Universidad Nacional de Tucumán
Facultad de Ciencias Exactas y Tecnología
Carrera: Ingeniería Electrónica - Año 2024
Asignatura: PROCESAMIENTO DIGITAL DE SEÑALES (E7S)
Trabajo Práctico de Laboratorio N°3

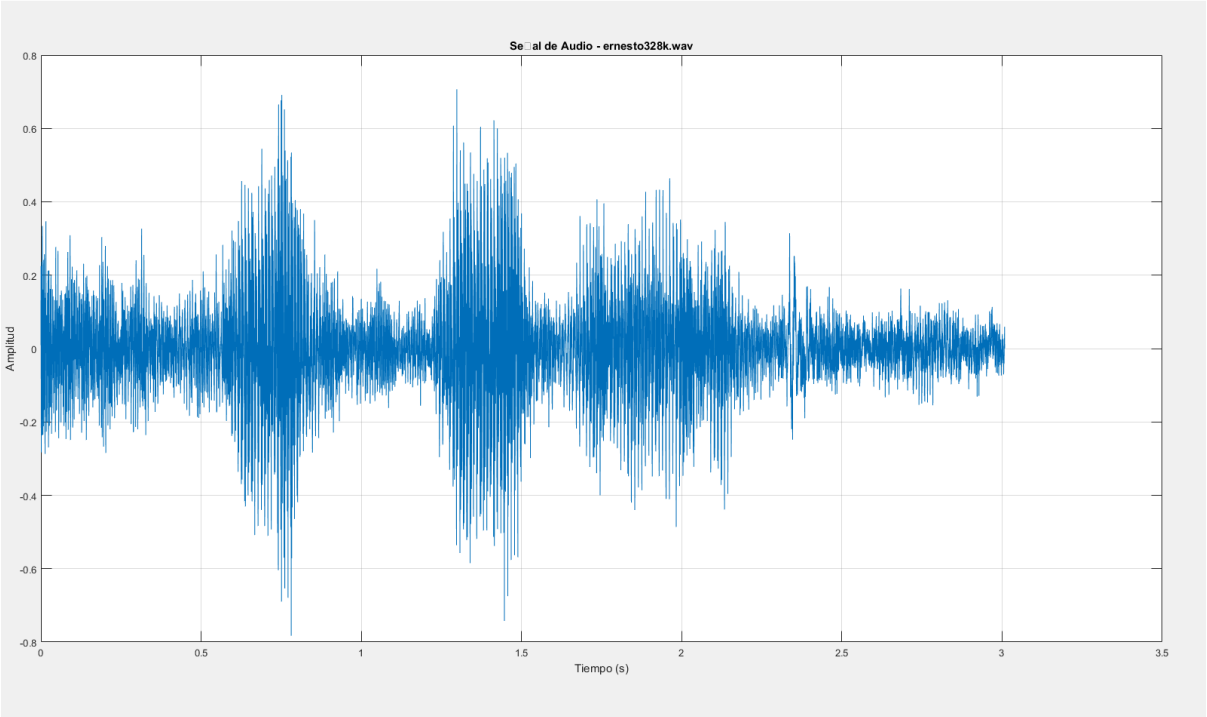
Integrantes: Ciccio Ernesto José
Geréz Jiménez Juan José Armando
Robles Héctor
Torres Juan Santiago Simón

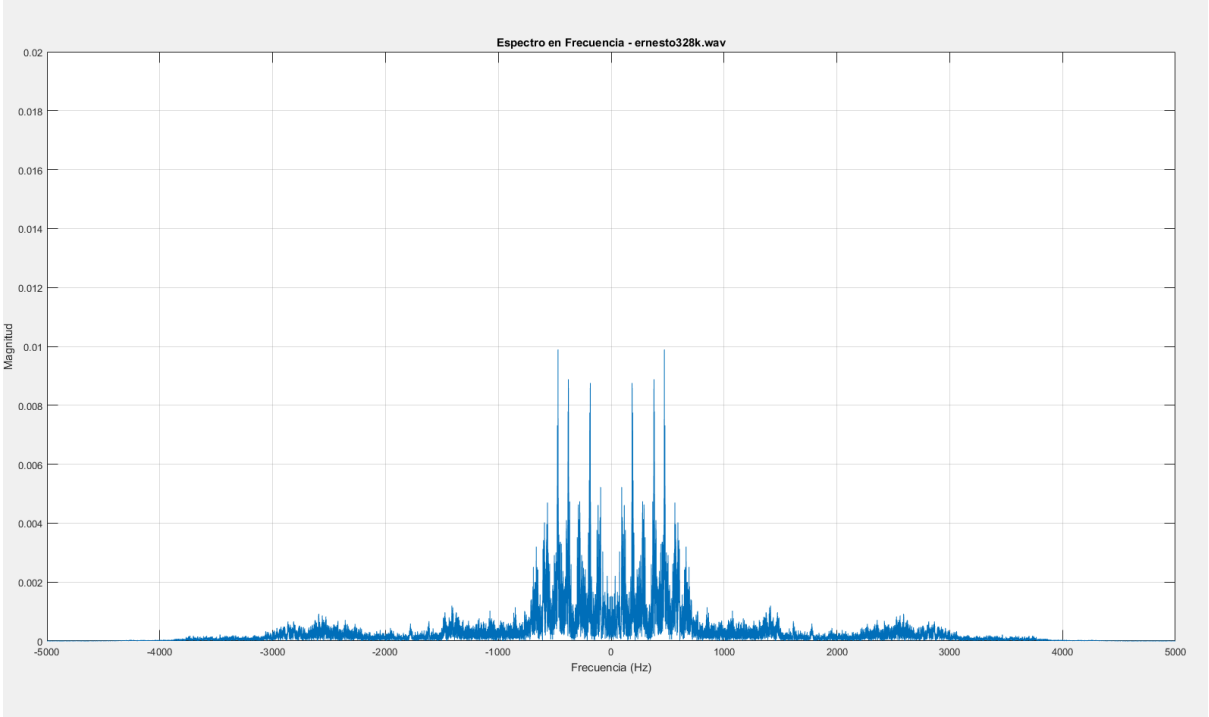
1. a) Para obtener los nuevos audios con la nueva frecuencia de muestreo de 328KHz escribimos el siguiente código utilizando la función “resample”:

```
[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);
```

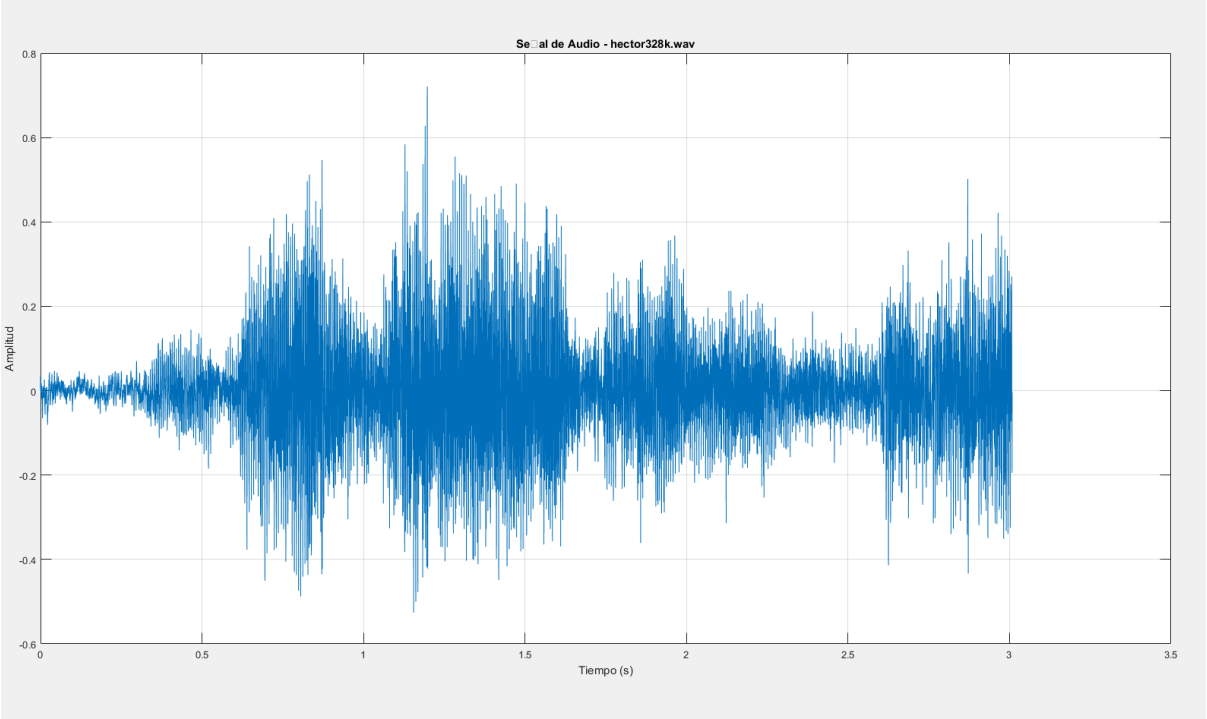
De esta forma obtenemos los archivos hector8k.wav, santi328k.wav, juanjo328k.wav y ernesto328k.wav.

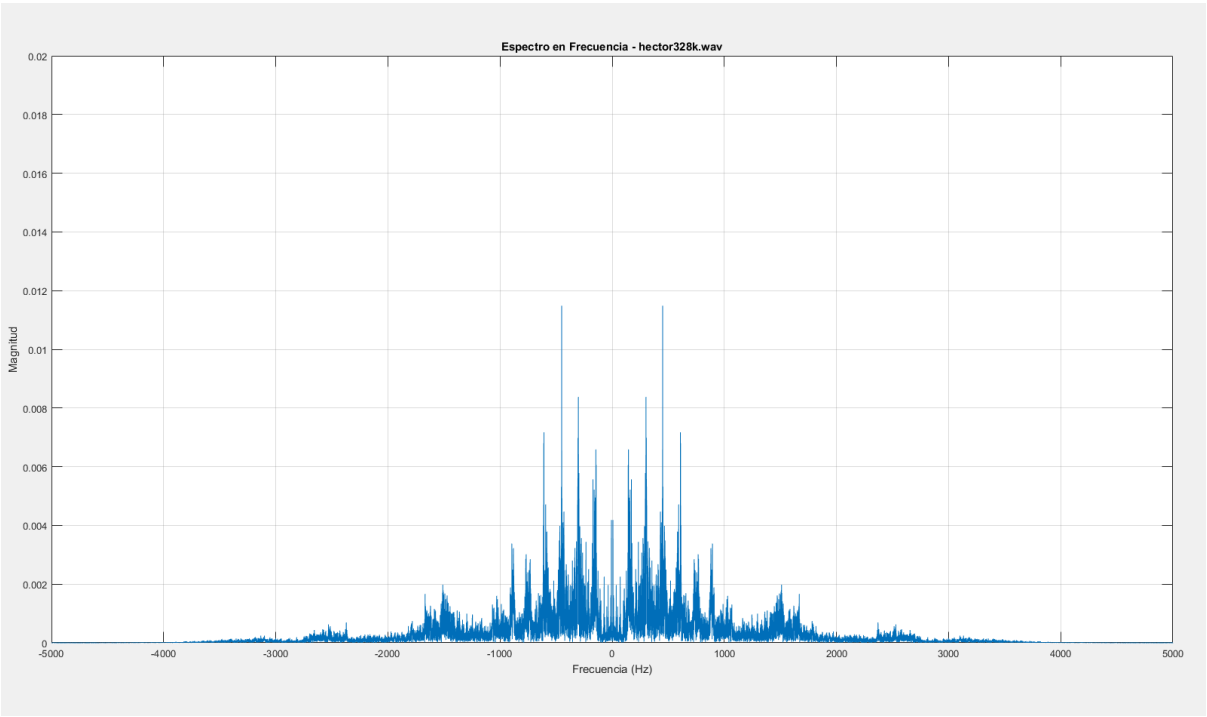
b)**ernesto328k.wav:**



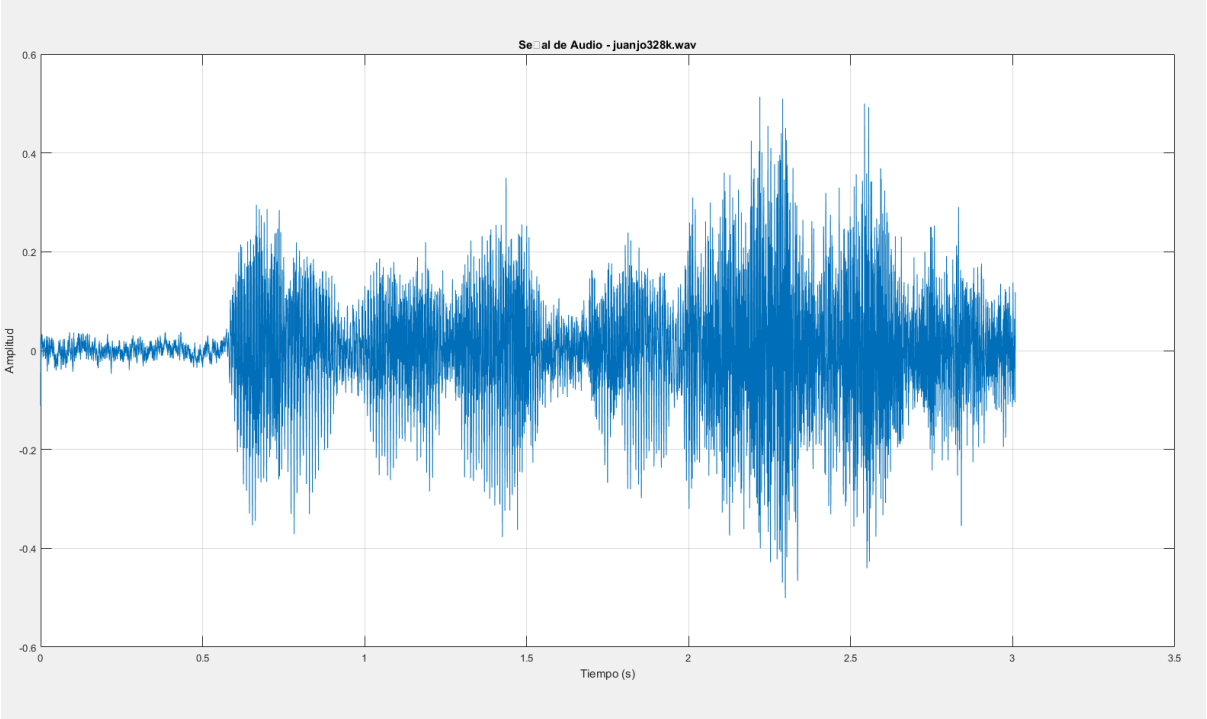


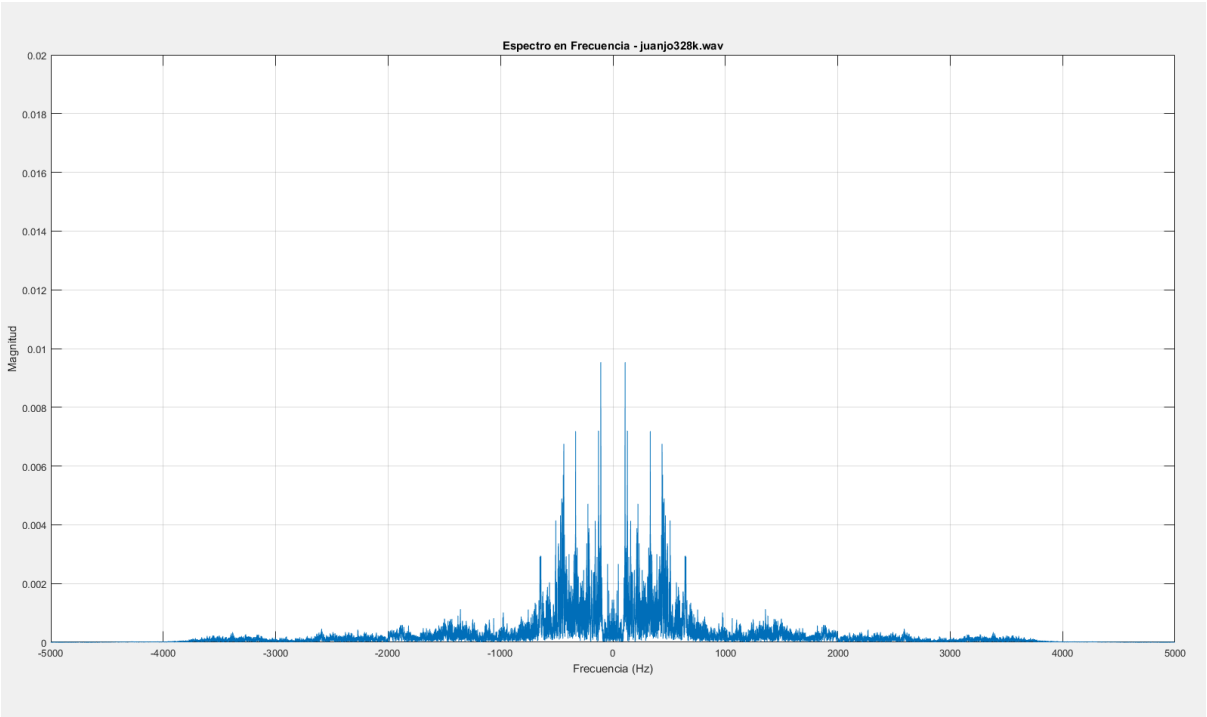
hector328k.wav:



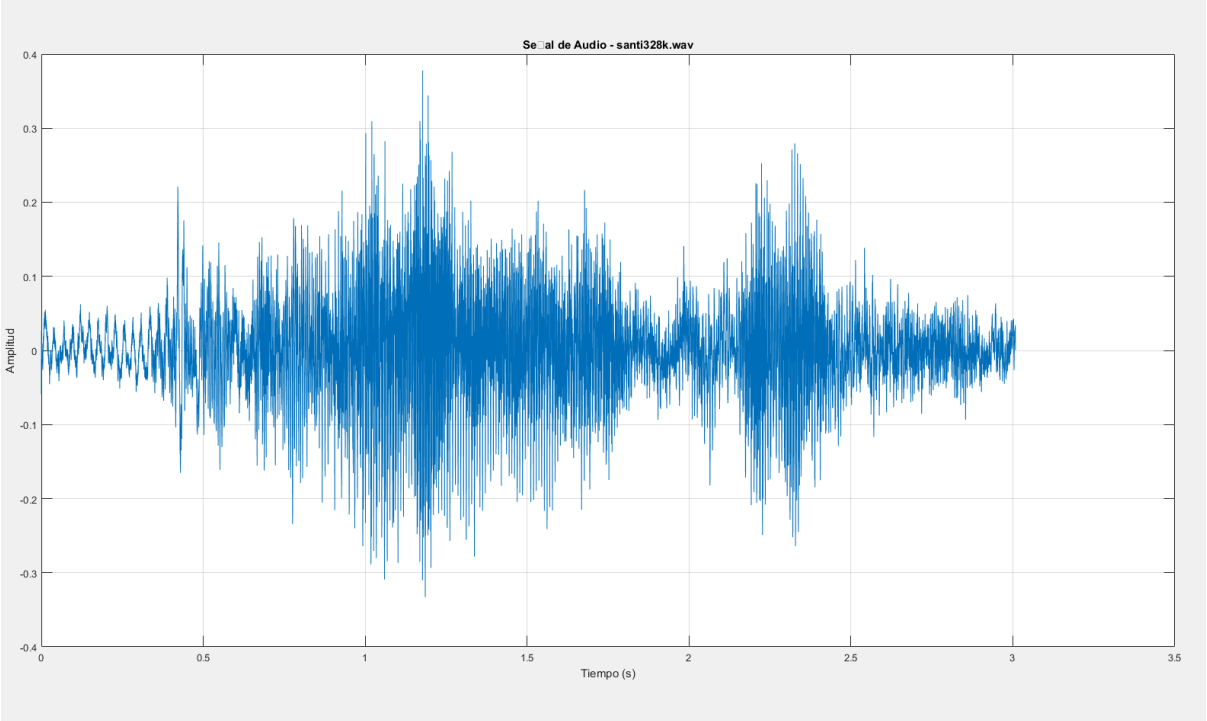


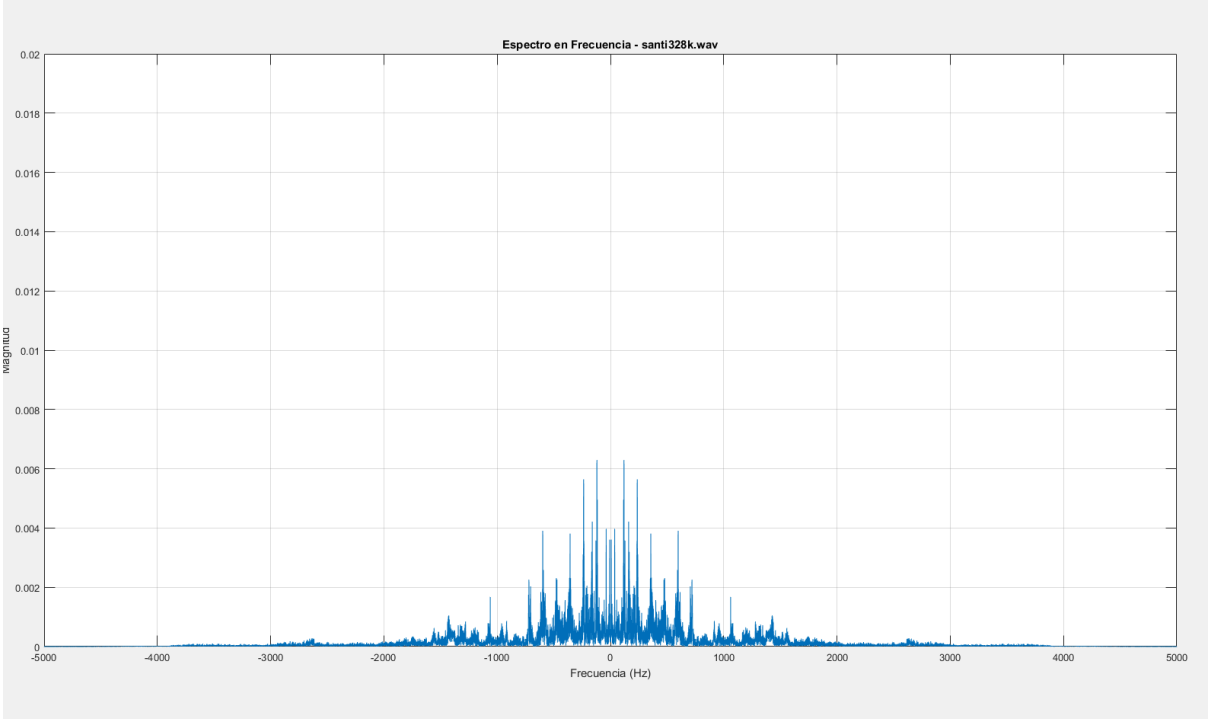
juanjo328k.wav:





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 $X'k[n] = Xk[n] * cos(Wk * n)$, Donde $X'k[n]$ representa a cada una de las señales moduladas, $Xk[n]$ 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)
```

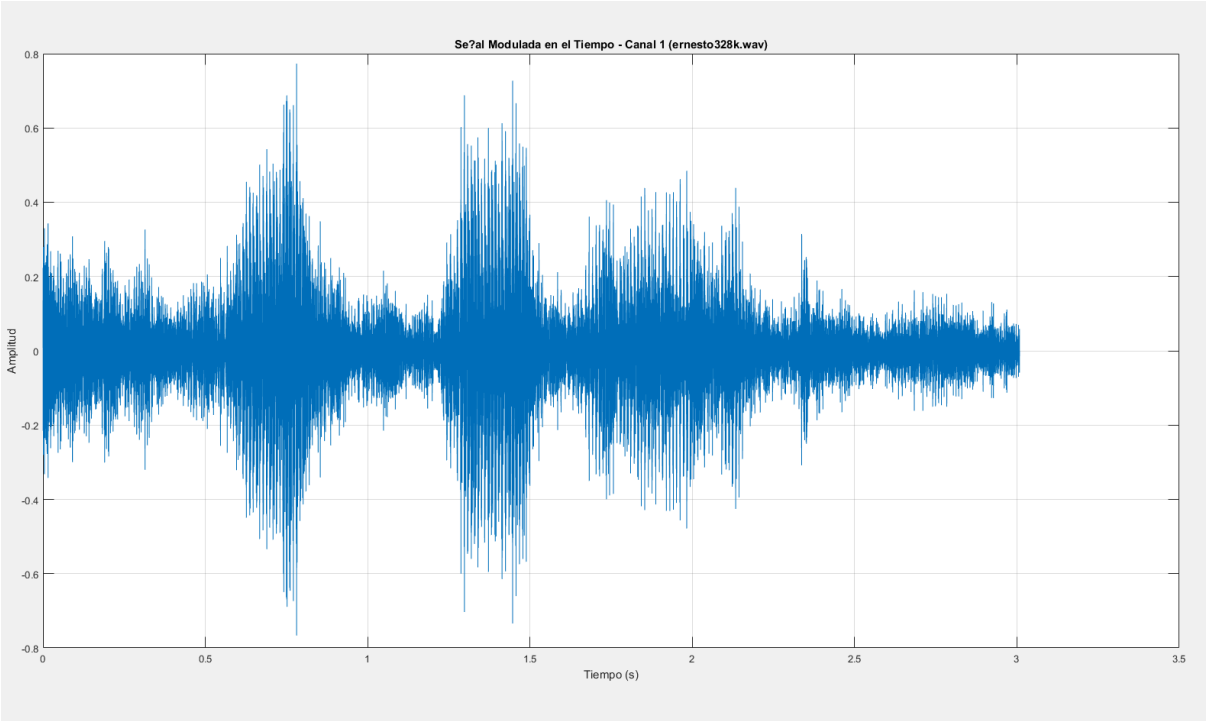
```
frecuenciasPortadoras = [60000, 64000, 68000, 72000];   % 60
kHz, 64 kHz, 68 kHz, 72 kHz

