**MEMORIA DE PROGRAMACIÓN**

****

**Índice**

1. Introducción………………………………………………………....2
2. Funciones para la gestión de los menús……………………………..2
3. Funciones globales………………………………………………..…8
4. Funciones asociadas a los componentes de la interfaz……………..14

**Introducción**

Para la realización del proyecto hemos desarrollado una serie de funciones. A continuación, se presenta una breve explicación de cada una de ellas junto con el código necesario para implementarlas. Además hemos incluido una multitud de comentarios aclaratorios en cada una de las líneas del código para hacer más fácil su entendimiento.

Este código sigue las enseñanzas de la escuela de Código Libre, por lo que estará disponible para cualquiera que desee solicitarlo en los correos electrónicos [*lauralesarri@alumnos.uva.es*](mailto:lauralesarri@alumnos.uva.es)y[*maria.herrero.tudela@alumnos.uva.es*](mailto:maria.herrero.tudela@alumnos.uva.es)

**Funciones para la gestión de los menús**

En primer lugar vamos a mencionar las funciones que corresponden a la gestión de los menús de la interfaz.

function BorraPantalla(app)

Esta función se encarga de borrar el menú principal. Se utiliza cada vez que cargamos un nuevo menú. A continuación se muestra una parte de la función por ser demasiado larga, pero teniendo en cuenta que esta idea se podría extrapolar a cualquier componente de la interfaz.

**function** BorraPantalla**(**app**)**

app**.**ExportarAudio**.**Visible**=**'off'**;**

app**.**CargarButton**.**Visible **=** 'off'**;**

app**.**ReproducirButton**.**Visible**=**'off'**;**

app**.**BotonCargar**.**Visible **=** 'off'**;**

app**.**LabelEnergia**.**Visible**=**'off'**;**

app**.**LabelPotencia**.**Visible**=**'off'**;**

app**.**LabelPeriodo**.**Visible**=**'off'**;**

app**.**FiltroPanel**.**Visible**=**'off'**;**

app**.**RuidoBlancoPanel**.**Visible**=**'off'**;**

….

**end**

function CargarMenuPrincipal(app)

Esta función se encarga de cargar los elementos que corresponden al menú principal como se muestra:

**function** CargarMenuPrincipal**(**app**)**

% Borramos las gráficas que no necesitamos

cla**(**app**.**EspectrogramaModificado**);**

cla**(**app**.**EspectrogramaOriginal**);**

% Redimensionamos las gráficas

% Create EspectrogramaModificado

app**.**EspectrogramaModificado**.**AmbientLightColor **=** **[**0.902 0.902 0.902**];**

app**.**EspectrogramaModificado**.**PlotBoxAspectRatio **=** **[**1.94444444444444 1 1**];**

app**.**EspectrogramaModificado**.**Visible **=** 'off'**;**

app**.**EspectrogramaModificado**.**BackgroundColor **=** **[**0.9412 0.9412 0.9412**];**

app**.**EspectrogramaModificado**.**Position **=** **[**496 8 365 72**];**

% Create EspectrogramaOriginal

app**.**EspectrogramaOriginal**.**Visible **=** 'off'**;**

app**.**EspectrogramaOriginal**.**Position **=** **[**53 8 366 73**];**

% Create GraficaTiempo

title**(**app**.**GraficaTiempo**,** 'Señal Cargada'**)**

xlabel**(**app**.**GraficaTiempo**,** 'Tiempo (s)'**)**

ylabel**(**app**.**GraficaTiempo**,** 'Amplitud'**)**

app**.**GraficaTiempo**.**PlotBoxAspectRatio **=** **[**1.8961038961039 1 1**];**

app**.**GraficaTiempo**.**Alphamap **=** **[**0 0.0159 0.0317 0.0476 0.0635 0.0794 0.0952 0.1111 0.127 0.1429 0.1587 0.1746 0.1905 0.2063 0.2222 0.2381 0.254 0.2698 0.2857 0.3016 0.3175 0.3333 0.3492 0.3651 0.381 0.39**];**

app**.**GraficaTiempo**.**BackgroundColor **=** **[**0.9412 0.9412 0.9412**];**

app**.**GraficaTiempo**.**Position **=** **[**13 413 454 288**];**

% Create GraficaFFToRuido

title**(**app**.**GraficaFFToRuido**,** 'FFT'**)**

xlabel**(**app**.**GraficaFFToRuido**,** 'Frecuencia (Hz)'**)**

ylabel**(**app**.**GraficaFFToRuido**,** 'Amplitud'**)**

app**.**GraficaFFToRuido**.**PlotBoxAspectRatio **=** **[**1.95625 1 1**];**

app**.**GraficaFFToRuido**.**ColorOrder **=** **[**0 0.451 0.7412**;**0.851 0.3255 0.098**;**0.9294 0.6941 0.1255**;**0.4941 0.1843 0.5569**;**0.4667 0.6745 0.1882**;**0.302 0.7451 0.9333**;**0.6392 0.0784 0.1804**];**

app**.**GraficaFFToRuido**.**BackgroundColor **=** **[**0.9412 0.9412 0.9412**];**

app**.**GraficaFFToRuido**.**HandleVisibility **=** 'off'**;**

app**.**GraficaFFToRuido**.**Position **=** **[**477 403 439 311**];**

% Create GraficaDEP

title**(**app**.**GraficaDEP**,** 'DEP'**)**

xlabel**(**app**.**GraficaDEP**,** 'Frecuencia (Hz)'**)**

ylabel**(**app**.**GraficaDEP**,** 'Amplitud'**)**

app**.**GraficaDEP**.**AmbientLightColor **=** 'none'**;**

app**.**GraficaDEP**.**PlotBoxAspectRatio **=** **[**1.87654320987654 1 1**];**

app**.**GraficaDEP**.**ColorOrder **=** **[**0 0.451 0.7412**;**0.851 0.3255 0.098**;**0.9294 0.6941 0.1255**;**0.4941 0.1843 0.5569**;**0.4667 0.6745 0.1882**;**0.302 0.7451 0.9333**;**0.6392 0.0784 0.1804**];**

app**.**GraficaDEP**.**BackgroundColor **=** **[**0.9412 0.9412 0.9412**];**

app**.**GraficaDEP**.**Position **=** **[**494 92 422 265**];**

% Create Espectograma

title**(**app**.**Espectograma**,** 'Espectrograma'**)**

xlabel**(**app**.**Espectograma**,** 'Tiempo (s)'**)**

ylabel**(**app**.**Espectograma**,** 'Frecuencia'**)**

app**.**Espectograma**.**AmbientLightColor **=** **[**0.902 0.902 0.902**];**

app**.**Espectograma**.**PlotBoxAspectRatio **=** **[**1.94444444444444 1 1**];**

app**.**Espectograma**.**BackgroundColor **=** **[**0.9412 0.9412 0.9412**];**

app**.**Espectograma**.**Position **=** **[**13 90 443 278**];**

app**.**BotonCargar**.**Visible **=** 'on'**;**

app**.**EspectrogramaModificado**.**Visible**=**'off'**;**

app**.**EspectrogramaOriginal**.**Visible**=**'off'**;**

app**.**ExportarAudio**.**Visible**=**'on'**;**

cla**(**app**.**GraficaDEP**);**

cla**(**app**.**GraficaFFToRuido**);**

cla**(**app**.**Espectograma**);**

title**(**app**.**GraficaTiempo**,** 'Señal Original'**)**

title**(**app**.**GraficaDEP**,** 'DEP'**)** %Ponemos un titulo a la gráfica que representará la DEP

title**(**app**.**GraficaFFToRuido**,** 'FFT'**)** %Ponemos un titulo a la gráfica que representará la FFT

title**(**app**.**Espectograma**,** 'Espectograma'**)** %Ponemos un titulo a la gráfica que representará el Espectograma

xlabel**(**app**.**Espectograma**,**'Tiempo (s)'**)**

app**.**GenerarSealPanel**.**Visible**=**'on'**;**

app**.**SealesPrecargadasPanel**.**Visible**=**'on'**;**

app**.**InformacinsenalCargadaPanel**.**Visible**=**'on'**;**

app**.**RestablecerMenuPrincipal**.**Visible**=**'on'**;**

app**.**GrabarSealPanel**.**Visible**=**'on'**;**

**end**

function CargarMenuFiltrado(app)

Se encarga de cargar los elementos que corresponden al menú de filtrado de señales y ruido.

**function** CargarMenuFiltrado**(**app**)**

Redimensionar**(**app**);**

app**.**ReproducirButton**.**Visible**=**'on'**;**

app**.**EspectrogramaModificado**.**Visible**=**'on'**;**

app**.**EspectrogramaOriginal**.**Visible**=**'on'**;**

cla**(**app**.**GraficaDEP**);**

title**(**app**.**GraficaDEP**,** 'Filtro y Señal filtrada'**)**

title**(**app**.**GraficaTiempo**,** 'Señal Original'**)**

title**(**app**.**Espectograma**,** 'FFT Señal Original'**)**

title**(**app**.**EspectrogramaModificado**,** 'Espectrograma Señal Modificada'**)**

app**.**FiltroPanel**.**Visible**=**'on'**;**

app**.**RuidoBlancoPanel**.**Visible**=**'on'**;**

xlabel**(**app**.**Espectograma**,** 'Frecuencia (Hz)'**)** %Cambiamos lasunidades al eje x

ylabel**(**app**.**Espectograma**,** 'Amplitud'**)** %Cambiamos las unidades al eje y

title**(**app**.**GraficaFFToRuido**,** 'Señal Modificada'**)**

xlabel**(**app**.**GraficaFFToRuido**,** 'Tiempo (s)'**)**

app**.**Espectograma**.**YDir **=** 'normal'**;** %Nos aseguramos de que los valores en el eje y se representen en orden creciente

xlim**(**app**.**Espectograma**,**'auto'**);**

ylim**(**app**.**Espectograma**,**'auto'**);**

app**.**CargarButton**.**Visible**=**'off'**;**

app**.**CargarFiltrar**.**Visible**=**'on'**;**

**end**

function CargarMenuModulacion(app)

Esta función se encarga de cargar los elementos que corresponden al menú de modulación y demodulación de audio.

**function** CargarMenuModulacion**(**app**)**

app**.**CargarButton**.**Visible**=**'on'**;**

title**(**app**.**Espectograma**,** 'FFT Señal Cargada'**)**

title**(**app**.**GraficaFFToRuido**,** 'Señal Modulada'**)**

title**(**app**.**GraficaTiempo**,** 'Señal Cargada'**)**

title**(**app**.**GraficaDEP**,** 'FFT Señal Modulada'**)**

xlabel**(**app**.**GraficaFFToRuido**,** 'Tiempo (s)'**)**

xlabel**(**app**.**Espectograma**,** 'Frecuencia (Hz)'**)**

xlabel**(**app**.**GraficaDEP**,** 'Frecuencia (Hz)'**)**

app**.**Espectograma**.**YDir **=** 'normal'**;**

app**.**EspectrogramaModificado**.**YDir **=** 'normal'**;**

app**.**EspectrogramaOriginal**.**YDir **=** 'normal'**;**

app**.**ModulacinAMPanel**.**Visible**=**'on'**;**

app**.**ModulacinFMPanel**.**Visible**=**'on'**;**

app**.**RestablecerAM**.**Visible**=**'on'**;**

xlim**(**app**.**Espectograma**,**'auto'**);**

ylim**(**app**.**Espectograma**,**'auto'**);**

cla**(**app**.**GraficaDEP**);**

cla**(**app**.**EspectrogramaModificado**);**

**end**

function CargarMenuSumar(app)

Se encarga de cargar los elementos de la interfaz del menú para sumar y concatenar señales.

