WAVEVISUALIZER

MEMORIA DE PROGRAMACIÓN

29 DE NOVIEMBRE DE 2018

MARIO ACEBES CALZADA - GONZALO PARDO VILLALIBRE
ETSIT – UNIVERSIDAD DE VALLADOLID

ÍNDICE

MEMORIA DE PROGRAMACIÓN

A continuación, se representa una breve explicación de la estructura de los ficheros de código que sustentan el programa **WAVEVISUALIZER.**

La división de funciones se ha dividido principalmente en cinco ficheros que corresponden con las cinco interfaces de las que consta el programa.

PROYECTOV2.M

Fichero que controla la interfaz principal del programa, contiene la mayor parte de las funciones y desde el se llama al resto de ficheros. A continuación, se adjunta una breve descripción de las funciones o los conjuntos de funciones.

```
function varargout = Proyectov2(varargin)
function Proyectov2_OpeningFcn(hObject, eventdata, handles, varargin)
function varargout = Proyectov2_OutputFcn(hObject, eventdata, handles)
```

Se trata de las funciones que nos sirven para cargar la interfaz GUIDE del sistema, en estas funciones se declaran también las constantes que vamos a utilizar a lo largo del programa.

```
handles.thereis=zeros(7,2)
handles.posicion=0
handles.numHerramientas=0;
```

Sirven para identificar qué herramienta de las 7 disponibles está siendo usada y en qué parte de la interfaz.

```
handles.escala='V';
```

Identifica la escala de representación.

```
handles.audio=0;
handles.fs=0;
```

Identifican las variables con las que vamos a trabajar a lo largo el programa.

```
handles.save=0;
handles.savefs=0;
```

Identifican variables importantes que se puede utilizar con motivos de backup.

```
function Help_Callback(hObject, eventdata, handles)
```

Cada vez que se pulsa el botón “help” esta función se encarga de redirigir al usuario a la página web con el canal de YouTube del programa.

```
function abrir_Callback(hObject, eventdata, handles)
```

Esta función se encarga de abrir los archivos que contienen las señales de manera correcta, de tal forma que no se produzca ningún tipo de error más tarde.

```
function MuestreoTiempo(handles)
```

Se trata de una función bastante socorrida y utilizada con frecuencia en el programa, su función en mostrar en el display en tiempo la señal principal del sistema.

```
function MuestreoFrec(handles)
```

Se trata de una función bastante socorrida y utilizada con frecuencia en el programa, su función en mostrar en el display en frecuencia la señal principal del sistema.

```
function Espec_Callback(hObject, eventdata, handles)
```

Se encarga de llamar a la función de MuestreoFrecuencia y además de gestionar si aparece o no el botón de mostar en dB, puesto que en el espectrograma nunca tendremos esta opción.

```
function [tf,f] = transformada(x,fs,handles)
```
Calcula la transformada rápida de Fourier de la señal de audio que se le introduce como parámetro.

```
function DEP(x,fs,handles)
```
Se encarga de calcular la densidad espectral de la función de que le llega como parámetro.

```
function FFTdB_Callback(hObject, eventdata, handles)
```
Este botón gestiona la escala de representación del dominio espectral (V o dB).

```
function[herramientaelimino,numHerramientas,posicion,s]=actualizarposicion(hObject,handles,herr)
```
Esta función es una de las más complejas del programa y se encarga de otorgar a cada nueva herramienta la posición de las tres que les corresponde. Si la herramienta ya está desplegada entonces no la cambia de posición.

```
function RepParam_Callback(hObject, eventdata, handles)
```
Es la función que coordina todos los eventos de la herramienta "Parámetros de Reproducción", en esta función básicamente se inicializan los parámetros más importantes además de llamar a la función "actualizaposición" para determinar la posición de la herramienta.

```
function sumador_Callback(hObject, eventdata, handles)
```
Es la función que coordina todos los eventos de la herramienta "Sumador", en esta función básicamente se inicializan los parámetros más importantes además de llamar a la función "actualizaposición" para determinar la posición de la herramienta.

```
function concatenador_Callback(hObject, eventdata, handles)
```
Es la función que coordina todos los eventos de la herramienta "Concatenador", en esta función básicamente se inicializan los parámetros más importantes además de llamar a la función "actualizaposición" para determinar la posición de la herramienta.

```
function Datos_Callback(hObject, eventdata, handles)
```
Es la función que coordina todos los eventos de la herramienta "Datos", en esta función básicamente se inicializan los parámetros más importantes además de llamar a la función "actualizaposición" para determinar la posición de la herramienta.

```
function interp_Callback(hObject, eventdata, handles)
```
Es la función que coordina todos los eventos de la herramienta "Interpolado y diezmado", en esta función básicamente se inicializan los parámetros más importantes además de llamar a la función "actualizaposición" para determinar la posición de la herramienta

```
function modular_Callback(hObject, eventdata, handles)
```
Es la función que coordina todos los eventos de la herramienta "Modular", en esta función básicamente se inicializan los parámetros más importantes además de llamar a la función "actualizaposición" para determinar la posición de la herramienta

```
function demodular_Callback(hObject, eventdata, handles)
```
Es la función que coordina todos los eventos de la herramienta "Demodular", en esta función básicamente se inicializan los parámetros más importantes además de llamar a la función "actualizaposición" para determinar la posición de la herramienta

```
function FiltradoSenales_Callback(hObject, eventdata, handles)
```
Llama al fichero que carga la interfaz de filtrado de señales.

```
function sumadorsX_Callback(hObject, eventdata, handles)
```
Se encarga de manejar cada una de las señales que va a ser sumada (carga de señales).

