DEPARTAMENTO DE ELECTRÓNICA Y AUTOMÁTICA

FACULTAD DE INGENIERÍA – UNIVERSIDAD NACIONAL DE SAN JUAN

Informe de Practica de DSP Nº4

Implementación de Filtros Digitales FIR e IIR

Asignatura: Procesamiento Digital de Señales

Ingeniería Electrónica

Autor:

Avila, Juan Agustin – Registro 26076

1º Semestre

Año 2020

1 Enunciado

1. Usando la herramienta FdaTool de Matlab implementar 3 filtros FIR de 4º orden de distintos tipos y 3 filtros IIR de 4º orden de distintos tipos. Experimentar primero fuera de línea con archivos de señales y luego en línea usando el oscilador que diseñaron y el Arduino Uno.

2. Probar los filtros del punto 1 con señales de sonido WAV.

3. Realizar un ecualizador de 5 bandas para una señal de sonido WAV, según se muestra en la figura.

4. Investigar alguna otras aplicaciones de los filtros FIR e IIR.

5. Realizar el informe detallando cada paso realizado e incorporando en el mismo todas las gráficas e información relevantes. Armar una pequeña presentación en PPT explicando el desarrollo de la práctica.

2 Resolución.

2.1 Punto 1

Se comenzó generando 3 filtros, un pasa bajos de 40Hz, un pasabanda entre 40 y 100Hz y un pasa altos de 150Hz, y estos 3 filtros se implementaron como filtros FIR e IIR, con un orden de 4.

Los filtros generados se exportaron al espacio de trabajo de matlab, y esas variables con las ecuaciones en diferencias de los filtros se exportaron a un archivo de texto con el nombre correspondiente del filtro. Estos archivos de texto se pasaron como variables junto a la señal a utilizar al programa escrito en C que aplica el filtro propiamente dicho. La forma de utilizar el programa en C es la siguiente:

El programa se debe correr de la siguiente manera:

> DSP3.exe filtro_a_aplicar archivo_a_analizar
> Por ejemplo para el archivo "Tono_50Hz.txt" con el filtro "FiltroFIR_LP50Hz.txt":
> "DSP3.exe FiltroFIR_LP50Hz Tono_50Hz"
> (No se debe incluir la terminacion .txt de los archivos)
>
> IMPORTANTE: el archivo de filtros debe tener la siguiente forma:
> b(n-5)
> b(n-4)
> ...
> b(n)
>
> Y si es IIR luego:
> a(n-5)
> ...
> a(n)

Y el algoritmo en C que aplica los filtros es el siguiente:

```
void filtrado(FILE *archivo_original, float num[], float den[], int N,
char nombre[], char extra[])
{
    float entrada[N], freq, out = 0, salida[N];
    int k = 0;
```

```
    FILE *archivo_nuevo;
    archivo_nuevo = abrir_archivo("w+", nombre, extra); //abre un nuevo
 archivo
    fscanf(archivo_original, "%f\n", &freq);            //obtiene la fr
ecuencia
    fprintf(archivo_nuevo, "%.1f\n", freq);             //y la guarda e
n el nuevo
    for (k = 0; k < N; k++)                             //inicializa el
 arreglo
    {
        entrada[k] = 0; //coloca todos los valores en cero
        salida[k] = 0;
    }

    while (!feof(archivo_original)) //lee una nueva linea
    {
        out = 0;
        for (k = 1; k < N; k++)
        {
            entrada[k - 1] = entrada[k]; //desplaza los valores
            salida[k - 1] = salida[k];
        }
        salida[N - 1] = 0;  //en principio se pone en cero para que no
se calcule
        fscanf(archivo_original, "%f", &entrada[N - 1]);
        for (k = 0; k < N; k++)
        {
            out = out + (num[k] * entrada[k]) - (den[k] * salida[k]); /
/realiza el calculo
        }
        salida[N - 1] = out;    //finalmente se asigna el valor de sali
da actual al arreglo
        fprintf(archivo_nuevo, "%.3f\n", out);
    }

    fclose(archivo_nuevo); //cierra el archivo nuevo
    rewind(archivo_original);
}
```

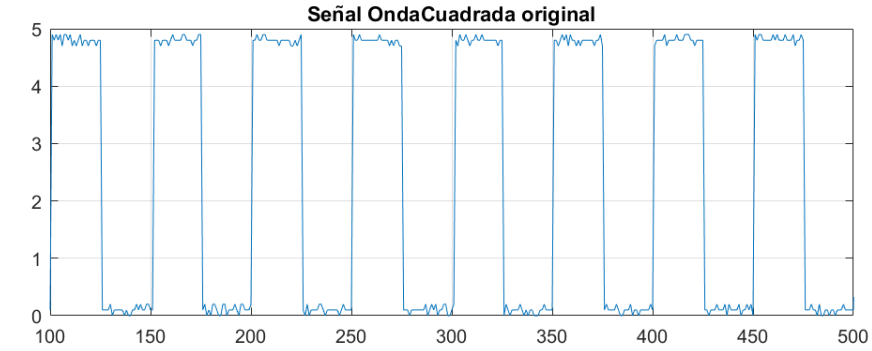
Con esto, se escribió un script en matlab (archivo adjunto) que realiza las siguientes acciones para cada filtro:

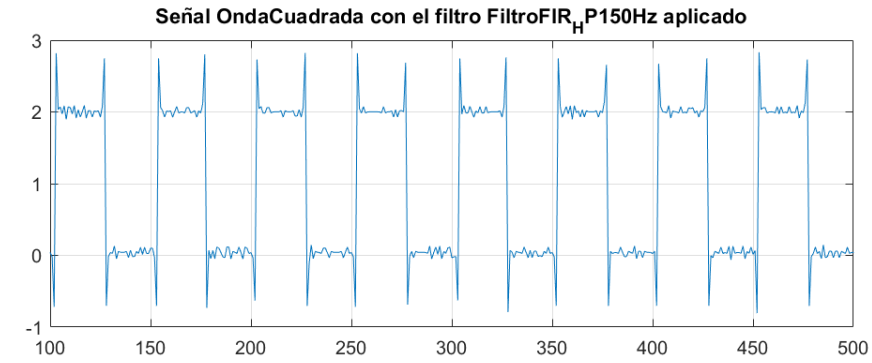
-Si es un filtro IIR, lo pasa de una matriz SOS a una función de transferencia
-Guarda los coeficientes en un archivo de texto en orden invertido
-Ejecuta el programa con el filtro dado y el archivo de la señal
-Grafica parte de la señal original y abajo la señal con el filtro aplicado
-Calcula la transformada con 512 puntos de ambas señales (original y filtrada) utilizando el programa creado en el practico de DSP3
-Grafica ambos espectros frecuenciales, el de la señal original y el de la señal filtrada.

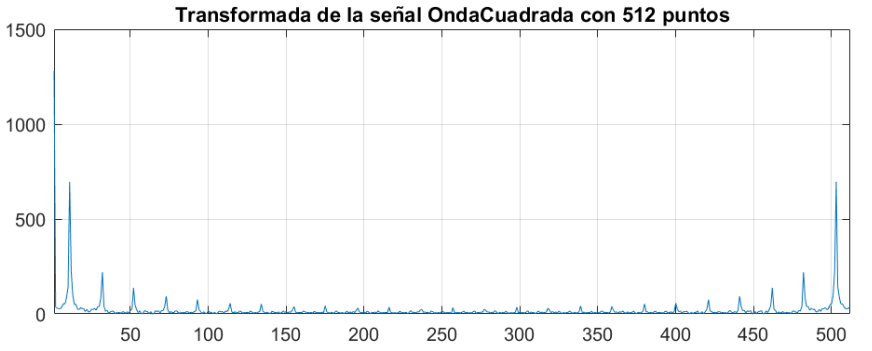
En un principio se trabajo con una señal realizada sumando los tonos de 20, 50 y 200Hz, pero luego se utilizó la señal “OndaCuadrada” provista por la cátedra. Los resultados fueron los siguientes:

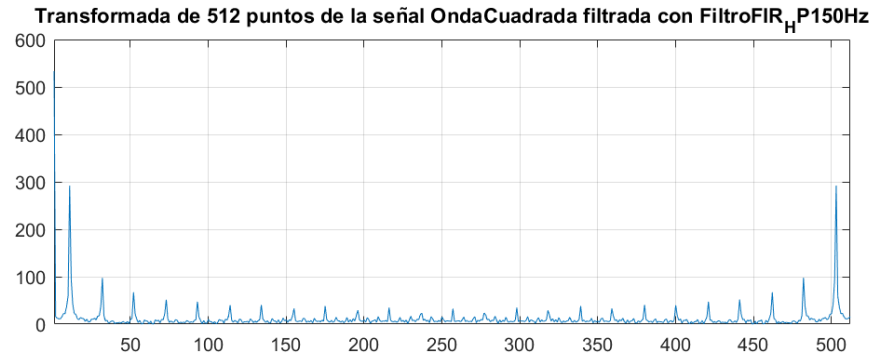
2.1.1 Filtros FIR:

2.1.1.1 Filtro pasa altos

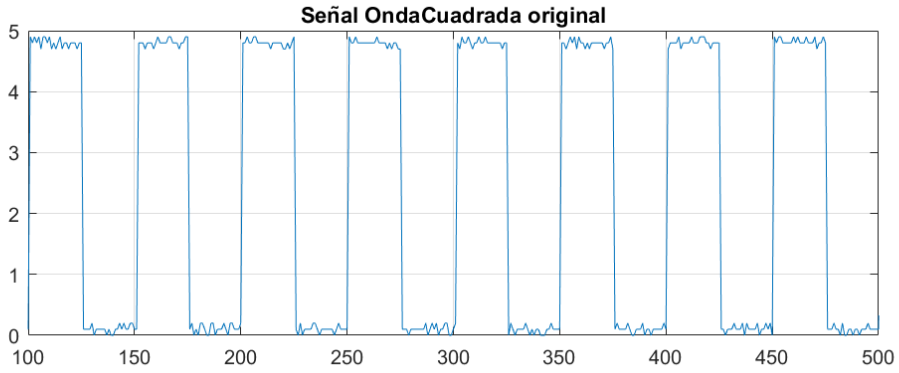


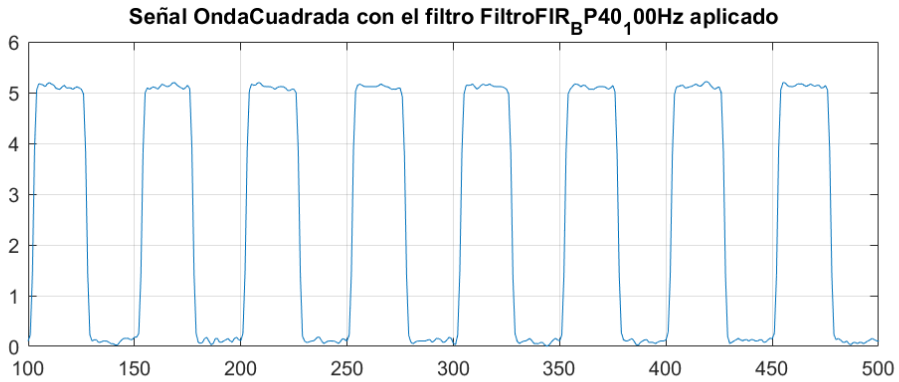


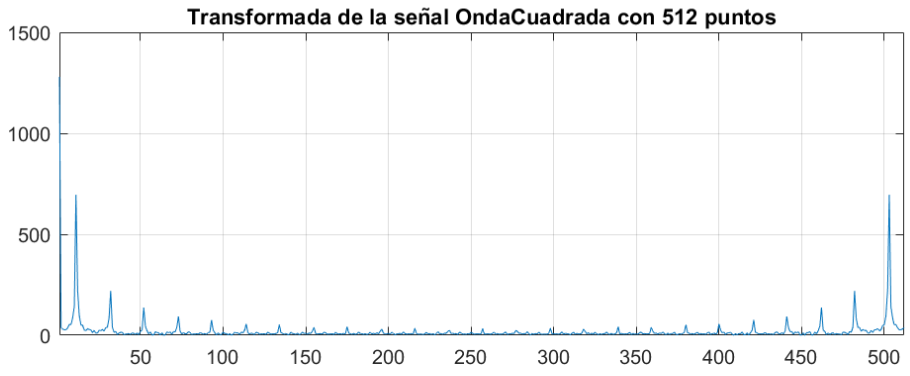


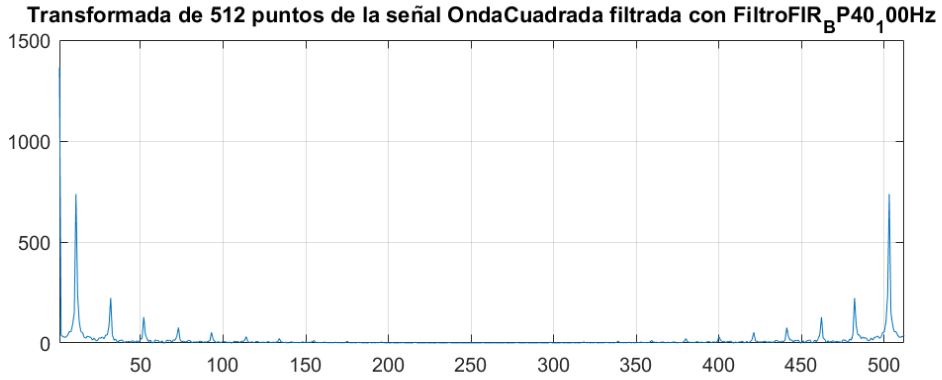


2.1.1.2 Filtro pasabanda

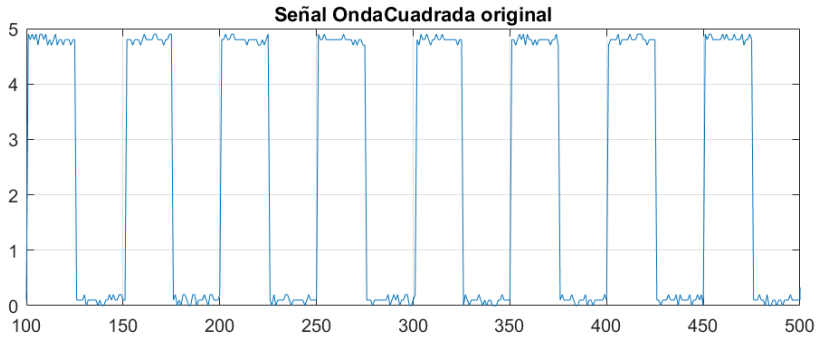


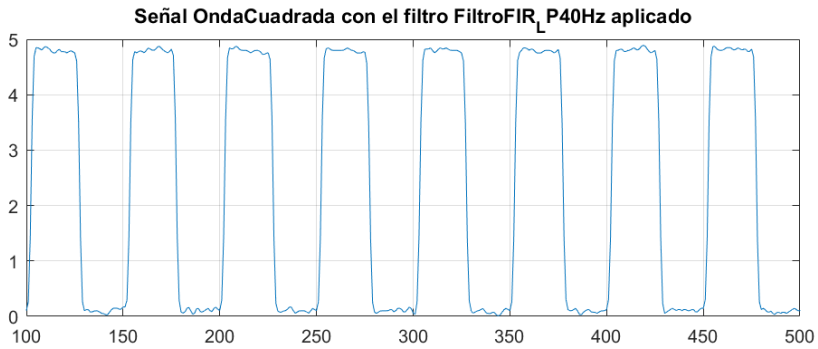


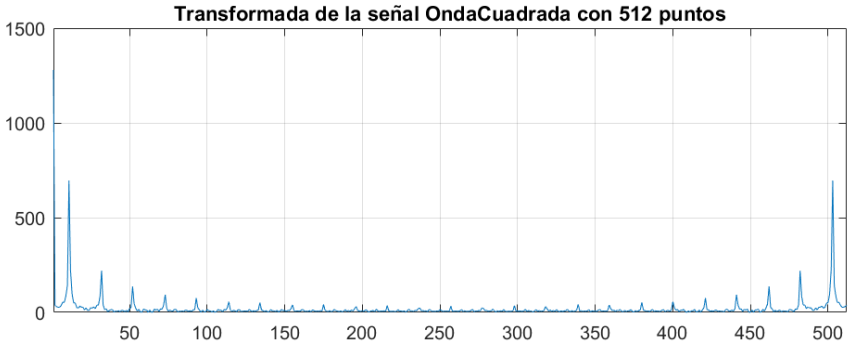


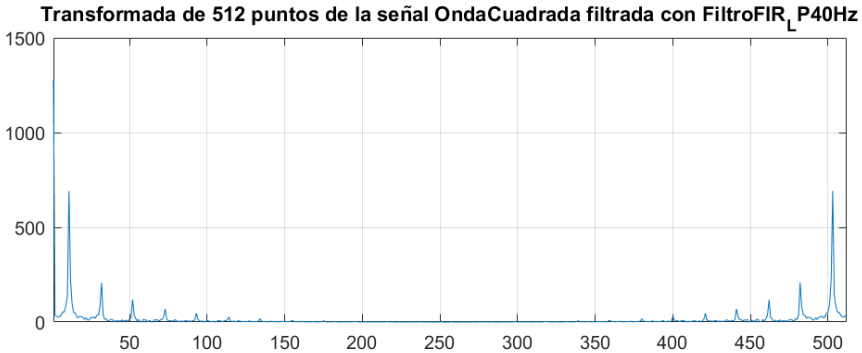


2.1.1.3 Filtro pasabajos



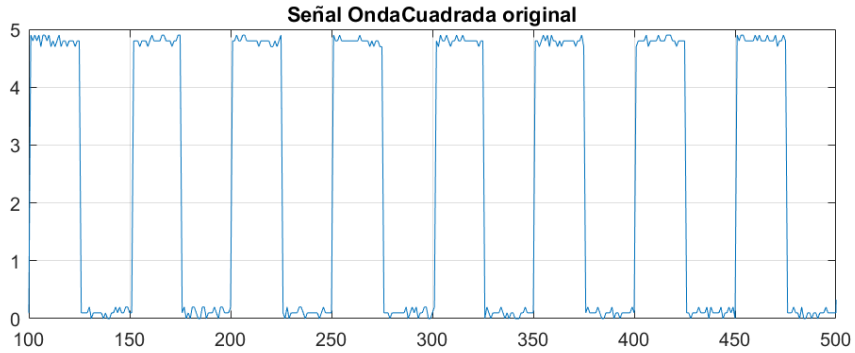


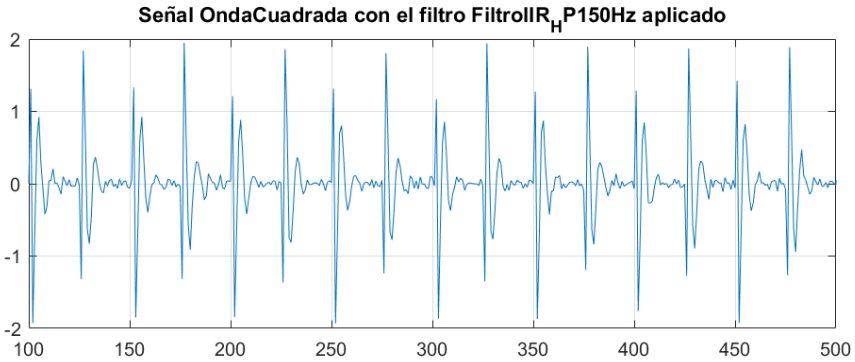


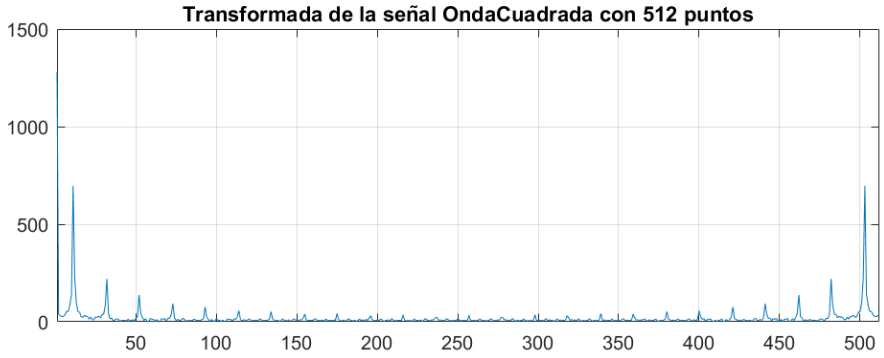


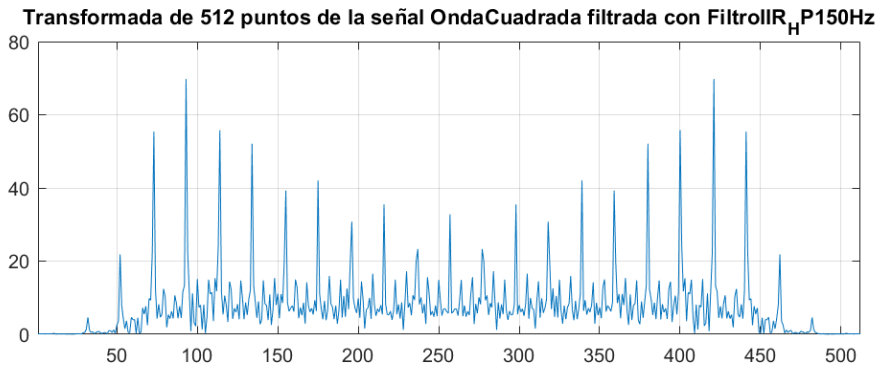
2.1.2 Filtros IIR:

2.1.2.1 Filtro pasaaltos

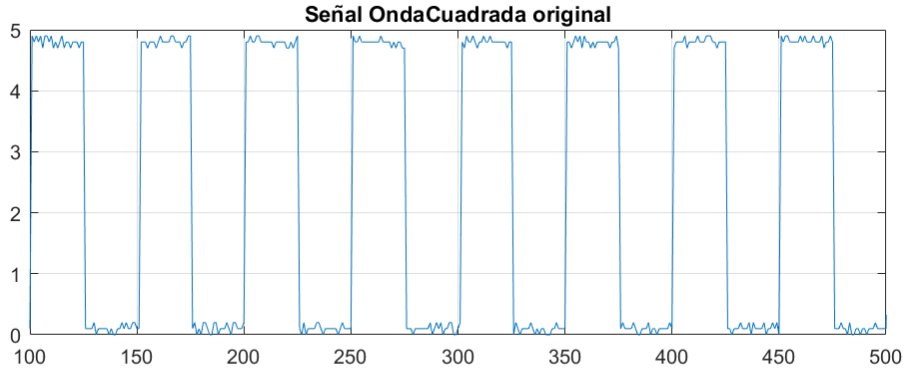


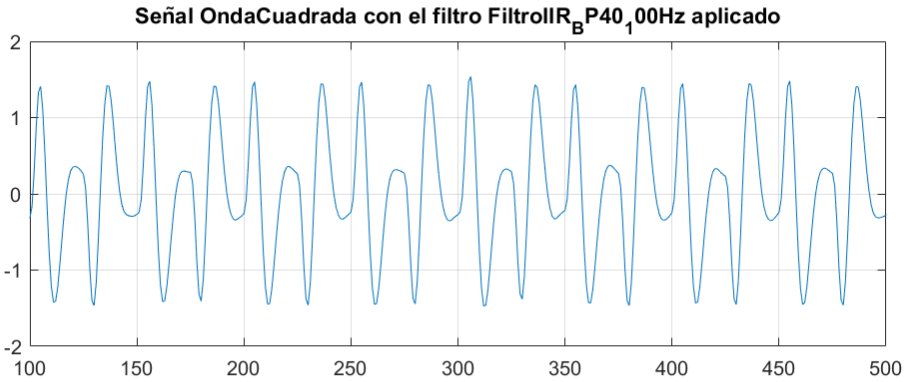


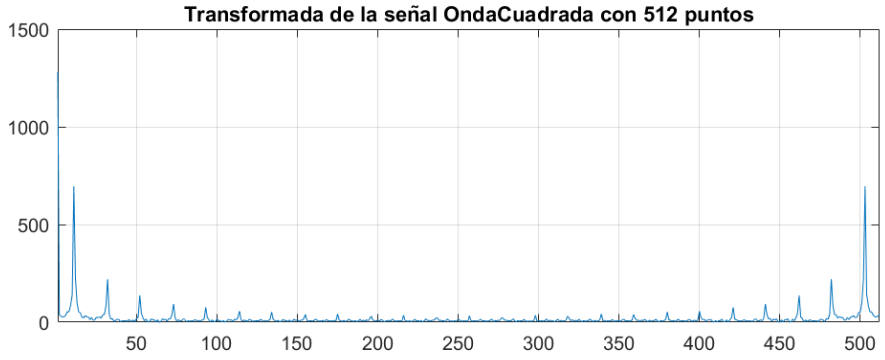


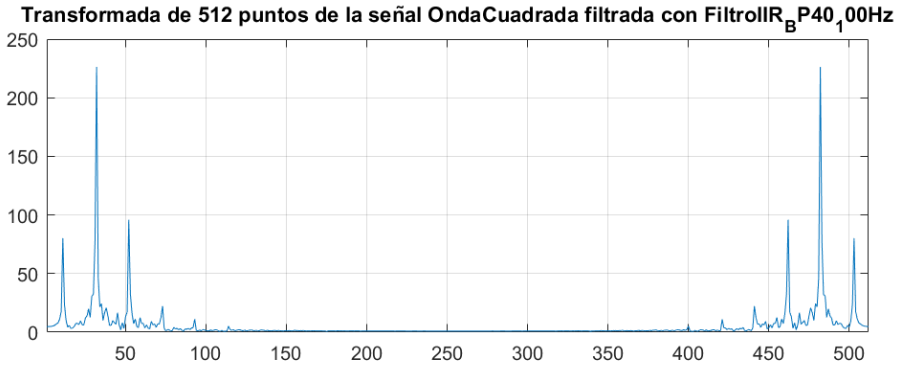


2.1.2.2 Filtro pasabanda

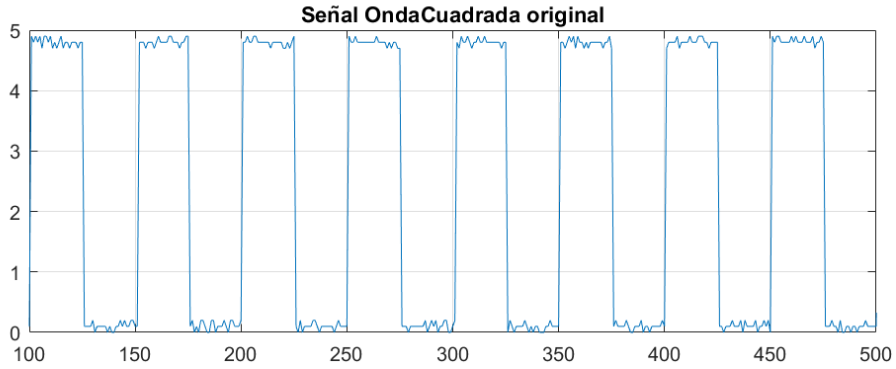


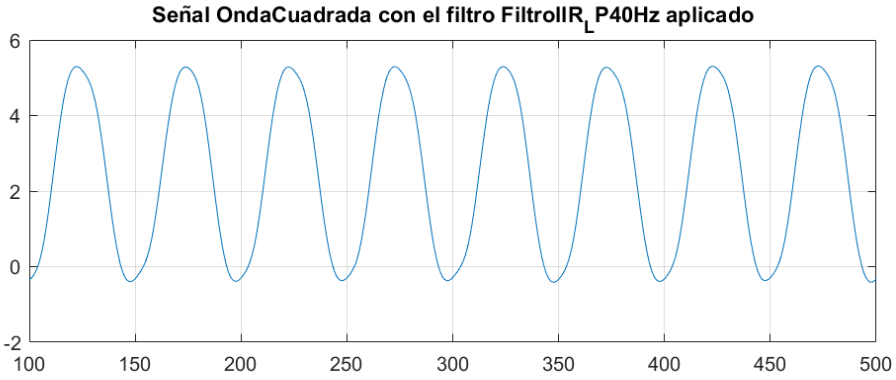


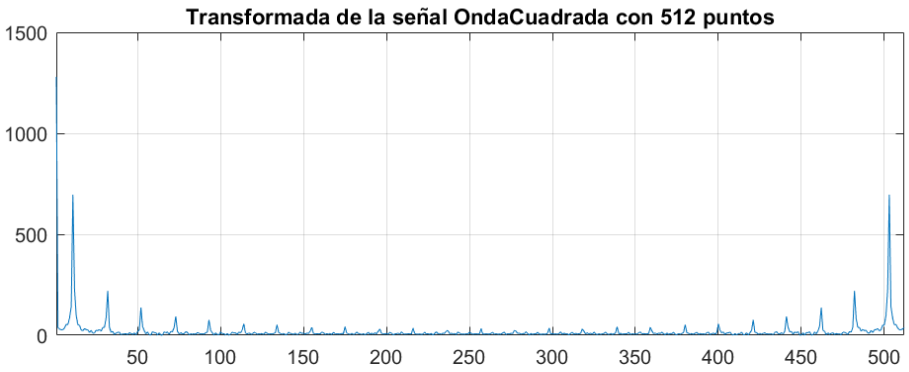


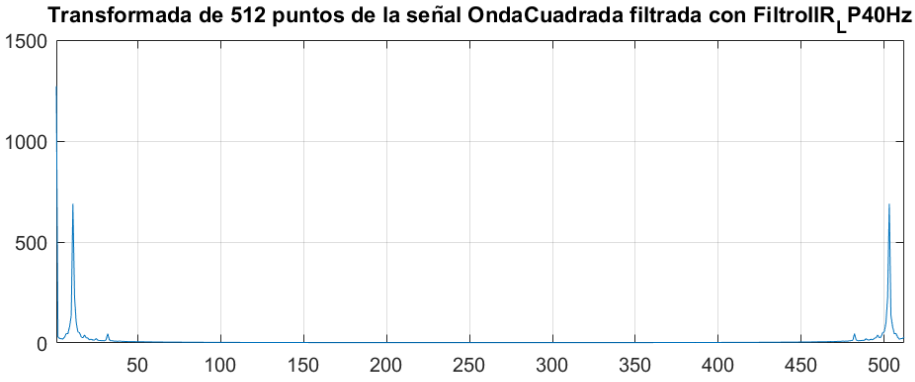


2.1.2.3 Filtro pasabajos







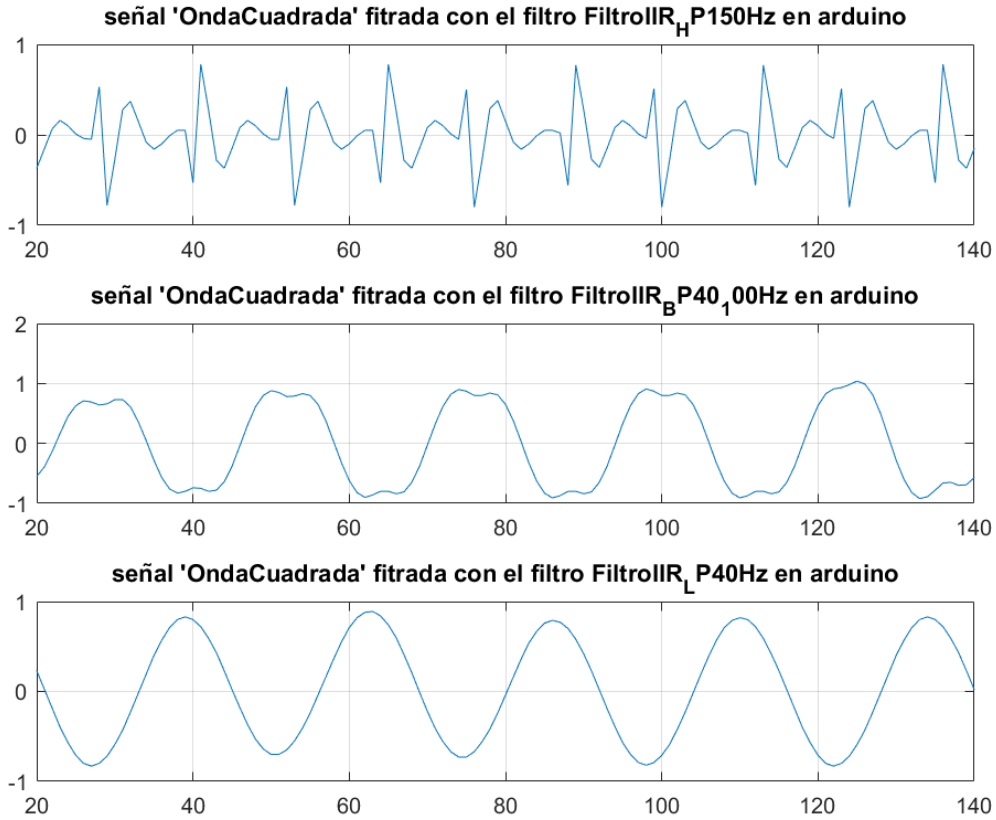


Una vez aplicados los filtros fuera de línea, se procedio a realizar el procesamiento en el simulador de arduino. El código utilizado es el siguiente:

```
//Juan Agustin Avila
//Julio 2020
//Reg 26076 - ELO
const int Ts = 1; //Tiempo de Muestreo en milisegundos
const int N = 5;  //cantidad de puntos
//numerador y denominador del filtro IIR pasabajo de 40Hz:
const float num[N] = {0.0001832160, 0.0007328641, 0.0010992961, 0.00073
28641, 0.0001832160};
const float den[N] = {0.5174781998, -2.4093428566, 4.2388639509, -
3.3440678377, 1.0000000000};
void setup()
{
    delay(10); //solo por si el generador tiene un transitorio
    Serial.begin(57600);
}

void loop()
{
    int k;
    static float entrada[N] = {0, 0, 0, 0, 0};
    static float salida[N] = {0, 0, 0, 0, 0};
    float out = 0;
    for (k = 1; k < N; k++)
    {
        entrada[k - 1] = entrada[k]; //desplaza los valores
        salida[k - 1] = salida[k];
    }
    salida[N - 1] = 0;                                          //en p
rincipio se pone en cero para que no se calcule
    entrada[N - 1] = (float)(analogRead(A0) - 512) * 2 / 1023.0; //gene
ra una señal similar a la provista por la catedra
    for (k = 0; k < N; k++)
    {
        out = out + (num[k] * entrada[k]) - (den[k] * salida[k]); //rea
liza el calculo
    }
    salida[N - 1] = out; //finalmente se asigna el valor de salida actu
al al arreglo
    Serial.println(out);
    delay(Ts); // Espera Ts
}
```

