Conversor de Frecuencia de Muestreo y Ecualizador para Señales de Audio en Tiempo Discreto

Pablo Bermeo
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
pablo.bermeo@ucuenca.edu.ec

Sebastián Guazhima
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
sebastian.guazhima@ucuenca.edu.ec

Tyrone Novillo
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
tyrone.novillo@ucuenca.edu.ec

***Resumen*—The following report presents the implementation of a sample frequency conversor and equalizer on MatLab. The process of converting sample frequency is achieved by decimating and expanding a discrete time signal. This signal is comoposed by the samples of an audio signal. On the other hand, equalizing the signal is composed by six possibilities: Sub-Bass (16Hz a 60 Hz), Bass (60Hz a 250Hz), Low-Mids (250Hz a 2KHz), High-Mids (2KHz a 4KHz), Presence (4KHz a 6KHz) y Brilliance (6KHz a 16KHz). Finally, the implementation offers a graphical user interface to interact with the options of the conversor and equalizing controls.**

***Keywords*—** Audio signals, MATLAB, Fourier Transform, Frequency domain filters, Signal processing, Spectral analysis

I. Introducción

El procesamiento de señales es un dominio que abarca múltiples disciplinas con distintas finalidades relacionadas a la manipulación de una señal continua muestreada: modificar ganancias, acortar la duración, suprimir o incrementar el peso de ciertas frecuencias, etc. Todo esto, a través de la dualidad entre el dominio del tiempo, o discreto, y el dominio de la frecuencia, relacionadas mediante la Transformada de Fourier.

Un proceso de suma utilidad en el procesamiento de señales, tiene que con la modificación de la amplitud del espectro para ciertas frecuencias. Específicamente, al efecto de variar las ganancias en ciertas bandas de frecuencia se le denomina Ecualización.

Por otra parte, la técnica de Muestreo, caracterizada por su frecuencia de muestreo, genera una señal discreta a partir de una continua; proceso que tiene consecuencias sobre el dominio de la frecuencia. Específicamente, se produce la generación de réplicas periódicas del espectro original, siendo este el punto de partida para el procesamiento de señales discretas.

A partir de este último punto de partida, es que se producen los procesamientos sobre la señal: modificar ganancias, eliminar frecuencias, acortar o expandir el espectro o modificar la frecuencia de muestreo. Este último procedimiento es de suma importancia, pues permite, justamente, variar la frecuencia de muestreo original a partir de muestrear los valores obtenidos de la función continua inicial. El muestreo de la señal discreta, y por lo tanto modificación de la frecuencia de muestreo, resulta en procesos de Expansión y Decimación de la señal.

En consideración de los conceptos nombrados con anterioridad, se ha planteado la implementación de un ecualizador y conversor de la tasa de muestreo, en MatLab. Específicamente, se ha de permitir al usuario elegir un factor de Decimación o Expansión, además de modificar la ganancia del espectro en seis bandas distintas: *Sub-Bass* (16Hz a 60 Hz), *Bass* (60Hz a 250Hz), *Low-Mids* (250Hz a 2KHz), *High-Mids* (2KHz a 4KHz), *Presence* (4KHz a 6KHz) y *Brilliance* (6KHz a 16KHz). Esto, con el objetivo de aplicar la teoría del análisis de Fourier para señales en tiempo discreto y muestreo, en un ejercicio práctico de interés.

II. Marco teórico

II-A. Teorema del muestreo

Suponga $x(t)$ como una señal con ancho de banda limitado con $X(jw) = 0$ para $|\omega| > \omega_M$. Entonces $x(t)$ es **únicamente** determinado por sus muestras $x(nT)$, $n = 0, \pm 1, \pm 2, \cdot$, si

$$\omega_s > 2\omega_M$$

donde

$$\omega_s = \frac{2\pi}{T}$$

Dadas la muestras, es posible reconstruir la señal a través de un tren de pulsos ideal y un filtro paso bajo. El filtro paso bajo tiene un valor de ganancia T y una frecuencia de corte $\omega_M < \omega_c < \omega_s - \omega_M$. [1]

II-B. Muestreo de señales discretas: conversión de la tasa de muestreo

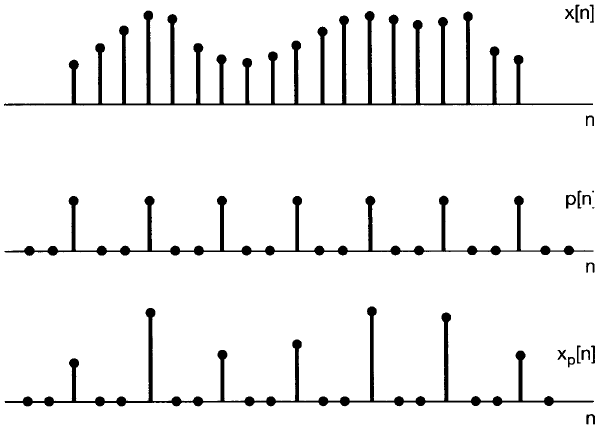

Figura 1: Expansión de la muestras a través del Muestreo mediante un tren de pulsos.[Tomado de [1]]

II-B1. Expansión: La Expansión de una señal discreta $x[n]$ con frecuencia de muestreo f_s, $x[n/L]$ puede ser vista como una forma de muestreo de la misma. En tanto que luego tomar, o

muestrear, un valor de $x[n]$ se dan $L-1$ valores iguales a cero, nuevamente se toma un valor de $x[n]$ y se colocan la misma cantidad de ceros hasta la siguiente muestra. Esto proceso se repite de tal manera que se ha muestreado $x[n]$ asumiendo valores de 0 entre cada muestra. Este efecto se observa en la Figura 1.

El efecto que tiene la expansión de las muestras sobre el espectro original $X(e^{j\omega})$ de $x[n]$ es la compresión del espectro por un factor L.

$$x[n/L] \longrightarrow X(e^{jL\omega})$$

El efecto de la expansión en el dominio del tiempo genera que frecuencia de muestro aumente por una factor L [2].

Si a este proceso de Expansión se le adjunta un procedimiento de filtrado paso bajo (que elimina las réplicas del espectro), se consigue la Interpolación de las muestras $x[n]$ con una frecuencia de muestreo de Lf_s.

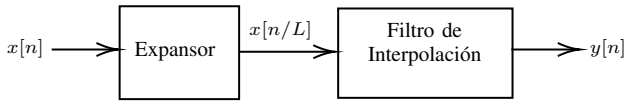

Figura 2: Esquema general de interpolación.

II-B2. Decimación: La Decimación, o Diezmado, selecciona valores de una colección de muestras $x[n]$ de tal forma que las muestras resultantes $x_b[n] = x[Mn]$. En tanto, el proceso provoca la pérdida de información: se seleccionan únicamente los valores ubicados en $n = kM$, es decir en múltiplos de M.

