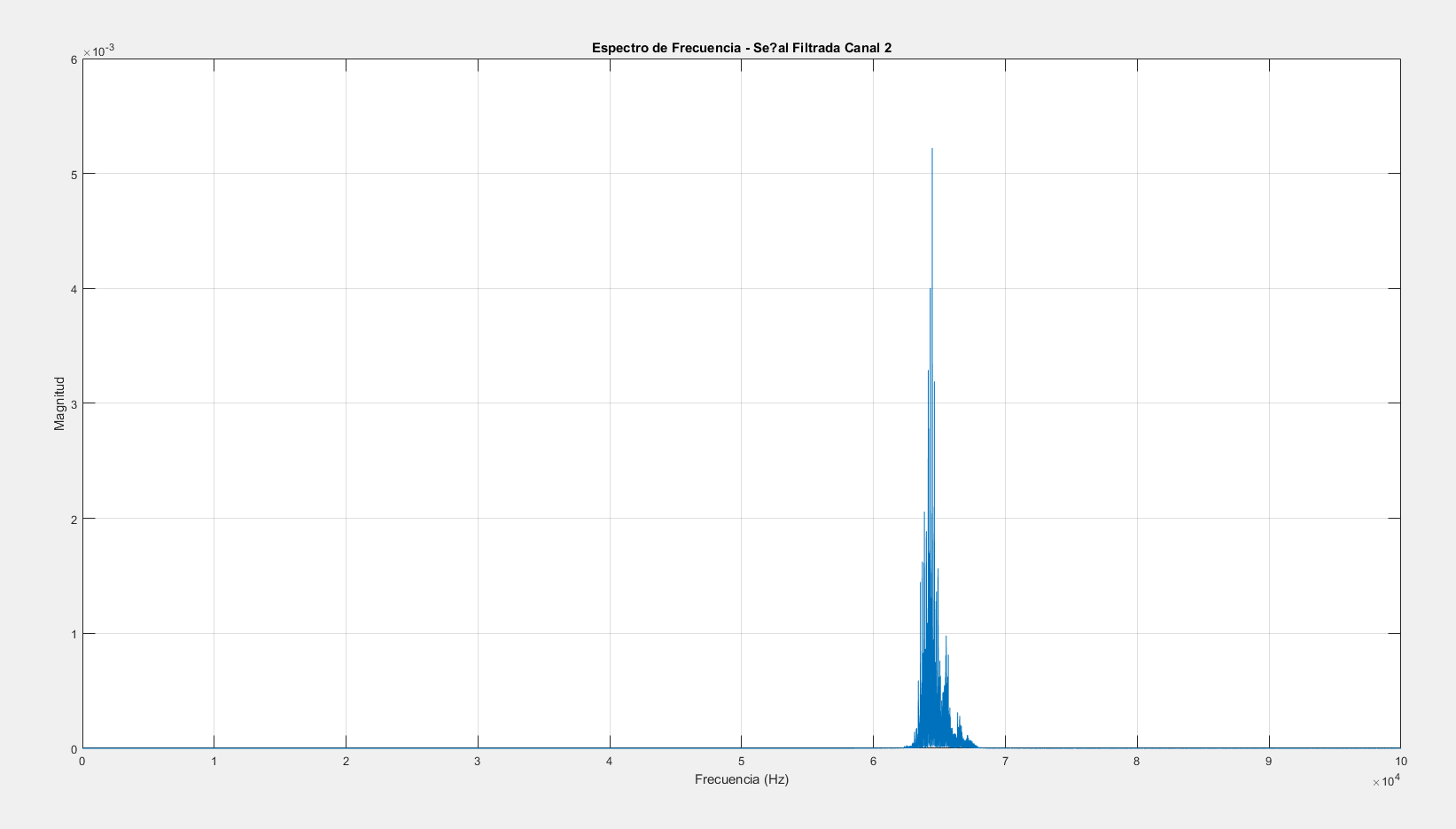
f-g)

Canal 1:  