**function** CargarMenuSumar**(**app**)**

Redimensionar**(**app**);**

app**.**ReproducirButton**.**Visible**=**'on'**;**

app**.**EspectrogramaModificado**.**Visible**=**'on'**;**

app**.**EspectrogramaOriginal**.**Visible**=**'on'**;**

title**(**app**.**GraficaTiempo**,** 'Señales Sumadas'**)**

title**(**app**.**GraficaFFToRuido**,** 'Señales Concatenadas'**)**

title**(**app**.**Espectograma**,** 'Señales Sumadas FFT'**)**

title**(**app**.**GraficaDEP**,** 'Señales Concatenadas FFT'**)**

title**(**app**.**EspectrogramaOriginal**,** 'Espectrograma suma'**)**

title**(**app**.**EspectrogramaModificado**,** 'Espectrograma concatenación'**)**

xlabel**(**app**.**Espectograma**,** 'Tiempo (s)'**)**

ylabel**(**app**.**Espectograma**,** 'Amplitud'**)**

xlabel**(**app**.**GraficaDEP**,** 'Frecuencia (Hz)'**)**

xlabel**(**app**.**GraficaFFToRuido**,**'Frecuencia(HZ)'**);**

app**.**Espectograma**.**YDir **=** 'normal'**;**

app**.**SumadordesenalesPanel**.**Visible**=**'on'**;**

app**.**ConcatenadordesenalesPanel**.**Visible**=**'on'**;**

xlim**(**app**.**Espectograma**,**'auto'**);**

ylim**(**app**.**Espectograma**,**'auto'**);**

cla**(**app**.**GraficaTiempo**);**

cla**(**app**.**Espectograma**);**

cla**(**app**.**GraficaFFToRuido**);**

cla**(**app**.**GraficaDEP**);**

cla**(**app**.**EspectrogramaOriginal**);**

**end**

function CargarMenuEstudio(app)

Se encarga de cargar los elementos del menú en el que realizamos modificaciones tales como comprimir, expandir, interpolar y diezmar, cuantificar, así como observar la modificación de la fs y distintos efectos de audio (Delay, Reverb, Invertir, Metálico, Shift-Pitching).

%Interfaz menú comprimir, expandir, interpolar y diezmar,

%cuantificar, estudio de la fs y efectos especiales

**function** CargarMenuEstudio**(**app**)**

Redimensionar**(**app**);**

app**.**CompresionExpansionPanel**.**Visible **=** 'on'**;**

app**.**EspectrogramaModificado**.**Visible**=**'on'**;**

app**.**EspectrogramaOriginal**.**Visible**=**'on'**;**

app**.**ReproducirButton**.**Visible**=**'on'**;**

app**.**CargarButton**.**Visible**=**'on'**;**

app**.**InterpolacinydiezmadoPanel**.**Visible**=**'on'**;**

cla**(**app**.**GraficaDEP**);**

title**(**app**.**GraficaDEP**,** 'FFT Señal Modificada'**)**

title**(**app**.**Espectograma**,** 'FFT Señal Original'**)**

title**(**app**.**EspectrogramaOriginal**,** 'Espectrograma Señal Original'**)**

title**(**app**.**EspectrogramaModificado**,** 'Espectrograma Señal Modificada'**)**

title**(**app**.**GraficaFFToRuido**,**'Señal Modificada'**)**

app**.**Espectograma**.**YDir **=** 'normal'**;**

app**.**EspectrogramaModificado**.**YDir **=** 'normal'**;**

app**.**EspectrogramaOriginal**.**YDir **=** 'normal'**;**

xlabel**(**app**.**GraficaFFToRuido**,**'Frecuencia (Hz)'**)**

app**.**RestablecerInterpolacion**.**Visible**=**'on'**;**

app**.**EfectosPanel**.**Visible**=**'on'**;**

app**.**FsPanel**.**Visible**=**'on'**;**

app**.**EfectosPanel**.**Visible**=**'on'**;**

xlim**(**app**.**Espectograma**,**'auto'**);**

ylim**(**app**.**Espectograma**,**'auto'**);**

**end**

function CargaDemodulacion(app)

Este menú sólo aparece si demodulamos la señal. Carga un panel con dos gráficas en las que aparece la señal demodulada tanto en tiempo como en frecuencia.

%Este menu solo aparece si demodulamos la señal. Carga un panel con

%dos gráficas n las que aparece la señal demoduladaa tanto en

%tiempo como en frecuencia

**function** CargaDemodulacion**(**app**)**

app**.**ResultadosDemodulacinMenu**.**Visible**=**'on'**;**

**end**

**Funciones globales**

A continuación vamos a mostrar las funciones globales que utilizamos en el programa.

function [tf,f] = Transformada(~,x,fs)

Función que calcula la transformada rápida de Fourier de la señal de audio que se le introduce como parámetro.

**function** **[**tf**,**f**]** **=** Transformada**(~,**x**,**fs**)**

N**=**length**(**x**);**

tf**=**fft**(**x**,**N**)/**N**;**

tf **=** abs**(**tf**(**1**:**floor**(**N**/**2**+**1**)));**

f**=**linspace**(**0**,**fs**/**2**,(**N**/**2**+**1**));**

**end**

function [tf,f] = DEP(~,x,fs,~)

Función que se encarga de calcular la densidad espectral de la señal de audio que se introduce como parámetro.

% Función que calcula la densidad espectral de potencia de una señal

**function** **[**tf**,**f**]** **=** DEP**(~,**x**,**fs**,~)**

N**=**length**(**x**);**

tf**=**fft**(**x**)/**N**;**

size**(**tf**);**

tf**=**abs**(**tf**(**1**:**floor**(**N**/**2**+**1**)));**

f**=**linspace**(**0**,**fs**/**2**,**N**/**2**+**1**);**

tf**=**tf**.\***conj**(**tf**);**

**end**

function Representar\_Senal(app)

Función que representa la FFT, DEP, el espectrograma y la señal original en sus respectivas gráficas.

% Función que representa la FFT, DEP, el espectograma y la

señal % original en sus respectivas gráficas

**function** Representar\_Senal**(**app**)**

% Guardamos el audio de la memoria a una nueva variable

audioR**=**app**.**a**;**

% Convertimos las señales estéreo a mono

audioR **=** sum**(**audioR**,** 2**)** **/** size**(**audioR**,** 2**);**

app**.**fs**=**app**.**fsCargado**;**

dt **=** 1**/**app**.**fs**;**

t **=** 0**:**dt**:(**length**(**audioR**)\***dt**)-**dt**;**

plot**(**app**.**GraficaTiempo**,**t**,**audioR**);**

% Llamamos a la función para calcular la transformada y la

% representamos

**[**TF**,**f**]=**Transformada**(**app**,**audioR**,**app**.**fsCargado**);**

plot**(**app**.**GraficaFFToRuido**,**f**,**abs**(**TF**))**

% LLamamos a la función para calcular la densidad

espectral de potencia % y la representamos

**[**tf**,**f**]** **=** DEP**(**app**,**audioR**,**app**.**fsCargado**,**app**.**audio**);**

plot**(**app**.**GraficaDEP**,**f**,**abs**(**tf**),**'Color'**,[**0.00 0.45 0.74**]);**

% LLamamos a la función para calcular el espectograma y lo

% representamos

**[**S**,**F**,**T**]** **=** spectrogram**(**audioR**,**800**,[],[],**app**.**fsCargado**);**

imagesc**(**app**.**Espectograma**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**Espectograma**.**YDir **=** 'normal'**;**

% Establecemos los limites de las gráficas

ylim**(**app**.**Espectograma**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**Espectograma**,** **[**min**(**T**),** max**(**T**)]);**

**end**

function [x,t,energiacad,potenciacad,periodocad] = Generar\_Senal(app)

Función que genera señales introduciendo sus parámetros.

**function** **[**x**,**t**,**energiacad**,**potenciacad**,**periodocad**]=**Generar\_Senal**(**app**)**

% Definimos las variables

tipo**=**app**.**SeleccionSenal**.**Value**;**

amplitud**=**app**.**Amplitud**.**Value**;**

frec**=**app**.**Frec**.**Value**;**

app**.**fs**=**app**.**Fs**.**Value**;**

duracion**=**app**.**Duracion**.**Value**;**

% Introducimos un caso de error cuando la frecuencia de

% muestreo es demasiado baja

**if(**app**.**fs**<**1500**)**

app**.**fs**=**1500**;**

app**.**Fs**.**Value**=**1500**;**

%Salta un aviso indicando al usuario que hemos truncado la

%frecuencia de muestreo

warndlg**(**"El valor de fs introducido es demasiado bajo, se ha truncado a un valor fs=1500 para un mejor resultado"**,**"ATENCION"**)**

**end**

% Creamos el vector tiempo

t **=** 0**:**1**/**app**.**fs**:**duracion**;**

% En funcion de la opción marcada por el usuario crea una señal % otra

**switch** tipo

**case** "Seno"

x**=**amplitud**\***sin**(**2**\***pi**\***frec**\***t**);** %Genera un Seno

**case** "Coseno"

x**=**amplitud**\***cos**(**2**\***pi**\***frec**\***t**);** %Genera un coseno

**case** "Cuadrada"

x**=**amplitud**\***square**(**2**\***pi**\***frec**\***t**);** %genera una señal cuadrada

**case** "Triangular"

x**=**amplitud**\***sawtooth**(**2**\***pi**\***frec**\***t**,**0.5**);** %Genera una funcion triangular

**case** "Sierra"

x **=** amplitud**\***sawtooth**(**2**\***pi**\***frec**\***t**);** %Genera una señal de tipo sierra

**end**

% Calculamos la energía

energia**=(**1**/**app**.**fs**)\***sum**(**x**.^**2**);**

% Calculamos la potencia

potencia**=**1**/(**length**(**x**)\***sum**(**x**.^**2**));**

% Calculamos el periodo

periodo**=**1**/**frec**;**

% Añadimos unidades

periodocad**=**strcat**(**num2str**(**periodo**),**" Segundos"**);**

energiacad**=**strcat**(**num2str**(**energia**),**" Julios"**);**

potenciacad**=**strcat**(**num2str**(**potencia**),**" Watios"**);**

% Sacamos la información por pantalla

app**.**LabelEnergia**.**Visible**=**'on'**;**

app**.**LabelPotencia**.**Visible**=**'on'**;**

app**.**LabelPeriodo**.**Visible**=**'on'**;**

app**.**LabelEnergia**.**Text**=**energiacad**;**

app**.**LabelPotencia**.**Text**=**potenciacad**;**

app**.**LabelPeriodo**.**Text**=**periodocad**;**

**end**

function Intro\_ruido(app)

Función que añade ruido blanco a la señal cargada con un valor de SNR introducido por el usuario.

%Funcion que añade ruido blanco a la señal cargada con un valor de SNR introducido por el ususario

**function** Intro\_ruido**(**app**)**

%Guardamos las variables globales en variables globales

senal **=** app**.**a**;**

app**.**fs **=** app**.**fsCargado**;**

snr**=**app**.**SNRdBKnob**.**Value**;**

%Añadimos el ruido

senalruidosa**=**awgn**(**senal**,**snr**);**

%Guardamos la señal ruidosa como variable global

app**.**a**=**senalruidosa**;**

app**.**audio**=**audioplayer**(**app**.**a**,**app**.**fsCargado**);**

% Pasamos la señal de stereo a mono y representamos

senalruidosa **=** sum**(**senalruidosa**,** 2**)** **/** size**(**senalruidosa**,** 2**);**

Muestreo**(**app**,** senalruidosa**,** app**.**fs**,** app**.**fs**,** 4**);**

**end**

function [file,audio,fs] = cargarsenal(~,~)

Función que se encarga de cargar las señales que vamos a sumar.

% Se encarga de cargar las señales que vamos a sumar

**function** **[**file**,**audio**,**fs**]=**cargarsenal**(~,~)**

%Abrir el archivo de audio

**[**file**,**path1**]** **=** uigetfile**(**'\*.wav'**,**'\*.mp3'**);**

**if(**file**~=**0**)**

app**.**filen **=** strcat**(**path1**,**file**);**

**[**audio**,**fs**]** **=** audioread**(**app**.**filen**);**

audio**=** double**(**audio**);**

audio **=** sum**(**audio**,** 2**)** **/** size**(**audio**,** 2**);**

**else**

%En el caso de que haya un problema al abrir un archivo salta %el error y dicho archivo no se carga

file**=**'Error'**;**

audio**=**0**;**

fs**=**0**;**

**end**

**end**

function num = CompruebaCargada(app)

Esta función comprueba si ya existe una señal cargada previamente.

%Esta funcion comprueba si ya existe una señal cargada previamente

**function** num**=**CompruebaCargada**(**app**)**

**if(**isempty**(**app**.**a**)==**true**)**

num**=**1**;** %si no existe una señal cargada devuelve un uno

**else**

num**=**0**;** %Si existe una señal cargada devuelve un cero

**end**

**end**

function Muestreo(app,audio,fsRep,fs,num)

En esta función se guarda el objeto que queramos reproducir en la variable global audio, se calcula la transformada y se representa la función generada que hayamos pasado en el argumento audio.