`function sumar_Callback(hObject, eventdata, handles)`
Se encarga de evaluar qué señales están cargadas de las opciones de suma y coordinar todos los eventos involucrados (guardado, muestreo de la solución).

`function [suma]=sumar(suma,senal,fs)`
Suma señales de manera recursiva, para ello se sirve de un resampling a una frecuencia común entre todas.

`function [file,audio,fs]=cargarsenal(hObject,handles)`
Se encarga de cargar las señales de audio que se van a usar para la suma.

`function PVPlay_Callback(hObject, eventdata,handles)`
Reoroduce la pista de la señal cargada.

`function PVResume_Callback(hObject, eventdata, handles)`
Continua con la reproducción de una pista tras una pausa momentánea.

`function PVPause_Callback(hObject, eventdata, handles)`
Pausa la reproducción de la canción.

`function PV2_Callback(hObject, eventdata, handles)`
Se encarga de cambiar el sentido de la señal.

`function PV1_Callback(hObject, eventdata, handles)`
Aumenta la velocidad de reproducción de la señal.

`function save_Callback(hObject, eventdata, handles)`
Se encarga de guardar la señal.

`function customwave_Callback(hObject, eventdata, handles)`
Llama al fichero OndaPersonalizada.

`function informe_Callback(hObject, eventdata, handles)`
Genera un informe con los datos de la canción, incluyendo potencia y energía.

`function [concat]=concatenacion(concat,senal,fs)`
Realiza la concatenación de las señales recursivamente. Para ello se sirve de un resampling para poder.

`function concatenar_Callback(hObject, eventdata, handles)`
Se encarga de evaluar qué señales están cargadas y cuales no y coordinar todas las operaciones de la concatenación.

`function concatsX_Callback(hObject, eventdata, handles)`
Se encarga de cargar de manera segura cada una de las señales que se van a concatenar.

`function dointerpol_Callback(hObject, eventdata, handles)`
Toma todos los parámetros necesarios para hacer la interpolación y la realiza.

`function dodiezmado_Callback(hObject, eventdata, handles)`
Toma todos los parámetros necesarios para hacer el diezmado y lo realiza.

`function am_Callback(hObject, eventdata, handles)`
Se encarga de hacer la modulación am de la señal seleccionada para una portadora preseleccionada por el programador.

`function fm_Callback(hObject, eventdata, handles)`
Realiza la modulación fm de la señal seleccionada para una portadora preseleccionada.

`function addnoise_Callback(hObject, eventdata, handles)`
Añade efecto de un ruido leve para observar qué significa.

`function domodulation_Callback(hObject, eventdata, handles)`
Muestra por pantalla el resultado de la modulación y modifica la señal principal.

`function selecsenal_Callback(hObject, eventdata, handles)`
Se encarga de cargar la selección de la señal a modular.