Debido al proceso de Diezmado, la frecuencia de muestreo es reducida en un factor M [2].

En cuanto, al efecto en la frecuencia es posible demostrar que dado el espectro de una señal discreta expandida, o muestreada mediante un tren de pulsos, $X_p(e^{j\omega})$, el espectro de la señal diezmada es la expansión del mismo. Es decir:

$$X_{\text{Diezmada}}(e^{jw}) = X_p(e^{\frac{j\omega}{M}})$$

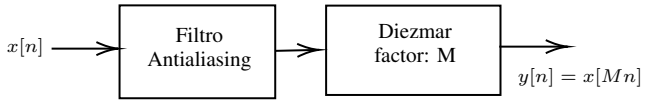

Figura 3: Esquema general para diezmar una señal.

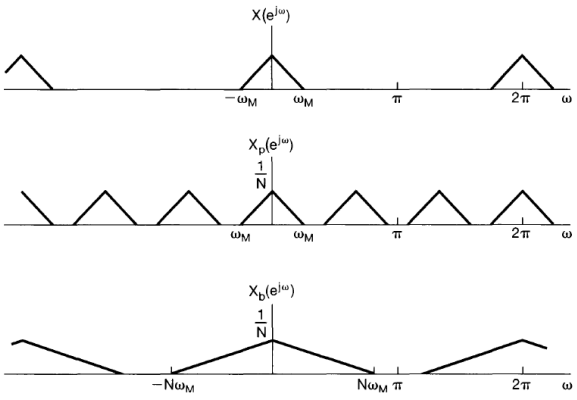

Figura 4: Espectro de la señal diezmada, por un factor N, a partir de la señal muestreada.[Tomado de [1]]

III. Desarrollo

A continuación, se detalla el proceso para implementar un script de Matlab el cual procesa una señal de audio $.wav$ para realizar: filtración, transformada de Fourier, decimación, expansión (interpolación) y ecualización.

El programa deberá, de procesar un audio, que pasará por un proceso de diezmado o de expansión (convertidor de frecuencias) para luego realizar un proceso de ecualización mediante la aplicación de filtros que son las bandas del ecualizador y finalmente obtener la salida del audio procesada para escuchar los cambios realizada en la misma.

III-A. Obtención de datos

Para realizar el programa, se considera trabajar con archivos de audio con extensión `.wav`, debido a que son un formato no codificado y sin pérdida.

Utilizando la función de Matlab se consigue la función de la señal de audio junto con su frecuencia de muestreo. Cabe recalcar que la señal está muestreada, es decir, está en tiempo discreto.

Luego, se procede a obtener los datos importantes de la señal de audio, como la duración del mismo y el número de muestras, además se convierte el audio estéreo de dos canales a una señal de un canal (mono).

El objetivo es generar una señal de una dimensión, que represente una mezcla ambos canales, se utiliza para simplificar procesamientos posteriores de la señal de audio.

III-B. Análisis del audio original

Para realizar el análisis de la señal de audio, se requiere obtener la señal en el dominio del tiempo (continuo y discreto) y además la señal de audio en el dominio de la frecuencia. Básicamente, para guiarnos en el procesamiento posterior de la señal.

Para graficar la señal de audio de entrada x(t), se construye un vector de tiempo que utiliza la duración del audio con pasos 1/fs. Y para la señal x[n], un vector utilizando el número de muestras.

En el dominio de frecuencia, se requiere realizar la Transformada rápida de Fourier (FFT), función disponible en Matlab, el cual obtenemos el espectro de frecuencias que contiene la señal de audio. Importante para observar en donde existe una mayor concentración de frecuencias.

En la Figura 5, se observa la señal en tiempo continuo de duración de 30seg y con su respectiva amplitud, en la señal de tiempo discreto dibujada con su respectiva cantidad de muestras (1404928 muestras) y finalmente en frecuencia observamos la extensión del rango de frecuencias de la señal y como las magnitudes mayores del espectro se concentran en las frecuencias bajas que en las altas (debido al ser una canción, la voz es la mas representativa).

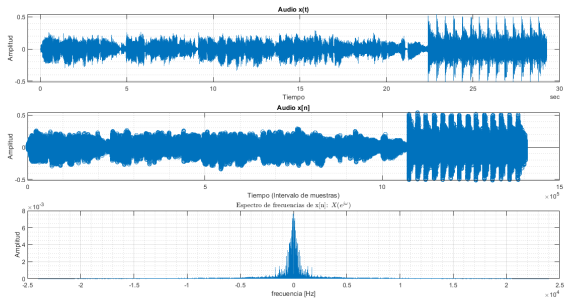

Figura 5: Gráfica de la señal de audio original: x(t), x[n] y el espectro de x[n]

A partir de la obtención de estas señales, se observará los cambios de la señal en los diferentes procesamientos realizados.

III-C. *Conversor de frecuencia de muestreo*

Este proceso permite aumentar o disminuir la frecuencia de muestreo de la señal original, esta operación se le conoce como decimación o expansión (muestreo en tiempo continuo o sobremuestreo). El trabajo consiste en aplicar ambas operaciones, por lo tanto se trabajó primeramente en la decimación de la señal.

III-D. *Decimación*

La operación de decimación consiste en disminuir la frecuencia de muestreo original de la señal por un factor M, esto causa que se expanda en frecuencia y se comprima en el tiempo, por lo tanto se busca implementar en código esta operación.

Ya que esta operación, modifica el espectro, puede producir aliasing, por lo tanto, es necesario realizar un filtrado y después la decimación.

Entonces, a la señal original se realiza su transformada en frecuencia, esa función será multiplicada por un fitro anti-aliasing, o un H_{LPF} para luego ser procesada la señal por la operación de decimación.

Para realizar un filtrado adecuado, el filtro pasabajos deberá tener una frecuencia de corte $\omega_c = \omega_s/2 * M$, introducido en la sección del marco teórico. Es decir, la frecuencia de corte del filtro es la frecuencia máxima de la señal (fs/2) dividido para un factor de decimación M.

Puesto en ecuación el filtro es:

$$H_{LPF} = \begin{Bmatrix} 1 & |\omega| <= \omega_c/M \\ 0 & o.w \end{Bmatrix} \quad (1)$$

Entonces, a la señal en frecuencia se realiza un filtrado anti-aliasing y obtener una señal adecuada para la decimación.

Para realizar la decimación, es mas sencillo programar la función en el tiempo, por lo tanto la señal filtrada pasa a tiempo nuevamente. Además, esta operación disminuye la frecuencia de muestreo (fs/M) y la cantidad de muestras para el mismo factor.

Como se indicó en la teoría, el diezmado en tiempo consiste en obtener un valor en la posición $x(n * M)$ de una señal y ubicarlo en una señal nueva en la posición $x_b[n]$. Para conseguir esto, se estableció un bucle *for* que itere desde n con posición 1 hasta la cantidad de muestras diezmadas, y los valores obtenidos de x[nM] se ubiquen en otro vector en la posición $x_{decimate}[n]$ y obteniendo una señal relacionada con la original.

