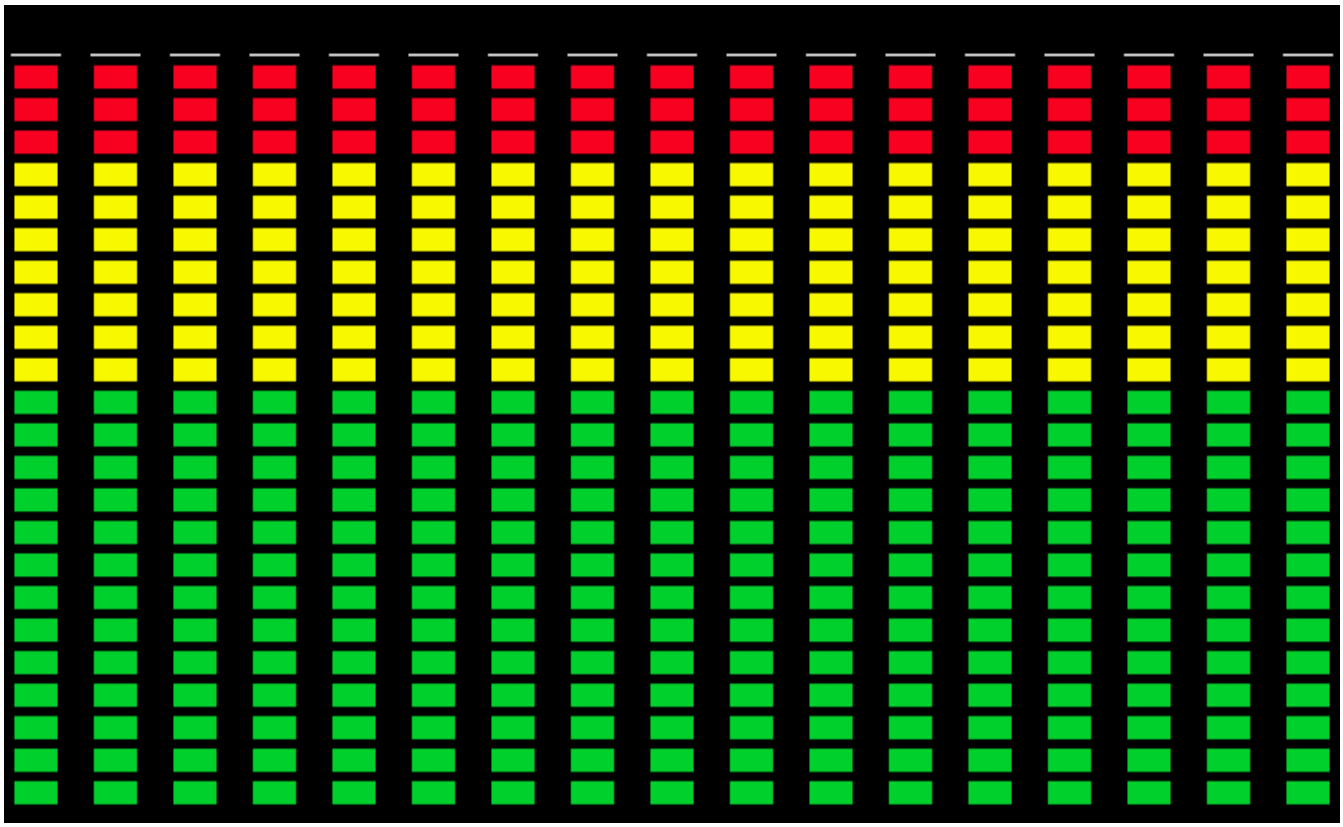
VÚMETRO



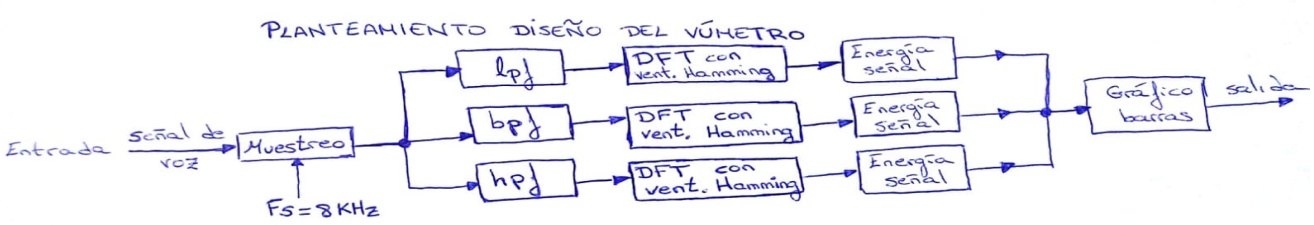
Alejandro Pérez Aguilera
Manuel Rebollo Gordillo
Isabel Rodríguez Ruiz
G-35