%En esta señal se guarda en la variable global audio el objeto que

%queramos reproducir se calcula la transformada y se representa la funcion generada que hayamos

%pasado en el argumento audio

**function** Muestreo**(**app**,**audio**,**fsRep**,**fs**,**num**)**

% Se guarda el objeto audioplayer se pasa a mono y se

% calcula la transformada

xlabel**(**app**.**Espectograma**,** 'Tiempo (s)'**)**

app**.**audio**=**audioplayer**(**audio**,**fsRep**);**

audioCompr **=** sum**(**audio**,** 2**)** **/** size**(**audio**,** 2**);**

dt **=** 1**/**fs**;**

t **=** 0**:**dt**:(**length**(**audioCompr**)\***dt**)-**dt**;**

app**.**TextArea\_14**.**Value**=**num2str**(**fsRep**);**

**[**TF**,**f**]=**Transformada**(**app**,**audioCompr**,**fsRep**);**

% LLamamos a la función para calcular el espectograma

**[**S**,**F**,**T**]** **=** spectrogram**(**audioCompr**,**800**,[],[],**fs**);**

Desde donde se llame a esta función querem os

%representar la información en una grafica determinada por

%lo que distinguimos diferentes casos. El numero indica

%desde donde se ha llamado a esta funcion

**if(**num**==**1**)**%Cuantificar,expandir,comprimir,interpolar,diezmar

plot**(**app**.**GraficaFFToRuido**,**t**,**audioCompr**);**

plot**(**app**.**GraficaDEP**,**f**,**abs**(**TF**));**

imagesc**(**app**.**EspectrogramaModificado**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**EspectrogramaModificado**.**YDir **=** 'normal'**;**

% Establecemos los limites de las gráficas

ylim**(**app**.**EspectrogramaModificado**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**EspectrogramaModificado**,** **[**min**(**T**),** max**(**T**)]);**

**end**

**if(**num**==**2**)**%Modular

plot**(**app**.**GraficaFFToRuido**,**t**,**audioCompr**);**

plot**(**app**.**GraficaDEP**,**f**,**abs**(**TF**));**

imagesc**(**app**.**EspectrogramaModificado**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**EspectrogramaModificado**.**YDir **=** 'normal'**;**

% Establecemos los limites de las gráficas

ylim**(**app**.**EspectrogramaModificado**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**EspectrogramaModificado**,** **[**min**(**T**),** max**(**T**)]);**

**end**

**if** **(**num**==**3**)**%Demodular

plot**(**app**.**Demodulada**,**t**,**audioCompr**);**

plot**(**app**.**FFTDemodulada**,**f**,**abs**(**TF**));**

imagesc**(**app**.**EspectrogramaModificado**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**EspectrogramaModificado**.**YDir **=** 'normal'**;**

% Establecemos los limites de las gráficas

ylim**(**app**.**EspectrogramaModificado**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**EspectrogramaModificado**,** **[**min**(**T**),** max**(**T**)]);**

**end**

**if(**num**==**4**)** % Ruido y concatenar

plot**(**app**.**GraficaFFToRuido**,**t**,**audioCompr**);**

plot**(**app**.**GraficaDEP**,**f**,**abs**(**TF**));**

imagesc**(**app**.**EspectrogramaModificado**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**EspectrogramaModificado**.**YDir **=** 'normal'**;**

% Establecemos los limites de las gráficas

ylim**(**app**.**EspectrogramaModificado**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**EspectrogramaModificado**,** **[**min**(**T**),** max**(**T**)]);**

**end**

**if(**num**==**5**)** % Sumar

plot**(**app**.**GraficaTiempo**,**t**,**audioCompr**);**

plot**(**app**.**Espectograma**,**f**,**abs**(**TF**));**

imagesc**(**app**.**EspectrogramaOriginal**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**EspectrogramaOriginal**.**YDir **=** 'normal'**;**

% Establecemos los limites de las gráficas

ylim**(**app**.**EspectrogramaOriginal**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**EspectrogramaOriginal**,** **[**min**(**T**),** max**(**T**)]);**

**end**

xlabel**(**app**.**GraficaDEP**,** 'Frecuencia (Hz)'**)** %Al final cambiamos las unidades a una de las graficas

**end**

**end**

**Funciones asociadas a componentes de la interfaz**

A continuación vamos a mostrar las funciones (Callbacks) que están asociadas a los componentes de la interfaz.

function MenuPrincipal(app, event)

Se encarga de llamar a la función BorraPantalla() para eliminar los elementos de otros menús, carga los elementos del menú principal y en caso de que haya un archivo cargado lo representa en el dominio temporal, frecuencial (FFT), así como su densidad espectral de potencia y su espectrograma.

% Menu selected function: MenPrincipalMenu

**function** MenuPrincipal**(**app**,** event**)**

% LLama a la función borra pantalla para eliminar los

elementos % del menú principal

BorraPantalla**(**app**)**

% Carga los diferentes elementos del menú cargar/grabar

CargarMenuPrincipal**(**app**);**

%En caso de que haya un archiv cargado lo representa

**if(**CompruebaCargada**(**app**)==**0**)**

Representar\_Senal**(**app**);**

**end**

**end**

function CargarSenal(app, event)

Función asociada al botón cargar. Se encarga de cargar una señal en caso de ser pulsado dicho botón.

% Button pushed function: BotonCargar

**function** CargarSenal**(**app**,** event**)**

%Carga una señal si se pulsa el boton cargar

**[**filename**,** pathname**]** **=** uigetfile**({**'\*.wav'**;**'\*.mp3'**;**'\*.wma'**},** 'Selecciona un archivo de audio'**);**

app**.**filen **=** filename**;**

app**.**path **=** pathname**;**

%Guardamos el nombre del fichero y su ruta

**[**aud**,**app**.**fs**]=**audioread**([**pathname filename**]);**

%Guardamos la informacion en las distintas variables locales

app**.**a **=** aud**;**

app**.**restablecer **=** app**.**a**;**

app**.**fsCargado **=** app**.**fs**;**

app**.**audio **=** audioplayer**(**aud**,**app**.**fs**);**

obj **=** audioplayer**(**aud**,**app**.**fs**);**

app**.**TextArea\_13**.**Value**=**num2str**(**app**.**fsCargado**);**

app**.**TextArea\_14**.**Value**=**num2str**(**app**.**fsCargado**);**

% Llamamos a la función Representar\_Señal para mostrar la señal

% en tiempo, la FFT, la DEP y el espectrograma

Representar\_Senal**(**app**);**

% Calculamos la energia y la potencia y sacamos la información % de la señal cargada por pantalla

aud **=** sum**(**app**.**a**,** 2**)** **/** size**(**app**.**a**,** 2**);**

energia **=** sum**(**abs**(**aud**).^**2**)/(**length**(**aud**));**

Pot**=**energia**/(**length**(**aud**)/**app**.**fsCargado**);**

Pav**=**Pot**/**length**(**aud**);**

nombre**=**strcat**(**"Nombre: "**,**filename**);** %Nombre del fichero

app**.**NombreLabel**.**Text **=** nombre**;**

app**.**MuestrastotalesLabel**.**Text **=** strcat**(**"Muestras totales: "**,**num2str**(**obj**.**TotalSamples**));**

frecmuestr**=** strcat**(**"Frecuencia de muestreo: "**,**num2str**(**app**.**fs**));** %Frecuencia de la señal

app**.**FrecuenciademuestreoLabel**.**Text **=** frecmuestr**;**

app**.**PeriodoLabel\_2**.**Text **=** strcat**(**"Periodo: "**,**num2str**(**obj**.**TimerPeriod**),** " segundos"**);** %Periodo de la señal

energiacad **=** strcat**(**"Energía: "**,**num2str**(**energia**),**" Julios"**);** %Energia de la Señal

app**.**EnergaLabel**.**Text **=** energiacad**;**

pavcad **=** strcat**(**"Potencia media: "**,**num2str**(**Pav**),**" Watios"**);** %Potencia de la señal

app**.**PotenciamediaLabel\_2**.**Text **=** pavcad**;**

**end**

function DejarGrabarSenal(app, event)

Función asociada al botón dejar de grabar, representado en la interfaz con un círculo rojo y negro. Cuando se pulsa el botón para la grabación para su posterior representación en los dominios temporal y frecuencial.

% Image clicked function: Image\_2

**function** DejarGrabarSenal**(**app**,** event**)**

%Para de grabar cuando se pulsa el boton

stop**(**app**.**audioRecord**);**

%Se guarda lo grabado

app**.**audio**=**audioplayer**(**app**.**audioRecord**);**

app**.**audioGrabado **=** getaudiodata**(**app**.**audioRecord**);**

app**.**a**=**app**.**audioGrabado**;**

app**.**restablecer**=**app**.**audioGrabado**;**

app**.**fsCargado**=**44100**;**

app**.**fs**=**44100**;**

%Convertimos señales estereo a mono

app**.**audioGrabado **=** sum**(**app**.**audioGrabado**,** 2**)** **/** size**(**app**.**audioGrabado**,** 2**);**

dt **=** 1**/**app**.**fs**;**

t **=** 0**:**dt**:(**length**(**app**.**audioGrabado**)\***dt**)-**dt**;**

plot**(**app**.**GraficaTiempo**,**t**,**app**.**audioGrabado**);**

%Llamamos a la funcion para calcular la transformada y la representamos

**[**TF**,**f**]=**Transformada**(**app**,**app**.**audioGrabado**,**app**.**fs**);**

plot**(**app**.**GraficaFFToRuido**,**f**,**abs**(**TF**));**

%Densidad espectral de potencia

**[**tf**,**f**]** **=** DEP**(**app**,**app**.**audioGrabado**,**app**.**fs**);**

plot**(**app**.**GraficaDEP**,**f**,**abs**(**tf**));**

% LLamamos a la función para calcular el espectograma y lo

% representamos

**[**S**,**F**,**T**]** **=** spectrogram**(**app**.**audioGrabado**,**800**,[],[],**app**.**fsCargado**);**

imagesc**(**app**.**Espectograma**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**Espectograma**.**YDir **=** 'normal'**;**

% Establecemos los limites de las gráficas

ylim**(**app**.**Espectograma**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**Espectograma**,** **[**min**(**T**),** max**(**T**)]);**

Funcion que calcula la densidad espectral de potencia de la % señal grabada

**function** **[**tf**,**f**]** **=** DEP**(~,**x**,**fs**)**

N**=**length**(**x**);**

tf**=**fft**(**x**)/**N**;**

size**(**tf**);**

tf**=**abs**(**tf**(**1**:**floor**(**N**/**2**+**1**)));**

f**=**linspace**(**0**,**fs**/**2**,**N**/**2**+**1**);**

tf**=**tf**.\***conj**(**tf**);**

**end**

**end**

function GenerarSenal(app, event)

Se encarga de generar una señal cuando pulsamos el botón generar. Los parámetros de esta señal: amplitud, frecuencia y duración, vienen determinados por el usuario. Una vez generada la señal se representa tanto en dominio temporal como frecuencial (FFT).

% Button pushed function: GenerarButton

**function** GenerarSenal**(**app**,** event**)**

%Se genera una señal cuando se pulsa el boton generar

amplitud**=**app**.**Amplitud**.**Value**;**

frec**=**app**.**Frec**.**Value**;**

duracion**=**app**.**Duracion**.**Value**;**

%Si ninguno de los valores introducidos para generar la señal %es cero

**if(**amplitud**>**0**&&**frec**>**0**&&**duracion**>=**1**)**

% Llamamos a la función Generar\_Senal que nos genera una señal % a partir de unos parámetros dados

**[**x**,**t**]=**Generar\_Senal**(**app**);**

% Representamos la señal generada en tiempo

plot**(**app**.**GraficaTiempo**,**t**,**x**);**

app**.**fs**=**app**.**Fs**.**Value**;**

%Llamamos a la funcion para calcular la transformada y la

% representamos

**[**tf**,**f**]=**Transformada**(**app**,**x**,**app**.**fs**);**

plot**(**app**.**GraficaFFToRuido**,**f**,**abs**(**tf**));**

% Calculamos la densidad espectral de potencia y la

% representamos

**[**tf**,**f**]** **=** DEP**(**app**,**x**,**app**.**fs**);**

plot**(**app**.**GraficaDEP**,**f**,**abs**(**tf**));**

% LLamamos a la función para calcular el espectograma y lo

% representamos

**[**S**,**F**,**T**]** **=** spectrogram**(**x**,**800**,[],[],**app**.**fsCargado**);**

imagesc**(**app**.**Espectograma**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**Espectograma**.**YDir **=** 'normal'**;**

% Establecemos los limites de las gráficas

ylim**(**app**.**Espectograma**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**Espectograma**,** **[**min**(**T**),** max**(**T**)]);**

**else**

warndlg**(**"Por favor, introduce valores correctos"**,**"ATENCION"**)**

**end**

**end**

function Interpolar(app, event)

Esta función asociada al botón interpolar calcula y representa la interpolación de una señal, cuyo factor es seleccionado por el usuario.