Los resultados obtenidos fueron los siguientes:



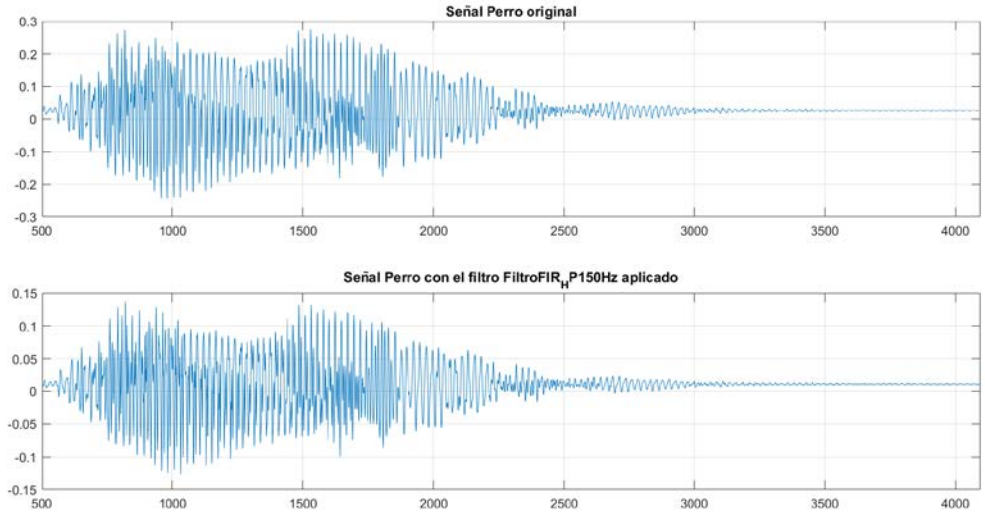
Cuyos resultados concuerdan con lo obtenido en C.

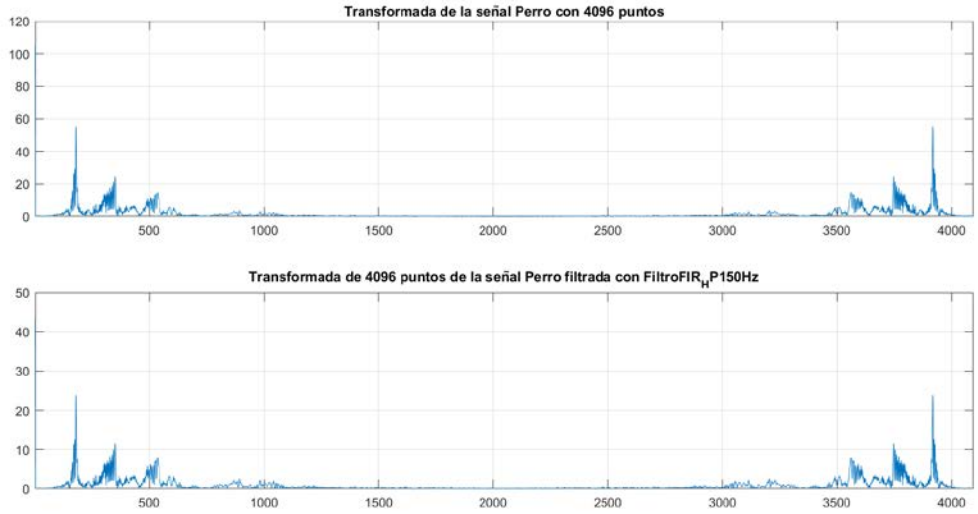
2.2 Punto 2

Se modifico el punto anterior para utilizar como señal a analizar la señal “Perro.wav”, que ya había sido convertida a archivo de texto en el laboratorio previo. Los resultados obtenidos fueron los siguientes:

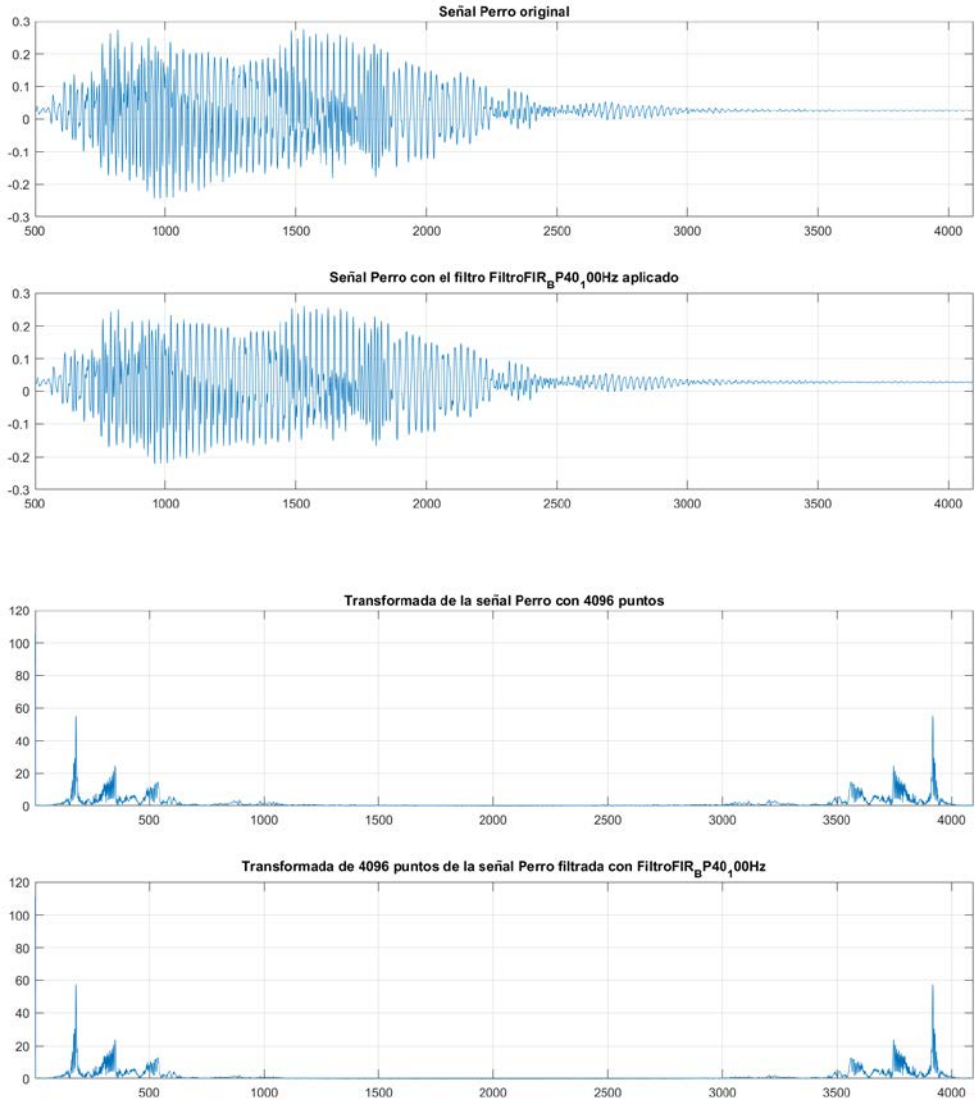
2.2.1 Filtros FIR:

2.2.1.1 Filtro pasaalto:

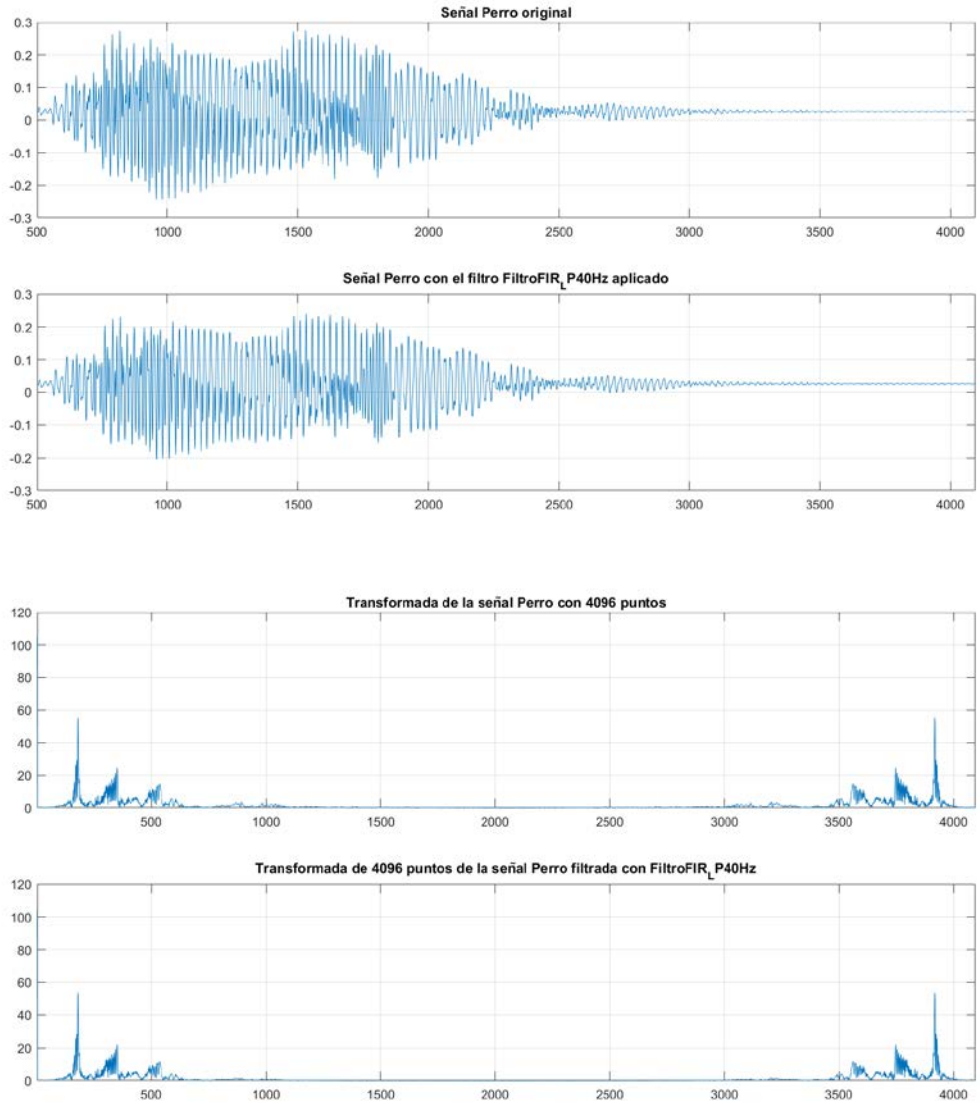




2.2.1.2 Filtro pasabanda:



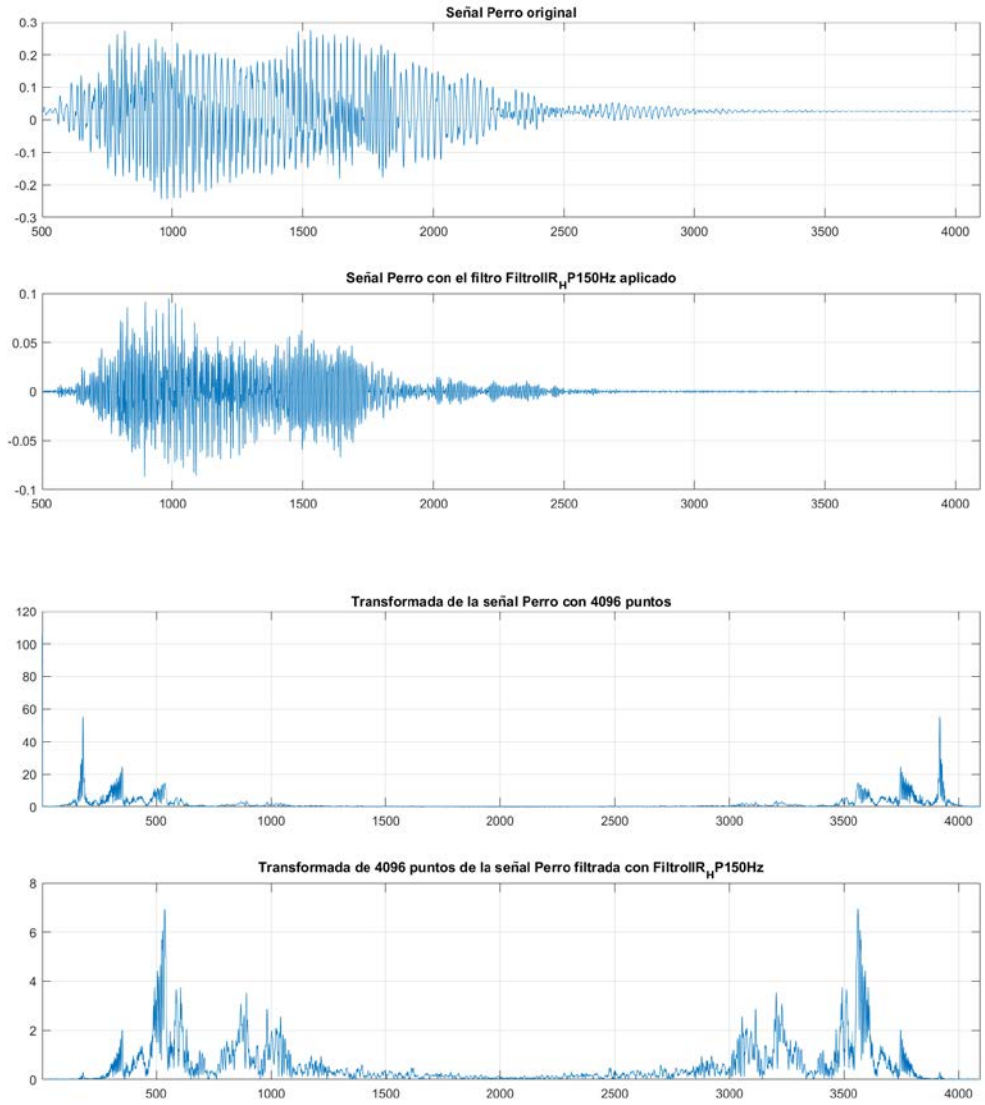
2.2.1.3 Filtro pasabajo:



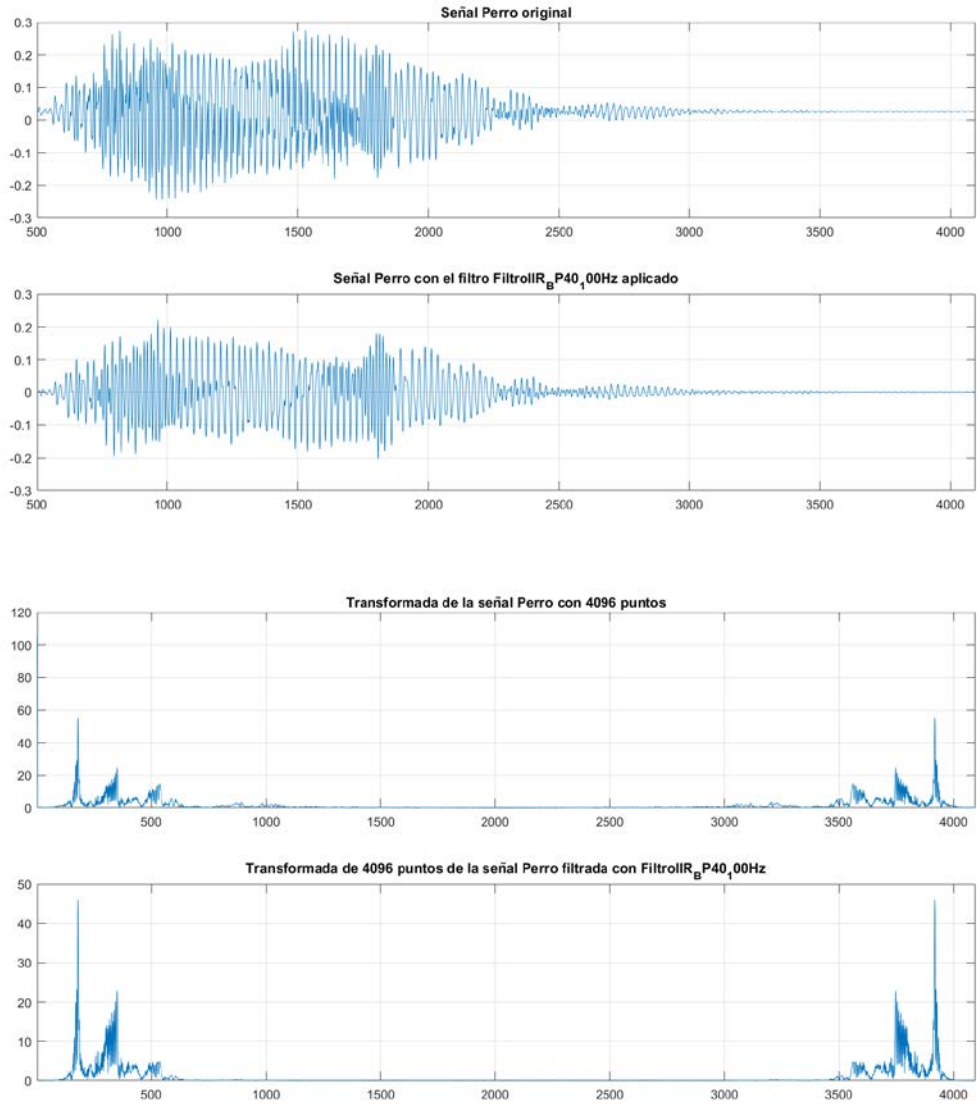
Se observa que los cambios realizados por los filtros FIR son prácticamente imperceptibles.

2.2.2 Filtros IIR

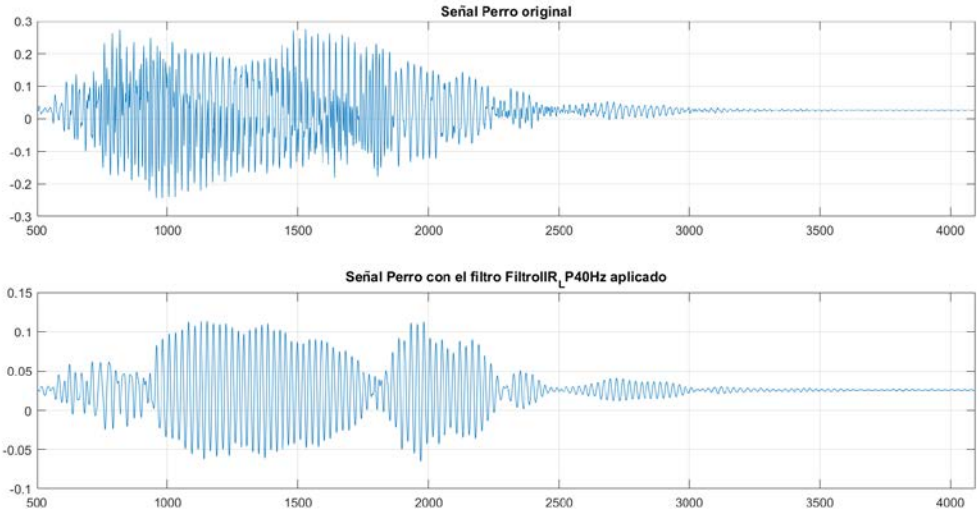
2.2.2.1 Filtro pasaalto:

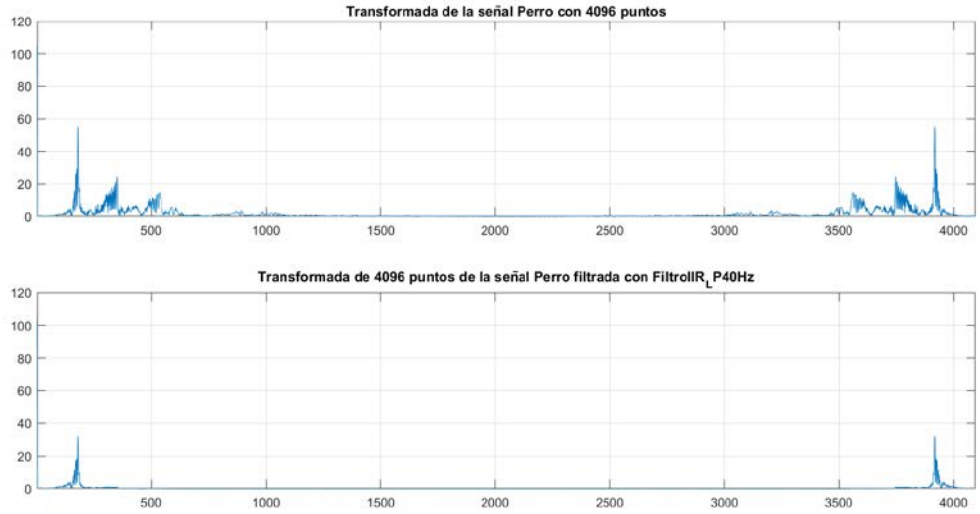


2.2.2.2 Filtro pasabanda:



2.2.2.3 Filtro pasabajos:



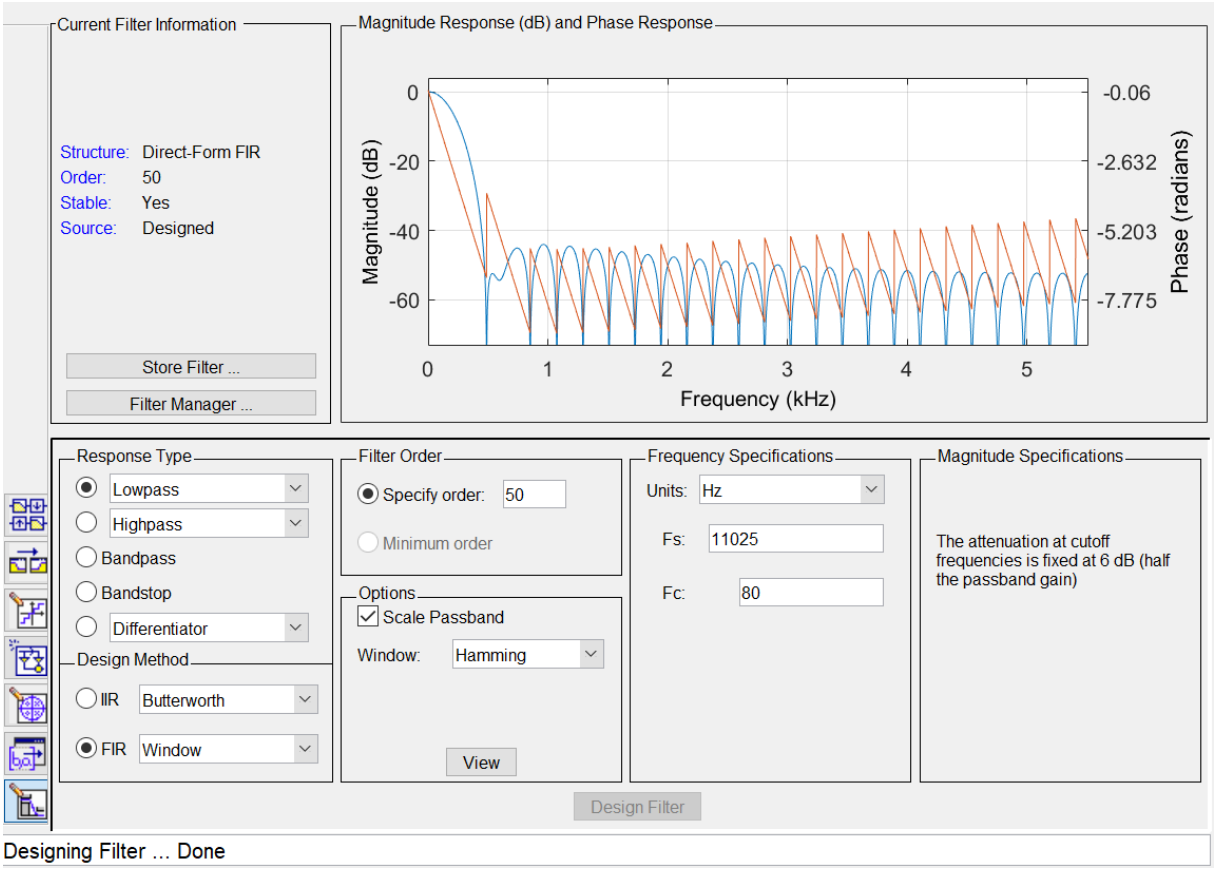


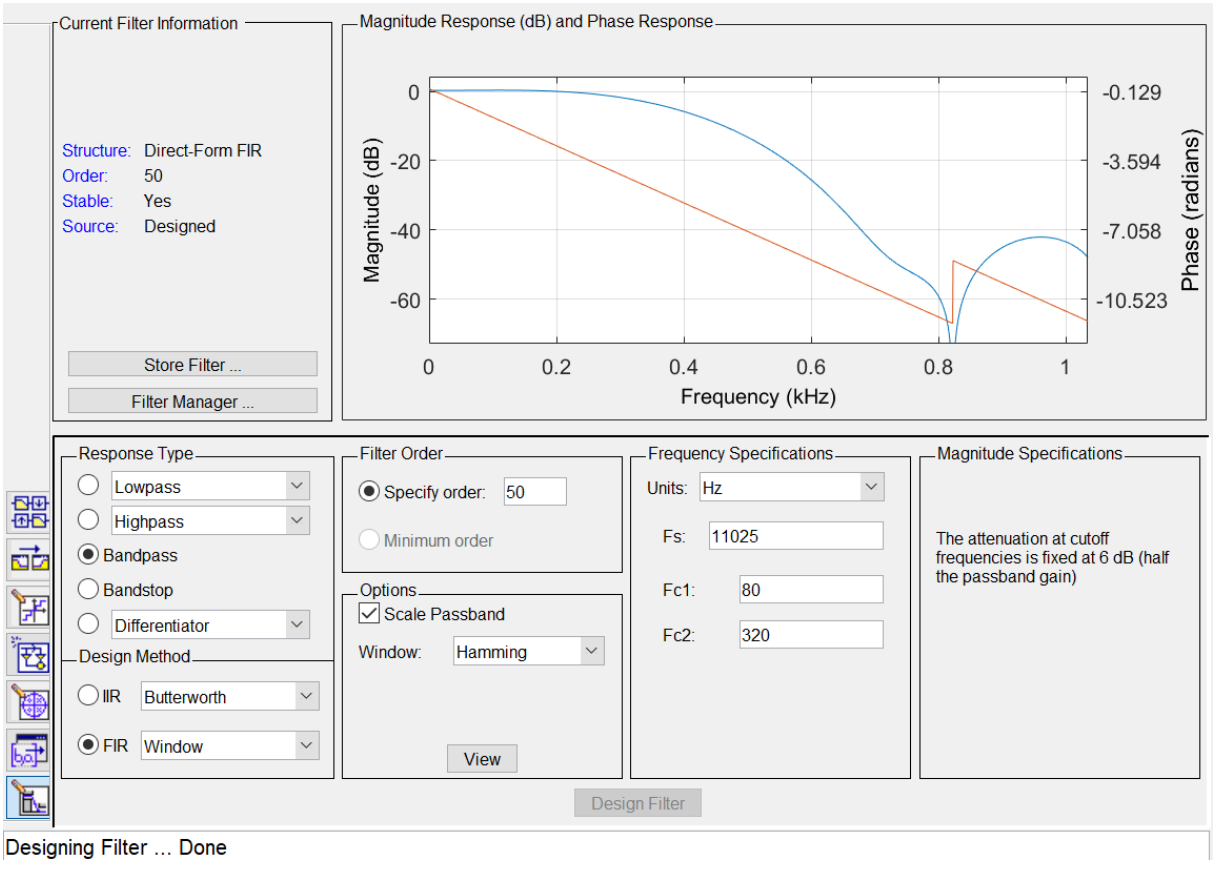
En el caso de los filtros IIR, la diferencia en el dominio tanto frecuencial como temporal es muy notable.