% 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
```

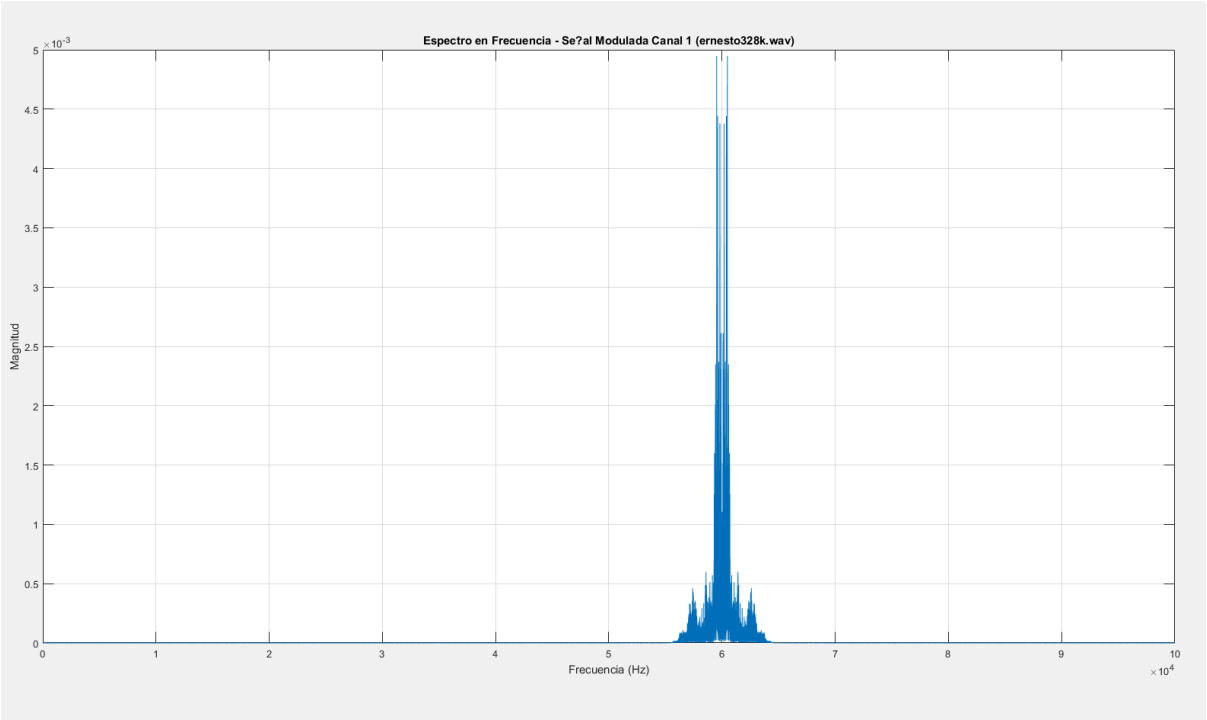
Con este código logramos multiplicar a cada una de las señales por $cos\,(Wk * n)$ logrando así desplazarlas en frecuencia.

d)

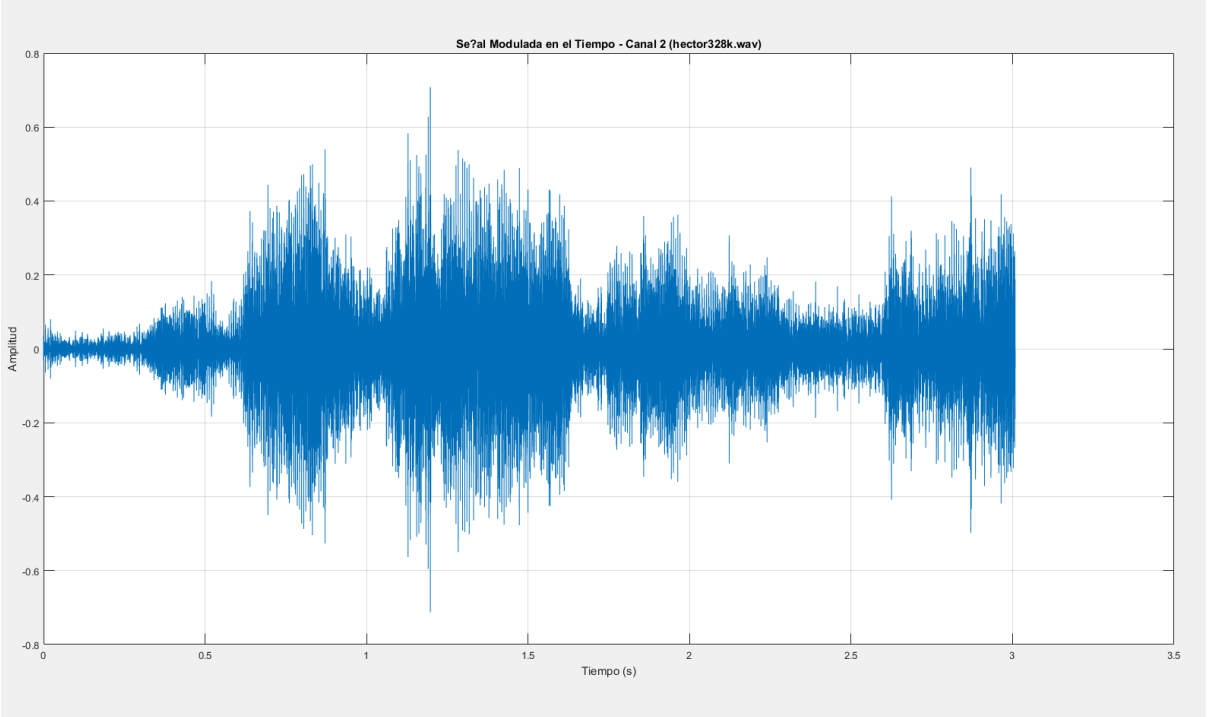
Las gráficas en tiempo y frecuencia de las señales moduladas en el apartado anterior serian:

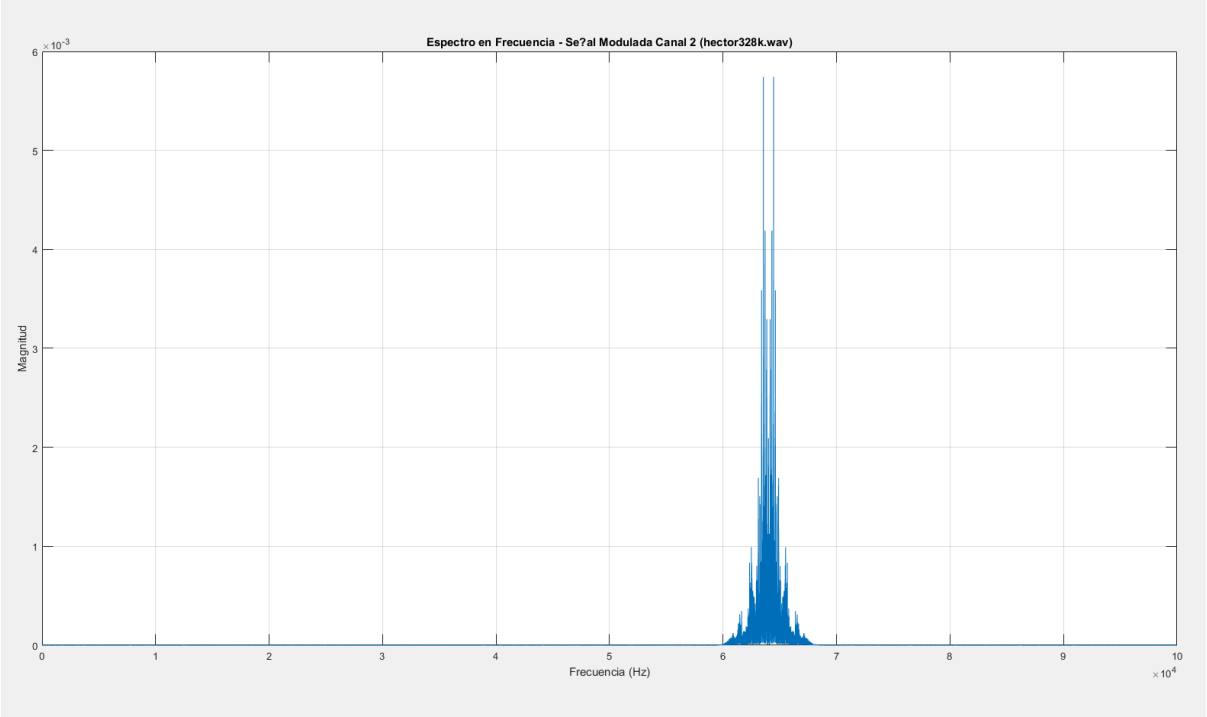
ernesto328K.wav modulado a 60 KHz:



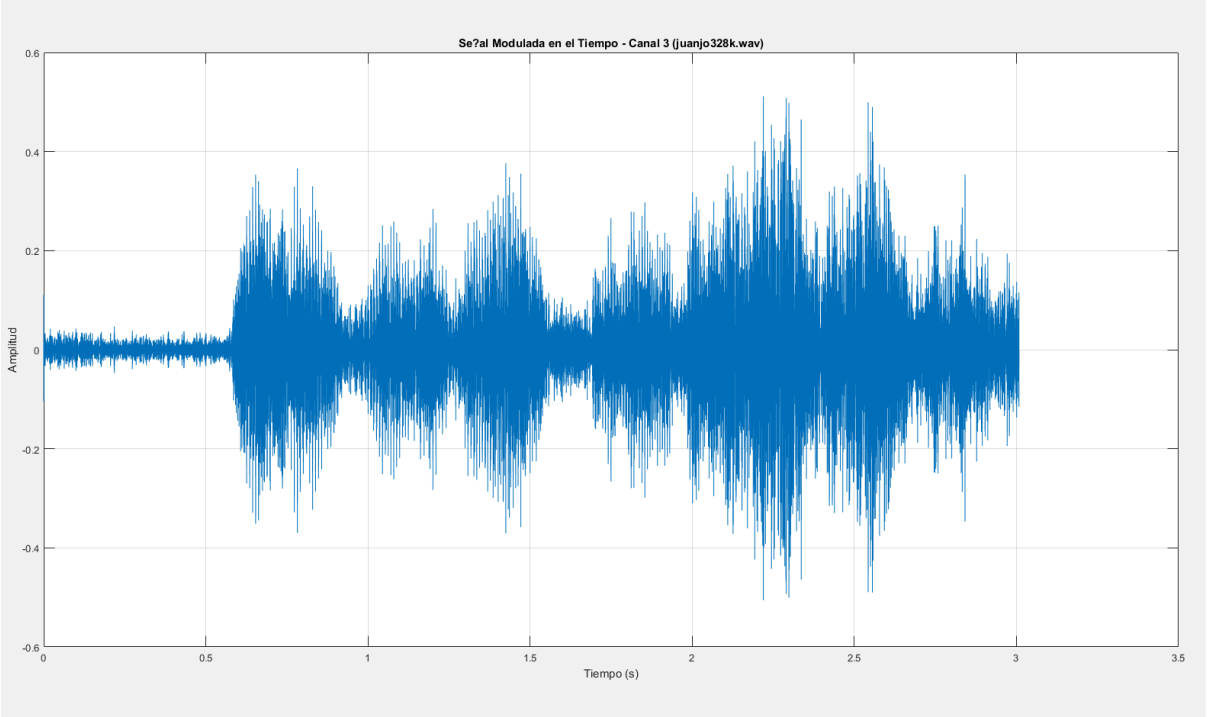


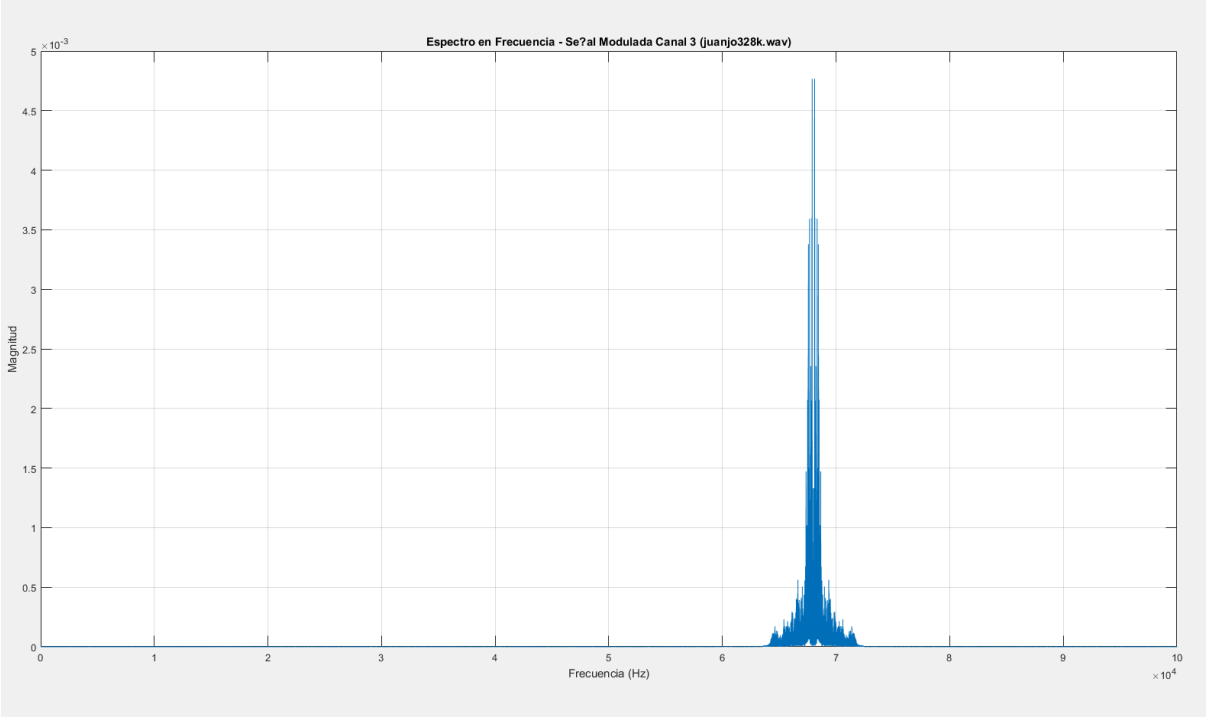
hector328k.wav modulado a 64 KHz:



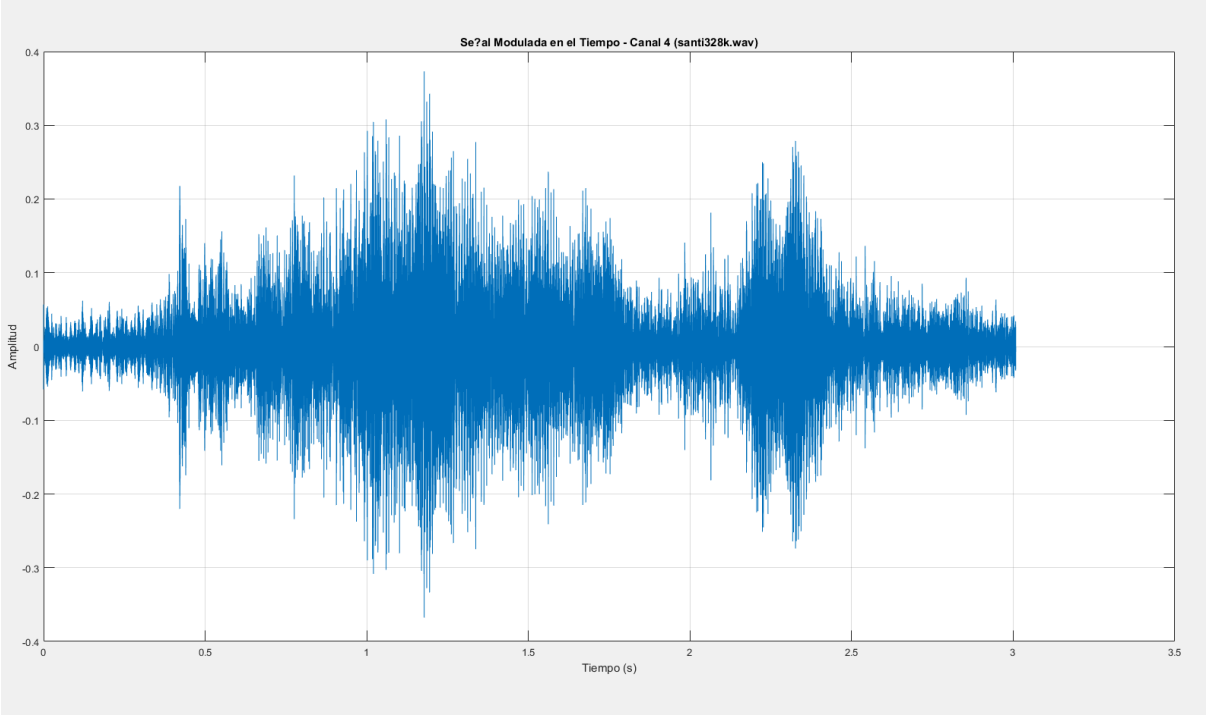


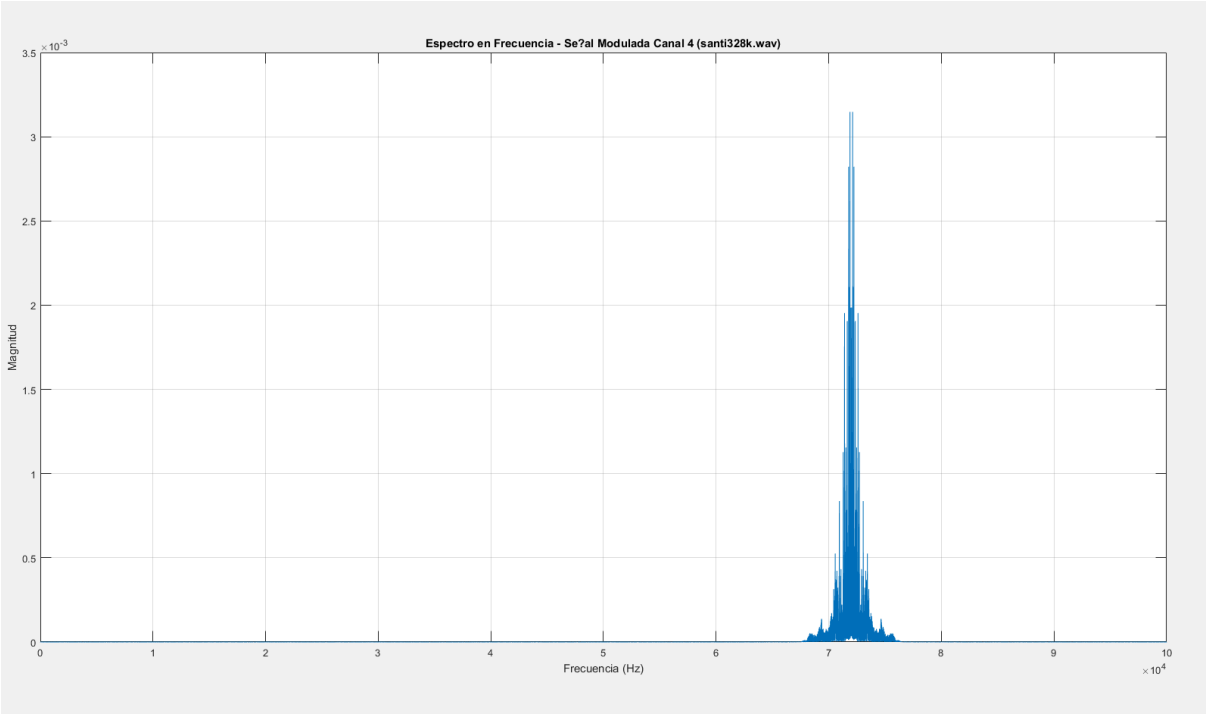
juanjo328K.wav modulado a 68 KHz:





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.

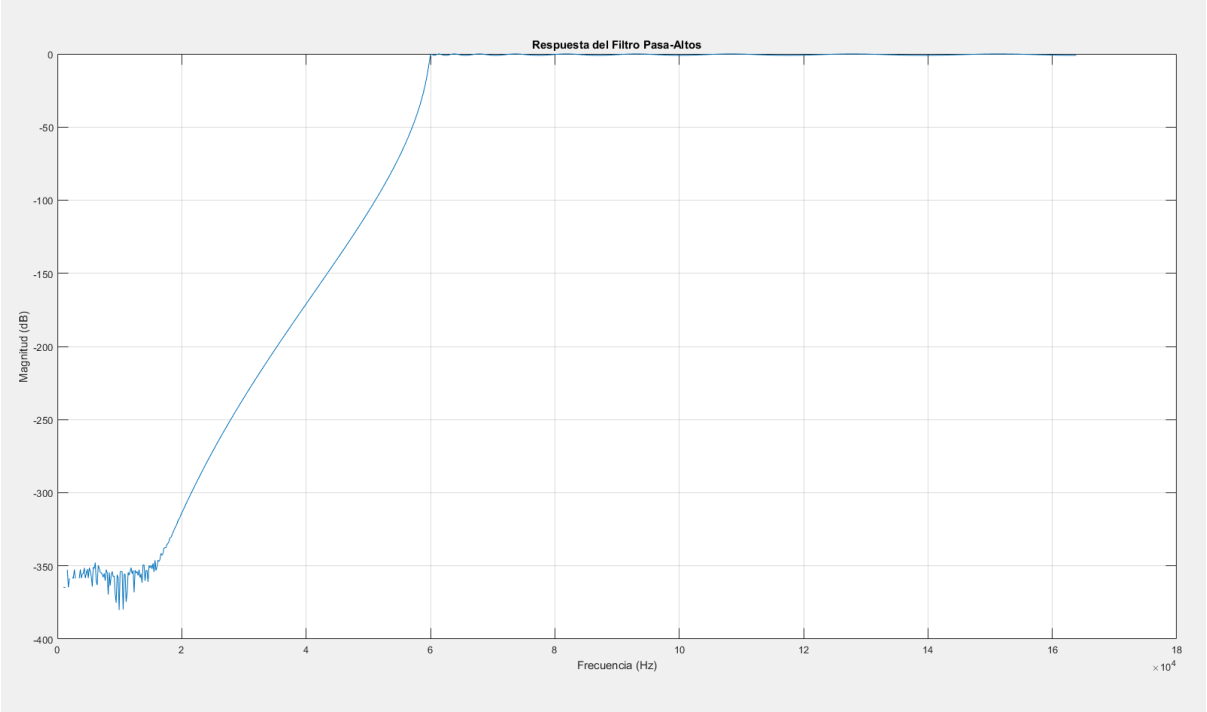
e)

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

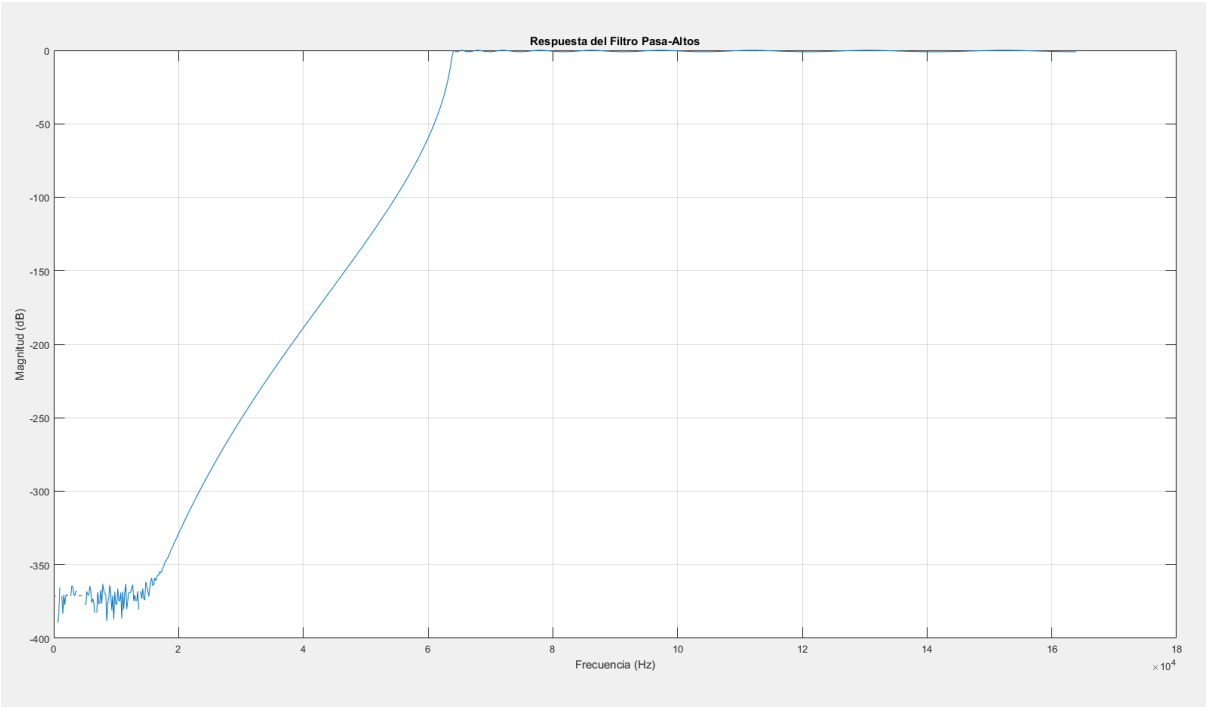
```
% Dise?o del filtro pasa-altos para eliminar la banda lateral inferior
Fc = 60000; % Frecuencia de corte (cambiar según corresponda)
Wn = Fc / (Fs / 2); % Normalizaci?n