Además, se realiza una transformada en frecuencia a la señal diezmada, para observar como el espectro se ensanchó.

En la Figura 6, se observa de manera gráfica todo el proceso de diezmado, en la superior se ubica la señal original filtrada por el filtro H_{LPF} anti-aliasing, por debajo se muestran las señales en tiempo continuo y luego en tiempo discreto.

En las gráficas en tiempo de la señal diezmada, para poder recuperarlas se debe tener en cuenta el nuevo número de muestras y la nueva tasa de frecuencia menor. Se observa que se tiene una menor cantidad de muestras en la señal discreta y la misma duración en continuo, esto se debe a que ahora se manipulan la señales con los nuevos parámetros de frecuencia y muestras para poder reconocer correctamente la señal de audio.

Finalmente, en el espectro en frecuencia de la señal diezmada, se observa un cambio expandiendo el espectro a mayor frecuencia, por lo que se entiende que esta operación fue realizada exitosamente

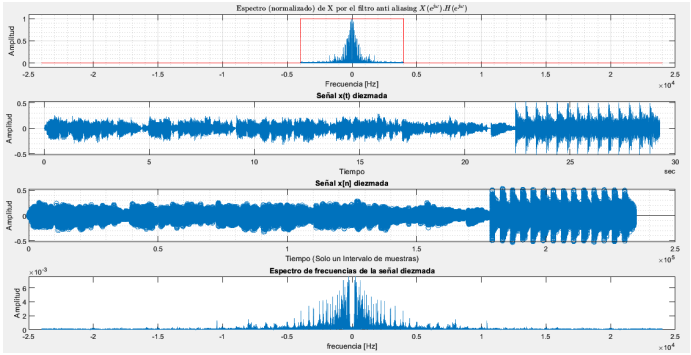

Figura 6: Gráfica de la señal Diezmada: filtrado anti-aliasing, gráficas en tiempo y el espectro de x[n] diezmada

III-E. *Expansión o Interpolación*

Ahora, nos centramos en realizar la operación opuesta a la decimación, la expansión. Esta operación busca expandir la señal en el dominio del tiempo y comprimir en frecuencia. Para ello se tiene un factor (L)de expansión que modifica los parámetros de la señal para que la frecuencia de sampleo sea (fs*L) y el número de muestras también se multiplique por un factor L.

Lo que busca la operación es introducir ceros a la señal original entre los valores de la misma y obtener una señal mas ancha, es decir: $x_{exp} = x[n/L]$ Si y solo si n/L es entero, y por lo tanto, los valores no múltiplos de esta división toman valor de 0.

Para implementar en el código, se guía de forma similar a la decimación, en este caso se usa un bucle *for* que va con la posición *n* desde 1 hasta el nuevo valor del numero de muestra. Evalúa con un if, si el valor es entero, entonces se almacena el valor en x[n/L] a un nuevo vector $x_exp[n]$ y si no es entero, cambia a valer 0.

Al realizar esta operación de interpolación, en frecuencia se pueden llegar a generar réplicas en el espectro, y eso es algo no deseado si se desea recuperar correctamente la señal de audio.

Por lo tanto, se procede a transformar en frecuencia la señal expandida y multiplicarla por un filtro de interpolación o H_{LPF} y con una frecuencia de corte adecuada para no perder datos importantes, los valores del filtro serán:

$$H_{LPF} = \begin{Bmatrix} 1 & |\omega| <= \omega_c/M \\ 0 & o.w \end{Bmatrix} \quad (2)$$

Aquí se observa que la frecuencia de corte sería la (fs/2) o frecuencia máxima de la señal / L (similar como en la decimación). Una vez filtrada la señal, se procede a obtener la señal en el tiempo y observar como cambió con respecto a la original.

En la Figura 7, se observa en la gráfica superior, el espectro de frecuencias comprimido y además replicado por efecto de la operación, luego en la siguiente gráfica se aplica un filtro para obtener el espectro interpolado original y finalmente obtener las funciones en el dominio del tiempo, se resalta que en tiempo discreto se nota como la longitud de las muestras aumento a un factor de $\#muestras * L$.

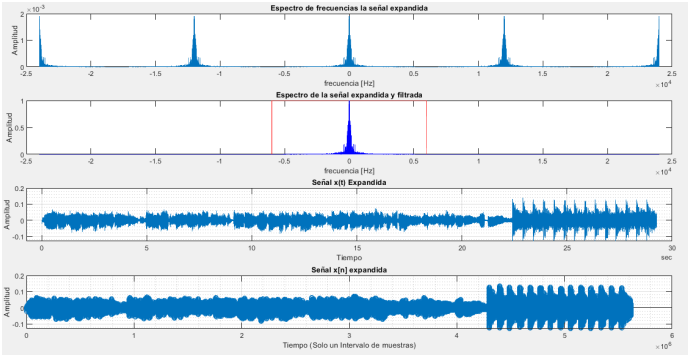

Figura 7: Gráfica de la señal Interpolada: señal interpolada y sus replicas y gráfica del filtrado en frecuencia, la señal expandida en el tiempo

Cabe resaltar que cada operación de FFT, se utiliza la frecuencia de muestre adecuada, original o por un factor de diezmado o interpolado, además que están normalizadas por las muestras y finalmente cuando se muestra un filtro con la señal filtrada, esta está normalizada también.

En esta parte se ha desarrollado el proceso de conversión de frecuencia de muestreo, se resalta que para el diezmado: la señal de entrada es en tiempo discreto, pasa a frecuencia para filtrarse (anti-aliasing), regresa al dominio del tiempo y realiza la operación, y finalmente pasa a frecuencia.

En el proceso de expansión: la señal en tiempo realiza la operación de interpolación, luego pasa a frecuencia para eliminar replicar mediante un filtro LPF.

Para el siguiente paso del procesado de la señal de audio que es la ecualización, el usuario si la señal que pasa es la diezmada o expandida, además, estas deben estar en el dominio de la frecuencia, para facilitar la ecualización.

III-F. Ecualización

Previo a la aplicación de los filtros definidos, se inicia definiendo las ganancias para cada banda de frecuencia, las cuales son ajustables por el usuario a través de la interfaz de la aplicación. Estas ganancias se utilizan posteriormente para modular la amplitud de las diferentes bandas de frecuencia en el proceso de ecualización.

A continuación, se establecen condiciones booleanas para definir filtros de banda específicos (Sub-Bass, Bass, Low Mids, High Mids, Presence, Brillance). Estos filtros se aplican a la señal expandida o decimada, permitiendo así modular selectivamente las componentes espectrales de interés. Cada banda se multiplica por su ganancia correspondiente, resultando en las señales ecualizadas individuales para cada banda.