OBJETIVOS:

El objetivo de este proyecto es la construcción de un vúmetro en Matlab. Un vúmetro es un dispositivo indicador en equipos de audio para mostrar el nivel de señal en unidades de volumen. Para ello, captaremos la voz en tiempo real a través de un micrófono y pasado a través de nuestro sistema diseñado en Matlab se representará en bajas, medias y altas frecuencias el volumen de la señal a la salida.

PLANTEAMIENTO:

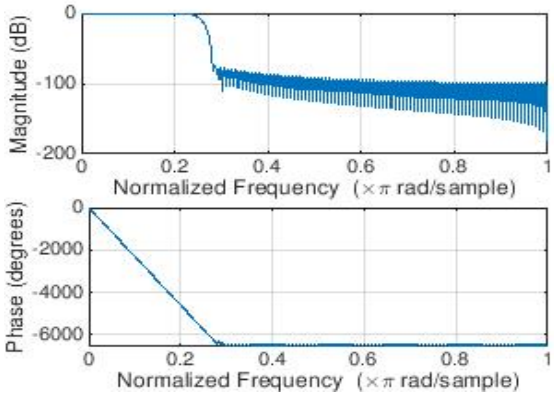
El planteamiento del diseño del vúmetro puede verse en la siguiente ilustración:

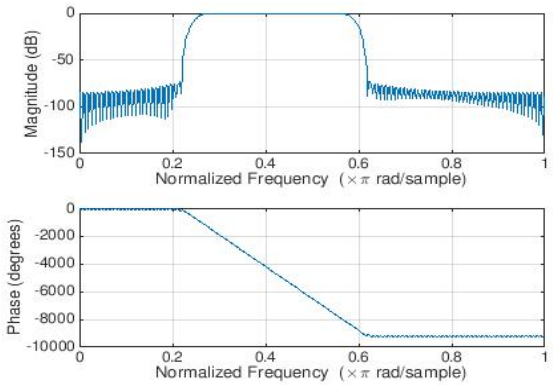


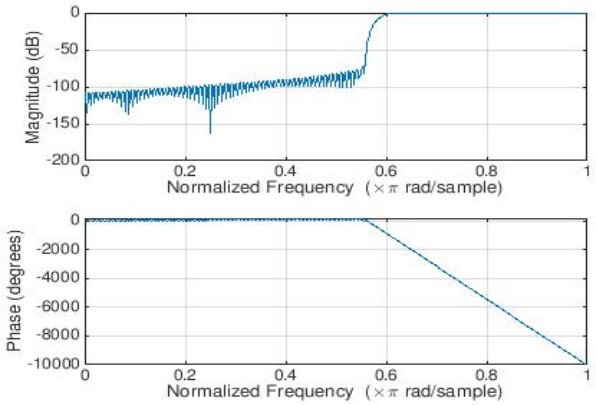
Como podemos observar, en la entrada tendremos la señal de voz que llega desde el micrófono. La señal será muestreada y a continuación será pasada por tres filtros (paso bajo, paso banda y paso alto) para dividir la señal en las tres franjas que nos interesa. Una vez hallamos las tres señales le hacemos la DFT a cada una de ellas, teniendo en cuenta que es muy recomendable enventanar la señal. Como en este caso nos interesa la amplitud de la señal, elegimos una ventana de Hamming. Además debido al bajo nivel de sus lóbulos secundarios, en frecuencia, el sistema tiene mayor sensibilidad para señales de bajo volumen. Después de ello calcularemos la potencia de cada una y finalmente, estas serán representadas en un gráfico de barras que represente el nivel de volumen de la señal de entrada.

DISEÑO DE FILTROS:

Como vemos en las ilustraciones siguientes hemos diseñado tres filtros: pasa bajo, paso banda y paso alto.