% Button pushed function: BotonInterpolar

**function** Interpolar**(**app**,** event**)**

%Calcula una interpolación si se pulsa el boton

**if(**CompruebaCargada**(**app**)==**0**)**

%Si ya tenemos la señal cargada podremos interpolarla

factor**=**app**.**FactorInterpolacion**.**Value**;**

**if(**factor**>**0**)**

%Ahora pasamos a interpolarla y guardamos los valores en las

%variables locales

app**.**audioInterpolado**=**upsample**(**app**.**a**,**factor**);**

app**.**fsREP**=** factor**\***app**.**fsCargado**;**

app**.**fsCargado **=** app**.**fsREP**;**

app**.**fs**=**app**.**fsREP**;**

app**.**a**=**app**.**audioInterpolado**;**

%Esta funcion representarala señal interpolada y la FFT de la

%señal interpolada Muestreo**(**app**,**app**.**audioInterpolado**,**app**.**fsREP**,**app**.**fs**,**1**);**

title**(**app**.**GraficaFFToRuido**,** 'Señal Interpolada'**);**

title**(**app**.**GraficaDEP**,** 'FFT Señal Interpolada'**);**

**else**

warndlg**(**"Debe introducir valores positivos"**,**"ATENCION"**)**

**end**

**else** %Si no esta cargada salta un error

warndlg**(**"Debe de cargar un fichero para poder reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function FiltrarButtonPushed(app, event)

Función asociada al botón “Filtrar”. Se encarga de llamar a la función que filtra señales.

% Callback function

function FiltrarSenal(app, event)

%Si se pulsa el boton filtrar se llama a esta funcion

if(CompruebaCargada(app)==0) %Comprobamos que la señal no esta cargada

if(app.FrecuenciadepasofinalEditField.Value>0 && app.FrecuenciadeparadafinalEditField.Value>0 && app.AtenuacinenbandadeparadadBEditField.Value >=0 && app.AtenuacinenbandadepasodBEditField.Value>=0)

% Recogemos la señal que queremos filtrar y su frecuencia de

% muestreo

title(app.GraficaDEP, 'Filtro y Señal Filtrada')

Filtrado.audio = app.a;

Filtrado.senalEntrada = app.a;

Filtrado.fs = app.fsCargado;

Fn = Filtrado.fs/2;

% Distinguimos varios casos en función de la aproximación que

% elija el usuario

aproximacion = app.AproximacionDropDown.Value;

filtr=app.FiltroDropDown.Value;

switch (aproximacion)

case "Butterworth" %Si queremos un filtro Butterworth

tipoFiltro=1;

case "Chebyshev" %Si queremos un filtro Chebyshev

tipoFiltro=2;

case "Cauer" %Si queremos un filtro Cuer

tipoFiltro=3;

end

% Distinguimos varios casos en función del tipo de filtro que

% elija el usuario

switch (filtr)

case "Paso bajo" %Si queremos un filtro paso bajo

Filtrado.frecFiltro = 1;

Filtrado.WpValue = app.FrecuenciadepasofinalEditField.Value/(Filtrado.fs/2);

Filtrado.WsValue = app.FrecuenciadeparadafinalEditField.Value/(Filtrado.fs/2);

case "Paso alto" %Si queremos un filtro paso alto

Filtrado.frecFiltro = 2;

Filtrado.WpValue = app.FrecuenciadepasofinalEditField.Value/(Filtrado.fs/2);

Filtrado.WsValue = app.FrecuenciadeparadafinalEditField.Value/(Filtrado.fs/2);

case "Paso banda" %si queremos un filtro paso banda

Filtrado.frecFiltro = 3;

Filtrado.WpValue = [app.FrecuenciadepasoinicialEditField.Value/(Fn) app.FrecuenciadepasofinalEditField.Value/(Fn)];

Filtrado.WsValue = [app.FrecuenciadeparadainicialEditField.Value/(Fn) app.FrecuenciadeparadafinalEditField.Value/(Fn)];

end

% Llamamos a la función que calcula la transformada y la

% representamos

[TF,f]=Transformada(app,Filtrado.audio,Filtrado.fs);

plot(app.Espectograma,f, TF)

filtro = (tipoFiltro - 1)\*3 + Filtrado.frecFiltro;

switch filtro

case 0

NUM=0; DEN=0;

case 1

[N,Wn]=buttord(Filtrado.WpValue,Filtrado.WsValue,app.AtenuacinenbandadepasodBEditField.Value, app.AtenuacinenbandadeparadadBEditField.Value);

[NUM,DEN]=butter(N,Wn);

case 2

[N,Wn]=buttord(Filtrado.WpValue,Filtrado.WsValue,app.AtenuacinenbandadepasodBEditField.Value, app.AtenuacinenbandadeparadadBEditField.Value);

[NUM,DEN]=butter(N,Wn,'high');

case 3

[N,Wn]=buttord(Filtrado.WpValue,Filtrado.WsValue,app.AtenuacinenbandadepasodBEditField.Value, app.AtenuacinenbandadeparadadBEditField.Value);

[NUM,DEN]=butter(N,Wn);

case 4

[N,Wn]=cheb1ord(Filtrado.WpValue,Filtrado.WsValue,app.AtenuacinenbandadepasodBEditField.Value, app.AtenuacinenbandadeparadadBEditField.Value);

[NUM,DEN]=cheby1(N,app.AtenuacinenbandadepasodBEditField.Value,Wn);

case 5

[N,Wn]=cheb1ord(Filtrado.WpValue,Filtrado.WsValue,app.AtenuacinenbandadepasodBEditField.Value, app.AtenuacinenbandadeparadadBEditField.Value);

[NUM,DEN]=cheby1(N,app.AtenuacinenbandadepasodBEditField.Value,Wn,'high');

case 6

[N,Wn]=cheb1ord(Filtrado.WpValue,Filtrado.WsValue,app.AtenuacinenbandadepasodBEditField.Value, app.AtenuacinenbandadeparadadBEditField.Value);

[NUM,DEN]=cheby1(N,app.AtenuacinenbandadepasodBEditField.Value,Wn);

case 7

[N,Wn]=ellipord(Filtrado.WpValue,Filtrado.WsValue,app.AtenuacinenbandadepasodBEditField.Value, app.AtenuacinenbandadeparadadBEditField.Value);

[NUM,DEN]=ellip(N,app.AtenuacinenbandadepasodBEditField.Value,app.AtenuacinenbandadeparadadBEditField.Value,Wn);

case 8

[N,Wn]=ellipord(Filtrado.WpValue,Filtrado.WsValue,app.AtenuacinenbandadepasodBEditField.Value, app.AtenuacinenbandadeparadadBEditField.Value);

[NUM,DEN]=ellip(N,app.AtenuacinenbandadepasodBEditField.Value,app.AtenuacinenbandadeparadadBEditField.Value,Wn,'high');

case 9

[N,Wn]=ellipord(Filtrado.WpValue,Filtrado.WsValue,app.AtenuacinenbandadepasodBEditField.Value, app.AtenuacinenbandadeparadadBEditField.Value);

[NUM,DEN]=ellip(N,app.AtenuacinenbandadepasodBEditField.Value,app.AtenuacinenbandadeparadadBEditField.Value,Wn);

end

[h,w]=freqz(NUM,DEN);

w1=w\*Filtrado.fs/8;

filtro = abs(h)/length(Filtrado.audio)\*max(max(abs(fft(Filtrado.audio))));

Filtrado.senalFiltrada = filter(NUM,DEN,Filtrado.senalEntrada);

app.a=Filtrado.senalFiltrada;

app.audio=audioplayer(app.a,app.fsCargado);

% Convertimos la señales estereo a mono

audioFiltrado = sum(Filtrado.senalFiltrada, 2) / size(Filtrado.senalFiltrada, 2);

app.fs=app.fsCargado;

dt = 1/app.fs;

t = 0:dt:(length(audioFiltrado)\*dt)-dt;

plot(app.GraficaFFToRuido,t,audioFiltrado);

% Llamamos a la funcion que calcula la transformada

[TF2,f2]=Transformada(app,audioFiltrado,Filtrado.fs);

%Representamos

plot(app.GraficaDEP,w1\*max(f)/max(w1),filtro, 'red')

hold(app.GraficaDEP,'on');

hold(app.Espectograma,'on');

plot(app.GraficaDEP,f2,TF2, "Color",[0.00 0.45 0.74]);

app.ocultarFiltro = plot(app.Espectograma,w1\*max(f)/max(w1),filtro, 'red');

title(app.Espectograma, 'Filtro y señal original')

% LLamamos a la función para calcular el espectograma y lo

% representamos

[S,F,T] = spectrogram(audioFiltrado,800,[],[],app.fsCargado);

imagesc(app.EspectrogramaModificado,T,F,log(abs(S)))

app.EspectrogramaModificado.YDir = 'normal';

% Establecemos los limites de las gráficas

ylim(app.EspectrogramaModificado, [min(F), max(F)]);

xlim(app.EspectrogramaModificado, [min(T), max(T)]);

else %Si emos introducidos valores nulos salta un aviso

warndlg("Debe introducir valores positivos","ATENCION")

end

else %Si la señal no esta cargada salta un error

warndlg("Debe de cargar un fichero para poder reproducirlo","ATENCION")

end

end

function FiltrarSenal(app, event)

Esta función se encarga de filtrar una señal previamente cargada. Podemos escoger, por una parte, entre filtro paso bajo, filtro paso alto o filtro paso banda; y por otra parte entre Butterworth, Chebyshev y Cauer. Posteriormente se representa la señal original, la señal filtrada y el filtro escogido.

% Button pushed function: FiltrarButton

**function** FiltrarButtonPushed**(**app**,** event**)**

% Llamamos a la función que filtra señales

title**(**app**.**GraficaFFToRuido**,** 'Señal Filtrada'**)**

FiltrarSenal**(**app**,** event**);**

**end**

function ModulacionAMButtonPushed(app, event)

Función que realiza una modulación AM de la señal de audio cargada si pulsamos el botón “Modulación AM”. Se utiliza una portadora preseleccionada.

% Button pushed function: ModulacionAMButton

**function** ModulacionAMButtonPushed**(**app**,** event**)**

%Se modula en am si se pulsa el boton modulacion am

**if(**CompruebaCargada**(**app**)==**0**)** %Se comprueba que existe una señal cargada

% Recogemos la señal que queremos modular

audioR **=** app**.**a**;**

app**.**fs **=** app**.**fsCargado**;**

%Convertimos las señales estereo a mono

audioR **=** sum**(**audioR**,** 2**)** **/** size**(**audioR**,** 2**);**

%Procedemos a la modulacion AM

**if(**length**(**audioR**)~=**1**)**

%Datos de la modulación AM

Ac**=**0.85**;**

ka **=** 1**;**

fc**=** 3**\***app**.**fs**/**10 **;** %frecuencia de la portadora

%Calculo de la señal modulada en am

audioRecogido**.**modulada **=** Ac**\***ka**\***modulate**(**audioR**,**fc**,**app**.**fs**,**'amdsb-tc'**,** **-**1**/**ka**);**

app**.**audioAM **=** audioRecogido**.**modulada**;**

Muestreo**(**app**,**audioRecogido**.**modulada**,**app**.**fs**,**app**.**fs**,**2**);**

%Representamos

title**(**app**.**GraficaDEP**,** 'FFT AM'**)**

title**(**app**.**GraficaFFToRuido**,** 'Señal modulada AM'**)**

**end**

**else** %Si no existe una señal cargada salta un error

warndlg**(**"Debe de cargar un fichero para poder reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function MenuModulacionyFiltrado(app, event)

Función que llama a la función BorraPantalla() para eliminar los elementos del menú anterior y carga los diferentes elementos del menú modulación.

% Menu selected function: ModulacinyFiltradoMenu

**function** MenuModulacionyFiltrado**(**app**,** event**)**

% LLama a la función borra pantalla para eliminar los elementos % del menú principal

BorraPantalla**(**app**)**

% Carga los diferentes elementos del menú

% Modulación/Demodulación

CargarMenuModulacion**(**app**);**

CargarMenuFiltrado**(**app**);**

**end**

function ModulacionFMButtonPushed(app, event)

Si pulsamos el botón “Modulación FM” realiza la modulación FM de la señal seleccionada para una portadora preseleccionada.