Finalmente, se combinan las señales ecualizadas de todas las bandas para obtener la señal ecualizada total. Esta señal es sometida a una transformada inversa de Fourier (IFFT) para recuperar la forma de onda en el dominio del tiempo. En resumen, este proceso de ecualización ajusta selectivamente diferentes bandas de frecuencia en una señal de audio, permitiendo así la personalización de la respuesta espectral según las preferencias del usuario.

IV. Resultados

IV-A. Gráficas del proceso de Ecualización

Se presenta el resultado final del ecualizador y conversor de frecuencia de muestreo en la Figura 8.

V. Conclusiones

Las operaciones de decimación y expansión tienen un impacto significativo en el muestreo y la respuesta espectral de las señales de audio.

La decimación, al disminuir la frecuencia de muestreo de la señal original, tiene el efecto de expandir la señal en el dominio de la frecuencia y comprimirla en el tiempo. Sin embargo, esta operación puede introducir el fenómeno de aliasing, que se contrarresta mediante la aplicación de un filtro anti-aliasing antes de la decimación. La reducción en la frecuencia de muestreo implica una disminución en la cantidad de muestras y un cambio en la representación temporal de la señal.

Por otro lado, la expansión o interpolación busca aumentar la frecuencia de muestreo, lo que tiene el efecto de comprimir la señal en el dominio del tiempo y expandirla en la frecuencia. Para evitar la generación de réplicas no deseadas en el espectro, se aplica un filtro de interpolación antes de la expansión. Esta operación aumenta la cantidad de muestras y modifica la duración temporal de la señal.

En cuanto al proceso de ecualización, las señales decimadas o expandidas se someten a una modificación selectiva de diferentes bandas de frecuencia.

En resumen, las acciones de decimación y expansión no solo afectan la representación temporal y frecuencial de la señal, sino que también influyen en la respuesta espectral, lo cual tiene implicaciones directas en la posterior ecualización.

Referencias

[1] A. V. Oppenheim and R. W. Schafer, *Discrete-time signal processing*, 3rd ed. Pearson Education, 2010.

[2] A. A. Nieto, *Realización Eficiente de Conversión de Frecuencia de Muestreo por un Factor Racional Mediante el Uso de Filtros CIC*. Instituto Nacional de Astrofísica Óptica y Electrónica, 2012.

[3] R. N. Bracewell, *The Fourier Transform and Its Applications*, 3rd ed. McGraw-Hill, 2000.

[4] J. B. Allen, *The Art and Science of Acoustic Design*. ..., 1997.

[5] P. Stoica and R. L. Moses, *Spectral Analysis of Signals*. Pearson Education, 2005.

[6] B. V. V. Youtube, "The fast fourier transform algorithm," https://www.youtube.com/watch?v=EsJGuI7e_ZQ&t=855s, 2012, accessed: January,4.

[7] A. M. Quiroz, *Tutorial de tópicos avanzados de procesamiento digital de señales con Matlab/Simulink*. Universidad de las Américas Puebla, 2006.

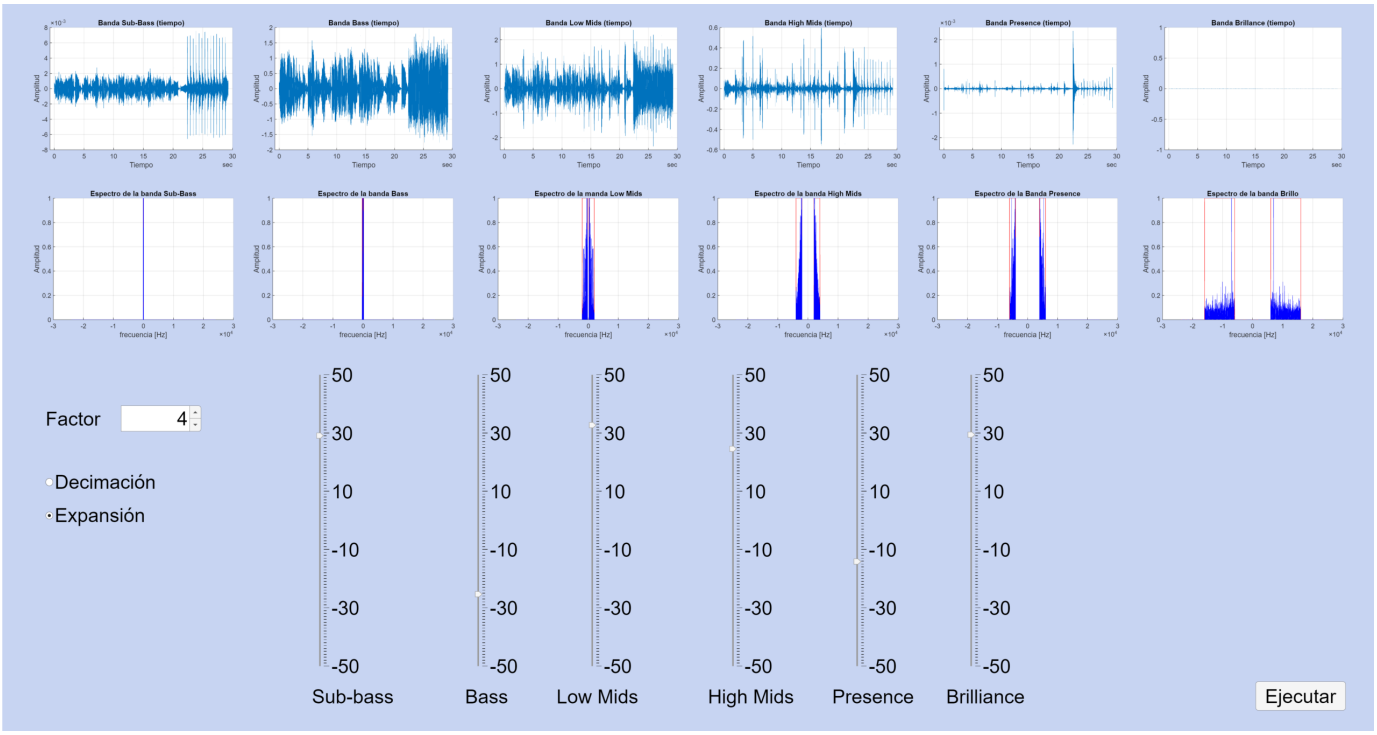

Figura 8: Salida del ecualizador para la expansión por un factor de 4 y diferentes ganancias de las bandas.

VI. Anexos

VI-A. Github

La dirección https://github.com/thyron001/sample_rate_converter_and_equalizer corresponde al repositorio que contiene los ficheros utilizados en este reporte como el script y el audio de prueba.