% 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:

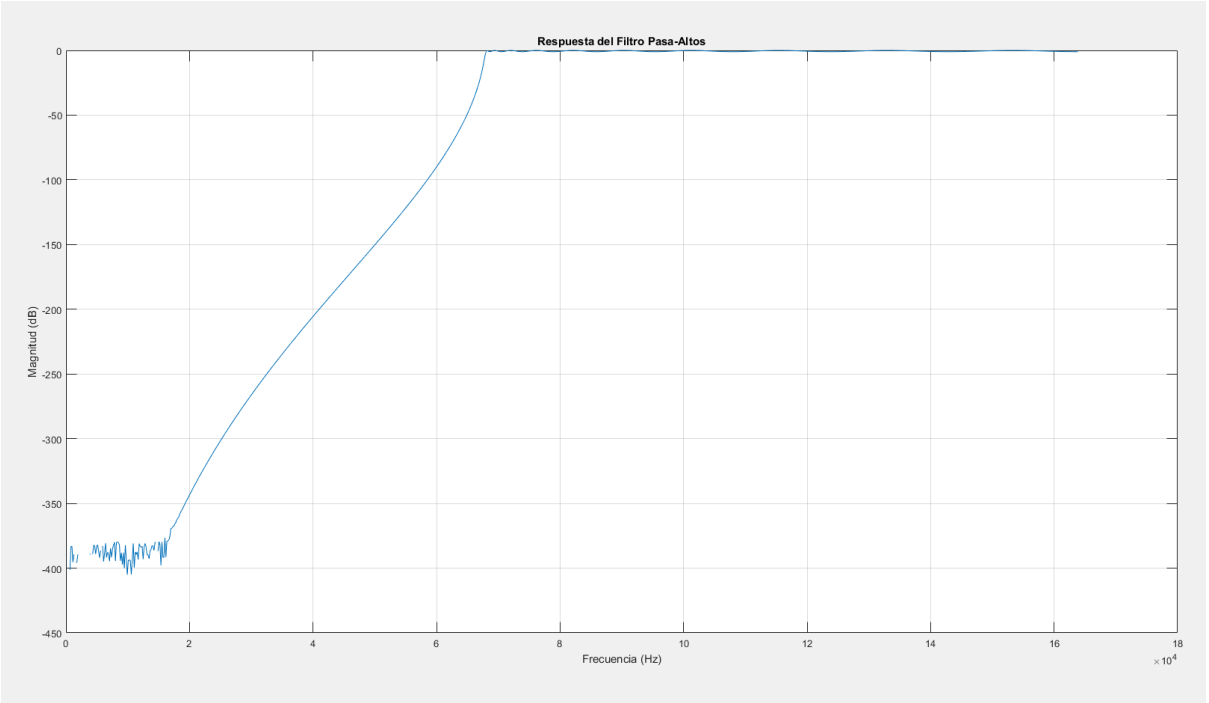
Filtro para canal 1 (Fcorte = 60Khz):



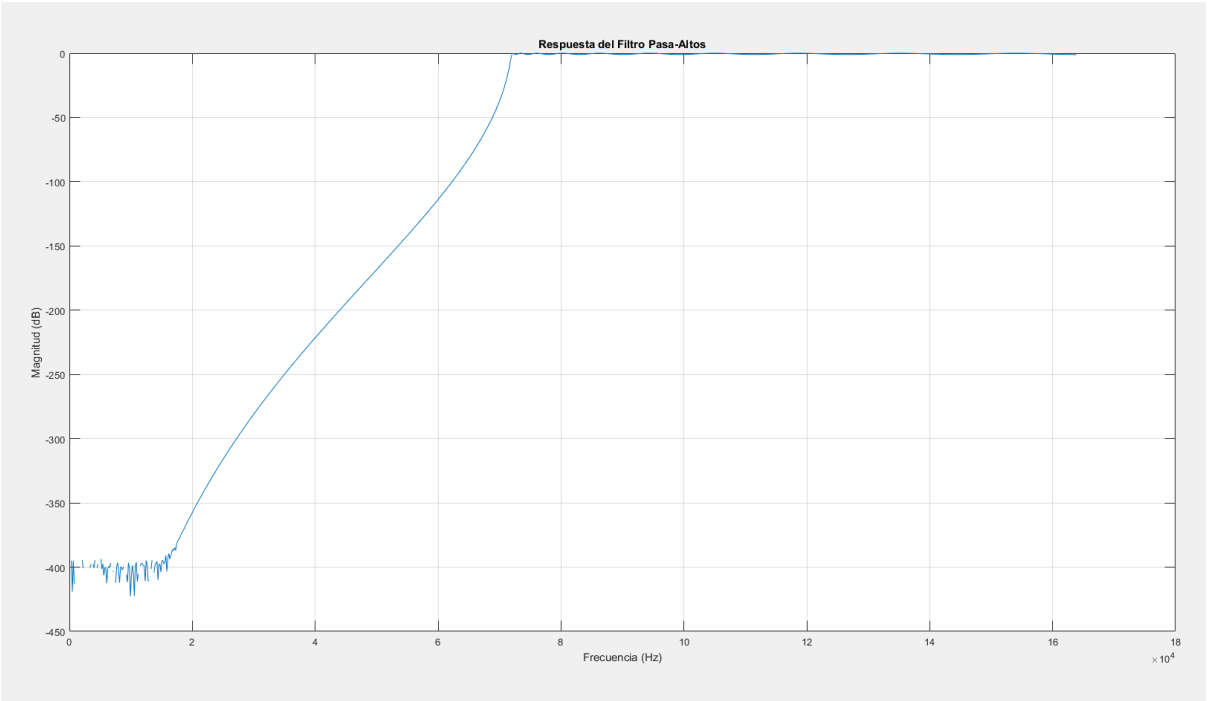
Filtro para canal 2 (Fcorte = 64Khz):



Filtro para canal 3 (Fcorte = 68Khz):

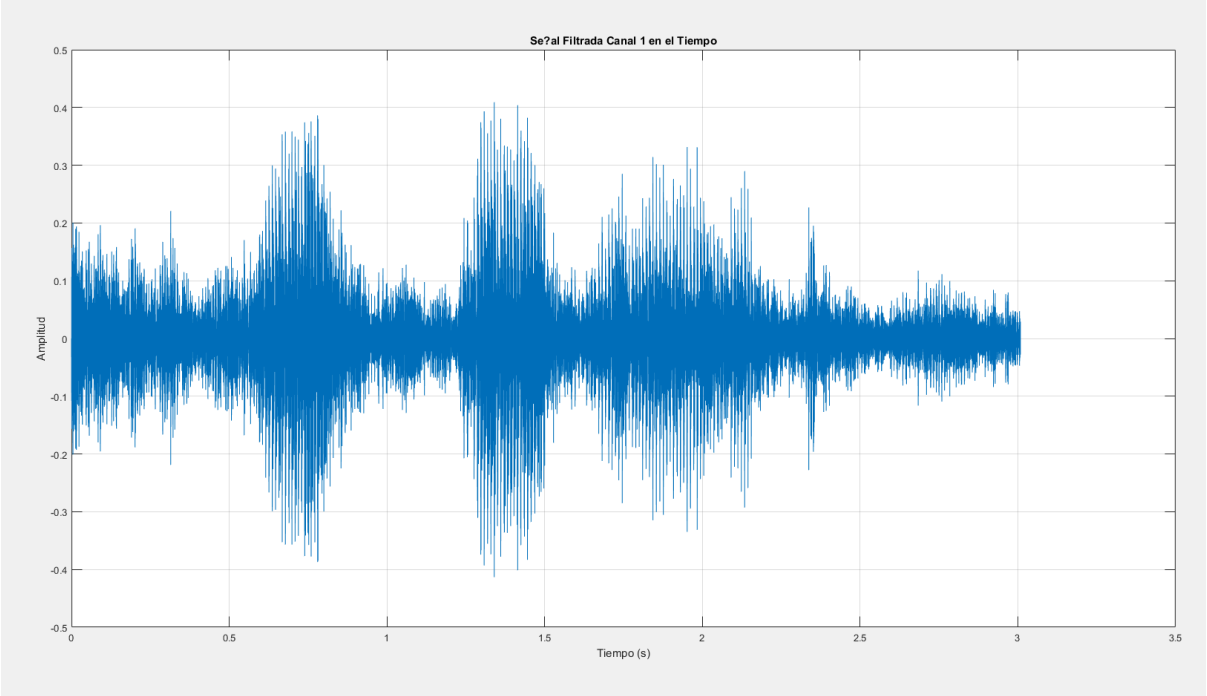


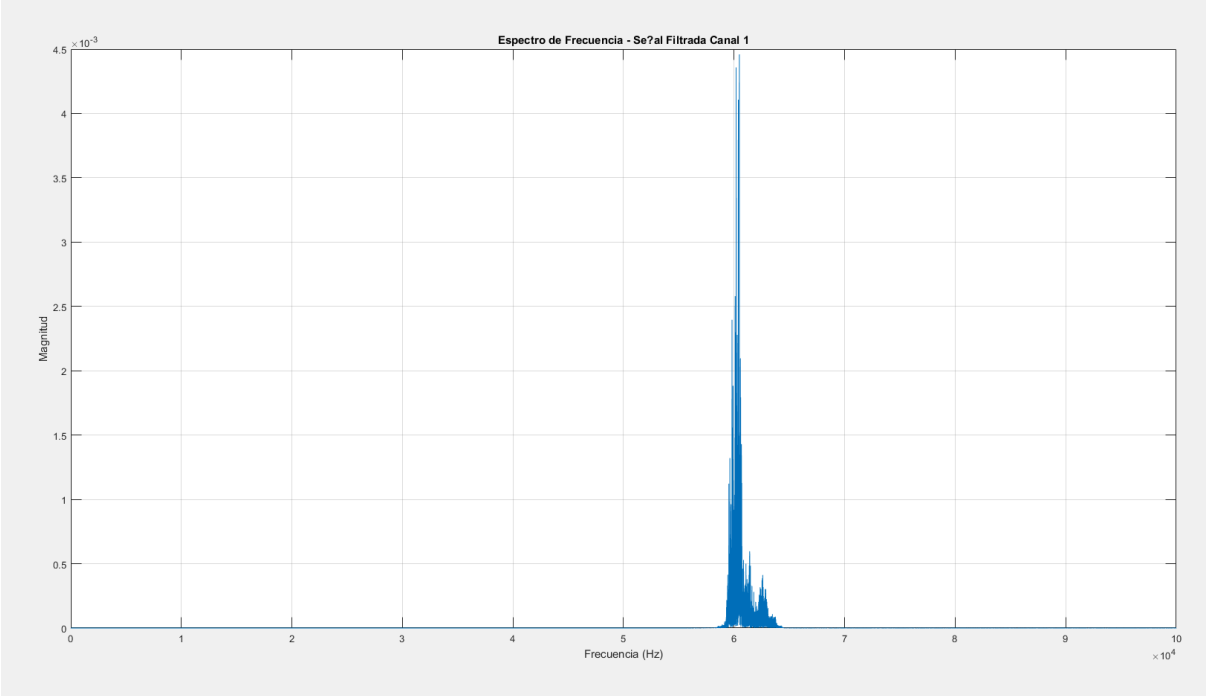
Filtro para canal 4 (Fcorte = 72Khz):



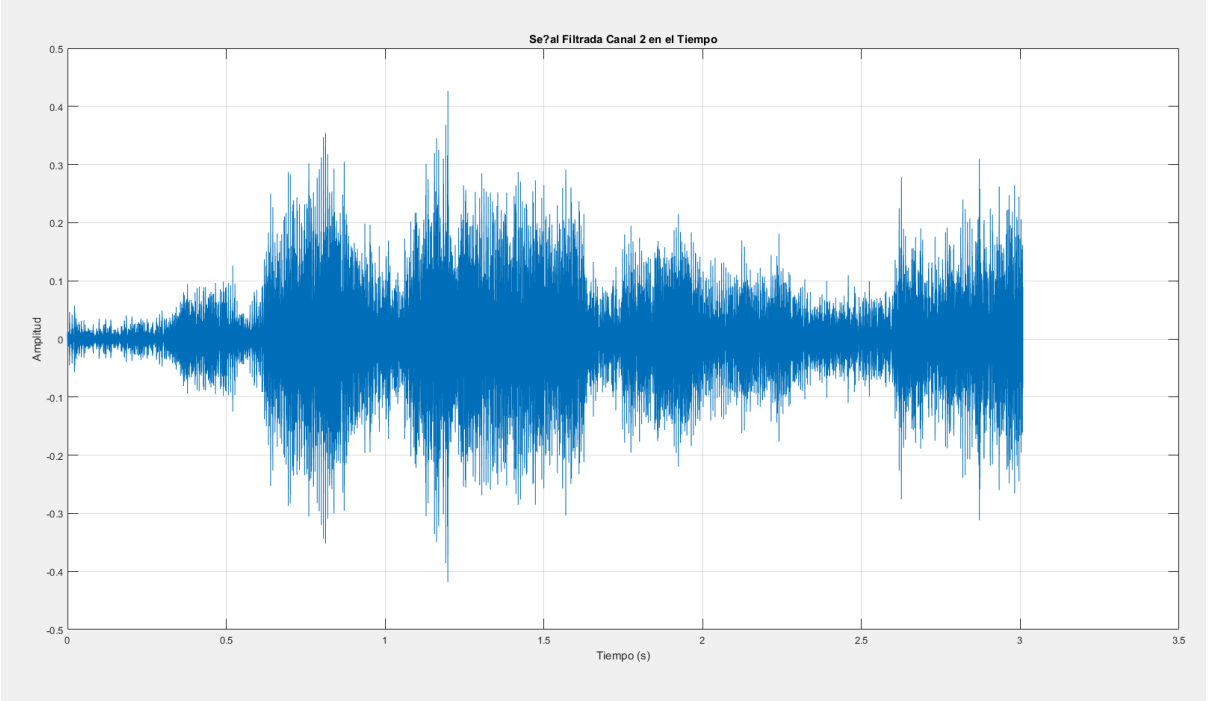
f-g)