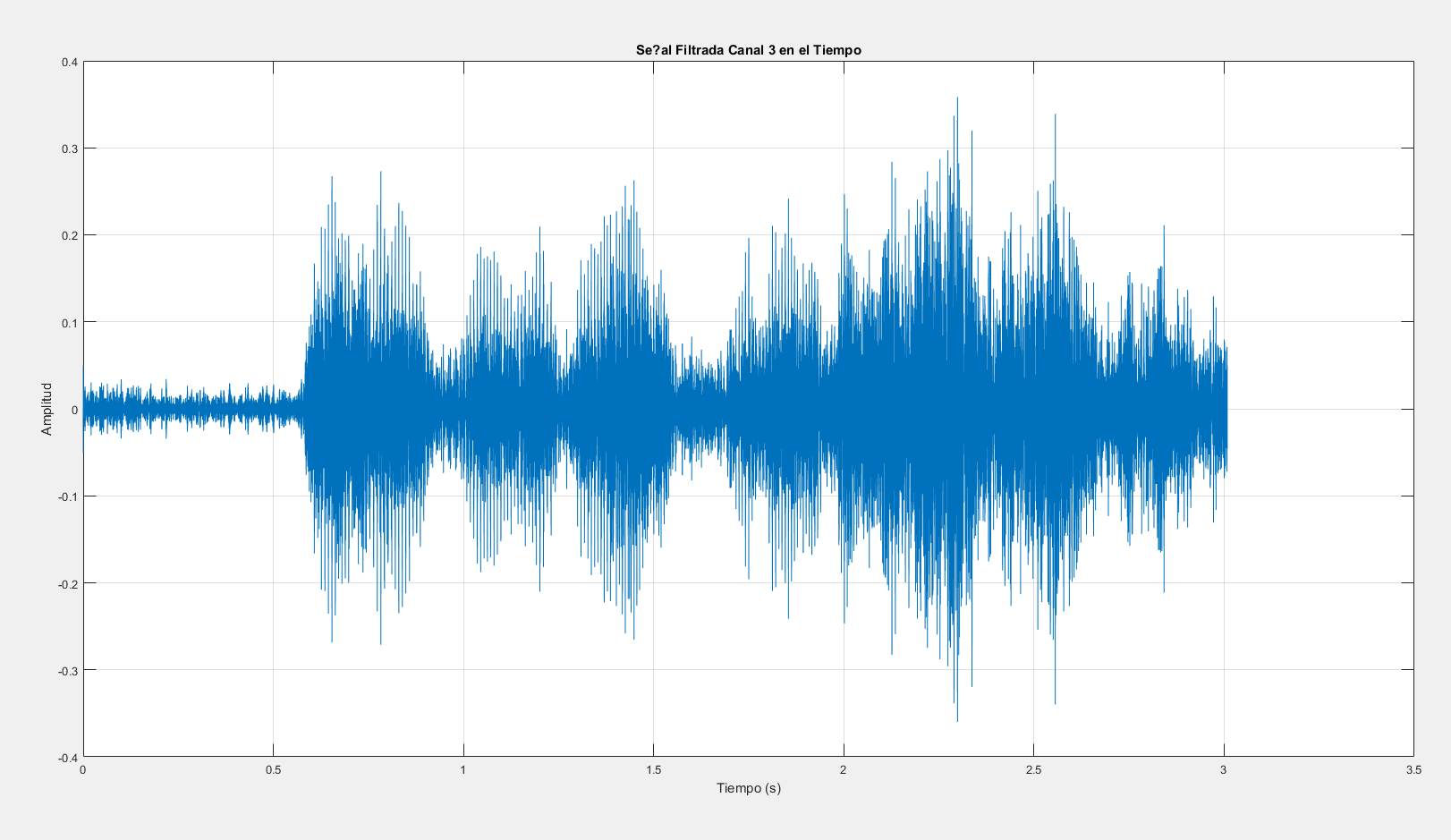


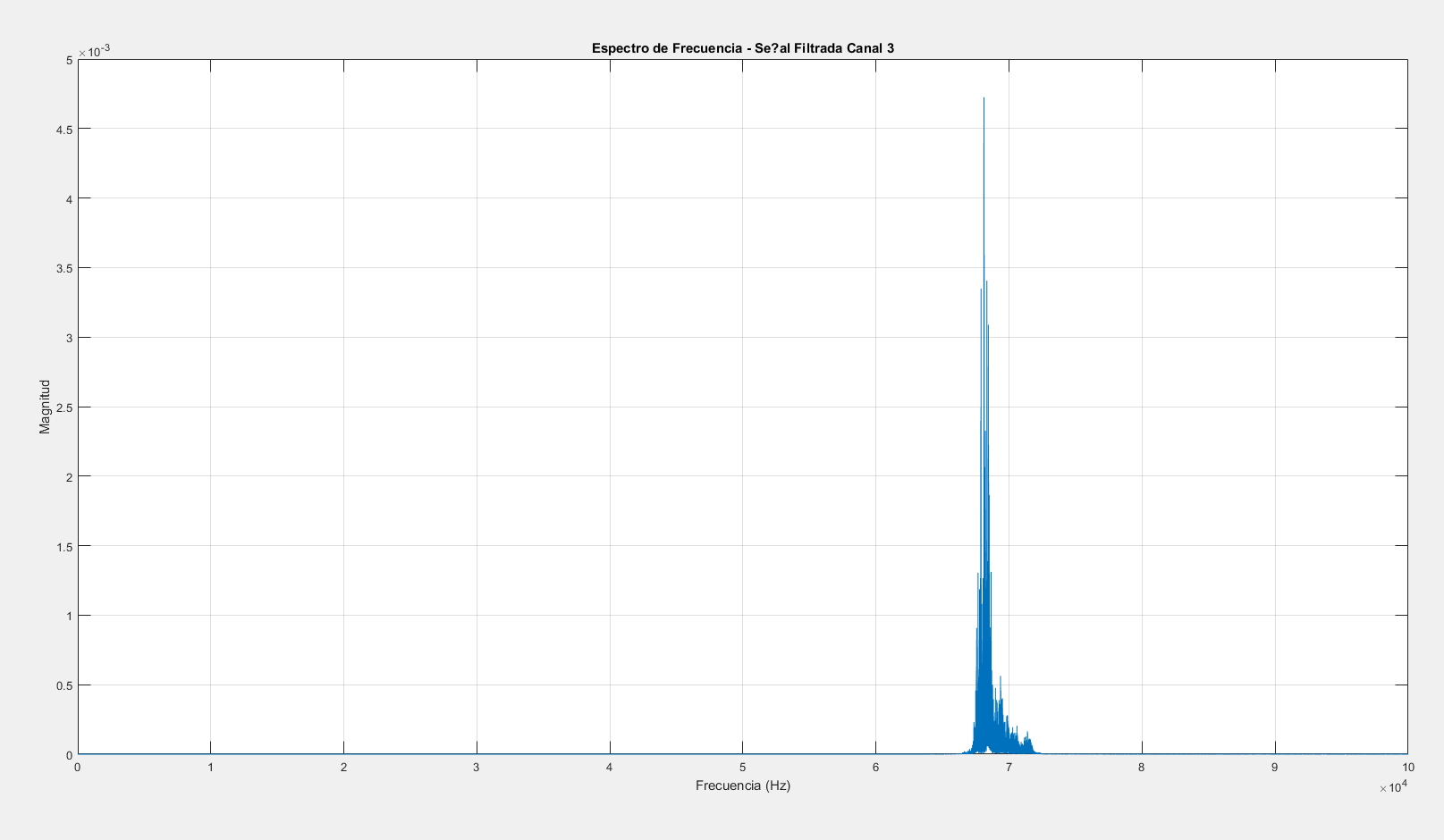
Canal 2:



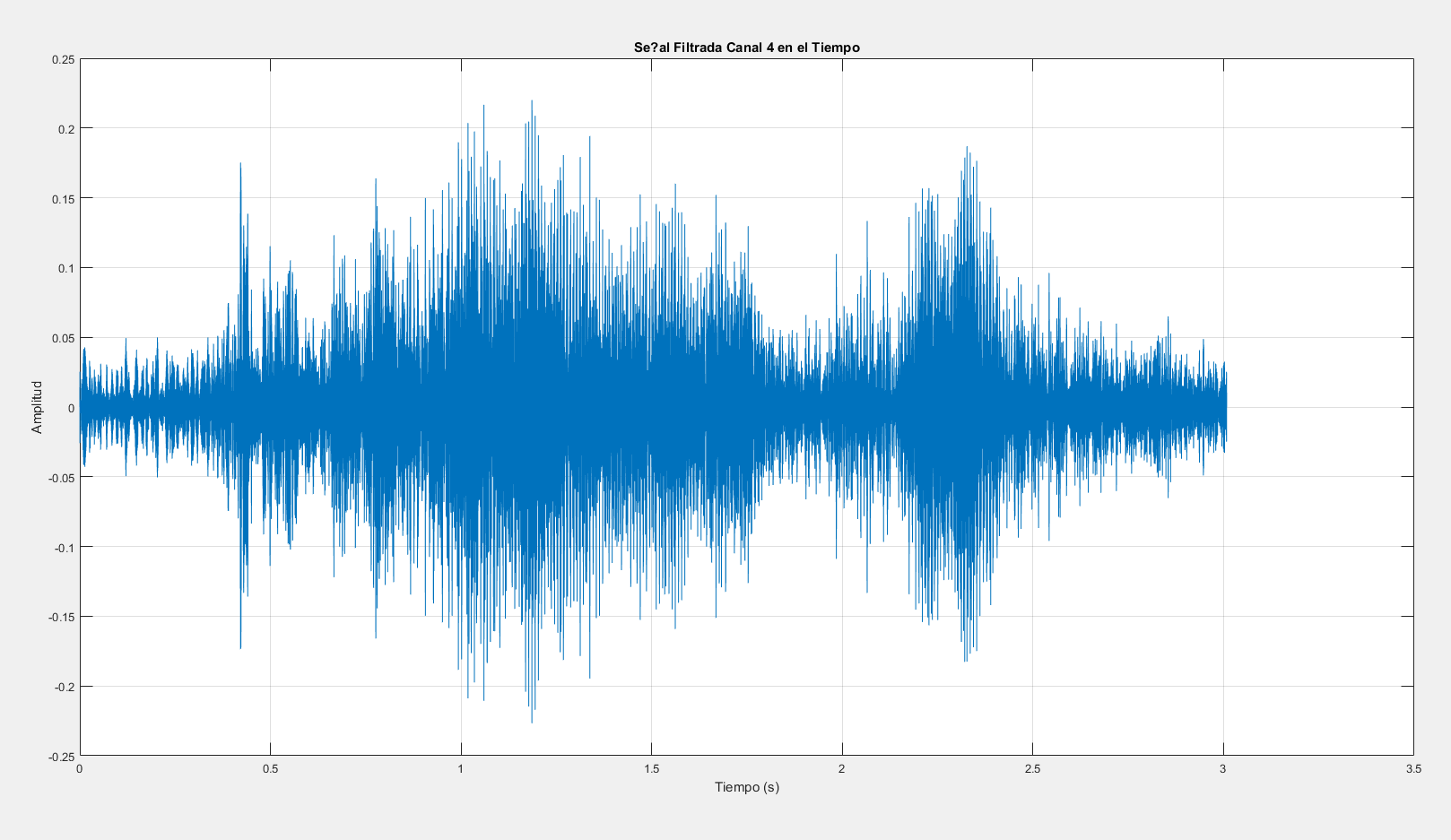


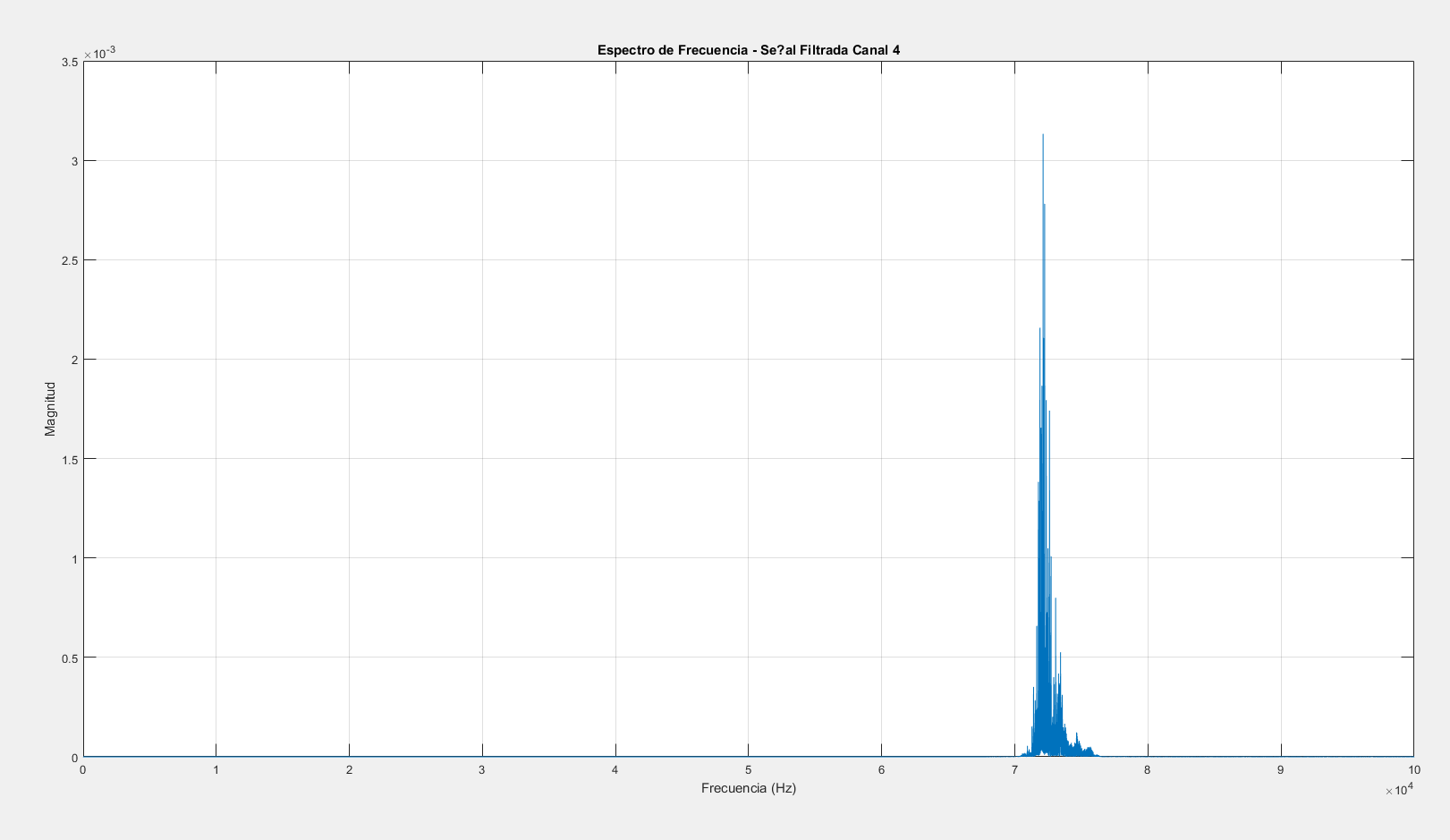
Canal 3:





Canal 4:





*Código utilizado:*

% Configuraci?n inicial

Fs = 328000; % Frecuencia de muestreo

archivosAudio = {'ernesto328k.wav', 'hector328k.wav', 'juanjo328k.wav', 'santi328k.wav'};

frecuenciasPortadoras = [60000, 64000, 68000, 72000]; % Frecuencias portadoras

% Modulaci?n y filtrado de las se?ales

senalesFiltradas = cell(1, length(archivosAudio)); % Inicializar

for k = 1:length(archivosAudio)

% Leer cada archivo de audio

[Xk\_n, ~] = audioread(archivosAudio{k});

n = (0:length(Xk\_n)-1)'; % Vector de muestras

% Calcular la frecuencia angular

omega\_k = 2 \* pi \* frecuenciasPortadoras(k) / Fs;

% Modulaci?n

senalModulada = Xk\_n .\* cos(omega\_k \* n);

% Dise?o del filtro pasa-altos para este canal

Fc = frecuenciasPortadoras(k); % Frecuencia de corte para este canal

Wn = Fc / (Fs / 2); % Normalizaci?n

orden = 20; % Orden del filtro

Rp = 1; % Ondulaci?n permitida en la banda de paso (en dB)

[b, a] = cheby1(orden, Rp, Wn, 'high'); % Filtro pasa-altos Chebyshev

% Aplicar el filtro pasa-altos

senalesFiltradas{k} = filter(b, a, senalModulada);

% Guardar las se?ales filtradas

audiowrite(['filtrada\_', archivosAudio{k}], senalesFiltradas{k}, Fs);

% Graficar amplitud vs tiempo

t = (0:length(senalesFiltradas{k})-1) / Fs; % Vector de tiempo

figure;

plot(t, senalesFiltradas{k});

xlabel('Tiempo (s)');

ylabel('Amplitud');

title(['Se?al Filtrada Canal ', num2str(k), ' en el Tiempo']);

grid on;

% Espectro de frecuencia de cada se?al filtrada

L = length(senalesFiltradas{k});

Y = fftshift(fft(senalesFiltradas{k}));

f = (-L/2:L/2-1) \* (Fs / L); % Eje de frecuencia en Hz

P = abs(Y) / L;

% Graficar espectro individual

figure;

plot(f, P);

xlabel('Frecuencia (Hz)');

ylabel('Magnitud');

title(['Espectro de Frecuencia - Se?al Filtrada Canal ', num2str(k)]);