VI-B. Código

```
classdef ecualizador < matlab.apps.AppBase

    % Properties that correspond to app
        components
    properties (Access = public)
        UIFigure                matlab.ui.Figure
        BrillianceLabel         matlab.ui.control.
            Label
        brilliance              matlab.ui.control.
            Slider
        PresenceLabel           matlab.ui.control.
            Label
        presence                matlab.ui.control.
            Slider
        HighMidsLabel           matlab.ui.control.
            Label
        highmids                matlab.ui.control.
            Slider
        LowMidsLabel            matlab.ui.control.
            Label
        lowmids                 matlab.ui.control.
            Slider
        BassLabel               matlab.ui.control.
            Label
        bass                    matlab.ui.control.
            Slider
        SubbassLabel            matlab.ui.control.
            Label
        subbass                 matlab.ui.control.
            Slider
        ButtonGroup             matlab.ui.container
            .ButtonGroup
        expansion               matlab.ui.control.
            RadioButton
        decimacion              matlab.ui.control.
            RadioButton
        FactorLabel             matlab.ui.control.
            Label
        factor                  matlab.ui.control.
            Spinner
        EjecutarButton          matlab.ui.control.
            Button
        brilliancefrecuencia    matlab.ui.control.
            UIAxes
        presencefrecuencia      matlab.ui.control.
            UIAxes
        highmidsfrecuencia      matlab.ui.control.
            UIAxes
        lowmidsfrecuencia       matlab.ui.control.
            UIAxes
        bassfrecuencia          matlab.ui.control.
            UIAxes
        subbassfrecuencia       matlab.ui.control.
            UIAxes
        brilliancetiempo        matlab.ui.control.
            UIAxes
        presencetiempo          matlab.ui.control.
            UIAxes
        highmidstiempo          matlab.ui.control.
            UIAxes
        lowmidstiempo           matlab.ui.control.
            UIAxes
        basstiempo              matlab.ui.control.
            UIAxes
        subbasstiempo           matlab.ui.control.
            UIAxes
    end

    % Callbacks that handle component events
```

```
    methods (Access = private)

        % Button pushed function: EjecutarButton
        function EjecutarButtonPushed(app, event)
            clc;
    close all;
N = app.factor.Value;
%% Señal Original x[n]
name_a="counting_stars_cut.wav";
%Informacion del audio original

info_x = audioinfo(name_a)


[audio, fs] = audioread(name_a);

%%
audio = 0.5*(audio(:, 1) + audio(:, 2)).';  %%
    Audio estereo a mono

duracion_audio = length(audio) / fs;   %%duración

n_samples = length(audio);  %%num_muestras


%%
vec_xt = 0:seconds(1/fs): seconds(duracion_audio)
    ;
vec_xt = vec_xt(1:end-1);

figure(1)
subplot(3,1,1);
plot(vec_xt,transpose(audio));
title("Audio x(t)");
xlabel("Tiempo");
ylabel("Amplitud");
grid on;
grid minor;

figure(1)
subplot(3,1,2);
vec_nx = 1:n_samples;
stem(vec_nx, audio);
title("Audio x[n]");
xlabel("Tiempo (Solo un Intervalo de muestras)");
%xlim([500000 500100])
ylabel("Amplitud");
grid on;
grid minor;


%% Transformada de Fourier del audio original
X = fft(audio, n_samples)/n_samples;
Xi = fftshift(X);

frec_X = linspace(-fs/2, fs/2, length(X));

figure(1);
subplot(3,1,3);
plot(frec_X, 2*abs(Xi));
title("Espectro de frecuencias de x[n]: $X(e^{j\
    omega})$", 'Interpreter', 'latex');
xlabel("frecuencia [Hz]");
ylabel("Amplitud");

grid on;
grid minor;

A = app.bass.Value;
B = app.subbass.Value;

%% Decimación

M = N;
```

```
% Filtro Anti-aliasing
Hdec_LPF = abs(frec_X)<= (fs/(2*M));

% Filtrado para decimación
X_filtered = (Xi .* Hdec_LPF);

x_filt = ifft(X_filtered*n_samples);

%x_filt = real(x_filt);

% Operacion de decimación
f_dec = fs / M;
muestras_dec = n_samples / M;

y_dec = zeros;
for n = 1:muestras_dec
    y_dec(n) = (x_filt(n*M));
end

%%%FFT de la señal diezmada
Y_dec = fft(y_dec)/round(muestras_dec);
Yi_dec = fftshift(Y_dec);

frecY_dec = linspace(-fs/2, fs/2, muestras_dec);


%%% Graficas de decimación

figure(2);
subplot(4,1,1);
plot(frec_X, abs(X_filtered)/max(abs(X_filtered))
    );
hold on;
plot(frec_X, Hdec_LPF,'r');
title("Espectro (normalizado) de X por el filtro
    anti aliasing $X(e^{j\omega}) . H(e^{j\omega
    })$", 'Interpreter', 'latex');
xlabel("Frecuencia [Hz]");
ylabel("Amplitud");
ylim([-0.1 1.1])
grid on; grid minor;
ax = gca;

tdec = 0:seconds(1/f_dec): seconds(duracion_audio
    );
tdec = tdec(1:end-1);
figure(2)
subplot(4,1,2);
plot(tdec, real(y_dec*(-1)));
title("Señal x(t) diezmada");
xlabel("Tiempo");
ylabel("Amplitud");
grid on;
grid minor;

figure(2)
subplot(4,1,3);
nx_dec = 1:muestras_dec;
stem(nx_dec, real(y_dec*(-1)));
title("Señal x[n] diezmada");
xlabel("Tiempo (Solo un Intervalo de muestras)");
%xlim([500000 500100]);
ylabel("Amplitud");
grid on;
grid minor;

figure(2);
subplot(4,1,4);
plot(frecY_dec, 2*abs(Yi_dec));
title("Espectro de frecuencias de la señal
    diezmada");
xlabel("frecuencia [Hz]");
ylabel("Amplitud");
```

```
%%%%%%

%% Expansión / Interpolación

L = N;  %%factor de interpolación

f_interp = fs*L;
muestras_interp = n_samples*L;

%%% Operación de interpolación

y_expand = zeros;

for n = 1:muestras_interp
    if mod(n,L) == 0
        y_expand(n) = audio(n/L);
    else
        y_expand(n) = 0;
    end
end



%%% FFT de la señal expandida y filtrado
Y_expand = fft(y_expand, muestras_interp)/
    muestras_interp;

Yi_expand = fftshift(Y_expand);

frecY_expand = linspace(-fs/2, fs/2,
    muestras_interp);

%%% Filtro de interpolación
Hexp_LPF = abs(frecY_expand) <= (fs/L)*(0.5);

Yexp_filt = Yi_expand .* Hexp_LPF;



figure(3);
subplot(4,1,1);
plot(frecY_expand, 2*abs(Yi_expand));
title("Espectro de frecuencias la señal expandida
    ");
xlabel("frecuencia [Hz]");
ylabel("Amplitud");