% Button pushed function: ModulacionFMButton

**function** ModulacionFMButtonPushed**(**app**,** event**)**

%Se calcula una modulacion Fm si se pulsa el boton modular FM

**if(**CompruebaCargada**(**app**)==**0**)** %Se carga la señal

% Recogemos la señal que queremos modular

audioR **=** app**.**a**;**

app**.**fs **=** app**.**fsCargado**;**

%Convertimos las señales estereo a mono

audioR **=** sum**(**audioR**,** 2**)** **/** size**(**audioR**,** 2**);**

**if(**length**(**audioR**)~=**1**)**

%Datos para la modulación am

fc **=** 2**\***app**.**fs**/**5**;**

fm **=** 1000**;**

Ac **=** 7**;**

Am **=** 3**;**

B **=** 2**;**

k**=(**2**\***pi**\***B**\***fm**)/(**app**.**fs**\***Am**);**

%Se pasa a calcular la señal modulada en fm

audioRecogido**.**modulada**=** Ac**\***modulate**(**audioR**,** fc**,** app**.**fs**,** 'fm'**,** k**);**

app**.**audioFM **=** audioRecogido**.**modulada**;**

Muestreo**(**app**,**audioRecogido**.**modulada**,**app**.**fs**,**app**.**fs**,**2**);**

%Se representa la señal modulada en fm

title**(**app**.**GraficaFFToRuido**,** 'Señal modulada FM'**)**

title**(**app**.**GraficaDEP**,** 'FFT FM'**)**

xlabel**(**app**.**GraficaDEP**,** 'Frecuencia (Hz)'**)**

**end**

**else** %Si no existe ninguna señal cargada salta un error

warndlg**(**"Debe de cargar un fichero para poder reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function DemodulacionAMButtonPushed(app, event)

Al pulsar el botón “Demodulación AM” se encarga de hacer la demodulación para la portadora AM seleccionada.

% Button pushed function: DemodulacionAMButton

**function** DemodulacionAMButtonPushed**(**app**,** event**)**

%Se demodula la señal si se pulsa el boton de demodular

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si existe una

señal cargada

**if(~**isempty**(**app**.**audioAM**))** % Solo demodulamos en AM si

previamente habíamos modulado en AM

CargaDemodulacion**(**app**);** %Se carga el menu en el que se representara la señal demoduladada

%Recogemos los datos de la señal modulada

audioR **=** app**.**audioAM**;**

app**.**fs**=** app**.**fsCargado**;**

**if(**length**(**audioR**)~=**1**)**

Ac**=**0.85**;**

ka **=** 1**;**

fc**=** 3**\***app**.**fs**/**10 **;** %frecuencia de la portadora

%Procedemos a demodular la señal

demodulada **=** **(**2**/(**ka**\***Ac**))\***demod**(**audioR**,** fc**,**

app**.**fs**,** 'amdsb-tc'**,** Ac**/**2**);**

Muestreo**(**app**,**demodulada**,**app**.**fs**,**app**.**fs**,**3**);**

app**.**a**=**demodulada**;**

**end**

**else** %Si no había una señal modulada en am entonces salta

un error

warndlg**(**"Para demodular una función AM primero

debe moduarla"**,**"ATENCION"**)**

**end**

**else** %si no habia una señal cargada entonces salta un

error

warndlg**(**"Debe de cargar un fichero para poder

reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function DemodulacionFMButtonPushed(app, event)

Al pulsar el botón “Demodulación FM” se encarga de hacer la demodulación para la portadora FM seleccionada.

% Button pushed function: DemodulacionFMButton

**function** DemodulacionFMButtonPushed**(**app**,** event**)**

% Se demodula la señal si se pulsa el boton de demodular

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si existe una

señal cargada

**if(~**isempty**(**app**.**audioFM**))** % Solo demodulamos FM si

previamente habíamos mdulado en FM

CargaDemodulacion**(**app**);** %Se carga el menu en el que se

representara la señal demoduladada

% Recogemos los datos de la señal modulada

audioR **=** app**.**audioFM**;**

app**.**fs**=** app**.**fsCargado**;**

**if(**length**(**audioR**)~=**1**)**

fc **=** 2**\***app**.**fs**/**5**;**

fm **=** 1000**;**

Am **=** 3**;**

B **=** 2**;**

k**=(**2**\***pi**\***B**\***fm**)/(**app**.**fs**\***Am**);**

% Procedemos a demodular la señal

demodulada**=** demod**(**audioR**,** fc**,** app**.**fs**,** 'fm'**,** k**);**

Muestreo**(**app**,**demodulada**,**app**.**fs**,**app**.**fs**,**3**);**

app**.**a**=**demodulada**;**

**end**

% Si no habia una señal modulada en FM entonces salta un

error

**else**

warndlg**(**"Para demodular una función FM primero debe

moduarla"**,**"ATENCION"**)**

**end**

% Si no habia una señal cargada entonces salta un error

**else**

warndlg**(**"Debe de cargar un fichero para poder

reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function ReproducirButtonPushed(app, event)

Función que se encarga de reproducir el audio cargado. Podemos reproducir tanto el audio original y el audio grabado como los efectos producidos por modulación, interpolación, diezmado…

% Callback function: Image2, ReproducirButton

**function** ReproducirButtonPushed**(**app**,** event**)**

**if(**CompruebaCargada**(**app**)==**0**)**

factorcompresion**=** app**.**FactorCompresion**.**Value**;**

factorInterpolacion**=**app**.**FactorInterpolacion**.**Value**;**

**if(**factorcompresion**<**5**&&**factorInterpolacion**<**5**)**

% Reproduce el audio cargado

play**(**app**.**audio**);**

**else**

warndlg**(**"La tarjeta de sonido no permite reproducir

la canción cargada con una frecuencia de muestreo

tan alta"**,**"ATENCION"**)**

**end**

**else**

warndlg**(**"Debe de cargar un fichero para poder

reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function BotonDiezmarButtonPushed(app, event)

Función que se encarga de diezmar una señal cargada por el factor de diezmado introducido por el usuario. Esto se realiza al pulsar el botón “Diezmar”.

% Button pushed function: BotonDiezmar

**function** BotonDiezmarButtonPushed**(**app**,** event**)**

% Recogemos la señal que queremos diezmar y su frecuencia

de % muestreo

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos que hay una

señal cargada % Recogemos el factor de diezmado introducido por el usuario

factor**=**app**.**Factordiezmado**.**Value**;**

**if** **(**factor**>**0**)**

app**.**audioDiezmado**=**downsample**(**app**.**a**,**factor**);**

% La nueva frecuencia de muestreo es la frecuencia de

muestreo % de la señal original dividida por el factor de

diezmado

app**.**fsREP**=** app**.**fsCargado**/**factor**;**

app**.**fsCargado **=** app**.**fsREP**;**

app**.**fs**=**app**.**fsREP**;**

app**.**a**=** app**.**audioDiezmado**;**

% Llamamos a la función Muestreo() para representar en tiempo y % frecuencia la señal diezmada

Muestreo**(**app**,**app**.**audioDiezmado**,**app**.**fsREP**,**app**.**fs**,**1**);**

title**(**app**.**GraficaFFToRuido**,** 'Señal Diezmada'**);**

title**(**app**.**GraficaDEP**,** 'FFT Señal Diezmada'**);**

**else**

warndlg**(**"Debe introducir valores positivos"**,**"ATENCION"**)**

**end**

% Si no habia una señal cargada entonces salta un error

**else**

warndlg**(**"Debe de cargar un fichero para poder reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function PausarButtonPushed(app, event)

Al pulsar el botón “Pause” pausamos el audio cargado.

% Image clicked function: Image4

**function** PausarButtonPushed**(**app**,** event**)**

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos que hay alguna

señal cargada

% Pausamos la reproducción del audio

pause**(**app**.**audio**);**

% Si no habia una señal cargada entonces salta un error

**else**

warndlg**(**"Debe de cargar un fichero para poder reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function ReanudarButtonPushed(app, event)

Al pulsar el botón “Reanudar” continúa con la reproducción de una pista tras una pausa momentánea.

% Image clicked function: Image2\_3

**function** ReanudarButtonPushed**(**app**,** event**)**

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si hay alguna

señal cargada % Reanudamos la reproducción del audio

resume**(**app**.**audio**);**

% Si no habia una señal cargada entonces salta un error

**else**

warndlg**(**"Debe de cargar un fichero para poder reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function EmpezaragrabarButtonPushed(app, event)

Al pulsar el botón con el icono del micrófono comenzamos a grabar a través del ordenador.

% Image clicked function: Image

**function** EmpezaragrabarButtonPushed**(**app**,** event**)**

% Comienza la grabación de audio

app**.**audioRecord **=** audiorecorder**(**44100**,**16**,**2**);**

record**(**app**.**audioRecord**);**

**end**

function CargarButtonPushed(app, event)

Función que carga una señal de audio, la representa en el dominio temporal y muestra su frecuencia de muestreo por pantalla.

% Button pushed function: CargarButton

**function** CargarButtonPushed**(**app**,** event**)**

% Recogemos la señal seleccionada por el usuario de un

archivo

**[**filename**,** pathname**]** **=** uigetfile**({**'\*.wav'**;**'\*.mp3'**;**'\*.wma'**},** 'Select audiofile'**);**

app**.**path **=** pathname**;**

app**.**filen **=** filename**;**

**[**aud**,**app**.**fs**]=**audioread**([**pathname filename**]);**

app**.**a **=** aud**;**

% Guardamos la señal original en la variable global

% app.reestablecer por si necesitamos reestablecerla en un

% futuro

app**.**restablecer **=** app**.**a**;**

app**.**fsCargado **=** app**.**fs**;**

% Creamos un objeto de tipo audio para poder reproducirlo

% posteriormente

app**.**audio **=** audioplayer**(**aud**,**app**.**fs**);**

app**.**TextArea\_13**.**Value**=**num2str**(**app**.**fsCargado**);**

app**.**TextArea\_14**.**Value**=**num2str**(**app**.**fsCargado**);**

% Representamos la señal cargada

Representar**(**app**);**

**end**

function ExportarButtonPushed(app, event)

Función que exporta los valores (tipo de onda, frecuencia, periodo, energía y potencia) de la señal generada a un archivo de texto.

% Button pushed function: ExportarButton

**function** ExportarButtonPushed**(**app**,** event**)**

% Función que te exporta los valores de la señal generada

a un % archivo de texto

% Recogemos los valores que queremos exportar

tipo**=**app**.**SeleccionSenal**.**Value**;**

amplitud**=**app**.**Amplitud**.**Value**;**

frec**=**app**.**Frec**.**Value**;**

app**.**fs**=**app**.**Fs**.**Value**;**

duracion**=**app**.**Duracion**.**Value**;**

energia**=**app**.**LabelEnergia**.**Text**;**

potencia**=**app**.**LabelPotencia**.**Text**;**

periodo**=**app**.**LabelPeriodo**.**Text**;**

**if(**strcmp**(**energia**,**"Label"**))**

warndlg**(**"Genere una señal para poder exportarla"**,**"ATENCION"**)**

**else**

% Abrimos el fichero que vamos a generar

**[**s**,**ruta**]=**uiputfile**(**'\*.txt'**,**'Guardar informe'**);**

fichero **=** fopen**(**strcat**(**ruta**,** s**),**'w'**);**

fprintf**(**fichero**,**'Para una onda de tipo %s con amplitud %d, frecuencia %d, \nfrecuencia de muestreo %d y duracion %d:\n'**,**tipo**,**amplitud**,**frec**,**app**.**fs**,**duracion**);**

fprintf**(**fichero**,**' Periodo: %s \n'**,**periodo**);**

fprintf**(**fichero**,**' Energia: %s \n'**,**energia**);**

fprintf**(**fichero**,**' Potencia: media %s \n'**,**potencia**);**

fclose**(**fichero**);**

**end**

**end**

function Ruido(app, event)

Al pulsar el botón “Añadir” se llama a la función que introduce ruido.

% Button pushed function: AadirButton

**function** Ruido**(**app**,** event**)**

% Llamamos a la funcion que introduce ruido

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos que haya alguna señal cargada

title**(**app**.**GraficaFFToRuido**,** 'Señal Ruidosa'**)**

title**(**app**.**GraficaDEP**,** 'FFT Señal Ruidosa'**)**

Intro\_ruido**(**app**)**

% Si no hay ninguna señal cargada salta un error

**else**

warndlg**(**"Debe de cargar un fichero para poder reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function RestablecerInterpolacionPushed(app, event)

Al pulsar el botón “Reestablecer” se eliminan los efectos producidos por la interpolación, el diezmado, la compresión o la expansión de la señal original. Además, se restablece la *frecuencia de muestreo actual* al valor de la frecuencia de muestreo de la señal cargada. Ahora, la señal con la que trabajemos será la cargada originalmente.