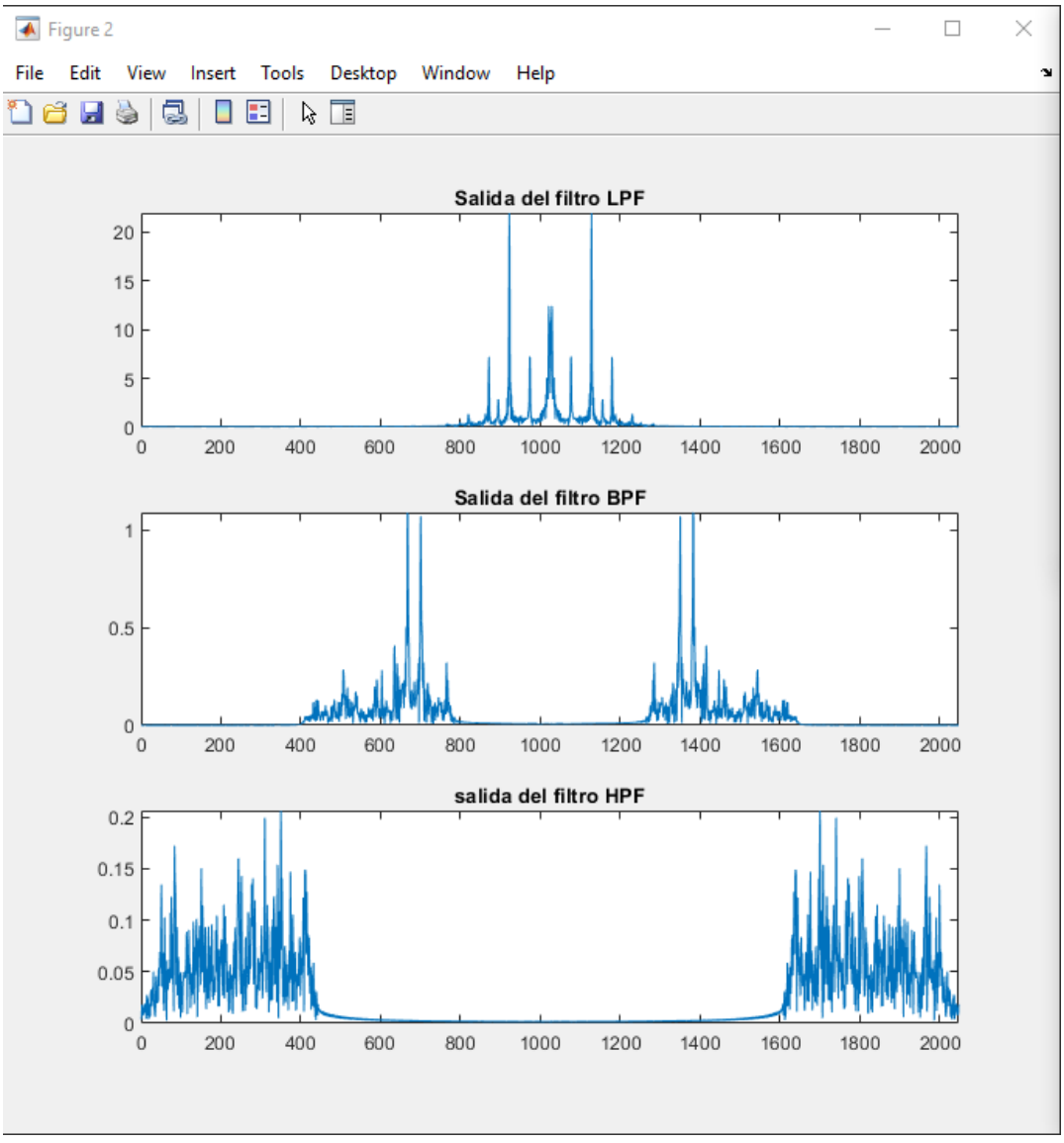




Paso bajo: Su frecuencia de corte ha sido establecida en 850 Hz como parte del espectro de voz audible en frecuencias bajas.
Paso banda: Su frecuencia en banda de paso va desde los 850 Hz hasta los 2000 Hz como parte del espectro audible en frecuencias medias.
Paso alto: Su frecuencia de corte ha sido establecida en 2000 Hz y va hasta los 3400 Hz (máximo audible en frecuencia) como parte del espectro de voz audible en frecuencias altas.

El diseño de los filtros se ha hecho con la función fir2. Esta función devuelve un filtro FIR del orden especificado (en nuestro caso 256 para acentuar la caída). Como parámetros hay que pasar a la función el orden del filtro y dos vectores. **Vector de frecuencias:** contendrá las frecuencias normalizadas en las que hay un cambio de amplitud; **vector de amplitudes:** contendrá los coeficientes de amplitud correspondientes al vector de frecuencias. En el código se puede observar el desplazamiento del vector de amplitudes para cada filtro, correspondiente a la banda de frecuencias que se desea obtener.
Una vez obtenido el filtro usamos la función filter para filtrar la señal de audio. Como es un FIR, los coeficientes del denominador serán 1. Cabe destacar, que para su representación gráfica, es necesario usar la función fftshift. Con esta función conseguimos la DFT enventanada a lo largo del tiempo ya que, al ser variante en el tiempo la señal, se necesita el desplazamiento temporal de la ventana.