figure(3);
subplot(4,1,2);
plot(frecY_expand, Hexp_LPF, 'r');
hold on;
plot(frecY_expand, abs(Yexp_filt)/max(abs(
    Yexp_filt)), 'b');
title("Espectro de la señal expandida y filtrada
    ");
xlabel("frecuencia [Hz]");
ylabel("Amplitud");


%% extrapolacion en el tiempo;
yexp_filtered = ifft(fftshift(Yexp_filt*
    muestras_interp));

texp = 0:seconds(1/f_interp): seconds(
    duracion_audio);
texp = texp(1:length(yexp_filtered));

figure(3)
subplot(4,1,3);
plot(texp, real(yexp_filtered));
title("Señal x(t) Expandida");
xlabel("Tiempo");
ylabel("Amplitud");
grid on;
grid minor;

figure(3)
subplot(4,1,4);
ny_exp = 1:muestras_interp;
stem(ny_exp, real(yexp_filtered));
title("Señal x[n] expandida");
xlabel("Tiempo (Solo un Intervalo de muestras)");
%xlim([500000 500100]);
ylabel("Amplitud");
grid on;
grid minor;


if app.decimacion.Value == 1
    %%Salida de las FFT de la señales expandidas
        o decimadas

    Y_in = Yi_dec;      %%decimación

    %%Frecuencias de sampleo de la decimación o
        interpolacion

    f_in = f_dec;       %%decimación

    %%Rango de frecuencias de la decimación o
        interpoladas

    frec_Y = frecY_dec;      %%decimación
    muestras_Y = muestras_dec;

end
%% Se escoge que datos salen decimacion o
    expansion para ecualizar

if app.expansion.Value == 1
    %%Salida de las FFT de la señales expandidas
        o decimadas
    Y_in = Yexp_filt;

    %%Frecuencias de sampleo de la decimación o
        interpolacion
    f_in = f_interp;

    %%Rango de frecuencias de la decimación o
        interpoladas
    frec_Y = frecY_expand;
    muestras_Y = muestras_interp;

end

%% Ecualizadores
%% Ganancias  que salen de la app
G1 =10^(app.bass.Value/20);  %% G = 10^(valor de
    la app/20);
G2 =10^(app.subbass.Value/20);
G3 =10^(app.lowmids.Value/20);
G4 =10^(app.highmids.Value/20);
G5 =10^(app.presence.Value/20);
G6 =10^(app.brilliance.Value/20);

%%Sub-Bass
Sub = ((abs(frec_Y)>=16).*(abs(frec_Y)<=60));
Y_sub = G1*(Y_in.* Sub);
%%Bass
Bass = ((abs(frec_Y)>=60).*(abs(frec_Y)<=250));
Y_bass = G2*(Y_in.* Bass);
%%Low Mid
L_Mid = ((abs(frec_Y)>=250).*(abs(frec_Y)<=2e3));
Y_Lmid = G3*(Y_in.* L_Mid);
%%High Mid
H_Mid = ((abs(frec_Y)>=2e3).*(abs(frec_Y)<=4e3));
Y_Hmid = G4*(Y_in.* H_Mid);
%%Presence
```

```
Pres = ((abs(frec_Y)>=4e3).*(abs(frec_Y)<=6e3));
Y_pres = G5*(Y_in.* Pres);
%Brrilliance
Brillo = ((abs(frec_Y)>=6e3).*(abs(frec_Y)<=16e3)
    );
Y_brillo = G6*(Y_in.* Brillo);

Z_EQ = Y_sub + Y_bass + Y_Lmid + Y_Hmid + Y_pres
    + Y_brillo; %FFT total del equalizador


z_eq_t = ifft(fftshift(Z_EQ*muestras_Y));  %%
    salida del ecualizador en el tiempo

t_eq = 0:seconds(1/f_in): seconds(duracion_audio)
    ;
t_eq = t_eq(1:length(z_eq_t));

%% Señal en el tiempo de cada banda
z_sub = ifft(fftshift(Y_sub*muestras_Y));
z_bass = ifft(fftshift(Y_bass*muestras_Y));
z_lmid = ifft(fftshift(Y_Lmid*muestras_Y));
z_hmid = ifft(fftshift(Y_Hmid*muestras_Y));
z_pres = ifft(fftshift(Y_pres*muestras_Y));
z_brillo = ifft(fftshift(Y_brillo*muestras_Y));

%%%graficas

plot(app.subbasstiempo,t_eq, real(z_sub*(-1)));
title(app.subbasstiempo, "Banda Sub-Bass (tiempo)
    ");
xlabel(app.subbasstiempo, "Tiempo");
ylabel(app.subbasstiempo, "Amplitud");
grid(app.subbasstiempo, 'on');
grid(app.subbasstiempo, 'minor');

plot(app.basstiempo,t_eq, real(z_bass*(-1)));
title(app.basstiempo, "Banda Bass (tiempo)");
xlabel(app.basstiempo, "Tiempo");
ylabel(app.basstiempo, "Amplitud");
grid(app.basstiempo, 'on');
grid(app.basstiempo, 'minor');

plot(app.lowmidstiempo,t_eq, real(z_lmid*(-1)));
title(app.lowmidstiempo, "Banda Low Mids (tiempo)
    ");
xlabel(app.lowmidstiempo, "Tiempo");
ylabel(app.lowmidstiempo, "Amplitud");
grid(app.lowmidstiempo, 'on');
grid(app.lowmidstiempo, 'minor');

plot(app.highmidstiempo,t_eq, real(z_hmid*(-1)));
title(app.highmidstiempo, "Banda High Mids (
    tiempo)");
xlabel(app.highmidstiempo, "Tiempo");
ylabel(app.highmidstiempo, "Amplitud");
grid(app.highmidstiempo, 'on');
grid(app.highmidstiempo, 'minor');


plot(app.presencetiempo,t_eq, real(z_pres*(-1)));
title(app.presencetiempo, "Banda Presence (tiempo
    )");
xlabel(app.presencetiempo, "Tiempo");
ylabel(app.presencetiempo, "Amplitud");
grid(app.presencetiempo, 'on');
grid(app.presencetiempo, 'minor');

plot(app.brilliancetiempo,t_eq, real(z_brillo
    *(-1)));
title(app.brilliancetiempo, "Banda Brillance (
    tiempo)");
xlabel(app.brilliancetiempo, "Tiempo");
ylabel(app.brilliancetiempo, "Amplitud");
grid(app.brilliancetiempo, 'on');
grid(app.brilliancetiempo, 'minor');



%% Espectros de cada una de las bandas del
    ecualizador (Valores Normalizados)

plot(app.subbassfrecuencia,frec_Y, Sub, 'r');
hold(app.subbassfrecuencia, 'on');
plot(app.subbassfrecuencia,frec_Y, abs(Y_sub)/max
    (abs(Y_sub)), 'b');
title(app.subbassfrecuencia,"Espectro de la banda
     Sub-Bass");
xlabel(app.subbassfrecuencia, "frecuencia [Hz]");
ylabel(app.subbassfrecuencia, "Amplitud");
grid(app.subbassfrecuencia, 'on');
grid(app.subbassfrecuencia, 'minor');

plot(app.bassfrecuencia,frec_Y, Bass, 'r');
hold(app.bassfrecuencia, 'on');
plot(app.bassfrecuencia,frec_Y, abs(Y_bass)/max(
    abs(Y_bass)), 'b');
title(app.bassfrecuencia,"Espectro de la banda
    Bass");
xlabel(app.bassfrecuencia,"frecuencia [Hz]");
ylabel(app.bassfrecuencia,"Amplitud");
grid(app.bassfrecuencia, 'on');
grid(app.bassfrecuencia, 'minor');


plot(app.lowmidsfrecuencia,frec_Y, L_Mid, 'r');
hold(app.lowmidsfrecuencia, 'on');
plot(app.lowmidsfrecuencia,frec_Y, abs(Y_Lmid)/
    max(abs(Y_Lmid)), 'b');
title(app.lowmidsfrecuencia,"Espectro de la manda
     Low Mids");
xlabel(app.lowmidsfrecuencia,"frecuencia [Hz]");
ylabel(app.lowmidsfrecuencia,"Amplitud");
grid(app.lowmidsfrecuencia, 'on');
grid(app.lowmidsfrecuencia, 'minor');


plot(app.highmidsfrecuencia,frec_Y, H_Mid, 'r');
hold(app.highmidsfrecuencia, 'on');
plot(app.highmidsfrecuencia,frec_Y, abs(Y_Hmid)/
    max(abs(Y_Hmid)), 'b');
title(app.highmidsfrecuencia,"Espectro de la
    banda High Mids");