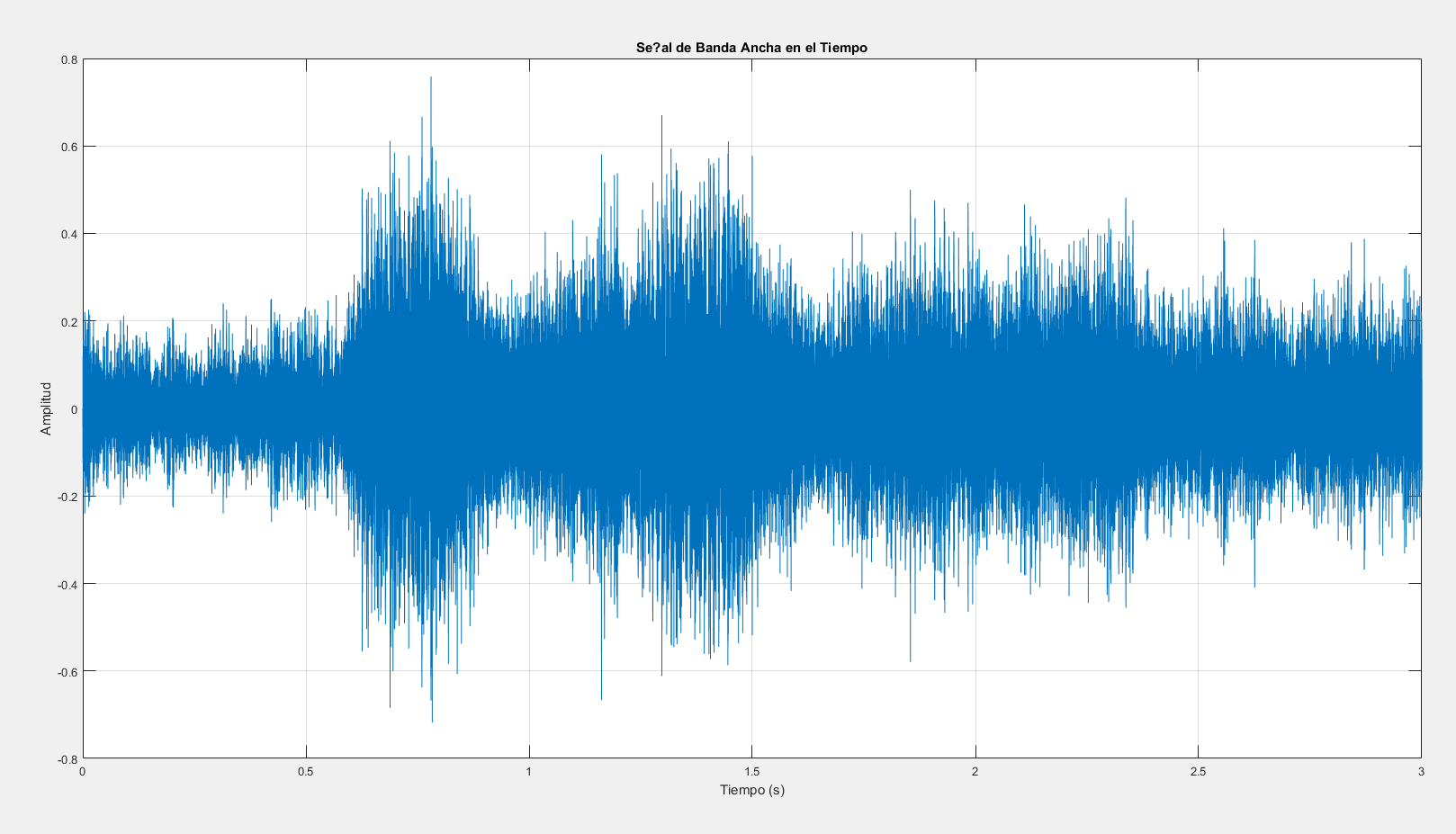
grid on;

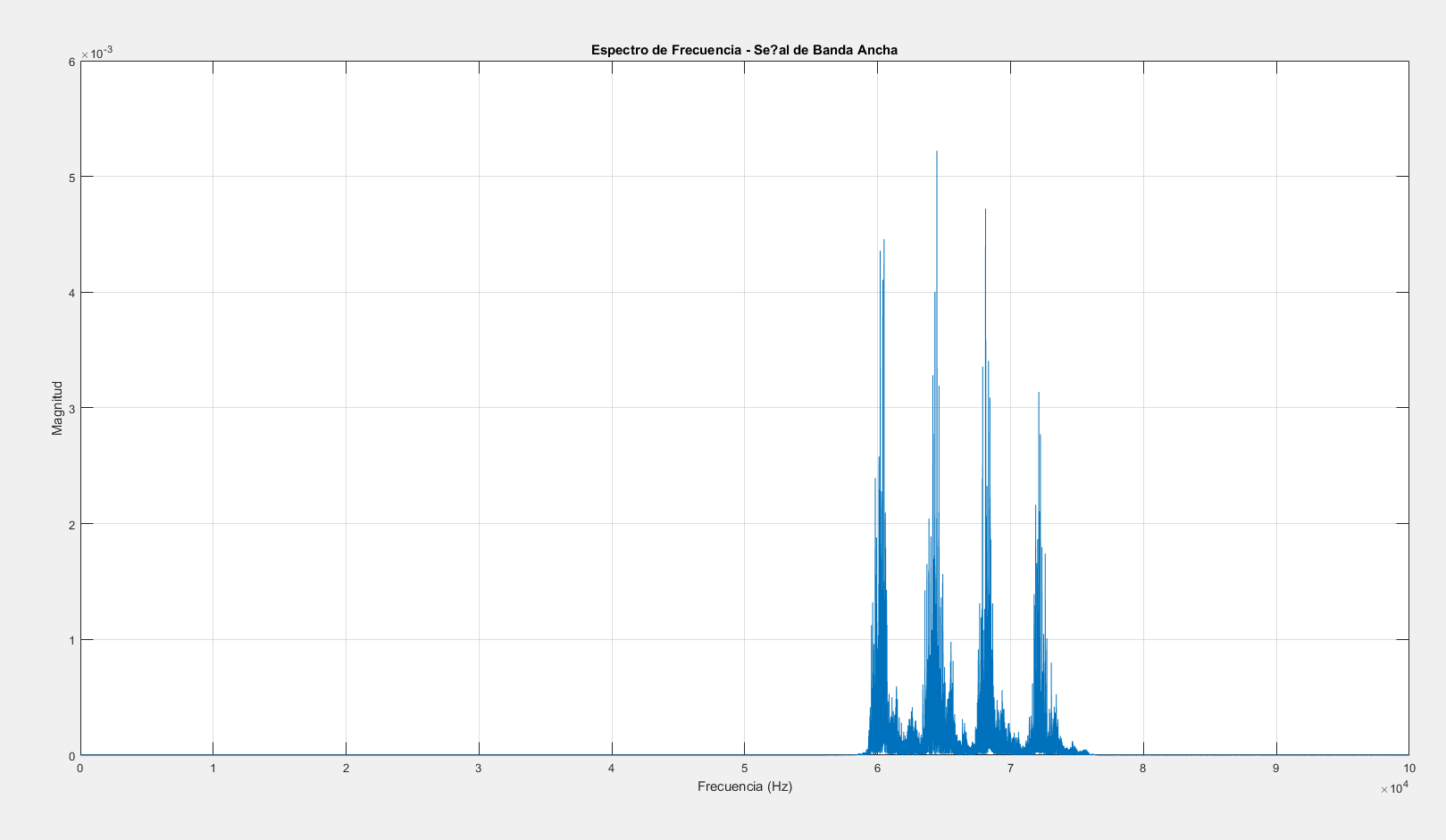
xlim([0, 100000]); % Ajuste

end

*Observaciones:* Efectivamente, como podemos apreciar (en mayor medida) en los gráficos de los espectros de frecuencia, la señal de cada canal tiene suprimida su BLI gracias al filtro que diseñamos. Cabe resaltar que suprimir dicha banda no afecta la información de nuestra señal ya que la misma se encuentra repetida en la banda lateral superior (BLS) que estamos transmitiendo. En todo caso, gracias a dicha supresión estamos evitando enviar información redundante y reducimos en gran medida la potencia necesaria para la transmisión.

h-i)





*Codigo utilizado:*

% Sumar las se?ales filtradas para obtener la se?al de banda ancha

senalBandaAncha = zeros(size(senalesFiltradas{1})); % Inicializar

for k = 1:length(senalesFiltradas)

senalBandaAncha = senalBandaAncha + senalesFiltradas{k};

end

% Guardar la se?al de banda ancha

audiowrite('senal\_banda\_ancha.wav', senalBandaAncha, Fs);

% Graficar la se?al de banda ancha y su espectro

t = (0:length(senalBandaAncha)-1) / Fs;

% Gr?fico en el dominio del tiempo

figure;

plot(t, senalBandaAncha);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Se?al de Banda Ancha en el Tiempo');

grid on;

xlim([0, 3]); % Ajuste

% Espectro de frecuencia

L = length(senalBandaAncha);

Y = fftshift(fft(senalBandaAncha));

f = (-L/2:L/2-1) \* (Fs / L); % Eje de frecuencia en Hz

P = abs(Y) / L;

figure;

plot(f, P);

xlabel('Frecuencia (Hz)');

ylabel('Magnitud');

title('Espectro de Frecuencia - Se?al de Banda Ancha');

grid on;

xlim([0, 100000]); % Ajuste

*Observaciones:* En la gráfica del espectro de frecuencia podemos apreciar nuestra nueva señal de banda ancha constituida por las señales de los cuatro canales filtrados.