Como se puede observar en todas las gráficas anteriores el eje está normalizado en frecuencia y por ello el eje va de 0 a 1. La salida de las señales en las salidas de los filtros serán:



Cálculo de energía de la señal:

Una vez se tiene la señal separada en tres bandas de frecuencia se pasa al cálculo de la potencia localizada para cada una de las tres bandas. Para ello haremos uso de la siguiente expresión:

$P_x = \frac{1}{N}\sum_{n=0}^{N-1} x^2(n),\ N = dur\left[x(n)\right]$ o bien en nuestro caso, calculamos la energía como:

$E_x = T * \sum_{n=0}^{N-1} x^2(n),\ N = dur\left[x(n)\right]$

Una vez obtenidos los tres valores se representan en un diagrama de barras.

CÓDIGO:

Main.m

```
%Programación de un vúmetro digital
clear
clc
%Estado de los filtros as global
global filterOrder;
global stateLpf;
global stateBpf;
global stateHpf;
global plot_handler_1;
global plot_handler_2_1;
global plot_handler_2_2;
global plot_handler_2_3;
global plot_handler_3;
global N;
global L;
% Datos a tener en cuenta
N = 1024; % Numero de muestras
Fs=8000; % Frecuencia de muestreo
L=1024; % Longitud de los bloques de la señal de entrada
filterOrder = 256; % Polos del filtro
stateLpf = zeros(1,filterOrder);
stateBpf = zeros(1,filterOrder);
stateHpf = zeros(1,filterOrder);
%Representacion de la señal original
figure(1);
title('Señal de entrada')
plot_handler_1 = plot(zeros(1,N));
xlim([0 N-1]);

%Representacion de la señal filtrada en las diferentes bandas
figure(2);
title('SEÑAL FILTRADA')
subplot(3,1,1);
plot_handler_2_1 = plot(zeros(1,2*N));
xlim([0 2*N-1]);
title('Salida del filtro LPF');
xlabel('Frecuencia en Hz');
ylabel('Amplitud');
subplot(3,1,2);
plot_handler_2_2 = plot(zeros(1,2*N));
xlim([0 2*N-1]);
title('Salida del filtro BPF');
xlabel('Frecuencia en Hz');
ylabel('Amplitud');
```

```
subplot(3,1,3);
plot_handler_2_3 = plot(zeros(1,2*N));
xlim([0 2*N-1]);
title('salida del filtro HPF');
xlabel('Frecuencia en Hz');
ylabel('Amplitud');
%Reprensentación en gráfico de barras de la señal
figure(3);
plot_handler_3 = bar(0);
title('Gráfico de barras en las tres bandas');
ylim([-60 60])
xlabel('Bandas baja, media y alta');
ylabel('Energía de la señal en dB. Volumen');
daqreset();
% Programación del canal de entrada analógica
s=daq.createSession('directsound');            % Se crea una sesión
ea=addAudioInputChannel(s, 'Audio1', 1,'Audio');  % Se añade un canal de entrada
s.Rate = Fs; % Frecuencia de muestreo
s.IsContinuous=true; % Operation continua
s.NotifyWhenDataAvailableExceeds = L; % Se avisa cuando hay mas de L muestras en la FIFO de entrada
% Listener
lh = addlistener(s,'DataAvailable',@(src,event) callback(event.Data, Fs));
% Comienzo de la operación
startBackground(s);  % Operación en Background
```