% Button pushed function: RestablecerInterpolacion

**function** RestablecerInterpolacionPushed**(**app**,** event**)**

%LLamamos a la funcion que reestablece la funcion original

%Como el audi se guarda en una variable a parte no la

habremos %sobreescrito. Lo unico que deberemos hacer es borrar la %pantalla

cla**(**app**.**GraficaDEP**);**

cla**(**app**.**GraficaFFToRuido**);**

cla**(**app**.**EspectrogramaModificado**);**

app**.**Factordiezmado**.**Value**=**0**;**

app**.**FactorInterpolacion**.**Value**=**0**;**

app**.**FactorCompresion**.**Value**=**0**;**

app**.**FactorExpansion**.**Value**=**0**;**

app**.**Niveles**.**Value**=**0**;**

app**.**fs**=**app**.**fsCargado**;**

**[**aud**,**app**.**fs**]=**audioread**([**app**.**path app**.**filen**]);**

app**.**a **=** aud**;**

% Guardamos la señal original en la variable global

% app.reestablecer por si necesitamos reestablecerla en un

% futuro

app**.**restablecer **=** app**.**a**;**

app**.**fsCargado **=** app**.**fs**;**

% Creamos un objeto de tipo audio con la señal original

% (reestablecida) para poder reproducirlo posteriormente

app**.**audio**=**audioplayer**(**app**.**restablecer**,**app**.**fsCargado**);**

title**(**app**.**GraficaFFToRuido**,** 'Señal Modificada'**)**

title**(**app**.**GraficaDEP**,** 'FFT Señal Modificada'**)**

app**.**TextArea\_14**.**Value**=**num2str**(**app**.**fsCargado**);**

**end**

function RestablecerRuidoPushed(app, event)

Al pulsar el botón “Reestablecer” se eliminan los efectos producidos por la introducción de ruido a la señal cargada.Además, se restablece el valor de la SNR a 1 dB. Ahora, la señal con la que trabajemos será la cargada originalmente.

% Button pushed function: RestablecerRuido

**function** RestablecerRuidoPushed**(**app**,** event**)**

% Reestablece el ruido a la funcion original

cla**(**app**.**GraficaFFToRuido**);**

cla**(**app**.**EspectrogramaModificado**);**

app**.**SNRdBKnob**.**Value**=**1**;**

app**.**a**=**app**.**restablecer**;**

app**.**audio**=**audioplayer**(**app**.**a**,**app**.**fsCargado**);**

**end**

function Restablecer\_AM(app, event)

Al pulsar el botón “Reestablecer” se eliminan los efectos producidos por la modulación o demodulación. Aunque lleva el nombre de *Restablecer\_AM* también borra los efectos producidos por la modulación FM. Ahora, la señal con la que trabajemos será la cargada originalmente.

% Button pushed function: RestablecerAM

**function** Restablecer\_AM**(**app**,** event**)**

% Reestablece los parametros originales de la señal antes

de % modular

cla**(**app**.**Demodulada**);**

cla**(**app**.**FFTDemodulada**);**

cla**(**app**.**GraficaDEP**);**

cla**(**app**.**GraficaFFToRuido**);**

cla**(**app**.**EspectrogramaModificado**);**

app**.**a**=**app**.**restablecer**;**

app**.**audio**=**audioplayer**(**app**.**restablecer**,**app**.**fsCargado**);**

**end**

function BotonSenal1Pushed(app, event)

Carga la primera señal que queremos sumar. Por ser todas las funciones siguientes iguales omitimos el código en las posteriores funciones.

**function** BotonSenal1Pushed**(**app**,** event**)**

% Cargamos la primera señal que vamos a sumar

**[**file**,**app**.**s1**,**app**.**fs1**]=**cargarsenal**(**app**,** event**);**

app**.**TextArea**.**Value **=** file**;**

**end**

function BotonSenal2ButtonPushed(app, event)

Carga la segunda señal que queremos sumar.

function BotonSenal3ButtonPushed(app, event)

Carga la tercera señal que queremos sumar.

function BotonSenal4ButtonPushed(app, event)

Carga la cuarta señal que queremos sumar.

function BotonSenal5ButtonPushed(app, event)

Carga la quita señal que queremos sumar.

function BotonSenal6ButtonPushed(app, event)

Carga la sexta señal que queremos sumar.

function BotonSumarButtonPushed(app, event)

Se encarga de sumar señales de forma recursiva.

% Button pushed function: BotonSumar

**function** BotonSumarButtonPushed**(**app**,** event**)**

% Suma las 6 muestras

suma**=**0**;**

% Vamos a trabajar a una frecuencia siempre de 44100 hz

**if** **(~**isempty**(**app**.**s1**))**

suma**=**sumar**(**suma**,**app**.**s1**,**app**.**fs1**);**

**end**

**if** **(~**isempty**(**app**.**s2**))**

suma**=**sumar**(**suma**,**app**.**s2**,** app**.**fs2**);**

**end**

**if** **(~**isempty**(**app**.**s3**))**

suma**=**sumar**(**suma**,**app**.**s3**,**app**.**fs3**);**

**end**

**if** **(~**isempty**(**app**.**s4**))**

suma**=**sumar**(**suma**,**app**.**s4**,**app**.**fs4**);**

**end**

**if** **(~**isempty**(**app**.**s5**))**

suma**=**sumar**(**suma**,**app**.**s5**,**app**.**fs5**);**

**end**

**if** **(~**isempty**(**app**.**s6**))**

suma**=**sumar**(**suma**,**app**.**s6**,**app**.**fs6**);**

**end**

**if(~**isempty**(**suma**))**

app**.**a**=** suma**;**

app**.**fsSuma**=**44100**;**

Muestreo**(**app**,**suma**,**app**.**fsSuma**,**app**.**fsSuma**,**5**);**

**end**

% Convertimos las señales a la misma frecuencia para poder

% sumarlas bien

**function** **[**suma**]=**sumar**(**suma**,**senal**,**fs**)**

**[**P**,**Q**]** **=** rat**(**44100**/**fs**);**

abs**(**P**/**Q**\***fs**-**44100**);**

% Remuestreamos

senalsuma **=** resample**(**senal**,**P**,**Q**);**

app**.**REP **=** audioplayer**(**senalsuma**,**44100**);**

app**.**fsREP**=** 44100**;**

**[**f1**,** **~]** **=** size**(**suma**);**

**[**f2**,** **~]** **=** size**(**senal**);**

**if** **(**f1**>**f2**)**

x**=**zeros**((**f1**-**f2**),**1**);**

senal **=** **[**senal**;**x**];**

**else**

x**=**zeros**((**f2**-**f1**),**1**);**

suma **=** **[**suma**;**x**];**

**end**

suma **=** senal**+**suma**;**

**end**

**end**

function MenuSumaConcatenacion(app, event)

Función que llama a la función BorraPantalla() para eliminar los elementos del menú anterior y carga los diferentes elementos del menú en que se suman y concatenan las señales escogidas.

% Menu selected function: SumayConcatenacinMenu

**function** MenuSumaConcatenacion**(**app**,** event**)**

% LLama a la función borra pantalla para eliminar los elementos % del menú principal

BorraPantalla**(**app**)**

% Carga los diferentes elementos del Menú Sumar

CargarMenuSumar**(**app**)**

**end**

function Concaten1ButtonPushed(app, event)

Carga la primera señal que queremos concatenar. Del mismo modo que sumar, solo mostramos un ejemplo de esta función.

% Image clicked function: Image5\_8

**function** Concaten1ButtonPushed**(**app**,** event**)**

% Cargamos la primera señal que vamos a concatenar

**[**file**,**app**.**sc1**,**app**.**fsc1**]=**cargarsenal**(**app**,** event**);**

app**.**TextArea\_7**.**Value **=** file**;**

**end**

function Concaten2ButtonPushed(app, event)

Carga la segunda señal que queremos concatenar.

function Concaten3ButtonPushed(app, event)

Carga la tercera señal que queremos concatenar.

function Concaten4ButtonPushed(app, event)

Carga la cuarta señal que queremos concatenar.

function Concaten5ButtonPushed(app, event)

Carga la quinta señal que queremos concatenar.

function Concaten6ButtonPushed(app, event)

Carga la sexta señal que queremos concatenar.

function ConcatenarBotonButtonPushed(app, event)

Función que se encarga de concatenar señales.

% Button pushed function: ConcatenarBoton

**function** ConcatenarBotonButtonPushed**(**app**,** event**)**

% Función que concatena señales

concat**=**0**;**

% Vamos a trabajar a una frecuencia siempre de 44100 hz

**if** **(~**isempty**(**app**.**sc1**))**

concat**=**concatenacion**(**concat**,**app**.**sc1**,**app**.**fsc1**);**

**end**

**if** **(~**isempty**(**app**.**sc2**))**

concat**=**concatenacion**(**concat**,**app**.**sc2**,** app**.**fsc2**);**

**end**

**if** **(~**isempty**(**app**.**sc3**))**

concat**=**concatenacion**(**concat**,**app**.**sc3**,**app**.**fsc3**);**

**end**

**if** **(~**isempty**(**app**.**sc4**))**

concat**=**concatenacion**(**concat**,**app**.**sc4**,**app**.**fsc4**);**

**end**

**if** **(~**isempty**(**app**.**sc5**))**

concat**=**concatenacion**(**concat**,**app**.**sc5**,**app**.**fsc5**);**

**end**

**if** **(~**isempty**(**app**.**sc6**))**

concat**=**concatenacion**(**concat**,**app**.**sc6**,**app**.**fsc6**);**

**end**

app**.**fs**=**44100**;**

Muestreo**(**app**,**concat**,**app**.**fs**,**app**.**fs**,**1**);**

app**.**fsREP**=** app**.**fs**;**

app**.**a**=**concat**;**

**function** **[**concat**]=**concatenacion**(**concat**,**senal**,**fs**)**

**[**P**,**Q**]** **=** rat**(**44100**/**fs**);**

abs**(**P**/**Q**\***fs**-**44100**);**

% Remuestreamos

senalconcat **=** resample**(**senal**,**P**,**Q**);**

concat **=[**concat**;**senalconcat**];**

app**.**REP **=** audioplayer**(**senalconcat**,**44100**);**

app**.**fsREP**=** 44100**;**

**end**

**end**

function BotonComprimirButtonPushed(app, event)

Se encarga de comprimir una señal seleccionada, escogiendo también el factor de compresión.

% Button pushed function: BotonComprimir

**function** BotonComprimirButtonPushed**(**app**,** event**)**

% Recogemos la señal que queremos comprimir y su

frecuencia de % muestreo

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si hay alguna

señal cargada

% Recogemos el factor de compresión introducido por el

% usuario

factor **=** app**.**FactorCompresion**.**Value**;**

**if** **(**factor**>**0**)**

% La nueva frecuencia será la original multiplicada

por el % factor introducido por el usuario

app**.**fsCargado**=** factor**\***app**.**fsCargado**;**

app**.**fsREP **=** app**.**fsCargado**;**

app**.**fs**=**app**.**fsREP**;**

% Llamamos a la función Muestreo() para representar

la nueva % señal en tiempo y frecuencia

Muestreo**(**app**,**app**.**a**,**app**.**fsREP**,**app**.**fs**,**1**);**

title**(**app**.**GraficaFFToRuido**,** 'Señal Comprimida'**);**

title**(**app**.**GraficaDEP**,** 'FFT Señal Comprimida'**);**

**else**

warndlg**(**"Debe introducir valores positivos"**,**"ATENCION"**)**

**end**

% Si no hay niguna señal cargada salta un error

**else**

warndlg**(**"Debe de cargar un fichero para poder reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function BotonExpandirButtonPushed(app, event)

Se encarga de expandir una señal seleccionada, escogiendo también el factor de compresión.

% Button pushed function: BotonExpandir

**function** BotonExpandirButtonPushed**(**app**,** event**)**

% Recogemos la señal que queremos comprimir y su

frecuencia de % muestreo

**if(**CompruebaCargada**(**app**)==**0**)**% Comprobamos si hay alguna

señal cargada % Recogemos el factor de compresión introducido por el% usuario

factor **=** app**.**FactorExpansion**.**Value**;**

**if** **(**factor**>**0**)**

% La nueva frecuencia será la original dividida por el %

factor introducido por el usuario

app**.**fsCargado**=** app**.**fsCargado**/**factor**;**

app**.**fsREP **=** app**.**fsCargado**;**

app**.**fs**=**app**.**fsREP**;**

% Creamos un objeto de tipo audio para reproducir el

audio % expandido

app**.**audio **=** audioplayer**(**app**.**a**,**app**.**fsREP**);**

% Llamamos a la función Muestreo() para representar

la nueva % señal en tiempo y frecuencia

Muestreo**(**app**,**app**.**a**,**app**.**fsREP**,**app**.**fs**,**1**);**

title**(**app**.**GraficaFFToRuido**,** 'Señal Expandida'**);**

title**(**app**.**GraficaDEP**,** 'FFT Señal Expandida'**);**

**else**

warndlg**(**"Debe introducir valores

positivos"**,**"ATENCION"**)**

**end**

% Si no hay niguna señal cargada salta un error

**else**

warndlg**(**"Debe de cargar un fichero para poder

reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function CuantificarButtonPushed(app, event)

Realiza la cuantificación de un audio seleccionado a través del número de niveles deseado.