`function demodam_Callback(hObject, eventdata, handles)`
Se encarga de hacer la demodulación para la portadora fm que concuerda con aquella seleccionada en el apartado de modular.

`function demodfm_Callback(hObject, eventdata, handles)`
Se encarga de hacer la demodulación para la portadora fm que concuerda con aquella seleccionada en el apartado de modular.

`function dodemodular_Callback(hObject, eventdata, handles)`
Muestra por pantalla el resultado de la demodulación y modifica la señal.

`function selecsnaldemod_Callback(hObject, eventdata, handles)`
Se encarga de cargar la selección de la señal a demodular.

OndaDeterminista.M

Este fichero controla una de las interfaces secundarias del programa, concretamente aquella que se abre cuando deseamos crear una nueva señal determinista.

```
function varargout = OndaDeterminista(varargin)
function OndaDeterminista_OpeningFcn(hObject, eventdata, handles, varargin)
function varargout = OndaDeterminista_OutputFcn(hObject, eventdata, handles)
```
Se trata de las funciones que nos sirven para cargar la interfaz GUIDE del fichero OndaDeterminista.

`function format_Callback(hObject, eventdata, handles)`
Se encarga de reconocer el formato de la señal que estamos diseñando.

`function [pista,fs]=dibujando_tiempo_Callback(hObject, eventdata, handles)`
Coordina la señal que se va a diseñar en tiempo.

`function [pista,fs]=buildcos(hObject,handles)`
Función que se encarga de diseñar el coseno.

`function [pista,fs]=buildsin(hObject,handles)`
Función que se encarga de diseñar el seno.

`function [pista,fs]=buildrectan(hObject,handles)`
Función que se encarga de diseñar la señal rectangular

`function [pista,fs]=buildtriang(hObject,handles)`
Función que se encarga de diseñar la señal en diente de sierra.

`function [pista,fs]= buildpulso(hObject,handles)`
Función que se encarga de diseñar un pulso discreto.

`function [pista,fs]=buildescalondisc(hObject,handles)`
Función que se encarga de diseñar una función escalón discreta.

```
function [pista,fs]=buildescalon(hObject,handles).
```
Función que se encarga de diseñar una función escalón. En realidad, como en Matlab a la hora de representar una señal lo que estamos haciendo es muestrear de forma discreta, ambos escalones van a tener la misma "información" aunque se representen en diseño de forma diferentes (uno con plot y otro con stem).

```
function [pista,fs]=buildsinc(hObject,handles)
```
Función que se encarga de diseñar la sinc centrada en el origen.

```
function changeInterface(handles,form)
```
Función que se encarga de cambiar la apariencia del apartado de parámetros para cada tipo de señal.

```
function dt_Callback(hObject, eventdata, handles)
```
Función que se encarga de actualizar la memoria del programa y de mostrar las señales creadas por pantalla.

```
function [tf,f] = transformada(x,fs,handles)
```
Función que se encarga de calcular la transformada de Fourier

```
function save_ClickedCallback(hObject, eventdata, handles)
```
Función que se encarga de guardar la señal creada.

Record.M

Este fichero controla una de las interfaces secundarias del programa, concretamente aquella que se abre cuando deseamos grabar una nueva señal determinista.

```
function varargout = Record(varargin)
function Record_OpeningFcn(hObject, eventdata, handles, varargin)
function varargout = Record_OutputFcn(hObject, eventdata, handles)
```
Se trata de las funciones que nos sirven para cargar la interfaz GUIDE del fichero Record.m.

```
function start_Callback(hObject, eventdata, handles)
```
Funcion que se encarga de empezar la grabación.

```
function finalizar_Callback(hObject, eventdata, handles)
```
Funcion que se encarga de finalizar la grabación.

```
function cancelar_Callback(hObject, eventdata, handles)
```
Funcion que se encarga de cancelar la grabación.

```
function guardar_Callback(hObject, eventdata, handles)
```
Funcion que se encarga de guardar la grabación.

```
function [tf,f] = transformada(x,fs,handles)
```
Funcion que se encarga de calcular la transformada.