xlabel(app.highmidsfrecuencia,"frecuencia [Hz]");
ylabel(app.highmidsfrecuencia,"Amplitud");
grid(app.highmidsfrecuencia, 'on');
grid(app.highmidsfrecuencia, 'minor');


plot(app.presencefrecuencia, frec_Y, Pres, 'r');
hold(app.presencefrecuencia, 'on');
plot(app.presencefrecuencia,frec_Y, abs(Y_pres)/
    max(abs(Y_pres)), 'b');
title(app.presencefrecuencia,"Espectro de la
    Banda Presence");
xlabel(app.presencefrecuencia,"frecuencia [Hz]");
ylabel(app.presencefrecuencia,"Amplitud");
grid(app.presencefrecuencia, 'on');
grid(app.presencefrecuencia, 'minor');


plot(app.brilliancefrecuencia, frec_Y, Brillo, 'r
    ');
hold(app.brilliancefrecuencia, 'on');
plot(app.brilliancefrecuencia,frec_Y, abs(
    Y_brillo)/max(abs(Y_brillo)), 'b');
title(app.brilliancefrecuencia,"Espectro de la
    banda Brillo");
```

```
xlabel(app.brilliancefrecuencia,"frecuencia [Hz
    ]");
ylabel(app.brilliancefrecuencia,"Amplitud");
grid(app.brilliancefrecuencia, 'on');
grid(app.brilliancefrecuencia, 'minor');




 figure(4);
 subplot(2,1,1);
 plot(frec_Y, 2*abs(Z_EQ) , 'b');
 title("Espectro de la señal ecualizada");
 xlabel("frecuencia [Hz]");
 ylabel("Amplitud");

 figure(4)
 subplot(2,1,2);
 plot(t_eq, real(z_eq_t*(-1)));
 title("Señal x(t) Ecualizada");
 xlabel("Tiempo");
 ylabel("Amplitud");
 grid on;
 grid minor;
sound(real(z_eq_t),f_in); % Audio filtrado (LPF)
pause( duracion_audio + 1 );
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all
                components are created
            app.UIFigure = uifigure('Visible', '
                off');
            app.UIFigure.Color = [0.7804 0.8314
                0.949];
            app.UIFigure.Position = [100 100 2179
                 947];
            app.UIFigure.Name = 'MATLAB App';

            % Create subbasstiempo
            app.subbasstiempo = uiaxes(app.
                UIFigure);
            title(app.subbasstiempo, 'Title')
            xlabel(app.subbasstiempo, 'X')
            ylabel(app.subbasstiempo, 'Y')
            zlabel(app.subbasstiempo, 'Z')
            app.subbasstiempo.Position = [57 741
                316 174];

            % Create basstiempo
            app.basstiempo = uiaxes(app.UIFigure)
                ;
            title(app.basstiempo, 'Title')
            xlabel(app.basstiempo, 'X')
            ylabel(app.basstiempo, 'Y')
            zlabel(app.basstiempo, 'Z')
            app.basstiempo.Position = [402 741
                316 174];

            % Create lowmidstiempo
            app.lowmidstiempo = uiaxes(app.
                UIFigure);
            title(app.lowmidstiempo, 'Title')
            xlabel(app.lowmidstiempo, 'X')
            ylabel(app.lowmidstiempo, 'Y')
            zlabel(app.lowmidstiempo, 'Z')
            app.lowmidstiempo.Position = [757 741
                 316 174];

            % Create highmidstiempo
            app.highmidstiempo = uiaxes(app.
                UIFigure);
            title(app.highmidstiempo, 'Title')
            xlabel(app.highmidstiempo, 'X')
            ylabel(app.highmidstiempo, 'Y')
            zlabel(app.highmidstiempo, 'Z')
            app.highmidstiempo.Position = [1102
                741 316 174];

            % Create presencetiempo
            app.presencetiempo = uiaxes(app.
                UIFigure);
            title(app.presencetiempo, 'Title')
            xlabel(app.presencetiempo, 'X')
            ylabel(app.presencetiempo, 'Y')
            zlabel(app.presencetiempo, 'Z')
            app.presencetiempo.Position = [1447
                741 316 174];

            % Create brilliancetiempo
            app.brilliancetiempo = uiaxes(app.
                UIFigure);
            title(app.brilliancetiempo, 'Title')
            xlabel(app.brilliancetiempo, 'X')
            ylabel(app.brilliancetiempo, 'Y')
            zlabel(app.brilliancetiempo, 'Z')
            app.brilliancetiempo.Position = [1802
                 741 316 174];

            % Create subbassfrecuencia
            app.subbassfrecuencia = uiaxes(app.
                UIFigure);
            title(app.subbassfrecuencia, 'Title')
            xlabel(app.subbassfrecuencia, 'X')
            ylabel(app.subbassfrecuencia, 'Y')
            zlabel(app.subbassfrecuencia, 'Z')
            app.subbassfrecuencia.Position = [57
                532 316 174];

            % Create bassfrecuencia
            app.bassfrecuencia = uiaxes(app.
                UIFigure);
            title(app.bassfrecuencia, 'Title')
            xlabel(app.bassfrecuencia, 'X')
            ylabel(app.bassfrecuencia, 'Y')
            zlabel(app.bassfrecuencia, 'Z')
            app.bassfrecuencia.Position = [402
                532 316 174];

            % Create lowmidsfrecuencia
            app.lowmidsfrecuencia = uiaxes(app.
                UIFigure);
            title(app.lowmidsfrecuencia, 'Title')
            xlabel(app.lowmidsfrecuencia, 'X')
            ylabel(app.lowmidsfrecuencia, 'Y')
            zlabel(app.lowmidsfrecuencia, 'Z')
            app.lowmidsfrecuencia.Position = [757
                 532 316 174];

            % Create highmidsfrecuencia
            app.highmidsfrecuencia = uiaxes(app.
                UIFigure);
            title(app.highmidsfrecuencia, 'Title'
                )
            xlabel(app.highmidsfrecuencia, 'X')
            ylabel(app.highmidsfrecuencia, 'Y')
            zlabel(app.highmidsfrecuencia, 'Z')
            app.highmidsfrecuencia.Position =
                [1102 532 316 174];
```

```
            % Create presencefrecuencia
            app.presencefrecuencia = uiaxes(app.
                UIFigure);
            title(app.presencefrecuencia, 'Title'
                )
            xlabel(app.presencefrecuencia, 'X')
            ylabel(app.presencefrecuencia, 'Y')
            zlabel(app.presencefrecuencia, 'Z')
            app.presencefrecuencia.Position =
                [1447 532 316 174];

            % Create brilliancefrecuencia
            app.brilliancefrecuencia = uiaxes(app
                .UIFigure);