**Actividad 2: Implementación del desmultiplexado**

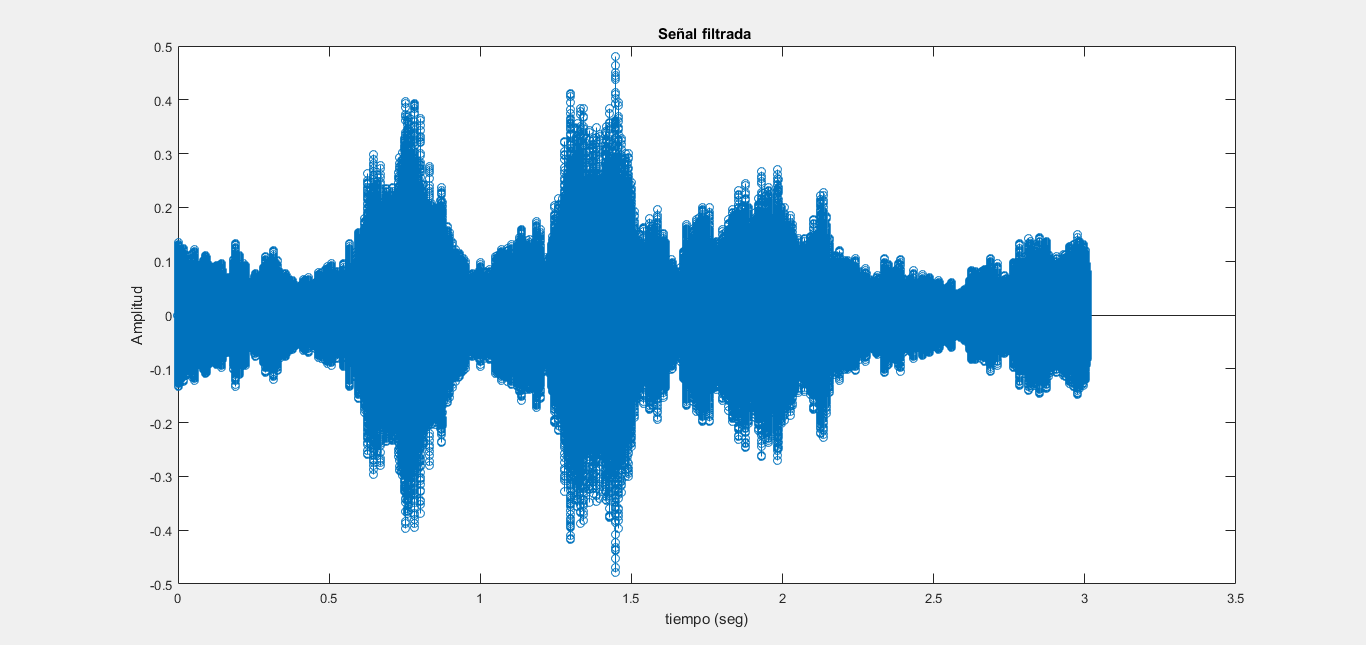
a) Diseñar el filtro digital recursivo (IIR) pasa banda para obtener la señal de banda lateral superior correspondiente cada canal de la portadora de banda ancha.

b) Utilizando la función “filter” de Matlab, procesar la señal portadora de banda ancha aplicándole los filtros diseñados para cada canal.

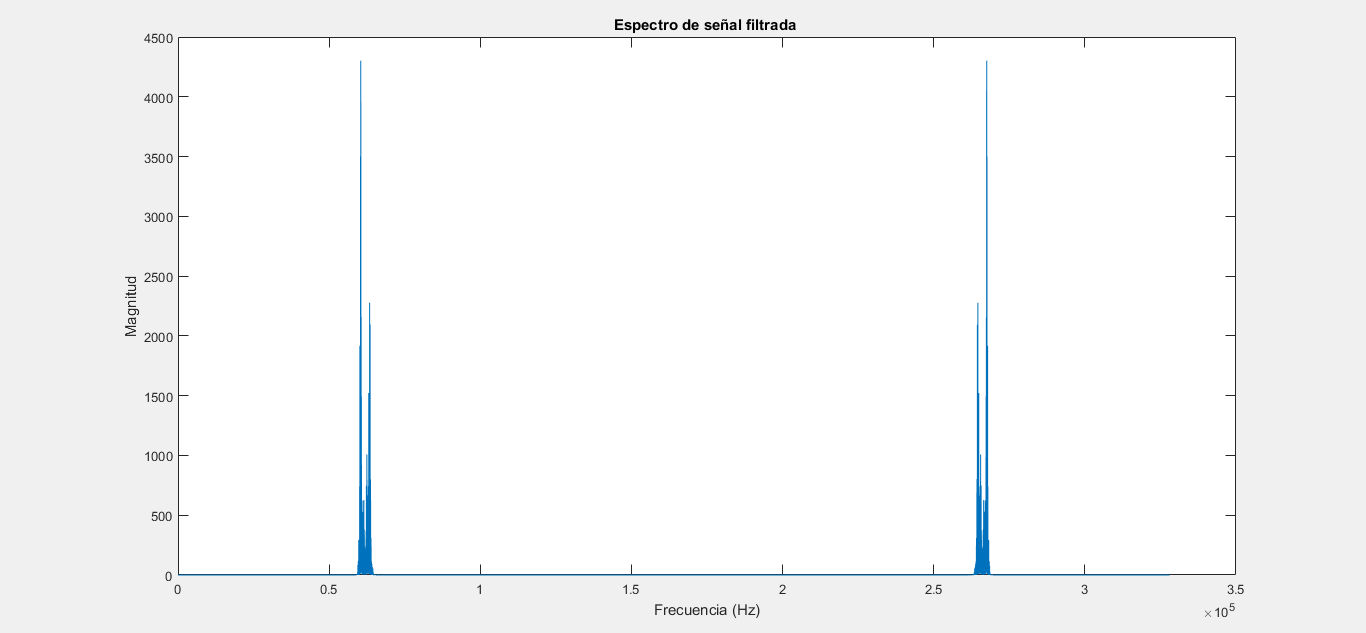
c) Representar gráficamente la señal obtenida y el espectro correspondiente para uno de los canales.

Canal 1:

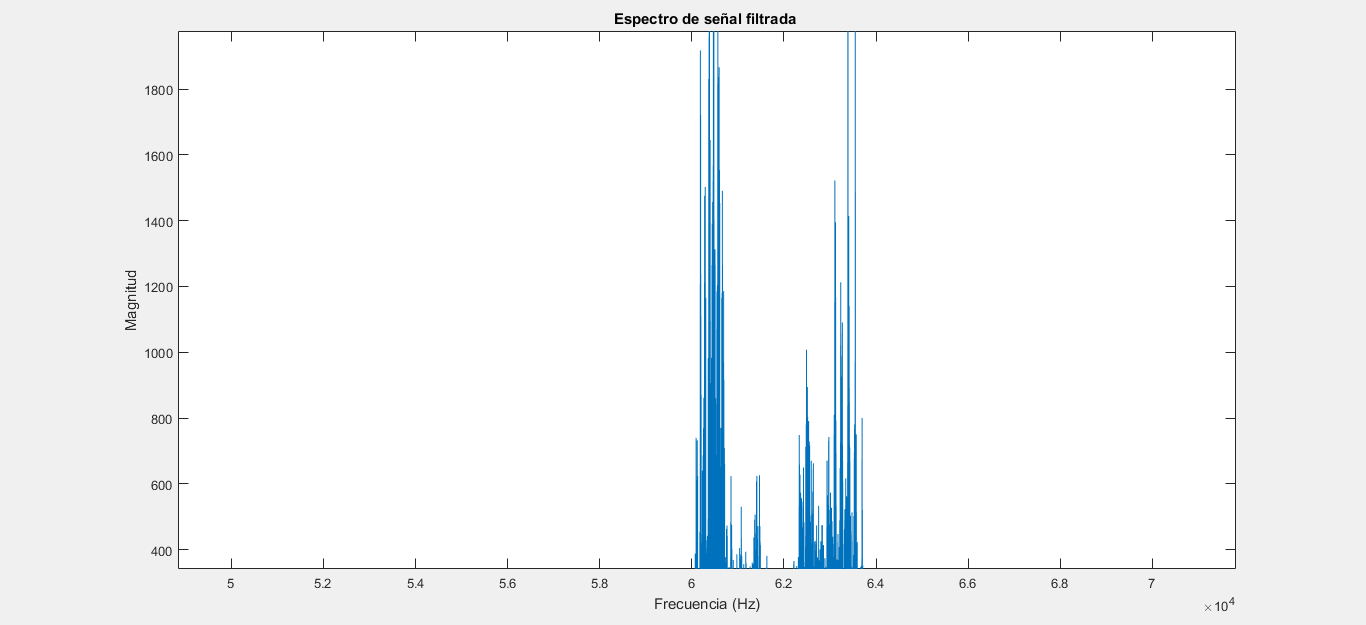
Señal de banda ancha filtrada



Espectro

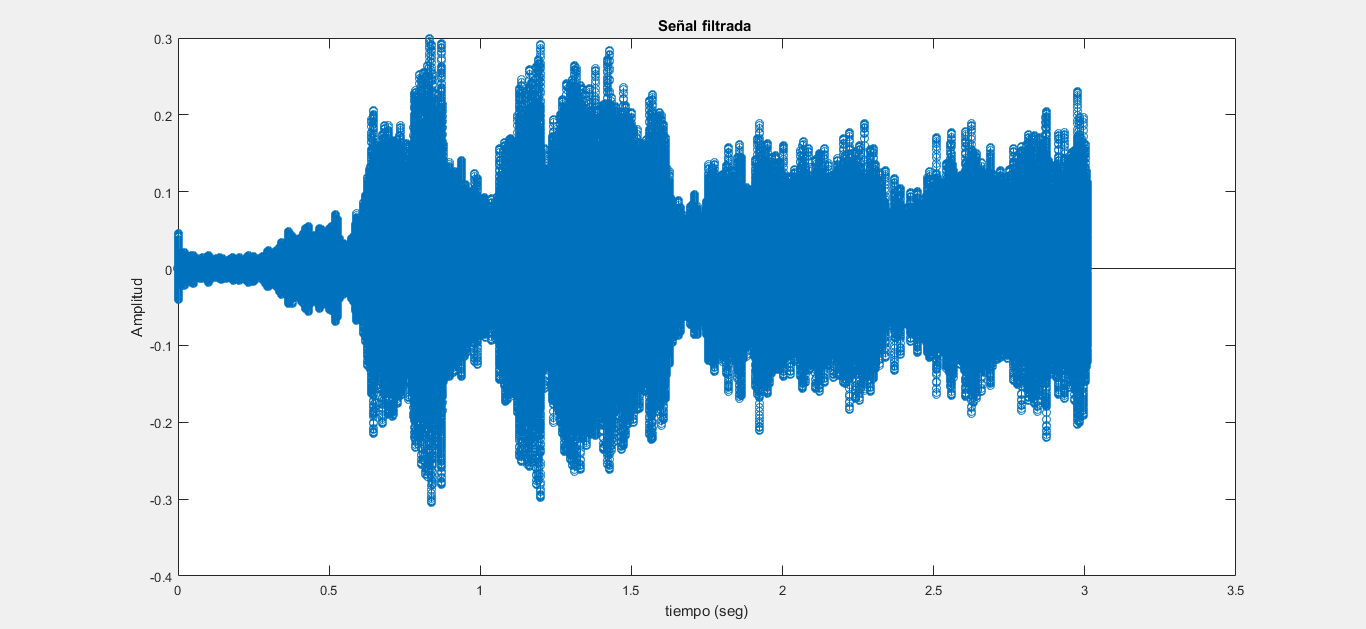


Zoom

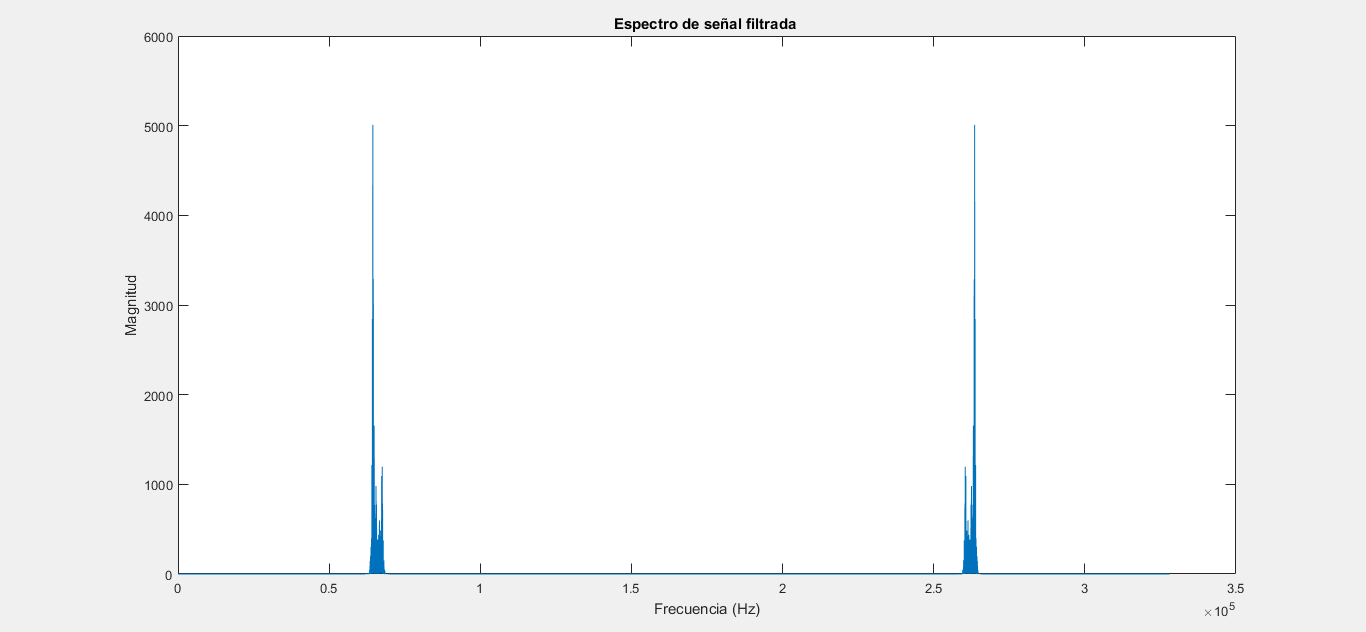


*CANAL 2*