**function** CuantificarButtonPushed**(**app**,** event**)**

% Función que nos permite cuantificar una señal

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si hay alguna señal cargada

% Llamamos la función para calcular la FFT y la

% representamos

**[**TF**,**f**]=**Transformada**(**app**,**app**.**a**,**app**.**fsCargado**);**

plot**(**app**.**GraficaFFToRuido**,**f**,**abs**(**TF**));**

% Convertimos la señal estéreo a mono

y **=** sum**(**app**.**a**,** 2**)** **/** size**(**app**.**a**,** 2**);**

dt **=** 1**/**app**.**fsCargado**;**

t **=** 0**:**dt**:(**length**(**y**)\***dt**)-**dt

% Recogemos el número de niveles que deseamos para cuantificar

nr **=** app**.**Niveles**.**Value**;**

**if** **(**nr**<=**0**)**

warndlg**(**"Debe introducir valores positivos"**,**"ATENCION"**)**

**end**

delt **=** 2**/**nr**;**

% Ajustamos los valores a los niveles que tenemos

v **=** max**(** max**(**y**),-**min**(**y**)** **);**

y **=** y**/**v**\***0.999999**;**

% Cuantificamos

y **=** delt**\***floor**((**y**+**1**)/**delt**)+**delt**/**2**-**1**;**

app**.**a**=**y**;**

app**.**audio**=**audioplayer**(**y**,**app**.**fsCargado**);**

% Representamos la señal cuantificada

plot**(**app**.**GraficaFFToRuido**,**t**,**y**);**

xlabel**(**app**.**Espectograma**,** 'Frecuencia (Hz)'**)**

ylabel**(**app**.**Espectograma**,** 'Amplitud'**)**

xlabel**(**app**.**GraficaFFToRuido**,** 'Tiempo (s)'**)**

% Calculamos la FFT de la señal cuantificada y la

% representamos

**[**TF**,**f**]=**Transformada**(**app**,**y**,**app**.**fsCargado**);**

plot**(**app**.**GraficaDEP**,**f**,**abs**(**TF**));**

% LLamamos a la función para calcular el espectograma y lo % representamos

**[**S**,**F**,**T**]** **=** spectrogram**(**y**,**800**,[],[],**app**.**fsCargado**);**

imagesc**(**app**.**EspectrogramaModificado**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**EspectrogramaModificado**.**YDir **=** 'normal'**;**

% Establecemos los limites de las gráficas

ylim**(**app**.**EspectrogramaOriginal**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**EspectrogramaOriginal**,** **[**min**(**T**),** max**(**T**)]);**

% Si no hay ningún audio cargado salta un error

**else**

warndlg**(**"Debe de cargar un fichero para poder reproducirlo"**,**"ATENCION"**)**

**end**

**end**

function MenuEstudio(app, event)

Función que llama a la función BorraPantalla() para eliminar los elementos del menú anterior y carga los diferentes elementos del menú en el que se realizan las operaciones de diezmado, interpolación, compresión, expansión, cuantificación.

**function** MenuEstudio**(**app**,** event**)**

% LLama a la función borra pantalla para eliminar los elementos % del menú principal

BorraPantalla**(**app**)**

% Carga los diferentes elementos del menú Estudio

CargarMenuEstudio**(**app**)**

**end**

function RestablecerMenuPrincipalButtonPushed(app, event)

Restablece los valores por defecto del panel de generación de ondas deterministas a parte de limpiar las gráficas. Ahora, la señal con la que trabajemos será la cargada originalmente.

**function** RestablecerMenuPrincipalButtonPushed**(**app**,** event**)**

% Función que reestablece la señal original en el menú principal

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si hay alguna señal cargada

app**.**a**=**app**.**restablecer**;**

app**.**audio**=**audioplayer**(**app**.**restablecer**,**app**.**fsCargado**);**

Representar\_Senal**(**app**);**

**end**

% Limpiamos las gráficas del menú y reestablecemos a 0 todos los % parámetros

app**.**NombreLabel**.**Text **=** 'Cargue una Señal'**;**

app**.**MuestrastotalesLabel**.**Text **=** 'Cargue una Señal'**;**

app**.**FrecuenciademuestreoLabel**.**Text **=** 'Cargue una Señal'**;**

app**.**PeriodoLabel\_2**.**Text **=** 'Cargue una Señal'**;**

app**.**EnergaLabel**.**Text **=** 'Cargue una Señal'**;**

app**.**PotenciamediaLabel\_2**.**Text **=** 'Cargue una Señal'**;**

cla**(**app**.**GraficaTiempo**);**

cla**(**app**.**GraficaDEP**);**

cla**(**app**.**Espectograma**);**

cla**(**app**.**GraficaFFToRuido**);**

app**.**Amplitud**.**Value**=**0**;**

app**.**Fs**.**Value**=**0**;**

app**.**Frec**.**Value**=**0**;**

app**.**Duracion**.**Value**=**0**;**

app**.**LabelEnergia**.**Visible**=**'off'**;**

app**.**LabelPotencia**.**Visible**=**'off'**;**

app**.**LabelPeriodo**.**Visible**=**'off'**;**

app**.**LabelEnergia**.**Text**=**'Label'**;**

app**.**LabelPotencia**.**Text**=**'Label'**;**

app**.**LabelPeriodo**.**Text**=**'Label'**;**

title**(**app**.**GraficaDEP**,**'DEP'**);**

title**(**app**.**Espectograma**,**'Espectograma'**);**

app**.**GrabacinentiemporealEditField**.**Value**=**0**;**

**end**

function RestablecerFiltradoButtonPushed(app, event)

Restablece los valores por defecto del filtro a parte de limpiar las gráficas de los valores del filtrado. Ahora, la señal con la que trabajemos será la cargada originalmente.

**function** RestablecerFiltradoButtonPushed**(**app**,** event**)**

% Función que reestablece la señal original en el menú de

% filtrado

% Limpiamos las gráficas

cla**(**app**.**GraficaDEP**);**

cla**(**app**.**GraficaFFToRuido**);**

app**.**ocultarFiltro**.**Visible **=** 'off'**;**

cla**(**app**.**EspectrogramaModificado**);**

% El audio cargado pasará a ser el audio reestablecido

app**.**a**=**app**.**restablecer**;**

app**.**audio**=**audioplayer**(**app**.**restablecer**,**app**.**fsCargado**);**

%Rstablecemos los valores por defecto

app**.**FrecuenciadepasofinalEditField**.**Value**=**0**;**

app**.**FrecuenciadeparadafinalEditField**.**Value**=**0**;**

app**.**AtenuacinenbandadeparadadBEditField**.**Value**=**0**;**

app**.**AtenuacinenbandadepasodBEditField**.**Value**=**0**;**

app**.**FrecuenciadeparadainicialEditField**.**Value**=**0**;**

app**.**FrecuenciadepasoinicialEditField**.**Value**=**0**;**

**end**

function RestablecerSuma(app, event)

Limpia las gráficas de los resultados de la suma.

**function** RestablecerSuma**(**app**,** event**)**

% Función que reestablece la señal original en el menú

% suma/concatenación

% Limpiamos las gráficas

cla**(**app**.**GraficaTiempo**);**

cla**(**app**.**Espectograma**);**

cla**(**app**.**EspectrogramaOriginal**);**

app**.**TextArea**.**Value **=**'Señal 1'**;**

app**.**TextArea\_2**.**Value **=**'Señal 2'**;**

app**.**TextArea\_3**.**Value **=**'Señal 3'**;**

app**.**TextArea\_4**.**Value **=**'Señal 4'**;**

app**.**TextArea\_5**.**Value **=**'Señal 5'**;**

app**.**TextArea\_6**.**Value **=**'Señal 6'**;**

**end**

function RestablecerConcatenacionButtonPushed(app, event)

Limpia las gráficas de los resultados de concatenación.

**function** RestablecerConcatenacionButtonPushed**(**app**,** event**)**

% Función que reestablece la señal original en el menú

% concatenación

% Limpiamos las gráficas

cla**(**app**.**GraficaDEP**);**

cla**(**app**.**GraficaFFToRuido**);**

cla**(**app**.**EspectrogramaModificado**);**

app**.**TextArea\_7**.**Value **=** 'Señal 1'**;**

app**.**TextArea\_8**.**Value **=** 'Señal 2'**;**

app**.**TextArea\_9**.**Value **=** 'Señal 3'**;**

app**.**TextArea\_10**.**Value **=**'Señal 4'**;**

app**.**TextArea\_11**.**Value **=**'Señal 5'**;**

app**.**TextArea\_12**.**Value **=**'Señal 6'**;**

**end**

function Guardar(app, event)

El botón “Guardar” permite guardar la señal modificada que deseemos. Por ejemplo, podemos guardar una señal modulada, diezmada, expandida…

**function** Guardar**(**app**,** event**)**

% Esta función nos permite guardar una señal

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si hay alguna señal cargada

audiograbado**=**app**.**a**;**

**[**s**,~]=**uiputfile**(**'\*.wav'**,**'Guardar señal de audio'**);** % Escogemos donde guardar el audio

**if(**s**~=**0**)**

audiowrite**(**s**,**audiograbado**,**44100**);**

**end**

% Si no hay ninguna señal cargada salta un mensaje de error

**else**

warndlg**(**"Debe de cargar un fichero para poder guardarlo"**,**"ATENCION"**)**

**end**

**end**

function ResultadosDemodulacinMenuSelected(app, event)

Esta función carga el panel en el que se encuentran las gráficas encargadas de mostrar los resultados de la modulación, ya sea AM o FM.

**function** ResultadosDemodulacinMenuSelected**(**app**,** event**)**

% El panel de resultados de demodulación se hace visible

app**.**DemodulacinPanel**.**Visible**=**'on'**;**

**end**

function DelayButtonPushed(app, event)

Al pulsar sobre el botón “Delay” podemos observar el clásico efecto de eco sonoro. Consiste en la multiplicación y retraso modulado de una señal sonora.

**function** DelayButtonPushed**(**app**,** event**)**

% Esta función nos permite añadir el efecto de delay a un audio

**if(**CompruebaCargada**(**app**)==**0**)** % Comprueba que haya un audio cargado

title**(**app**.**GraficaFFToRuido**,** 'Señal con Delay'**)**

title**(**app**.**GraficaDEP**,** 'FFT Señal con Delay'**)**

% Recogemos la señal y su frecuencia

**[**x**,**fsdelay**]=**audioread**([**app**.**path app**.**filen**]);**

% Fijamos los parámetros necesarios para realizar el efecto % delay

delay **=** 0.5**;** % Retardo

alpha **=** 0.65**;**

D **=** delay**\***fsdelay**;**

y **=** zeros**(**size**(**x**));**

b **=** size**(**x**);**

y**(**1**:**D**)** **=** x**(**1**:**D**);**

tiempo **=** 0**:**1**/**fsdelay**:(**b**-**1**)/**fsdelay**;**

% Recorremos todo el audio añadiendo el efecto

**for** i**=**D**+**1**:**length**(**x**)**

y**(**i**)** **=** x**(**i**)** **+** alpha**\***x**(**i**-**D**);**

**end**

app**.**a**=**y**;**

% Creamos un objeto audio para poder reproducirlo

app**.**audio**=**audioplayer**(**y**,**fsdelay**);**

% Representamos

plot**(**app**.**GraficaFFToRuido**,**tiempo**,**y**);**

% Calculamos la FFT y la representamos

**[**tf**,**f**]** **=** Transformada**(**app**,**y**,**fsdelay**);**

title**(**app**.**GraficaDEP**,** 'FFT Señal con Delay'**)**

xlabel**(**app**.**GraficaDEP**,** 'Frecuencia (Hz)'**)**

ylabel**(**app**.**GraficaDEP**,** 'Amplitud'**)**

plot**(**app**.**GraficaDEP**,**f**,**tf**);** % LLamamos a la función para calcular el espectograma y lo % representamos

**[**S**,**F**,**T**]** **=** spectrogram**(**y**(:,**1**),**800**,[],[],**fsdelay**);**

imagesc**(**app**.**EspectrogramaModificado**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**EspectrogramaModificado**.**YDir **=** 'normal'**;**

% Establecemos los limites de las gráficas

ylim**(**app**.**EspectrogramaModificado**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**EspectrogramaModificado**,** **[**min**(**T**),** max**(**T**)]);**

% Si no hay ninguna señal cargada salta un mensaje de

error

**else**

warndlg**(**"Debe de cargar un fichero para poder añadir efectos"**,**"ATENCION"**)**

**end**

**end**

function MetlicoButtonPushed(app, event)

Al pulsar el botón “Metálico” somos capaces de añadir un efecto de metal a nuestro audio seleccionado.