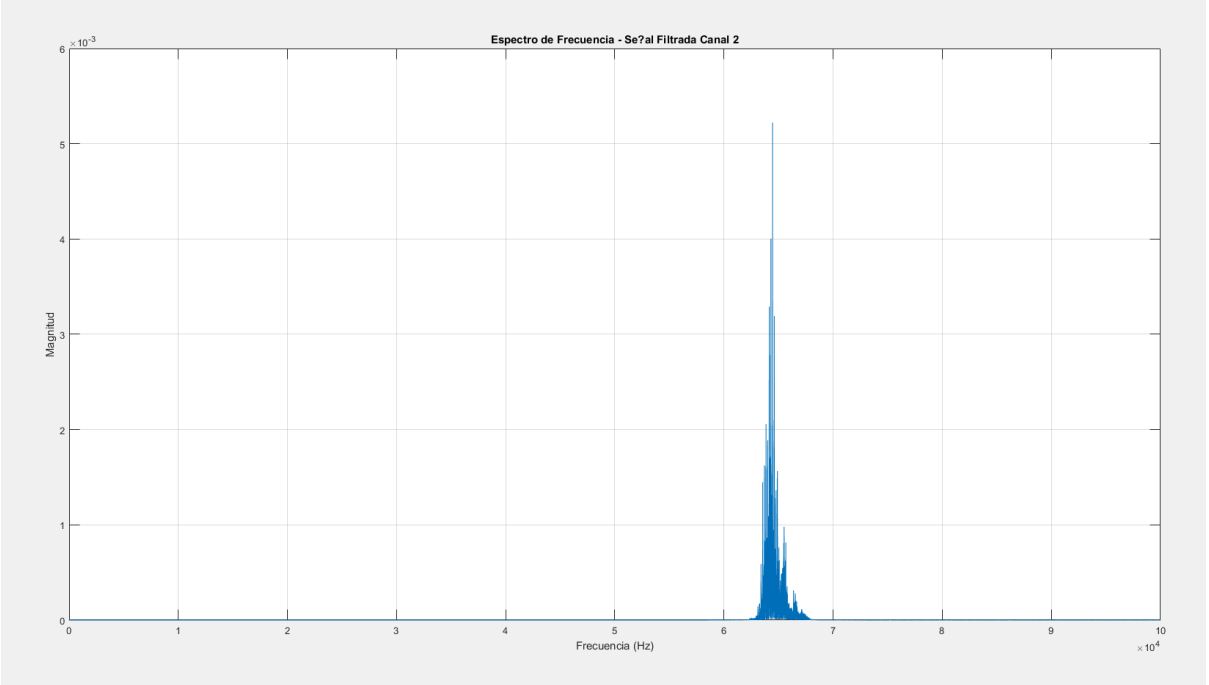
Canal 1:



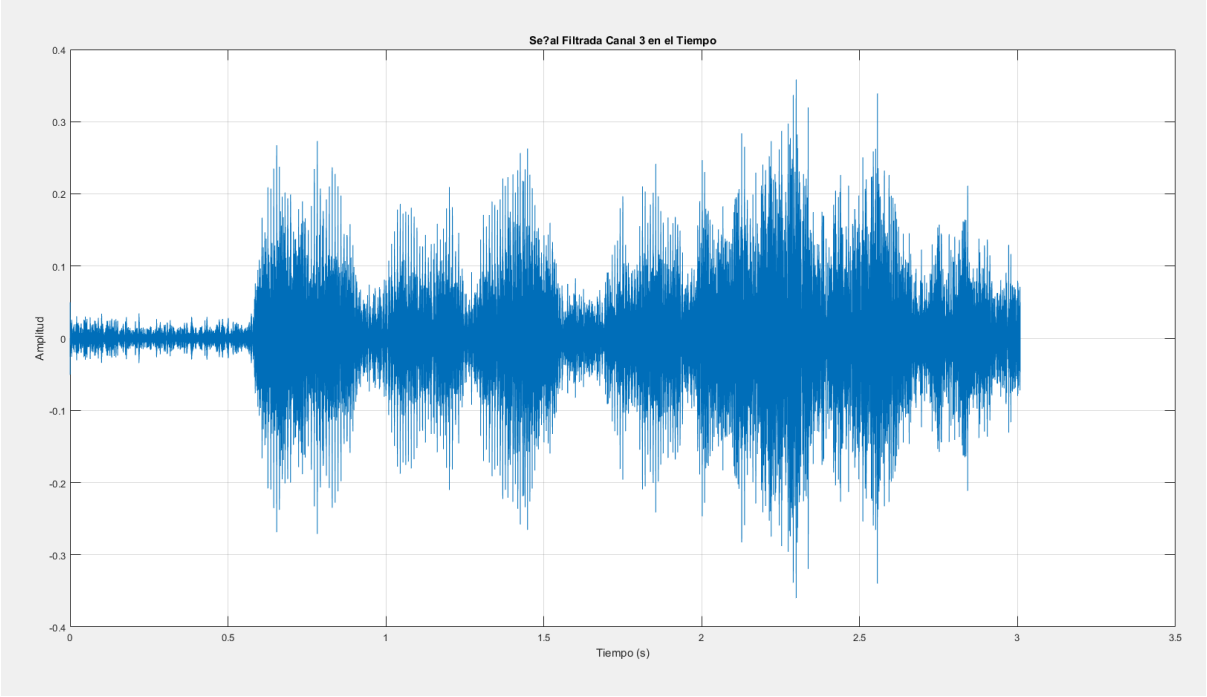


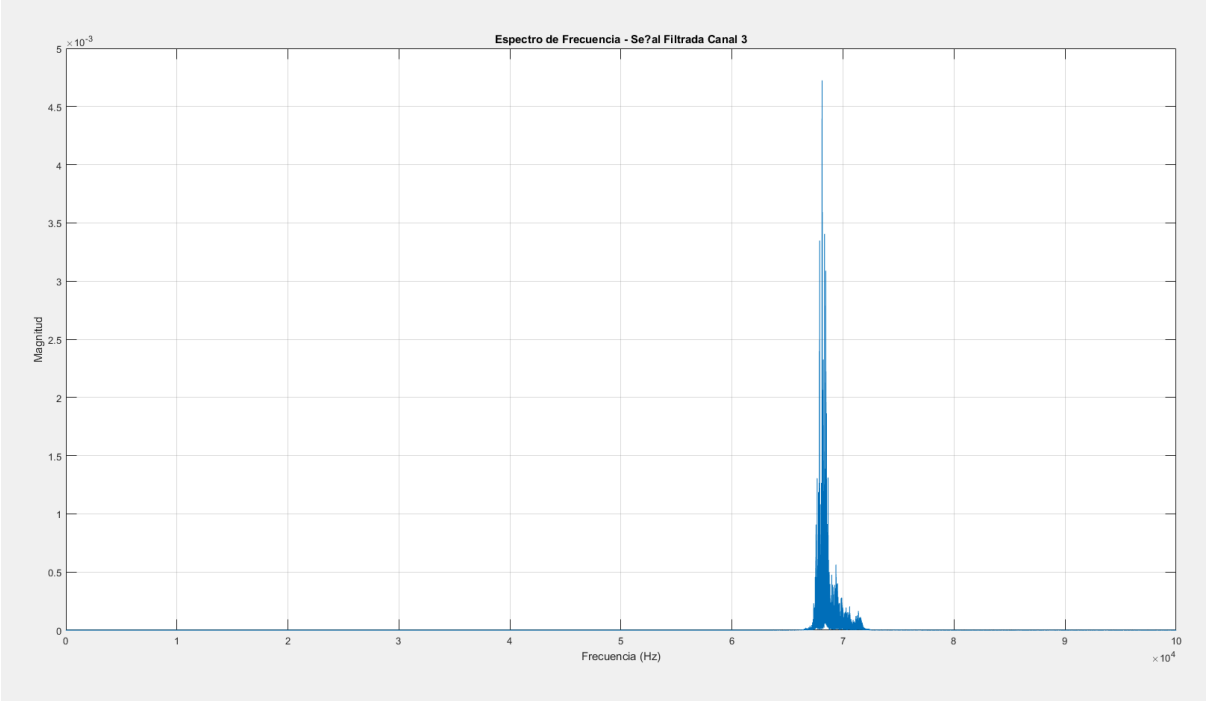
Canal 2:


Se?al Filtrada Canal 2 en el Tiempo
Amplitud
Tiempo (s)

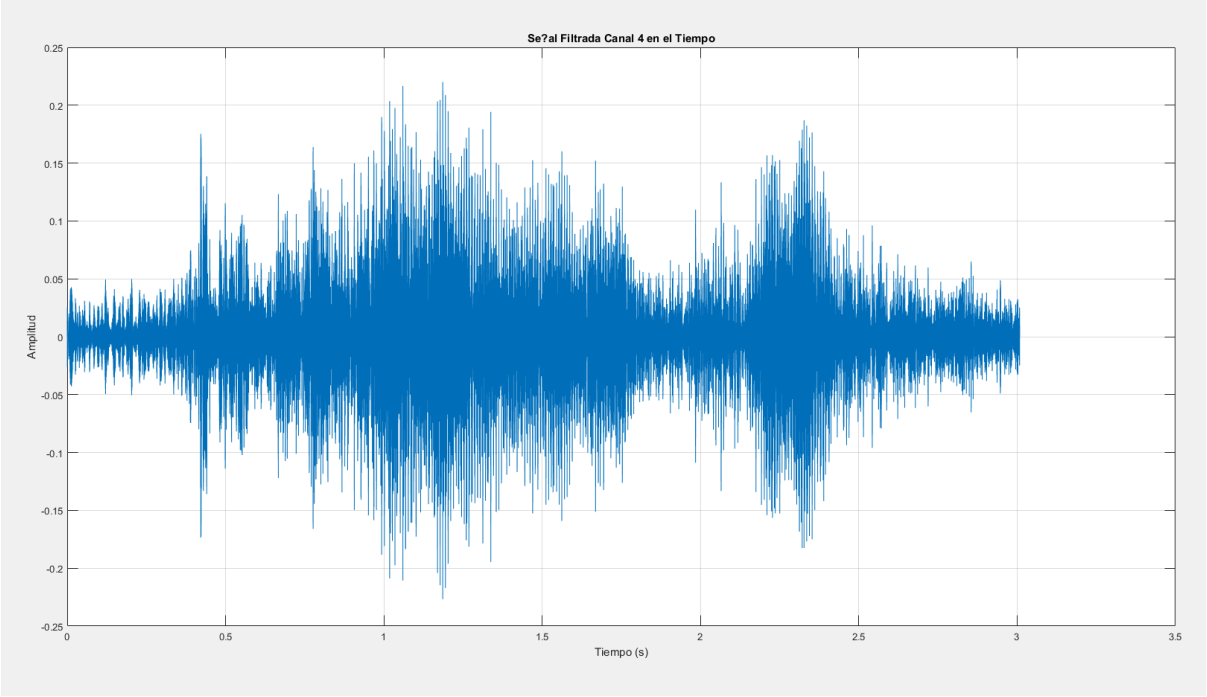

Espectro de Frecuencia - Se?al Filtrada Canal 2
Magnitud
Frecuencia (Hz)

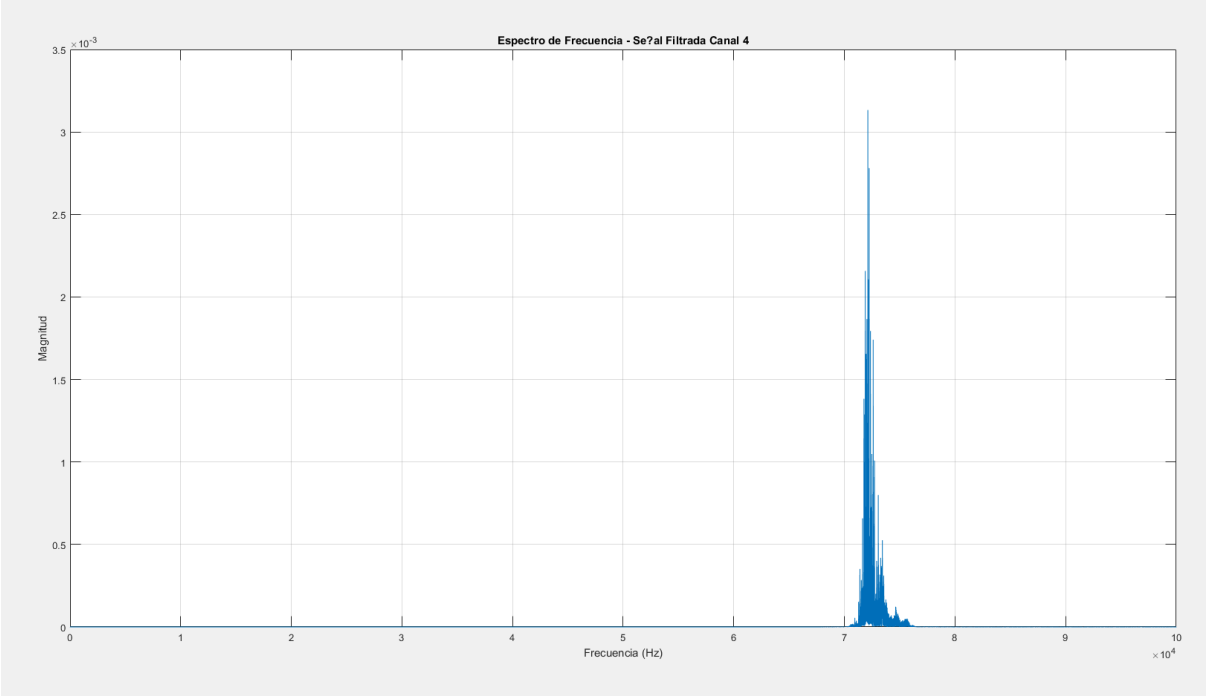
Canal 3:


Se?al Filtrada Canal 3 en el Tiempo
Amplitud
Tiempo (s)


Espectro de Frecuencia - Se?al Filtrada Canal 3
Magnitud
Frecuencia (Hz)

Canal 4:





Código utilizado:

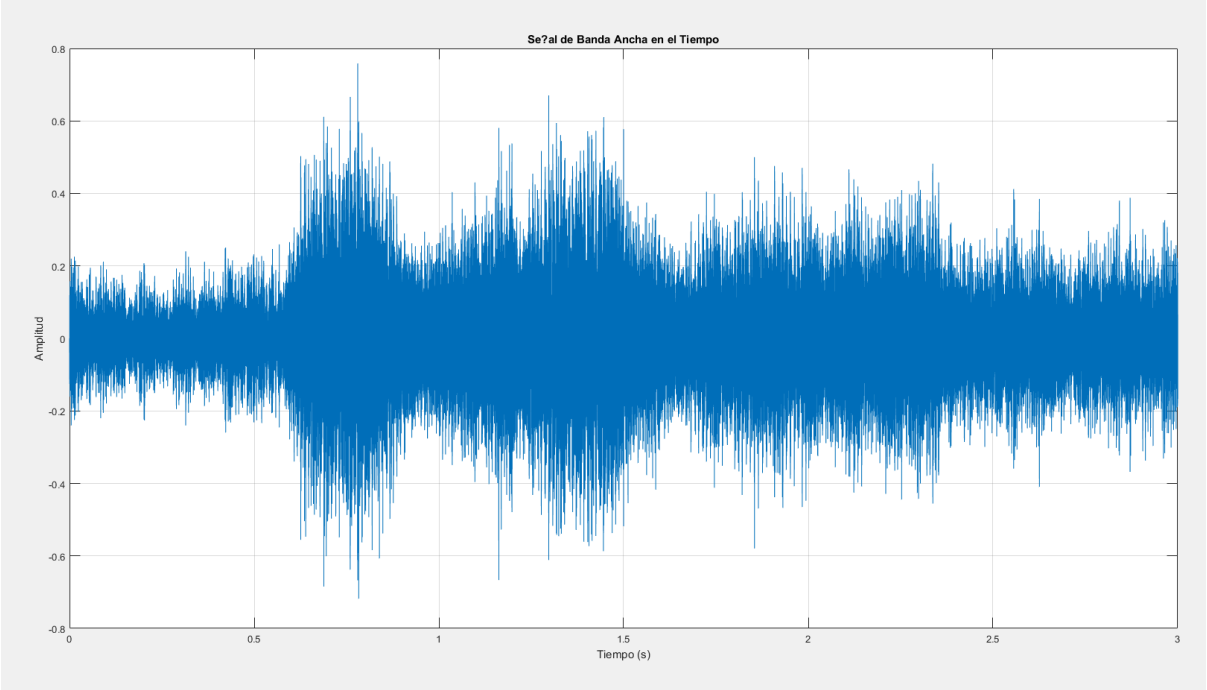
```
% Configuraci?n inicial
Fs = 328000; % Frecuencia de muestreo
archivosAudio = {'ernesto328k.wav', 'hector328k.wav', 'juanjo328k.wav', 'santi328k.wav'};
frecuenciasPortadoras = [60000, 64000, 68000, 72000]; % Frecuencias portadoras

% Modulaci?n y filtrado de las se?ales
senalesFiltradas = cell(1, length(archivosAudio)); % Inicializar

for k = 1:length(archivosAudio)
```