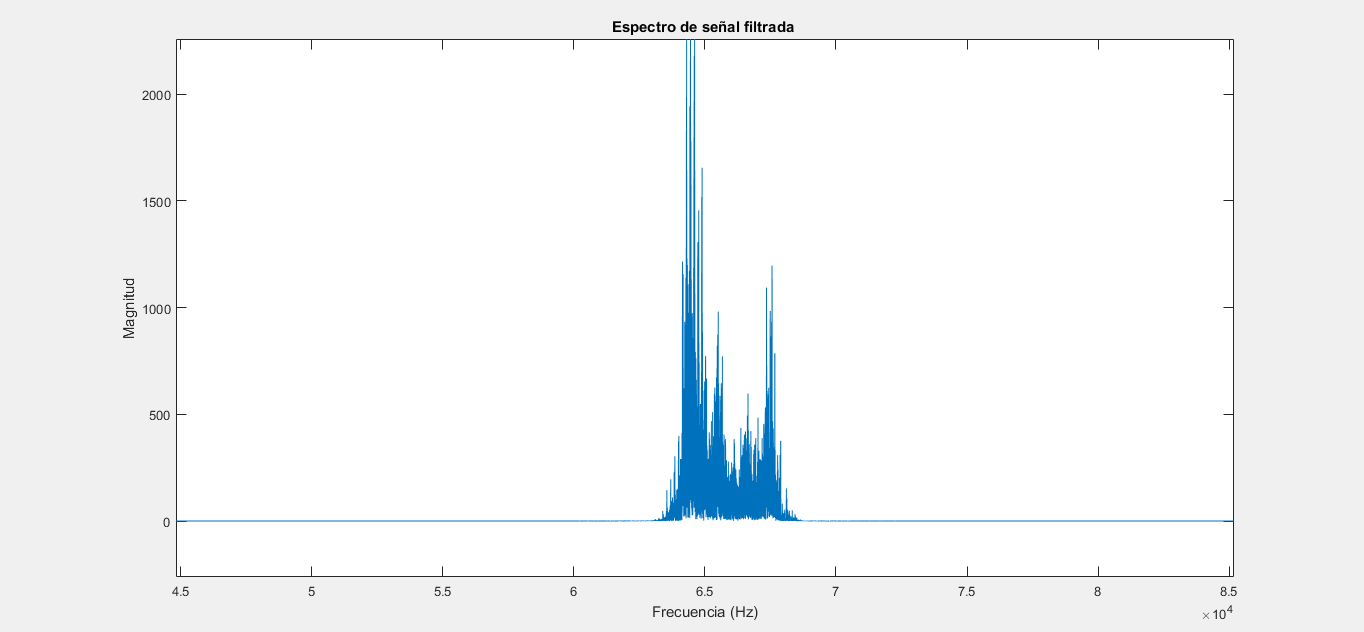
Señal de banda ancha filtrada



Espectro

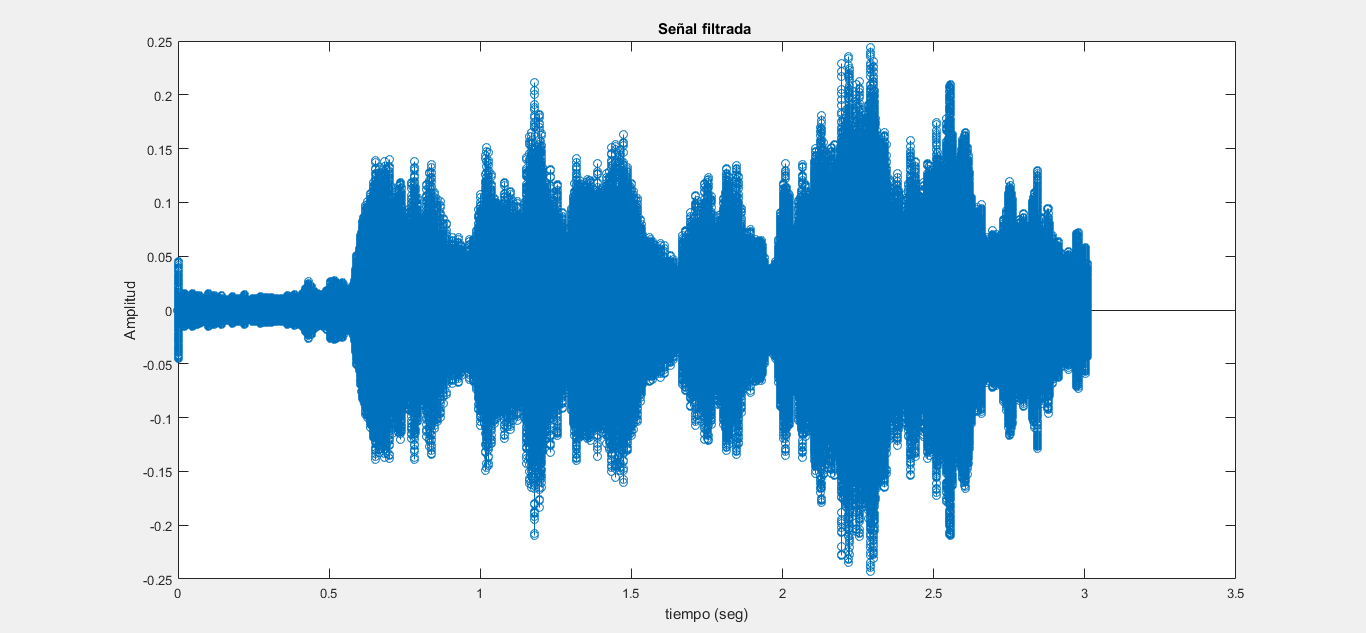


Zoom

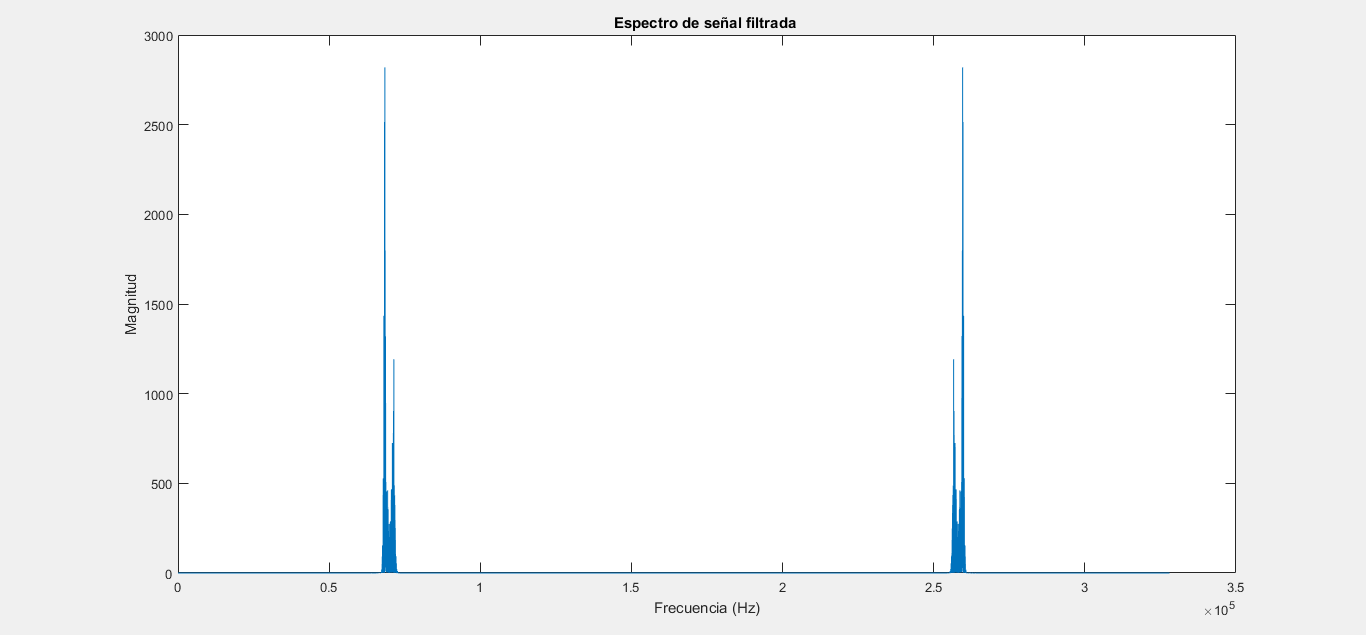


*CANAL 3*

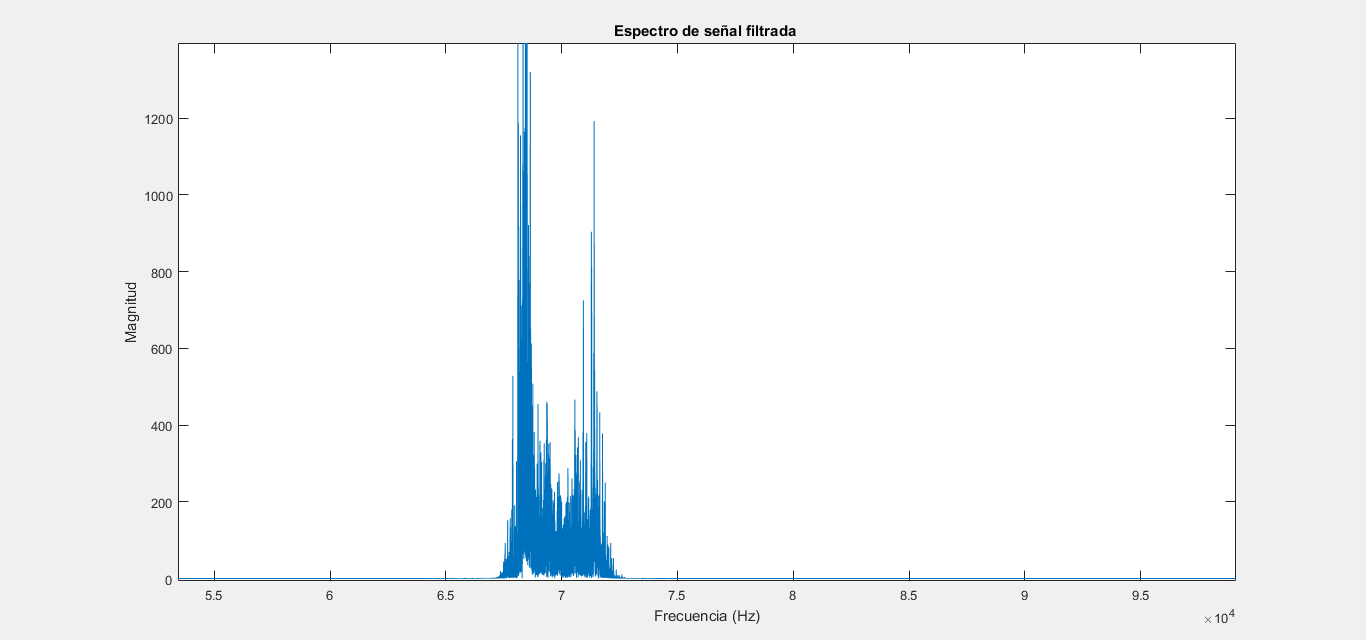