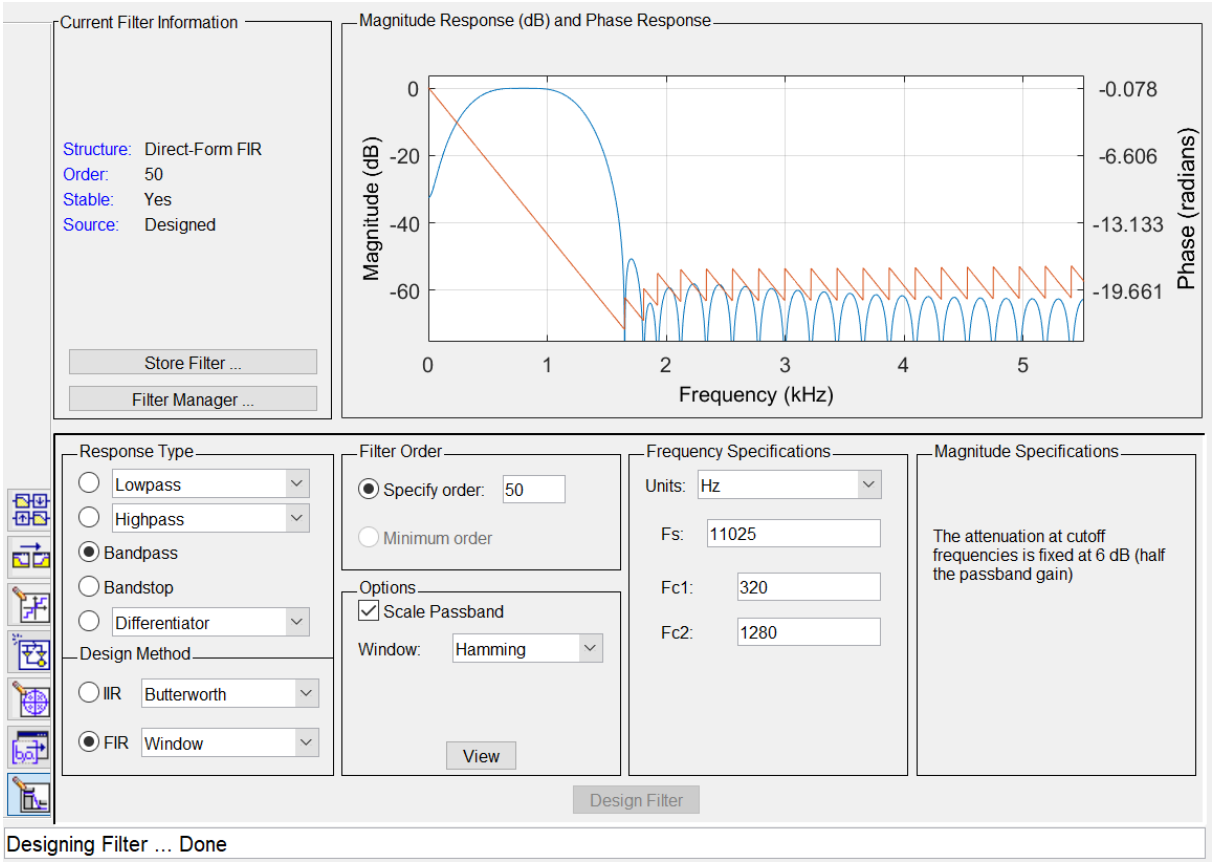
2.3 Punto 3

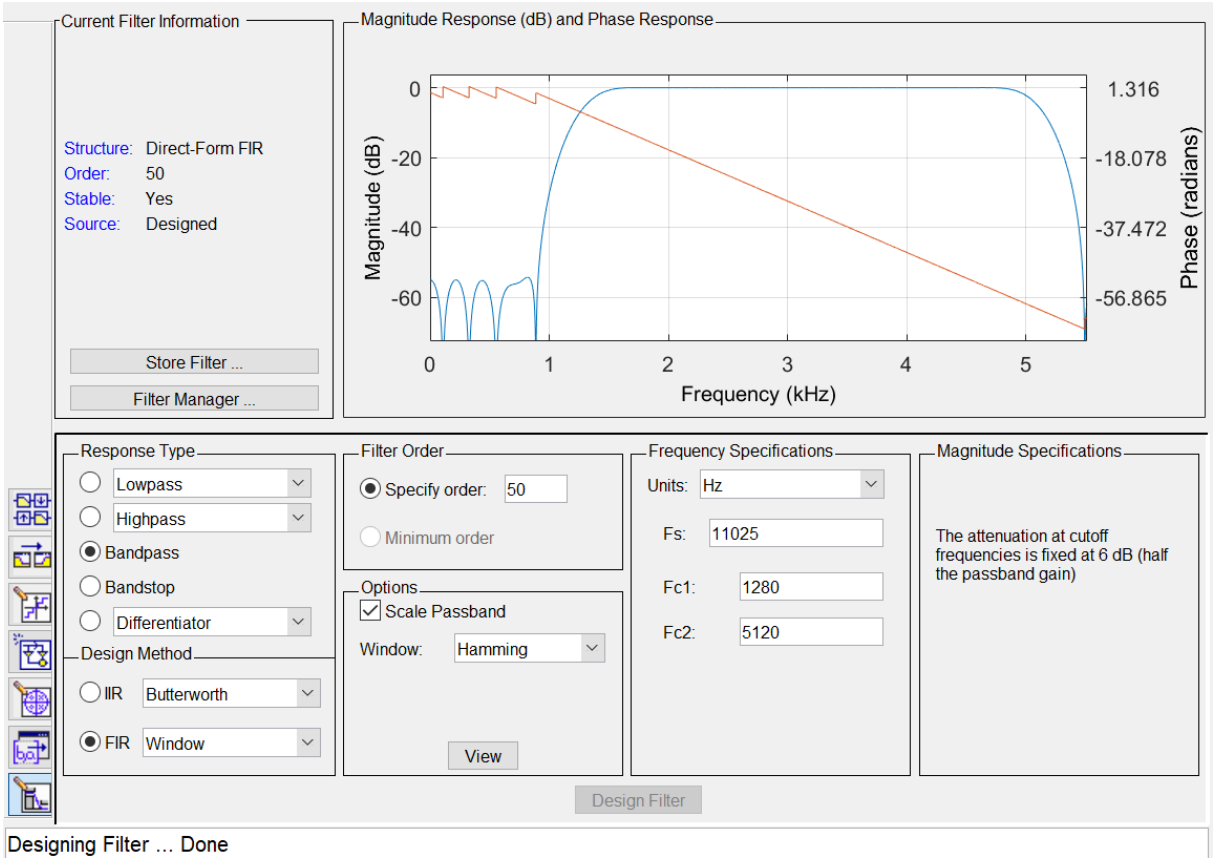
Para la realización del ecualizador de cinco bandas, se utilizaron bandas de dos octavas, arrancando en 20Hz y finalizando en 20KHz. La primer banda es un filtro LP de 80Hz la segunda banda un filtro BP entre 80Hz y 320Hz, la tercer banda un filtro BP entre 320Hz y 1280Hz, la cuarta banda un filtro BP entre 1280 y 5120Hz, y la quinta banda es un filtro HP de 5112Hz.

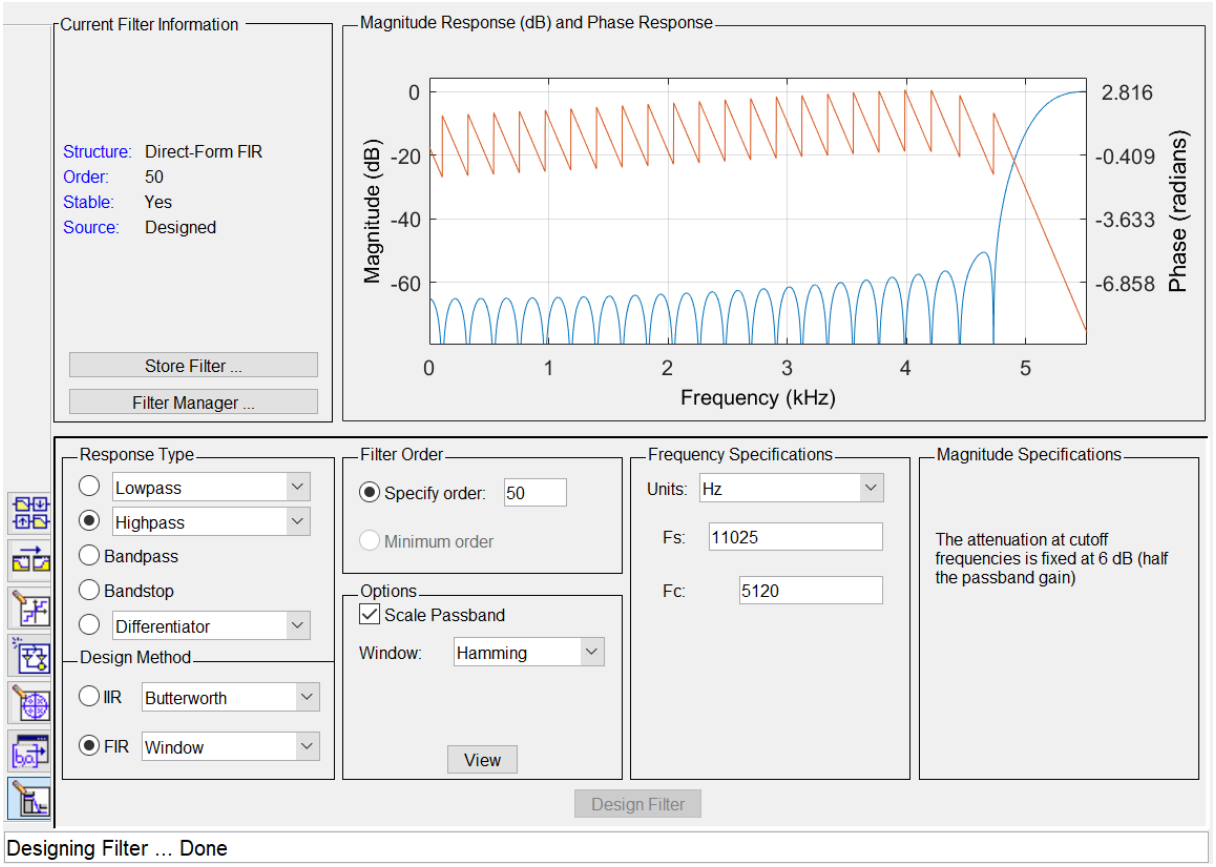
En este caso, se utilizaron filtros FIR para mantener la fase constante, y se realizaron los filtros con un orden de n=50. A continuación de reproduce la configuración de los distintos filtros:


Current Filter Information
Structure: Direct-Form FIR
Order: 50
Stable: Yes
Source: Designed
Store Filter ...
Filter Manager ...
Magnitude Response (dB) and Phase Response
Magnitude (dB)
Phase (radians)
Frequency (kHz)
Response Type
Lowpass
Highpass
Bandpass
Bandstop
Differentiator
Design Method
IIR
Butterworth
FIR
Window
Filter Order
Specify order: 50
Minimum order
Options
Scale Passband
Window: Hamming
View
Frequency Specifications
Units: Hz
Fs: 11025
Fc: 80
Magnitude Specifications
The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)
Design Filter
Designing Filter ... Done

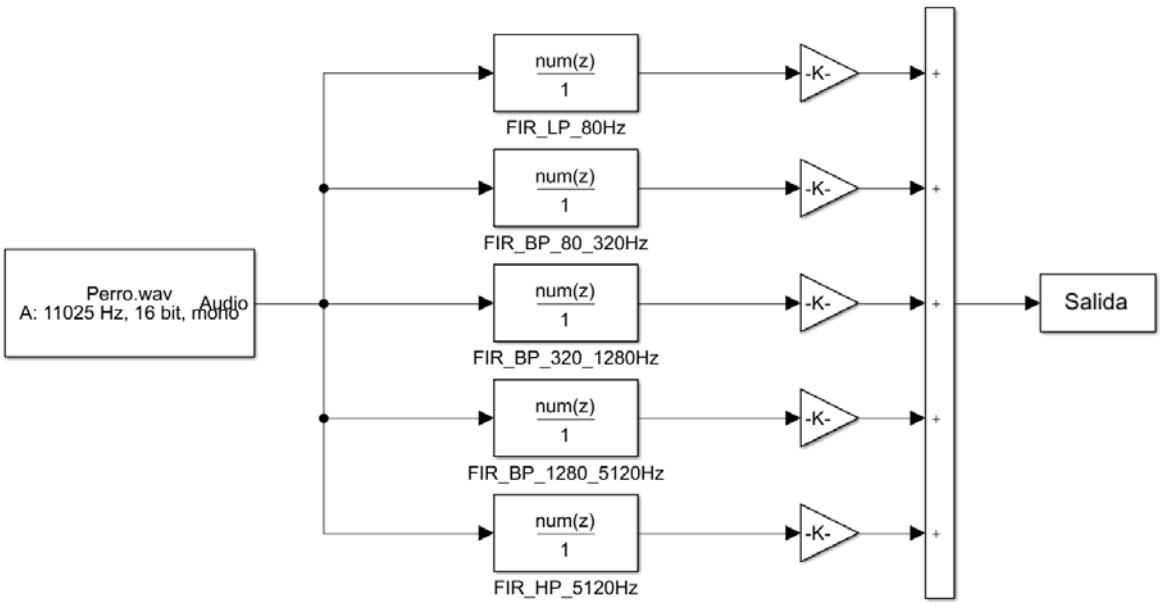

Current Filter Information
Structure: Direct-Form FIR
Order: 50
Stable: Yes
Source: Designed
Store Filter ...
Filter Manager ...
Magnitude Response (dB) and Phase Response
Magnitude (dB)
Phase (radians)
Frequency (kHz)
Response Type
Lowpass
Highpass
Bandpass
Bandstop
Differentiator
Design Method
IIR
Butterworth
FIR
Window
Filter Order
Specify order: 50
Minimum order
Options
Scale Passband
Window: Hamming
View
Frequency Specifications
Units: Hz
Fs: 11025
Fc1: 80
Fc2: 320
Magnitude Specifications
The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)
Design Filter
Designing Filter ... Done


Current Filter Information
Structure: Direct-Form FIR
Order: 50
Stable: Yes
Source: Designed
Store Filter ...
Filter Manager ...
Magnitude Response (dB) and Phase Response
Magnitude (dB)
Phase (radians)
Frequency (kHz)
0
-20
-40
-60
-0.078
-6.606
-13.133
-19.661
1
2
3
4
5
Response Type
Lowpass
Highpass
Bandpass
Bandstop
Differentiator
Design Method
IIR
Butterworth
FIR
Window
Filter Order
Specify order: 50
Minimum order
Options
Scale Passband
Window: Hamming
View
Frequency Specifications
Units: Hz
Fs: 11025
Fc1: 320
Fc2: 1280
Magnitude Specifications
The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)
Design Filter
Designing Filter ... Done