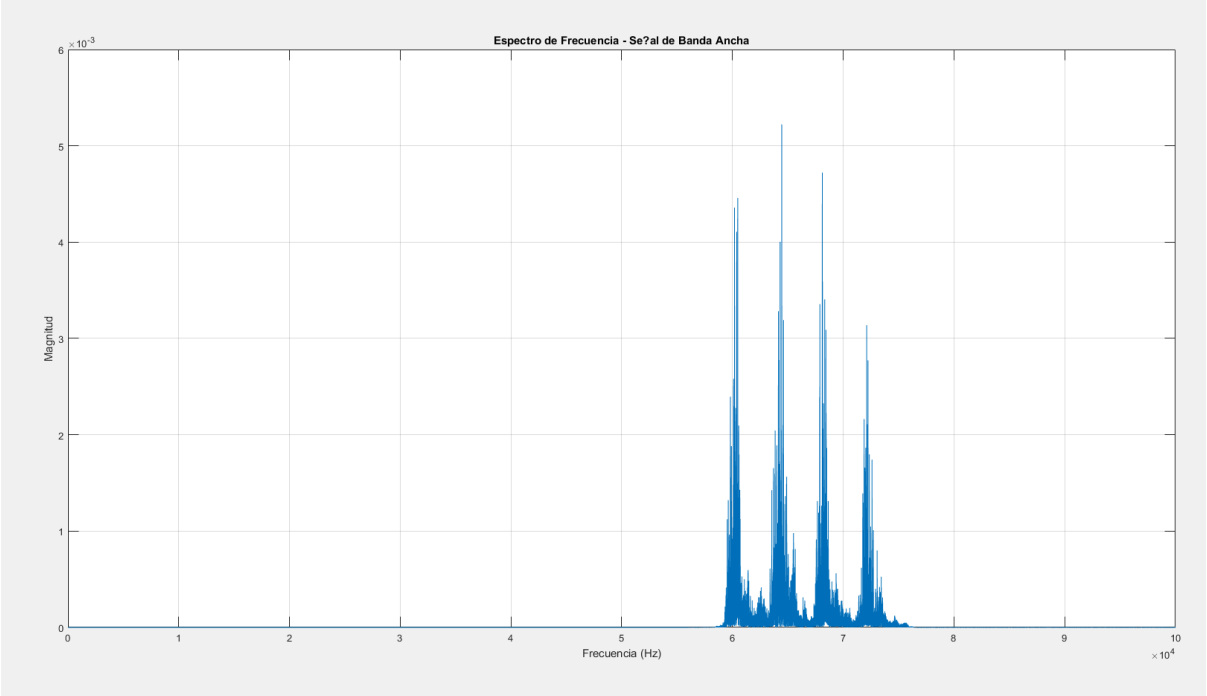
```
    % Leer cada archivo de audio
    [Xk_n, ~] = audioread(archivosAudio{k});
    n = (0:length(Xk_n)-1)'; % Vector de muestras

    % 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 demultiplexado

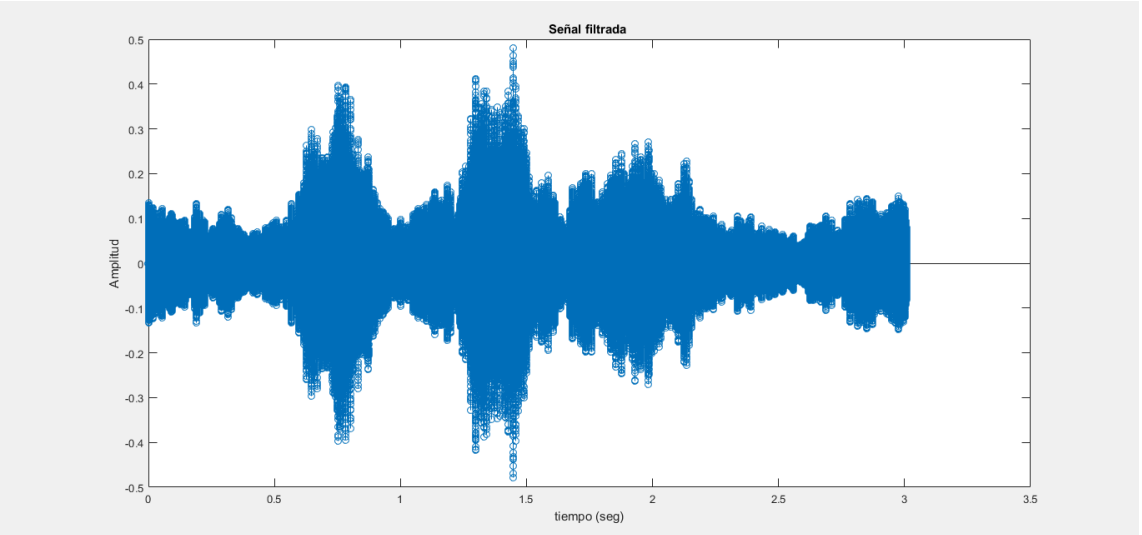
a) Diseñar el filtro digital recursivo (IIR) pasa banda para obtener la señal de banda lateral superior correspondiente a cada canal de la portadora de banda ancha.

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

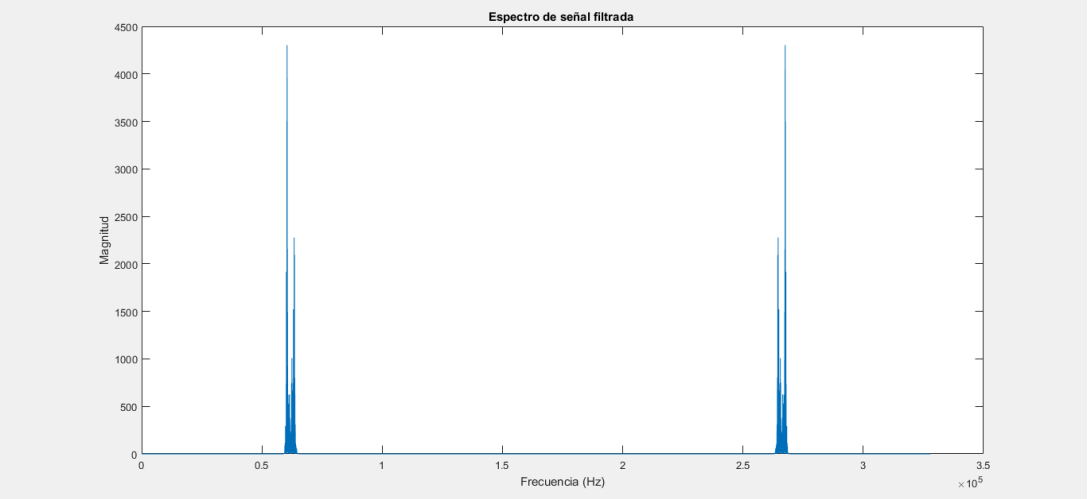
c) Representar gráficamente la señal obtenida y el espectro correspondiente para uno de los canales.

<u>Canal 1:</u>

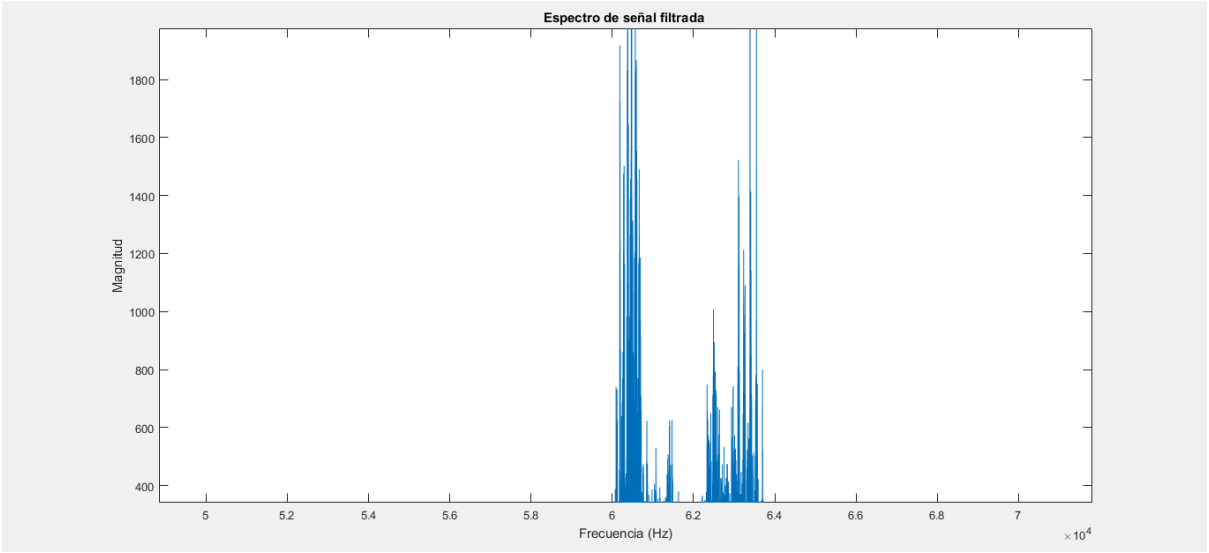
Señal de banda ancha filtrada



Espectro

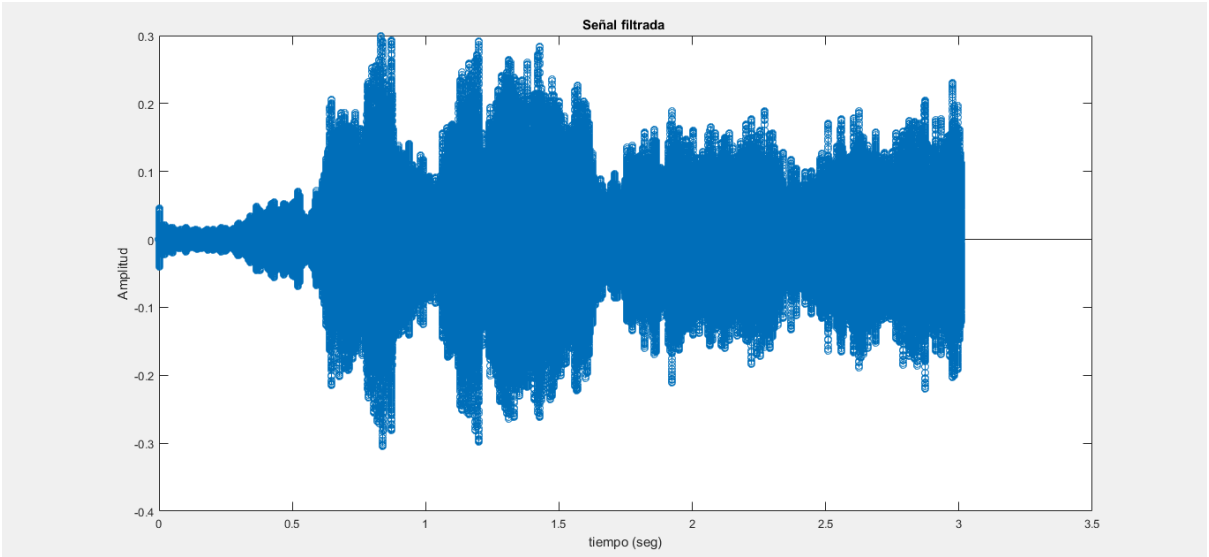


Zoom

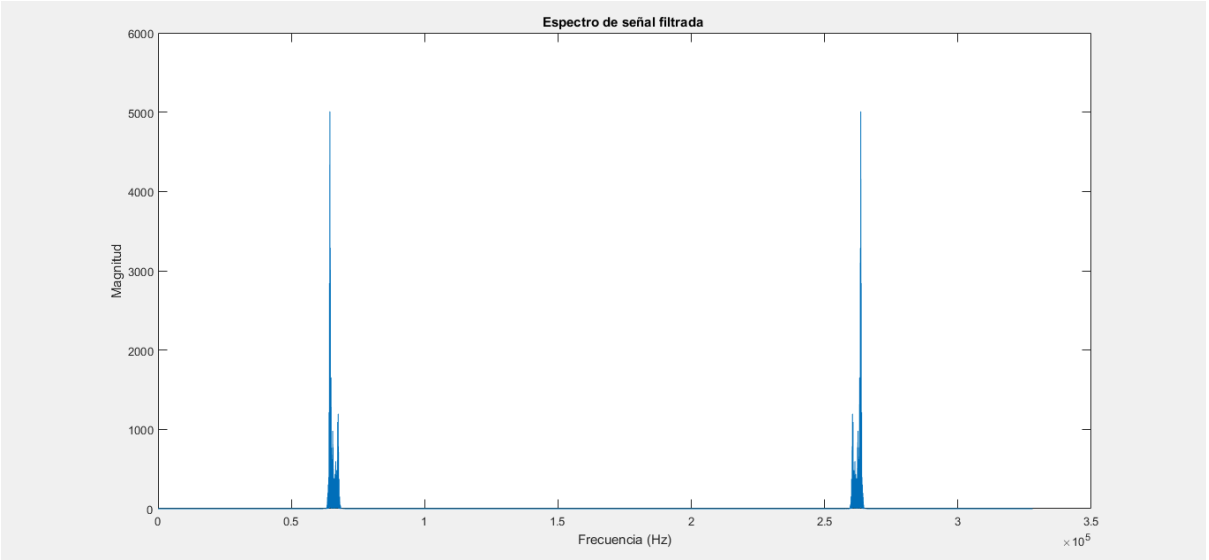


CANAL 2

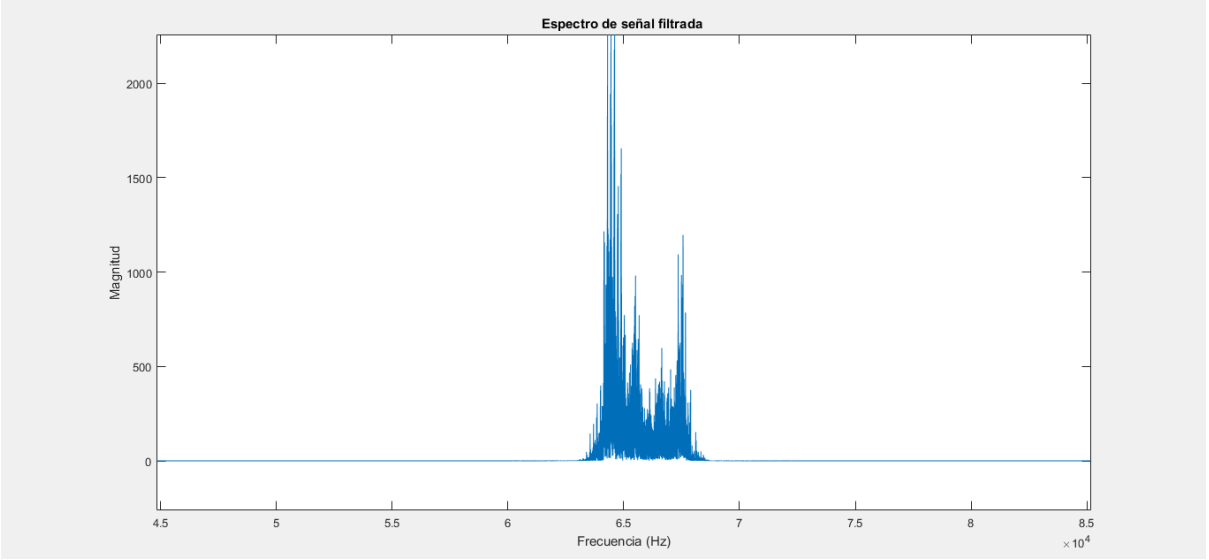
Señal de banda ancha filtrada



Espectro

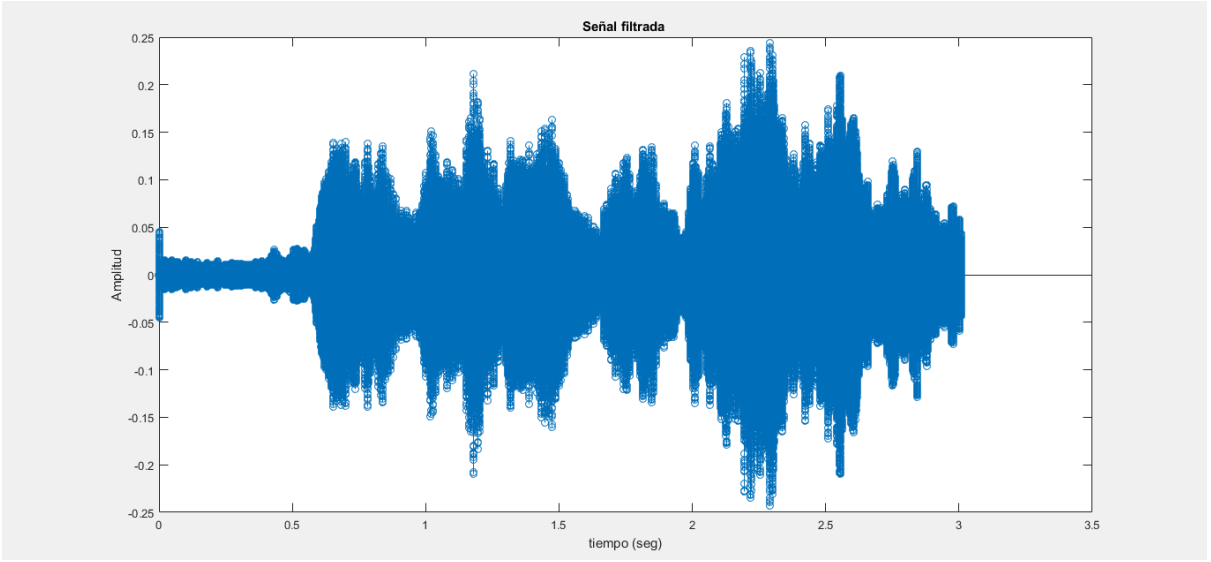


Zoom

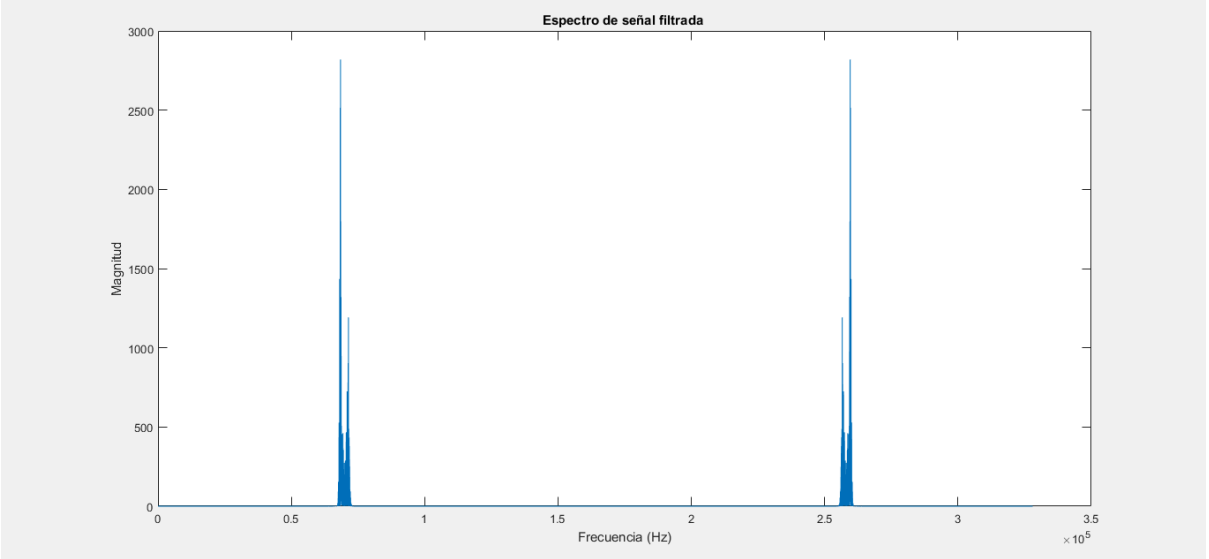


CANAL 3

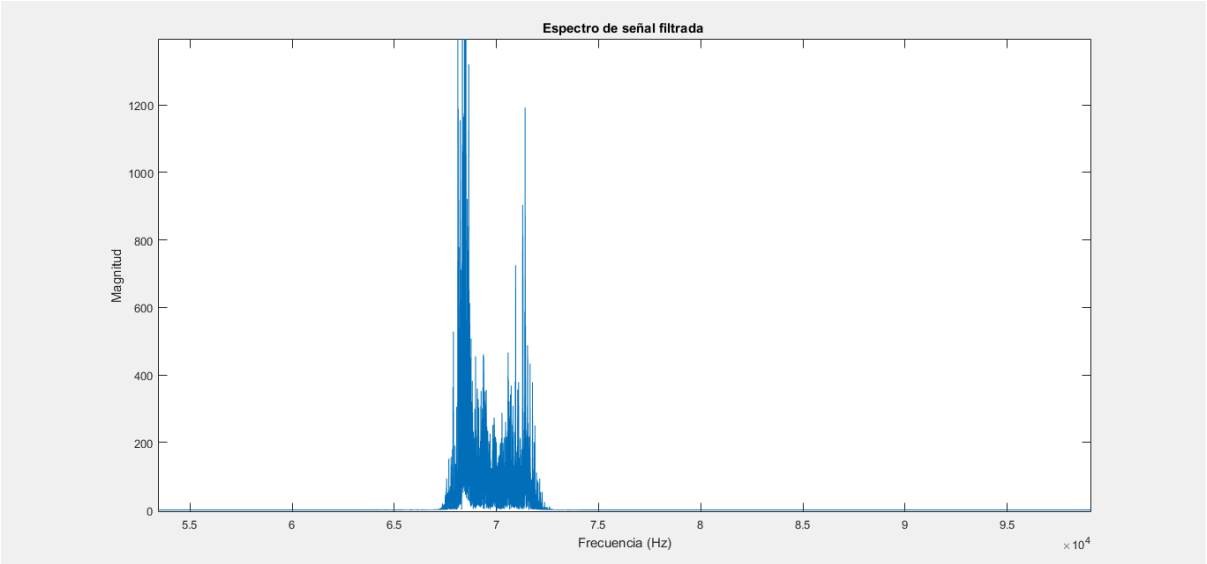
Señal de banda ancha filtrada



Espectro

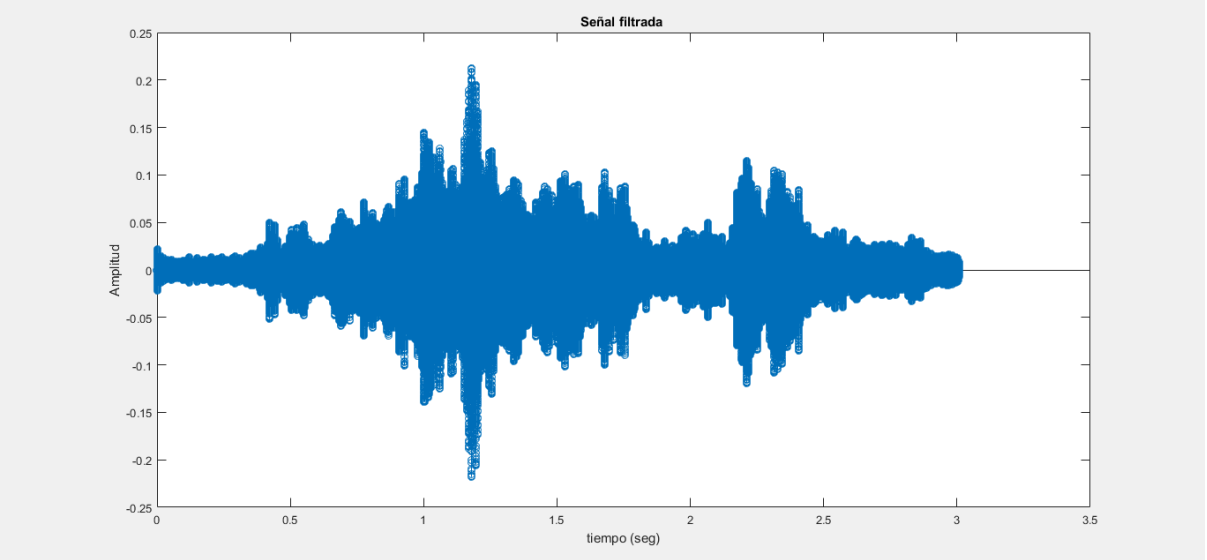


Zoom

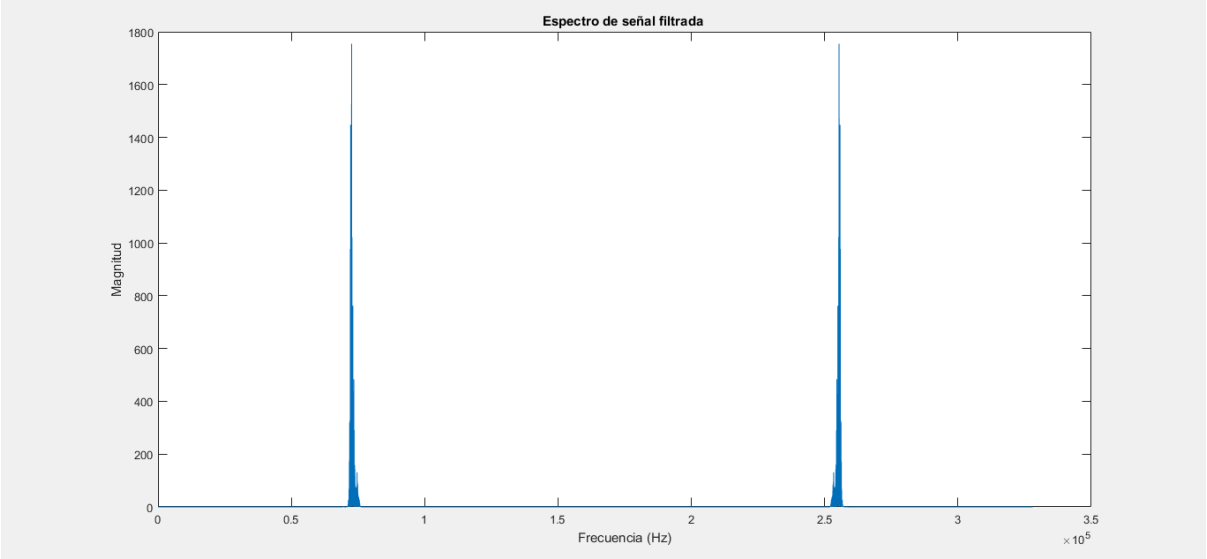


CANAL 4

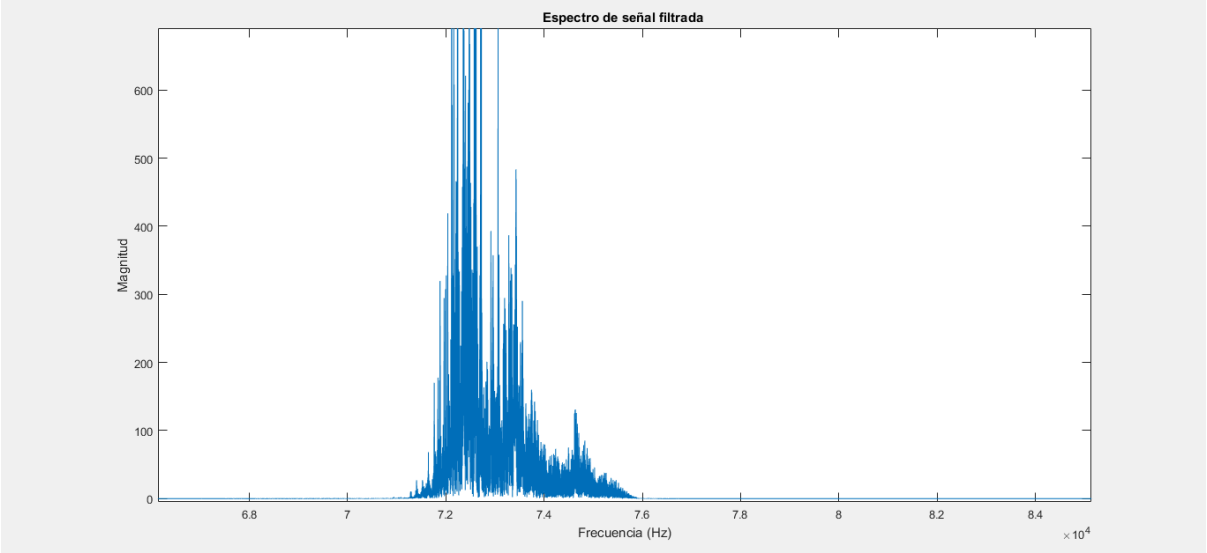
Señal de banda ancha filtrada



Espectro



Zoom

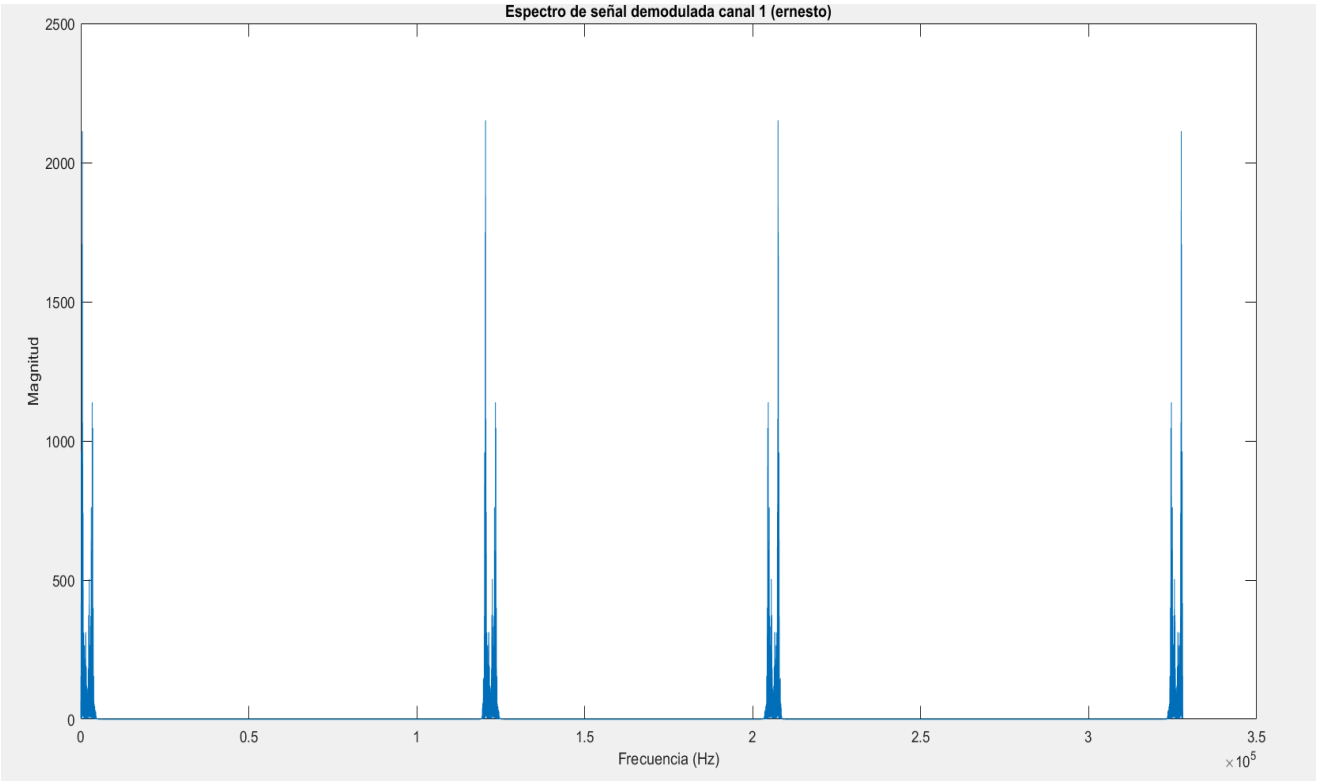


d) Realizar el desplazamiento en frecuencia de la señal de banda lateral única correspondiente a cada uno de los canales.

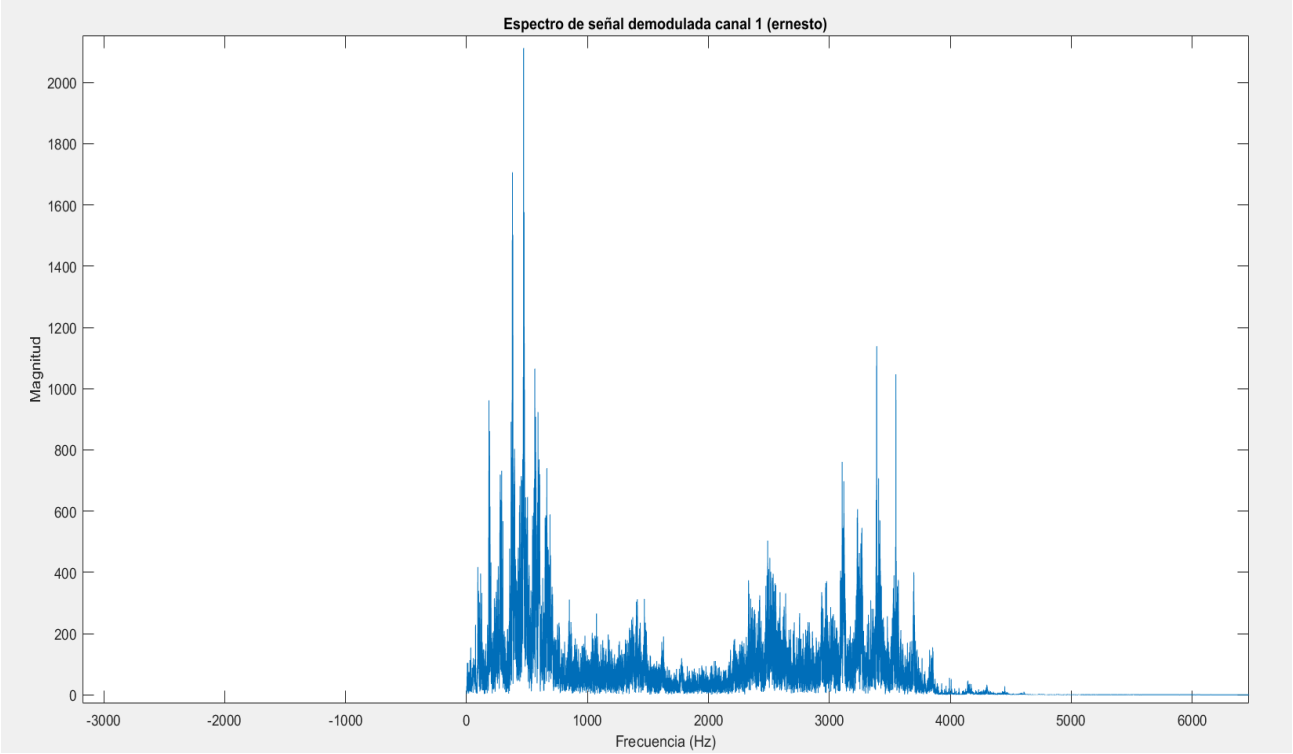
A continuación se muestra la sección del código en Matlab que realiza el desplazamiento en frecuencia de cada canal que anteriormente fue pasado por un filtro pasa banda y demultiplexado de acuerdo a la frecuencia correspondiente