**function** MetlicoButtonPushed**(**app**,** event**)**

% Función que permite añadir el efecto metalizado a un audio

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si hay algún

audio cargado

% Leemos el audio

**[**x**,**fsMetal**]=**audioread**([**app**.**path app**.**filen**]);**

title**(**app**.**GraficaDEP**,**'FFT Señal Metálica'**)**

title**(**app**.**GraficaFFToRuido**,**'Señal Metálica'**)**

% Definimos los parámetros necesarios para realizar el

% efecto

x **=** x**(:,**1**);**

delay **=** 15**;**

range **=** 12**;**

sweepfreq **=** 0.5**;**

b **=** length**(**x**)-**delay**-**range**;**

y **=** zeros**(**1**,**b**);**

% Recorremos el audio para añadir el efecto

**for** i **=** 1**:**length**(**x**)-**delay**-**range

y**(**i**)=**x**(**i**)+**x**(**i**+**delay**+**round**(**range**\***sin**(**2**\***pi**\***i**\***sweepfreq**/**fsMetal**)));**

**end**

app**.**a**=**y**;**

app**.**audio**=**audioplayer**(**app**.**a**,**fsMetal**);**

tiempo **=** 0**:**1**/**app**.**fsCargado**:(**b**-**1**)/**app**.**fsCargado**;**

plot**(**app**.**Espectograma**,**tiempo**,**y**);**

**[**tf**,**f**]** **=** Transformada**(**app**,**y**,**app**.**fs**);**

plot**(**app**.**GraficaDEP**,**f**,**tf**);**

% Si no hay ninguna señal cargada salta un mensaje de error

**else**

warndlg**(**"Debe de cargar un fichero para poder añadir efectos"**,**"ATENCION"**)**

**end**

**end**

function InvertirEjeTemporal(app, event)

Al pulsar el botón “Invertir” invertimos la pista de audio escogida.

**function** InvertirEjeTemporal**(**app**,** event**)**

% Mostramos las gráficas que vamos a necesitar

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si hay alguna señal cargada

b**=**length**(**app**.**a**);**

y **=** app**.**a**;**

app**.**fs **=** app**.**fsCargado**;**

tiempo **=** 0**:**1**/**app**.**fs**:(**b**-**1**)/**app**.**fs**;**

y1 **=** zeros**(**1**,**b**);**

%Inversion de señal, recorremos el audio desde el final hasta el principio con un bucle for

**for** i**=**1**:(**b**)**

%Guardamos nuestra señal invertida en otra variable llamada

%y1

y1**(**i**)=**y**(**b**-**i**+**1**);**

**end**

% Reproducimos y representamos la señal invertida

app**.**a**=**y1**;**

% Creamos un objeto de audio para reproducirlo

app**.**audio**=**audioplayer**(**y1**,**app**.**fsCargado**);**

plot**(**app**.**GraficaFFToRuido**,**tiempo**,**y1**);**

% Calculamos la transformada de la señal invertida y la

% repsentamos

**[**tf**,**f**]** **=** Transformada**(**app**,**y1**,**app**.**fs**);**

title**(**app**.**GraficaDEP**,** 'FFT Señal invertida'**)**

xlabel**(**app**.**GraficaDEP**,** 'Frecuencia (Hz)'**)**

ylabel**(**app**.**GraficaDEP**,** 'Amplitud'**)**

plot**(**app**.**GraficaDEP**,**f**,**tf**);**

% LLamamos a la función para calcular el espectograma y lo

% representamos

**[**S**,**F**,**T**]** **=** spectrogram**(**y1**,**800**,[],[],**app**.**fs**);**

imagesc**(**app**.**EspectrogramaModificado**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**EspectrogramaModificado**.**YDir **=** 'normal'**;**

% Establecemos los limites de las gráficas

ylim**(**app**.**EspectrogramaModificado**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**EspectrogramaModificado**,** **[**min**(**T**),** max**(**T**)]);**

% Si no hay ningún fichero cargado salta un mensaje de error

**else**

warndlg**(**"Debe de cargar un fichero para poder añadir efectos"**,**"ATENCION"**)**

**end**

**end**

function ReverbButtonPushed(app, event)

Al pulsar sobre el botón “Reverb” podemos apreciar el fenómeno de la reverberación que consiste en una ligera permanencia del sonido una vez que la fuente original ha dejado de emitirlo.

**function** ReverbButtonPushed**(**app**,** event**)**

% Función que nos permite añadir el efecto Reverb a un audio

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si hay algún audio cargado

**[**x**,**fsReverb**]=**audioread**([**app**.**path app**.**filen**]);**

title**(**app**.**GraficaDEP**,** 'FFT Señal con Reverb'**);**

title**(**app**.**GraficaFFToRuido**,** 'Señal con Reverb'**);**

% Calculamos la transformada de la señal original y la

% representamos

**[**TF**,**f**]=**Transformada**(**app**,**app**.**a**,**app**.**fsCargado**);**

plot**(**app**.**GraficaFFToRuido**,**f**,**abs**(**TF**))**

reverb **=** reverberator**;**

audioWithReverb **=** reverb**(**x**);**

app**.**a**=**audioWithReverb**;**

% Llamamos a la función Muetreo() para calcular y

% representar el audio con Reverb en tiempo y en frecuencia

Muestreo**(**app**,**audioWithReverb**,**fsReverb**,**fsReverb**,**1**)**

% Si no hay ningún fichero cargado salta un mensaje de error

**else**

warndlg**(**"Debe de cargar un fichero para poder añadir efectos"**,**"ATENCION"**)**

**end**

**end**

function PitchShiftingButtonPushed(app, event)

Al pulsar el botón Pitch-Shifting podemos subir o bajar el tono original de un audio. El número de semitonos que queremos subir o bajar podemos introducirlo por pantalla.

**function** PitchShiftingButtonPushed**(**app**,** event**)**

% Esta función nos permite añadir el efecto Pitch-Shifting a un

% audio

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si hay alguna señal cargada

**[**x**,**fsShift**]=**audioread**([**app**.**path app**.**filen**]);**

% Representamos la fft de la señal original

title**(**app**.**GraficaDEP**,** 'FFT Señal con Pitch-Shifting'**);**

title**(**app**.**GraficaFFToRuido**,** 'Señal con Pitch-Shifting'**);**

**[**TF**,**f**]=**Transformada**(**app**,**app**.**a**,**app**.**fsCargado**);**

plot**(**app**.**GraficaFFToRuido**,**f**,**abs**(**TF**))**

% Recogemos el número de semitonos que el usuario desea

nsemitones **=** app**.**SemitonosEditField**.**Value**;**

audioOut **=** shiftPitch**(**x**,**nsemitones**);**

app**.**a**=**audioOut**;**

% Llamamos a la función Muesreo() para calcular y

% representar la señal con el efecto en tiempo y en

% frecuencia

Muestreo**(**app**,**audioOut**,**fsShift**,**fsShift**,**1**)**

% Si no hay ninguna señal cargada salta un mensaje de error

**else**

warndlg**(**"Debe de cargar un fichero para poder añadir efectos"**,**"ATENCION"**)**

**end**

**end**

function GrabarTiempoReal(app, event)

Función que permite grabar y graficar en tiempo real al pulsar el icono del micrófono en el que se lee “ON AIR”.

**function** GrabarTiempoReal**(**app**,** event**)**

% Función para grabar y graficar en tiempo real

app**.**audioRecord **=** audiorecorder**(**44100**,**16**,**2**);**

recorder **=** audiorecorder**(**44100**,**16**,**2**);**

app**.**fsCargado**=**44100**;**

app**.**fs**=**44100**;**

% Recogemos el tiempo que el usuario desea visualizar en tiempo% real

tiempo **=** app**.**GrabacinentiemporealEditField**.**Value**;**

**if(**tiempo**>**0**)**

recorder**.**record**(**tiempo**);**

**while** recorder**.**isrecording**()**

pause**(**0.1**);**

% Convertimos señales estereo a mono

app**.**audioGrabado **=** sum**(**recorder**.**getaudiodata**(),** 2**)** **/** size**(**recorder**.**getaudiodata**(),** 2**);**

dt **=** 1**/**app**.**fs**;**

t **=** 0**:**dt**:(**length**(**app**.**audioGrabado**)\***dt**)-**dt**;**

plot**(**app**.**GraficaTiempo**,**t**,**app**.**audioGrabado**);**

% Llamamos a la función para calcular la transformada y la representamos

**[**TF**,**f**]=**Transformada**(**app**,**app**.**audioGrabado**,**app**.**fs**);**

plot**(**app**.**GraficaFFToRuido**,**f**,**abs**(**TF**));**

% Calculamos y representamos la densidad espectral de potencia

**[**tf**,**f**]** **=** DEP**(**app**,**app**.**audioGrabado**,**app**.**fs**);**

plot**(**app**.**GraficaDEP**,**f**,**abs**(**tf**));**

% LLamamos a la función para calcular el espectograma y lo

% representamos

**[**S**,**F**,**T**]** **=** spectrogram**(**app**.**audioGrabado**,**800**,[],[],**app**.**fsCargado**);**

imagesc**(**app**.**Espectograma**,**T**,**F**,**log**(**abs**(**S**)))**

app**.**Espectograma**.**YDir **=** 'normal'**;**

% Establecemos los límites de las gráficas

ylim**(**app**.**Espectograma**,** **[**min**(**F**),** max**(**F**)]);**

xlim**(**app**.**Espectograma**,** **[**min**(**T**),** max**(**T**)]);**

**end**

app**.**audioRecord **=** recorder**;**

% Se guarda lo grabado

app**.**audio**=**audioplayer**(**app**.**audioRecord**);**

app**.**audioGrabado **=** getaudiodata**(**app**.**audioRecord**);**

app**.**a**=**app**.**audioGrabado**;**

app**.**restablecer**=**app**.**audioGrabado**;**

% Si no hay ningún audio cargado salta un mensaje de error

**else**

warndlg**(**"Debe introducir un tiempo de grabación mayor que cero"**,**"ATENCION"**)**

**end**

**end**

function BotonAyuda (app,event)

Redirecciona al usuarios a una playlist de youtube en donde podrá visualizar diferentes tutoriales.

**function** BotonAyuda**(**app**,** event**)**

% Callback que te redirige al canal de youtube en el que podemos % encontrar todos los tutoriales

url**=**'https://www.youtube.com/playlist?list=PLk4s5VRPewKi3mipGOHDRjz07K-GS1wI2'**;** %url que te redirige al canal de youtube en el que estan cargados los tutoriales

web**(**url**);**

**end**

function ExportarAudioPushed(app, event)

Función que te exporta los parámetros principales de la señal cargada a un archivo de texto.

**function** ExportarAudioPushed**(**app**,** event**)**

% Función que te exporta los parámetros principales de la señal

cargada a un % archivo de texto

**if(**CompruebaCargada**(**app**)==**0**)** % Comprobamos si hay alguna señal cargada % Abrimos el fichero que vamos a generar

**[**s**,**ruta**]=**uiputfile**(**'\*.txt'**,**'Guardar informe'**);**

fichero **=** fopen**(**strcat**(**ruta**,** s**),**'w'**);**

fprintf**(**fichero**,**'Para el archivo de audio %s\n'**,**

app**.**filen**);**

fprintf**(**fichero**,**' %s \n '**,**app**.**MuestrastotalesLabel**.**Text**);**

fprintf**(**fichero**,**' %s \n '**,**app**.**FrecuenciademuestreoLabel**.**Text**);**

fprintf**(**fichero**,**' %s \n'**,**app**.**PeriodoLabel\_2**.**Text**);**

fprintf**(**fichero**,**' %s \n'**,**app**.**EnergaLabel**.**Text**);**

fprintf**(**fichero**,**' %s \n'**,**app**.**PotenciamediaLabel\_2**.**Text**);**

fclose**(**fichero**);**

% Si no habia una señal cargada entonces salta un error

**else**

warndlg**(**"Debe de cargar un fichero para poder exportarlo"**,**"ATENCION"**)**

**end**

**end**