Current Filter Information
Structure: Direct-Form FIR
Order: 50
Stable: Yes
Source: Designed
Store Filter ...
Filter Manager ...
Magnitude Response (dB) and Phase Response
Magnitude (dB)
Phase (radians)
Frequency (kHz)
0
-20
-40
-60
1.316
-18.078
-37.472
-56.865
1
2
3
4
5
Response Type
Lowpass
Highpass
Bandpass
Bandstop
Differentiator
Design Method
IIR
Butterworth
FIR
Window
Filter Order
Specify order: 50
Minimum order
Options
Scale Passband
Window: Hamming
View
Frequency Specifications
Units: Hz
Fs: 11025
Fc1: 1280
Fc2: 5120
Magnitude Specifications
The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)
Design Filter
Designing Filter ... Done



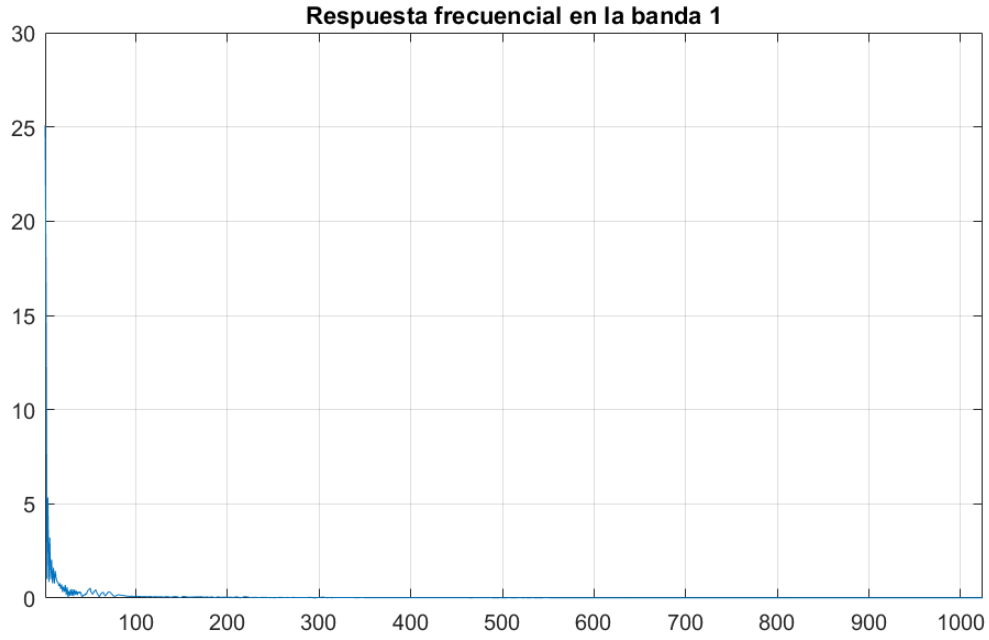
Los 5 filtros se exportaron al entorno de simulink, en el cual se armó el siguiente diagrama:

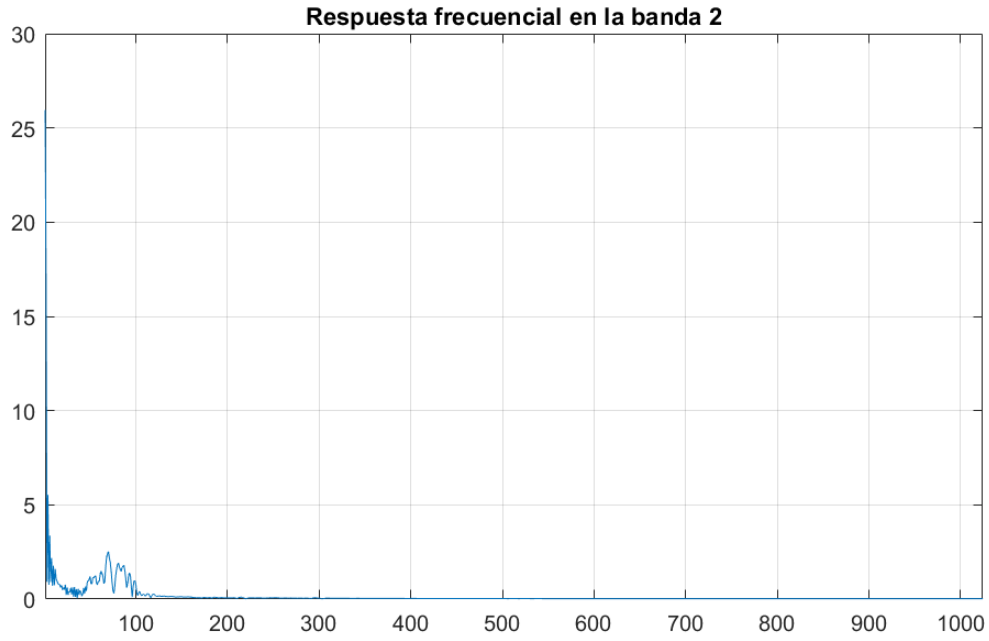


Las 5 ganancias dependen de un vector “ganancia”. Se realizó una matriz identidad de 5x5 y se simulo el sistema con cada fila de la matriz para probar individualmente los filtros:

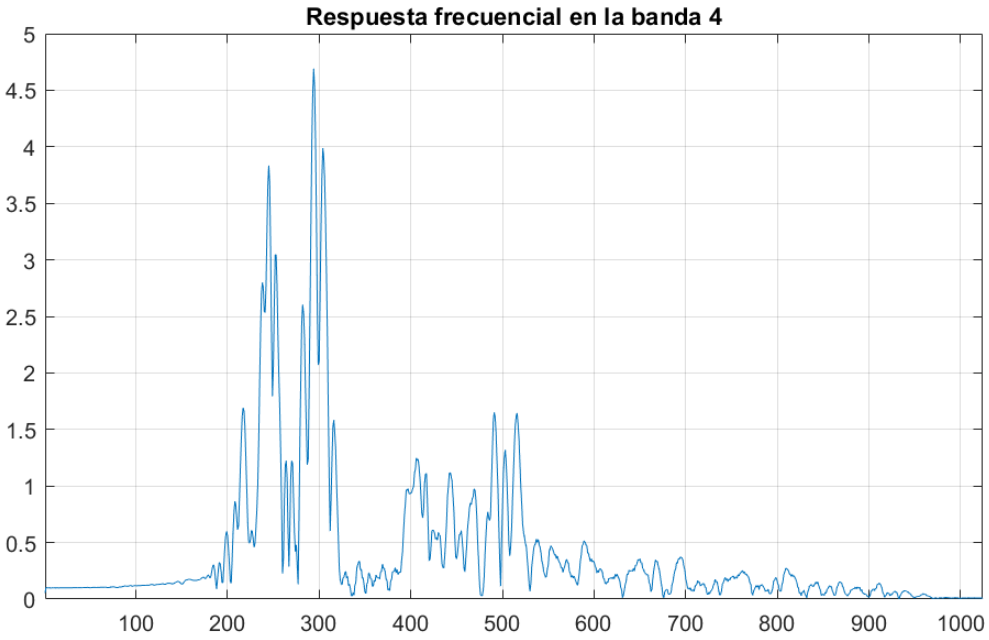
```
%% Realizacion del ecualizador
puntos=2048; ts=11025;
Gtotal=eye(5);  %Matriz identidad para probar cada filtro
for i=1:5
    ganancia=Gtotal(i,:);   %selecciona la columna I
    sim('ecualizador.slx');
    transformada=fft(Salida.Data,puntos);
```

```
    figure();
    plot((1:puntos),abs(transformada)); grid; xlim([1 puntos/2]);
    title("Respuesta frecuencial en la banda "+i);
end
```



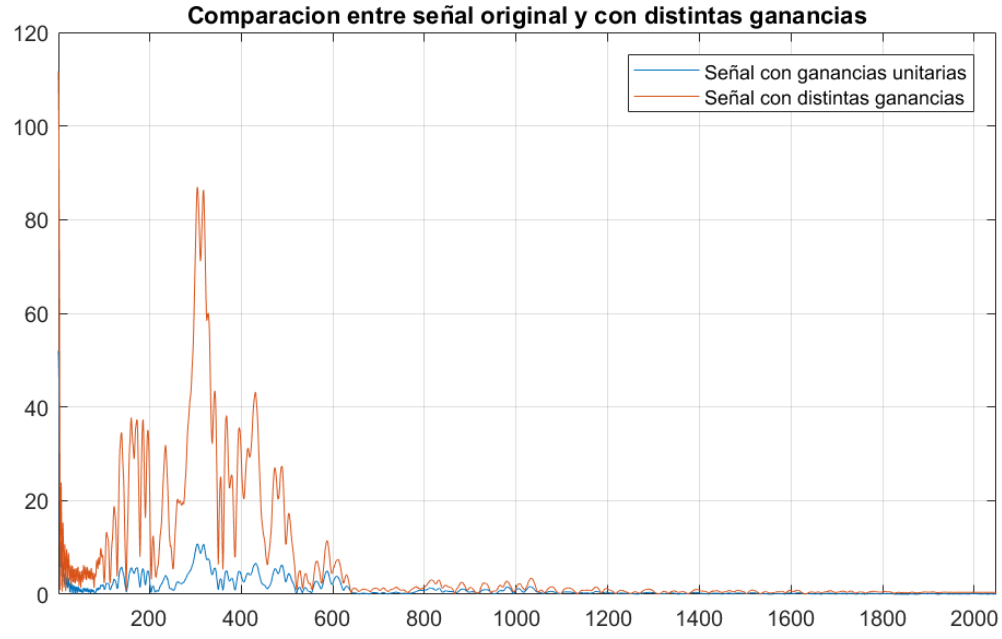



Respuesta frecuencial en la banda 3
12
10
8
6
4
2
0
100
200
300
400
500
600
700
800
900
1000


Respuesta frecuencial en la banda 4
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
100
200
300
400
500
600
700
800
900
1000

A continuación, se realizaron dos simulaciones, una con todas las ganancias unitarias y otra con las ganancias modificadas. Ambos resultados se graficaron superpuestos:

```
puntos=4096;
ganancia=[1 1 1 1 1];       %Ganancias unitarias
sim('ecualizador.slx');
transformada=fft(Salida.Data,puntos);
figure();
plot((1:puntos),abs(transformada));
ganancia=[.001 4 8 2 .1]; %distintas ganancias por banda
sim('ecualizador.slx');
transformada=fft(Salida.Data,puntos);
hold on; grid on;
plot((1:puntos),abs(transformada));xlim([1 puntos/2]);
legend("Señal con ganancias unitarias","Señal con distintas
ganancias");
title("Comparacion entre señal original y con distintas ganancias");
```



2.4 Punto 4

Algunas otras aplicaciones de filtros FIR e IIR:

.-Síntesis de sonido: creación o modificación de señales para moldear espectros o formas de onda y lograr un efecto auditivo deseado.

.-Separación de señales. ej; ruido, interferencias provenientes de otros sistemas, etc.

.-Recuperación de señales distorsionadas de alguna forma (distorsión pro transmisión).

.-Efectos de audios: chorus, flanger, phaser, reverb, etc.