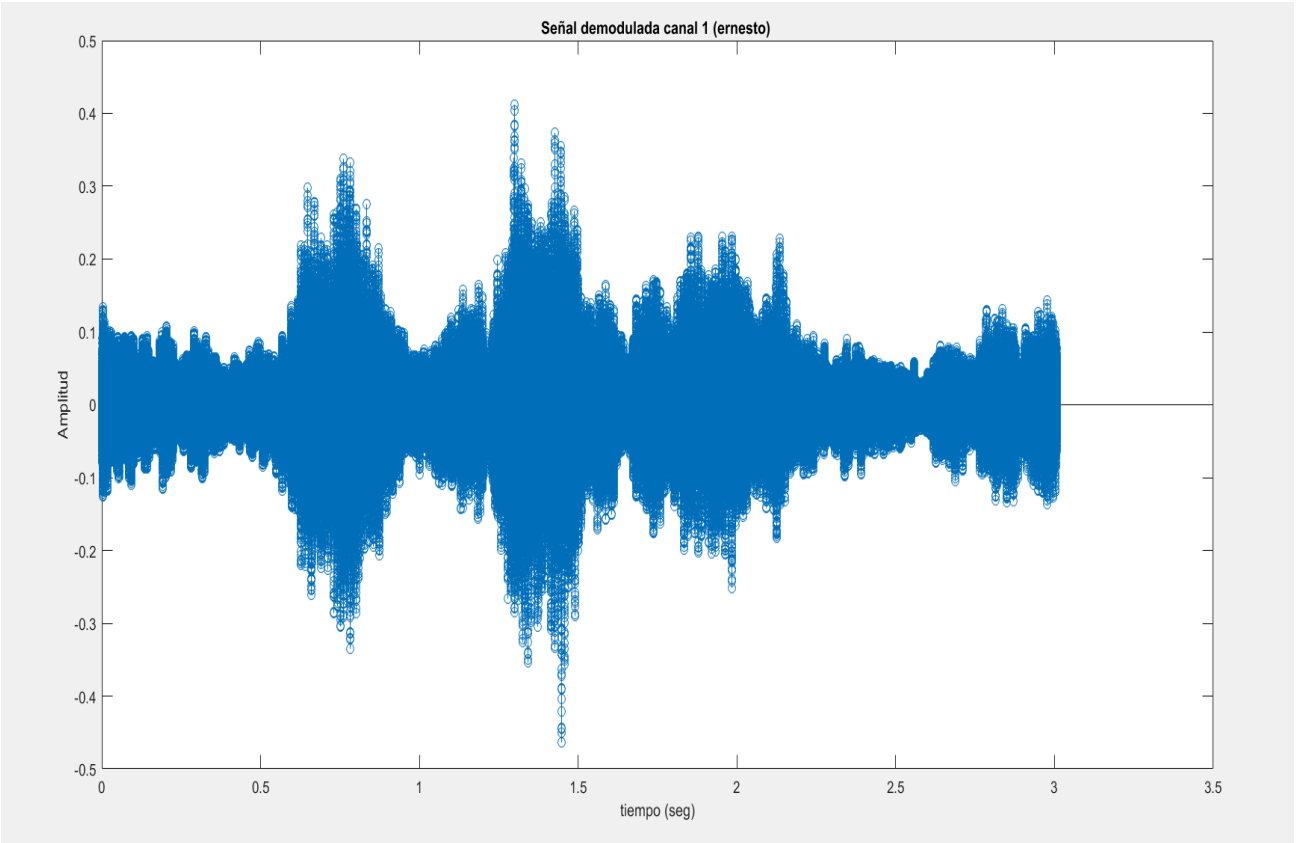
```
210 -       N = length(canal1filt); % Número de muestras
211 -       t = (0:N-1) / Fs; % Vector de tiempo, basado en la longitud de las señales
212
213     % Generar tonos portadores para cada señal
214 -   portadora1 = cos(2*pi*60000*t);
215 -   portadora2 = cos(2*pi*64000*t);
216 -   portadora3 = cos(2*pi*68000*t);
217 -   portadora4 = cos(2*pi*72000*t);
218
219     % Demodulación: multiplicar las señales por sus respectivas portadoras
220 -   demod1 = canal1filt .* portadora1';
221 -   demod2 = canal2filt .* portadora2';
222 -   demod3 = canal3filt .* portadora3';
223 -   demod4 = canal4filt .* portadora4';
```

e) Representar gráficamente la señal obtenida y el espectro correspondiente para uno de los registros.



Ampliando la imagen





f) Diseñar un filtro digital recursivo pasa bajos para obtener la señal de voz banda base.

Para El diseño del filtro utilizamos la herramienta fdatool de Matlab el cual nos diseñó un filtro pasabajos de orden 10, con la siguiente función de transferencia

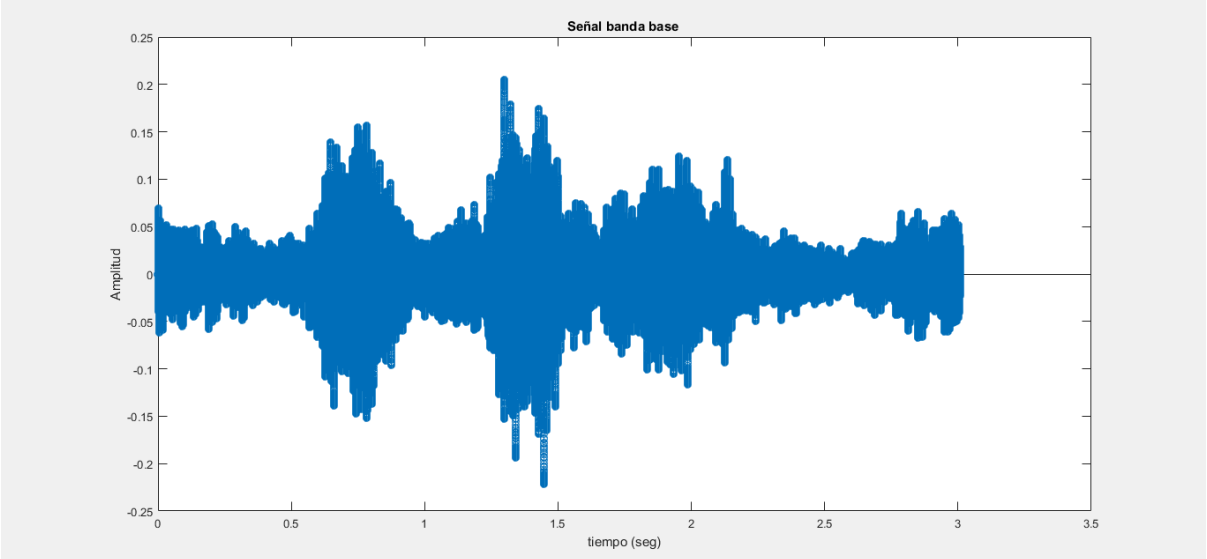
$$\frac{1.093e\text{-}15\ s^{10} + 1.093e\text{-}14\ s^{9} + 4.919e\text{-}14\ s^{8} + 1.312e\text{-}13\ s^{7} + 2.295e\text{-}13\ s^{6} + 2.755e\text{-}13\ s^{5} + 2.295e\text{-}13\ s^{4} + 1.312e\text{-}13\ s^{3} + 4.919e\text{-}14\ s^{2} + 1.093e\text{-}14\ s + 1.093e\text{-}15}{s^{10} - 9.584\ s^{9} + 41.34\ s^{8} - 105.7\ s^{7} + 177.4\ s^{6} - 204.1\ s^{5} + 163.2\ s^{4} - 89.48\ s^{3} + 32.2\ s^{2} - 6.868\ s + 0.6594}$$

g) Utilizando la función “filter” de Matlab, procesar la señal obtenida en el punto d.

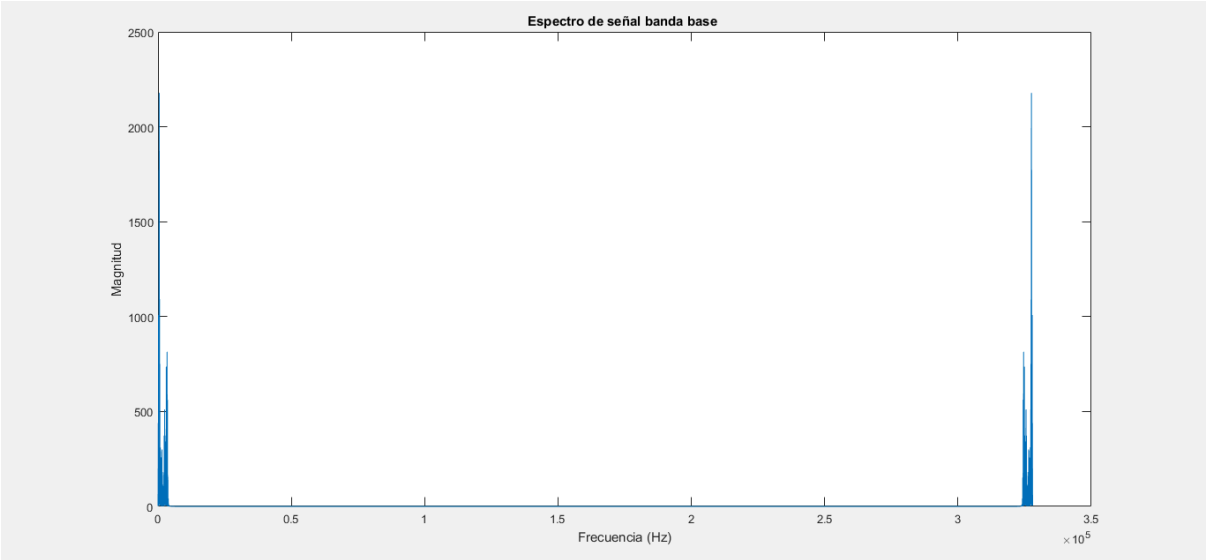
h) Representar gráficamente la señal obtenida y el espectro correspondiente para uno de los registros.