            title(app.brilliancefrecuencia, '
                Title')
            xlabel(app.brilliancefrecuencia, 'X')
            ylabel(app.brilliancefrecuencia, 'Y')
            zlabel(app.brilliancefrecuencia, 'Z')
            app.brilliancefrecuencia.Position =
                [1802 532 316 174];

            % Create EjecutarButton
            app.EjecutarButton = uibutton(app.
                UIFigure, 'push');
            app.EjecutarButton.ButtonPushedFcn =
                createCallbackFcn(app,
                @EjecutarButtonPushed, true);
            app.EjecutarButton.FontSize = 36;
            app.EjecutarButton.Position = [1951
                51 167 54];
            app.EjecutarButton.Text = 'Ejecutar';

            % Create factor
            app.factor = uispinner(app.UIFigure);
            app.factor.Step = 2;
            app.factor.Limits = [2 10];
            app.factor.FontSize = 36;
            app.factor.Position = [222 362 88
                48];
            app.factor.Value = 2;

            % Create FactorLabel
            app.FactorLabel = uilabel(app.
                UIFigure);
            app.FactorLabel.FontSize = 36;
            app.FactorLabel.Position = [82 363
                117 47];
            app.FactorLabel.Text = 'Factor ';

            % Create ButtonGroup
            app.ButtonGroup = uibuttongroup(app.
                UIFigure);
            app.ButtonGroup.AutoResizeChildren =
                'off';
            app.ButtonGroup.ForegroundColor =
                [0.7804 0.8314 0.949];
            app.ButtonGroup.BorderType = 'none';
            app.ButtonGroup.BackgroundColor =
                [0.7804 0.8314 0.949];
            app.ButtonGroup.FontSize = 36;
            app.ButtonGroup.Position = [67 170
                296 136];

            % Create decimacion
            app.decimacion = uiradiobutton(app.
                ButtonGroup);
            app.decimacion.Text = 'Decimación';
            app.decimacion.FontSize = 36;
            app.decimacion.Position = [16 77 211
                44];
            app.decimacion.Value = true;

            % Create expansion
            app.expansion = uiradiobutton(app.
                ButtonGroup);
            app.expansion.Text = 'Expansión';
            app.expansion.FontSize = 36;
            app.expansion.Position = [16 15 191
                44];

            % Create subbass
            app.subbass = uislider(app.UIFigure);
            app.subbass.Limits = [-50 50];
            app.subbass.Orientation = 'vertical';
            app.subbass.FontSize = 36;
            app.subbass.Position = [527 133 3
                334];

            % Create SubbassLabel
            app.SubbassLabel = uilabel(app.
                UIFigure);
            app.SubbassLabel.FontSize = 36;
            app.SubbassLabel.Position = [482 54
                157 47];
            app.SubbassLabel.Text = 'Sub-bass';

            % Create bass
            app.bass = uislider(app.UIFigure);
            app.bass.Limits = [-50 50];
            app.bass.Orientation = 'vertical';
            app.bass.FontSize = 36;
            app.bass.Position = [758 133 3 334];

            % Create BassLabel
            app.BassLabel = uilabel(app.UIFigure)
                ;
            app.BassLabel.FontSize = 36;
            app.BassLabel.Position = [735 54 85
                47];
            app.BassLabel.Text = 'Bass';

            % Create lowmids
            app.lowmids = uislider(app.UIFigure);
            app.lowmids.Limits = [-50 50];
            app.lowmids.Orientation = 'vertical';
            app.lowmids.FontSize = 36;
            app.lowmids.Position = [970 133 3
                334];

            % Create LowMidsLabel
            app.LowMidsLabel = uilabel(app.
                UIFigure);
            app.LowMidsLabel.FontSize = 36;
            app.LowMidsLabel.Position = [905 54
                157 47];
            app.LowMidsLabel.Text = 'Low Mids';

            % Create highmids
            app.highmids = uislider(app.UIFigure)
                ;
            app.highmids.Limits = [-50 50];
            app.highmids.Orientation = 'vertical'
                ;
            app.highmids.FontSize = 36;
            app.highmids.Position = [1168 133 3
                334];

            % Create HighMidsLabel
            app.HighMidsLabel = uilabel(app.
                UIFigure);
            app.HighMidsLabel.FontSize = 36;
            app.HighMidsLabel.Position = [1123 54
                 165 47];
            app.HighMidsLabel.Text = 'High Mids';

            % Create presence
```

```
            app.presence = uislider(app.UIFigure)
                ;
            app.presence.Limits = [-50 50];
            app.presence.Orientation = 'vertical'
                ;
            app.presence.FontSize = 36;
            app.presence.Position = [1399 133 3
                334];

            % Create PresenceLabel
            app.PresenceLabel = uilabel(app.
                UIFigure);
            app.PresenceLabel.FontSize = 36;
            app.PresenceLabel.Position = [1354 54
                 157 47];
            app.PresenceLabel.Text = 'Presence';

            % Create brilliance
            app.brilliance = uislider(app.
                UIFigure);
            app.brilliance.Limits = [-50 50];
            app.brilliance.Orientation = '
                vertical';
            app.brilliance.FontSize = 36;
            app.brilliance.Position = [1611 133 3
                 334];

            % Create BrillianceLabel
            app.BrillianceLabel = uilabel(app.
                UIFigure);
            app.BrillianceLabel.FontSize = 36;
            app.BrillianceLabel.Position = [1566
                54 151 47];
            app.BrillianceLabel.Text = '
                Brilliance';

            % Show the figure after all
                components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = ecualizador

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```