FiltradoSenales.M

Este fichero controla una de las interfaces secundarias del programa, concretamente aquella que se abre cuando deseamos utilizar la herramienta de filtrado de señales.

```
function archivo_Callback(hObject, eventdata, handles)
```
Función que despliega el submenú referente un nuevo archivo o al procesado.

```
function help_Callback(hObject, eventdata, handles)
```
Función que despliega la página de ayuda creada en el canal de youtube.

```
function abrir_Callback(hObject, eventdata, handles)
```

Función que carga un audio como señal que vamos a procesar.

```
function guardar_Callback(hObject, eventdata, handles)
```

Función que guarda la señal que acabamos de procesar en un archivo .wav.

```
function Wp_Callback(hObject, eventdata, handles)
```

Función que toma el valor de la caja referente a la frecuencia de paso final.

```
function Ws_Callback(hObject, eventdata, handles)
```

Función que toma el valor de la caja referente a la frecuencia de parada final.

```
function Rp_Callback(hObject, eventdata, handles)
```

Función que toma el valor de la caja referente a la atenuación máxima en la banda de paso.

```
function Rs_Callback(hObject, eventdata, handles)
```

Función que toma el valor de la caja referente a la atenuación mínima en la banda de parada.

```
function Wp0_Callback(hObject, eventdata, handles)
```

Función que toma el valor de la caja referente a la frecuencia de paso inicial.

```
function Ws0_Callback(hObject, eventdata, handles)
```

Función que toma el valor de la caja referente a la frecuencia de parada inicial.

```
function FILTRO_Callback(hObject, eventdata, handles)
```

Esta función realiza dos tareas, calcula un filtro equivalente al dibujado y desencadena la acción de filtrado mediante la función"Filter".

```
function ActualizarFiltro_Callback(hObject, eventdata, handles)
```

Esta función realiza varias funciones, lo primero recoge el tipo de filtro con el que estamos tratando, visualiza si trabajamos con un filtro paso banda y si es así, recoge los dos valores correspondientes a las frecuencias iniciales de paso y parada. Por último, coge los valores del filtro que corresponde al menor orden dentro de las características escogidas y lo imprime sobre la señal de trabajo.

```
function [tf,f] = transformada(x,fs)
```

Función que realiza la transformada de Fourier.

```
function MuestreoFrec(handles)
```

Función que imprime la señal escogida en el dominio de la frecuencia a través de la ventana 1.

```
function MuestreoFrec2(handles)
```

Función que imprime la señal procesada en el dominio de la frecuencia a través de la ventana 2.

```
function [NUM,DEN] = calculaNumDen(handles)
```

Función que calcula el numerador y el denominador del filtro escogido.

```
function BPF_Callback(hObject, eventdata, handles)
```

Función que activa la opción de filtro paso bajo.

```
function LPF_Callback(hObject, eventdata, handles)
```

Función que activa la opción de filtro paso alto.

```
function HPF_Callback(hObject, eventdata, handles)
```

Función que activa la opción de filtro paso banda.

OndaPersonalizada.M

Este fichero controla una de las interfaces secundarias del programa, concretamente aquella que se abre cuando deseamos crear una onda arbitraria personalizada.

```
function varargout = OndaPersonalizada(varargin)
function OndaPersonalizada_OpeningFcn(hObject, eventdata, handles, varargin)
function varargout = OndaPersonalizada_OutputFcn(hObject, eventdata, handles)
```

Se trata de las funciones que nos sirven para cargar la interfaz GUIDE del fichero OndaPersonalizada.m

```
function funtionhandle_Callback(hObject, eventdata, handles)
```

Se encarga de recoger el valor valor del function handle y transmitir al usuario que lo ha introducido de manera correcta.

```
function puntofinal_Callback(hObject, eventdata, handles)
```

Se encarga de recoger el valor del punto final de nuestra evaluación y transmitir al usuario que lo ha introducido de manera correcta.

```
function anadir_Callback(hObject, eventdata, handles)
```

Mediante un sofisticado cálculo matemático, esta función concatena la anteriormente creada, si hubiera, con la nueva evaluación. Todo ello se hace sobre una frecuencia de muestreo común de 44100 Hz.

```
function save_Callback(hObject, eventdata, handles)
```

Esta función se encarga de guardar la señal que hemos creado.