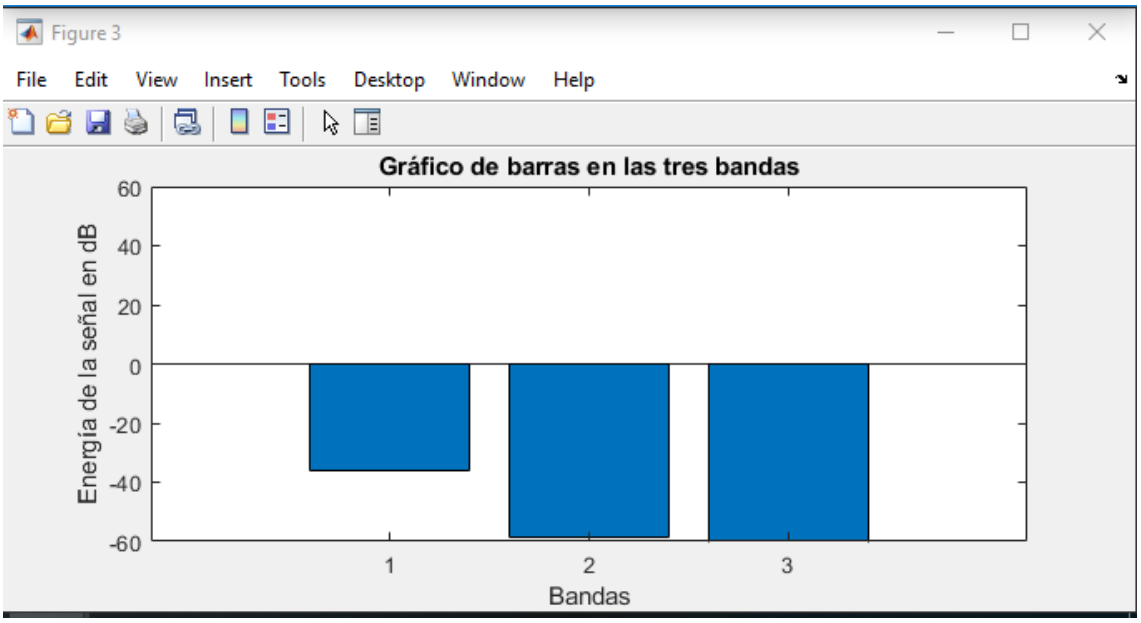
Callback.m

```
function callback(data,fs)
global filterOrder;
global stateLpf;
global stateBpf;
global stateHpf;
global plot_handler_1;
global plot_handler_2_1;
global plot_handler_2_2;
global plot_handler_2_3;
global plot_handler_3;
global N;
set(plot_handler_1,'YData',data);
% Corta en 850 Hz haciendo las bajas frecuencias
blpf = [0 0.25 0.25 0.588 0.588 1];
mli = [1 1 0 0 0 0];
bli = fir2(filterOrder,blpf,mli);
%freqz(bli,1);
% Corta en 2000 Hz aproximadamente haciendo las frecuencias medias
bbpf = [0 0.25 0.25 0.588 0.588 1];
mpi = [0 0 1 1 0 0];
bpi = fir2(filterOrder,bbpf,mpi);
%freqz(bpi,1);
% hace las frecuencias altas llegando a 3400 Hz
bhpf = [0 0.25 0.25 0.588 0.588 1];
mhi = [0 0 0 0 1 1];
bhi = fir2(filterOrder,bhpf,mhi);
%freqz(bhi,1);
%Filtramos la señal de entrada con los filtros diseñados
[slpf, stateLpf]=filter(bli, 1, data, stateLpf);
[sbpf, stateBpf]=filter(bpi, 1, data, stateBpf);
[shpf, stateHpf]=filter(bhi, 1, data, stateHpf);
%hacemos la fft y el plot de los filtros
```

```
fftLpf = fftshift(abs(fft(slpf.*hamming(N),N*2)));
fftBpf = fftshift(abs(fft(sbpf.*hamming(N),N*2)));
fftHpf = fftshift(abs(fft(shpf.*hamming(N),N*2)));
set(plot_handler_2_1,'YData',fftLpf);
set(plot_handler_2_2,'YData',fftBpf);
set(plot_handler_2_3,'YData',fftHpf);
%Cálculo de la energía de las señales
elpf = (1/fs)*sum(abs(slpf.^2));
ebpf = (1/fs)*sum(abs(sbpf.^2));
ehpf = (1/fs)*sum(abs(shpf.^2));
%Definimos los datos que lleva dentro nuestro gráfico de barras y lo
%convertimos en dBs
barsData = [elpf ebpf ehpf];
set(plot_handler_3,'YData',10*log10(barsData));
end
```

RESULTADO:

Representación, en gráfico de barras, de la energía en dB.



CONCLUSIONES:

Después de las pruebas realizadas podemos decir que el sistema desarrollado es un modelo totalmente funcional de un vúmetro. Creemos que el código y algoritmos usados serían factibles en otros entornos, ya que en este caso estamos limitados por el hardware usado (PC y dispositivos de captación de señal poco profesionales). Además, aparte de tener el nivel de señal, conseguimos visualización en tiempo real de la señal en el dominio del tiempo y transformado, lo que nos sirve para comprobar que el volumen de señal está calculado y concentrado en las bandas correctas. En definitiva, este trabajo nos ha servido para comprobar una de las aplicaciones del tratamiento de señal en tiempo real y que tenga una aplicación práctica.