Señal de banda ancha filtrada



Espectro

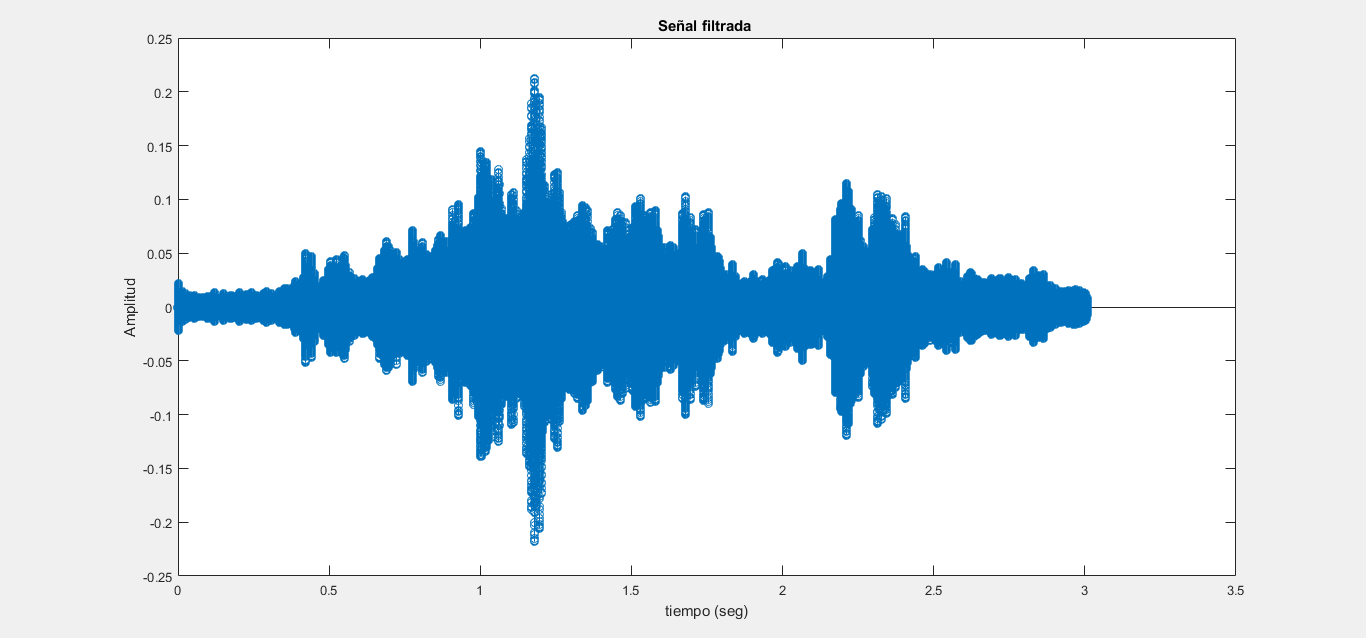


Zoom

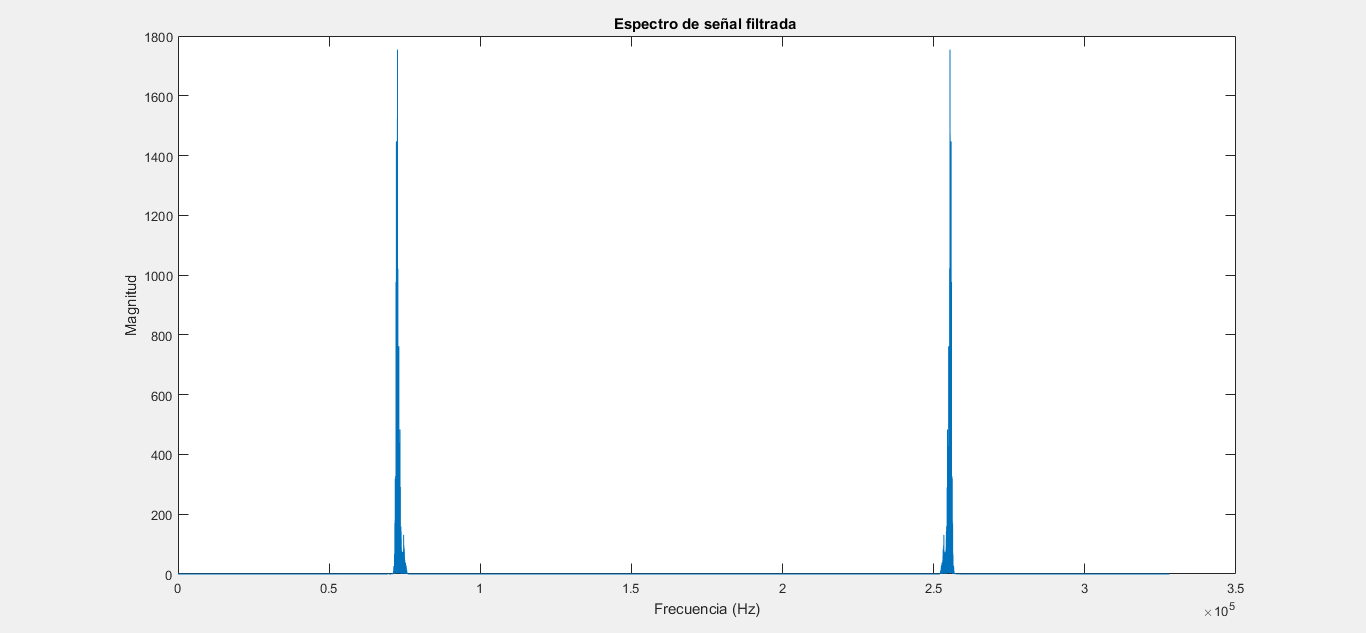


*CANAL 4*

Señal de banda ancha filtrada



Espectro



Zoom