```
% Filtro
BBcanal1= filter(I,J,demod1);
BBcanal2= filter(I,J,demod2);
BBcanal3= filter(I,J,demod3);
BBcanal4= filter(I,J,demod4);
% Graficas
figure;
t=1/Fs;
n=length(BBcanal1);
z=(0:n-1)*t;
stem(z,BBcanal1);
title('Señal banda base ');
xlabel('tiempo (seg)');
ylabel('Amplitud');

%espectro
Resp= fft(BBcanal1);
n_1=length(BBcanal1);
M=abs(Resp);
f_1=(0:n_1-1)*(Fs/n_1);
figure;
plot(f_1, M);
title('Espectro de señal banda base ');
xlabel('Frecuencia (Hz)');
ylabel('Magnitud');
```

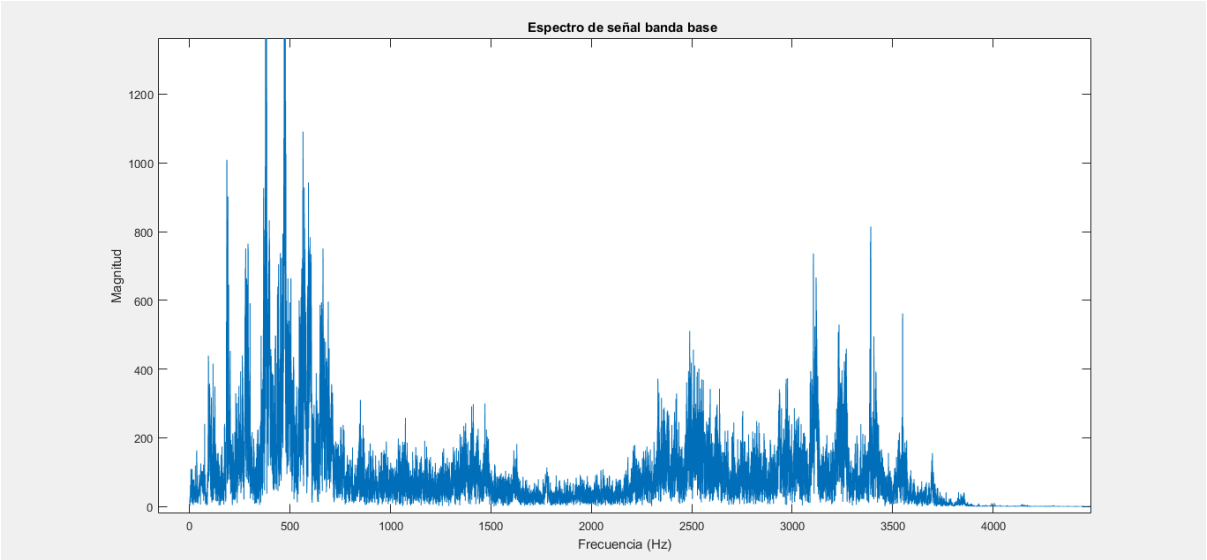
Señal de Banda Base obtenida:



Espectro:



Zoom:



i) Bajar la frecuencia de muestreo a fs= 8kHz de las señales de voz en banda base.

```
audiowrite('BBcanal1.wav', BBcanal1, Fs);
audiowrite('BBcanal2.wav', BBcanal2, Fs);
audiowrite('BBcanal3.wav', BBcanal3, Fs);
audiowrite('BBcanal4.wav', BBcanal4, Fs);

[audio, Fs]= audioread('BBcanal1.wav');
Fs_new= 8000;
audio_fsnew = resample(audio, Fs_new, Fs);
audiowrite('ernesto8Kdem.wav',audio_fsnew, Fs_new);

  [audio, Fs]= audioread('BBcanal2.wav');
Fs_new= 8000;
audio_fsnew = resample(audio, Fs_new, Fs);
audiowrite('hector8Kdem.wav',audio_fsnew, Fs_new);

  [audio, Fs]= audioread('BBcanal3.wav');
Fs_new= 8000;
audio_fsnew = resample(audio, Fs_new, Fs);
audiowrite('juanjo8Kdem.wav',audio_fsnew, Fs_new);

  [audio, Fs]= audioread('BBcanal4.wav');
Fs_new= 8000;
audio_fsnew = resample(audio, Fs_new, Fs);
audiowrite('santi8Kdem.wav',audio_fsnew, Fs_new);
```

```
% Lista de archivos de audio
audioFiles = {'ernesto8Kdem.wav', 'hector8Kdem.wav', 'juanjo8Kdem.wav', 'santi8Kdem.wav'};

% Número 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 señal de audio
    plot(t, audioData);
    xlabel('Tiempo (s)');
    ylabel('Amplitud');

    % Añadir el título con el nombre del archivo de audio
    title(['Señal de Audio - ', audioFiles{i}]);
    grid on;
end
```

Señal de Banda Base resampleada a fs = 8 kHz: