**MEMORIA**

**DE**

**PROGRAMACIÓN**

**Interfaz Gráfico de Análisis y Procesado de Señales de Sonido en Matlab: App Designer**

Fsi005

Arturo Labajo

Rubén Urraca

**ÍNDICE**

[**1-** **Introducción** 3](#_Toc185539149)

[**2-** **Funciones** 4](#_Toc185539150)

[1- startupFcn 4](#_Toc185539151)

[2- Cargar\_Archivo 4](#_Toc185539152)

[3- UpdatePlot 6](#_Toc185539153)

[4- Grabar 6](#_Toc185539154)

[5- Parar\_Grabación 7](#_Toc185539155)

[6- Reproducir 8](#_Toc185539156)

[7- Pausa 9](#_Toc185539157)

[8- Stop 9](#_Toc185539158)

[9- Señal\_Generada 10](#_Toc185539159)

[10- Invertir\_Senal 12](#_Toc185539160)

[11- Histograma 13](#_Toc185539161)

[12- CompresionExpansion 14](#_Toc185539162)

[13- Efectos 14](#_Toc185539163)

[14- Ruido 16](#_Toc185539164)

[15- CambiarPanel 18](#_Toc185539165)

# **Introducción**

El objetivo de esta memoria consta de documentar nuestro desarrollo para la realización de una interfaz de sonido creada a través de MATLAB utilizando AppDesigner.

La finalidad del proyecto es proporcionar una interfaz gráfica interactiva diseñada para ser intuitiva y accesible con fácil manejo (ver Figura 1).

Esta interfaz permitirá al usuario además de cargar archivos de audio y reproducirlos, pueden ver el espectrograma y el histograma correspondiente, añadir efectos, ruidos y muchas más cosas que iremos viendo tanto en el manual de usuario como adelante en este manual de programación.

Captura de pantalla de computadora

Descripción generada automáticamente con confianza media

*Figura 1: Interfaz*

Para realizar esto hemos añadido objetos para poder interaccionar y pulsando encima de ellos con el clic derecho y añadiendo un “callback” hemos podido hacer que cada objeto con el que el usuario pueda interactuar tenga una funcionalidad distinta al resto de manera muy sencilla

# **Funciones**

## startupFcn

Esta función se ejecuta al iniciar la aplicación y lo que queremos es que nos maximice la ventana para que se ajuste al tamaño de la pantalla que reproduce la aplicación (ver Figura 2)

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Figura 2: Función startupFcn*

## Cargar\_Archivo

Con esta función buscamos que el usuario cargue un archivo de audio desde su dispositivo.

En la primera parte de esta función (ver Figura 3), lo que realizamos es la selección del archivo mediante **uigetfile** obteniendo el nombre y camino del archivo, tras esto comprobamos que se ha seleccionado correctamente el archivo y se procede a cargarlo mediante **audioread**, almacenando la pista de audio y la frecuencia de muestreo.

Texto, Aplicación

Descripción generada automáticamente

*Figura 3: Primera parte función Cargar\_Archivo*

En la segunda parte de esta función (ver Figura 4), lo que hacemos es obtener los datos interesantes del archivo para mostrárselo al usuario en la interfaz (ver Figura 5). Algunos datos los obtenemos a través de **audioinfo.**

Otros datos como la extensión del archivo lo obtenemos a través de **fileparts** y después vemos si el archivo es .MP3, .WAV u otro archivo que hemos puesto como “Desconocido”.

Luego la potencia la calculamos de forma manual y tras esto vamos aplicando los valores calculados al correspondiente campo de la interfaz que va a ver el usuario.

Escala de tiempo

Descripción generada automáticamente con confianza media

*Figura 4: Segunda parte función Cargar\_Archivo*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Figura 5: Panel con información del archivo*

Esta función se arranca cuando el usuario pulsa el botón de correspondiente (ver Figura 6).

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

*Figura 6: Botón de cargar archivo*

## UpdatePlot

Esta función lo que hace es actualizar la grafica en tiempo real durante la grabación de un audio a través del micrófono pudiendo ver en todo momento la señal que esta captando la aplicación

Como código importante (ver Figura 7) a través de **getaudiodata** vamos a obtener los datos de lo que se esta grabando y luego mediante **drawnow** la gráfica se va actualizando inmediatamente recogiendo y mostrando lo almacenado en Data

Es importante calcular el tiempo para luego mientras se grafica la grabación esta se muestre de forma correcta

Texto

Descripción generada automáticamente

*Figura 7: Función updatePlot*

Con esto el usuario puede ver una representación en vivo de la grabación que se está llevando acabo

## Grabar

Esta función (ver Figura 8) va a iniciar la grabación de audio creando un objeto con **audirecorder** donde nosotros pasamos los parámetros de frecuencia de muestreo y bits que puede introducir el usuario a través de la interfaz.

Antes de nada, vamos a limpiar las gráficas para el caso en el que haya algún audio cargado y nos de problemas para después configurar el uso de la función updatePlot para ver en tiempo real lo que se está grabando.

Tras todo eso con **record** iniciamos la grabación

Texto

Descripción generada automáticamente

*Figura 8: Función Grabar*

Esta función se arranca cuando el usuario pulsa el botón correspondiente (ver Figura 9)

Icono

Descripción generada automáticamente

*Figura 9: Botón Grabar*

## Parar\_Grabación

Con esta función para la grabación que se está en curso y que se muestre ajustada en el dominio del tiempo y su espectrograma.

Para esta función (ver Figura 10), después de comprobar que por lo menos haya alguna grabación en curso paramos la grabación mediante **stop** y obtenemos los datos grabados y los almacenamos para que si luego los modificamos no perdamos los originales.

Tras esto ya solo queda mostrar la señal en el dominio del tiempo y su espectrograma al usuario.

Texto

Descripción generada automáticamente con confianza media

*Figura 10: Función Parar\_Grabacoion*

## Reproducir

En esta función el usuario tras pulsar el botón correspondiente (ver Figura 11) empezara la reproducción del audio que este cargado en la interfaz ya sea una grabación, un archivo de audio cargado o una señal simple generada.

Icono

Descripción generada automáticamente

*Figura 11: Botón de reproducción*

Dentro de esta función (ver Figura 12) comprobamos primero que existe un audio cargado para reproducir en **app.audiodata**, luego tenemos que conocer si tenemos alguna pista de audio pausada y debemos retomar la reproducción donde se pauso mediante **resume(app.player).**

Tras estas comprobaciones si tenemos algún audio cargado y no ha sido pausado creamos el reproductor mediante **audioplayer** introduciendo la pista de audio y la frecuencia de muestreo almacenadas, y iniciamos la reproducción con **play(app.player).**

**Texto

Descripción generada automáticamente**

*Figura 12: Función Reproducir*

## Pausa

Esta función procede a pausar el audio que esta en reproducción en el momento que el usuario pulsa el botón correspondiente (ver Figura 13)

Icono

Descripción generada automáticamente

*Figura 13: Botón de pausa*

Para esta función (ver Figura 14) hemos comprobado primero que hay un audio para reproducir y luego tras comprobar que el audio se está reproduciendo y no está vacío pausamos la reproducción con **pause(app.player)**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

*Figura 14: Función Pause*

## Stop

Esta función procede a detener el audio que esta en reproducción y reiniciar el reproductor desde el inicio cuando el usuario pulsa el botón correspondiente (ver Figura 15)

Icono

Descripción generada automáticamente

*Figura 15: Botón de stop*

Dentro de esta función (ver Figura 16) tras comprobar que hay algún audio reproduciendo detenemos la reproducción con **stop(app.player)** y por ultimo reiniciamos el reproductor para que empiece desde cero mediante **audioplayer.**

**Captura de pantalla de un celular

Descripción generada automáticamente**

*Función 16: Función Stop*

## Señal\_Generada

Esta función va a generar señales simples mediante los parámetros que introduce el usuario (ver Figura 17), dentro de los tipos de señales tenemos el seno, coseno, cuadrática, triangular y dientes de sierra

Interfaz de usuario gráfica

Descripción generada automáticamente

*Función 17: Parámetros*

Dentro de esta función (ver Figura 18.1) primero vamos a recoger los parámetros que introduce el usuario a través de la interfaz. Tras esto calculamos los tiempos a través de las frecuencias introducidas.

Con un **switch** vamos a comprobar que tipo de señal ha seleccionado el usuario y dependiendo de la elección vamos a definir nuestra señal (signal), luego después de tener todos los parámetros antes de graficarlos vamos a asignarlos a las propiedades **app.audioData** y **app.fs** y los vamos a almacenar en **app.OriginalSignal** y **app.OriginalTime** para no perder las propiedades en caso de alguna posterior modificación.

Imagen que contiene Tabla

Descripción generada automáticamente

*Figura 18.1: Función Señal\_Generada*

Por último, vamos a graficarlo (ver Figura 18.2) tras mostrar la señal en el dominio del tiempo vamos a calcular el espectrograma, para esto hemos tenido que comprobar la longitud de la ventana adecuada a la duración de la señal que se ha escogido debido a que si la duración es muy pequeña da error.

Luego vamos a crear el espectrograma a través de la función **spectrogram** y lo graficamos.

Texto

Descripción generada automáticamente

*Figura 18.2: Función Señal\_Generada*

## Invertir\_Senal

Esta función procede a la inversión del eje temporal a través de la activación de un ‘Check Box’ por parte del usuario (ver Figura 19)

Imagen que contiene Logotipo

Descripción generada automáticamente

*Figura 19: Check Box*

Primero (ver Figura 20) vamos a recoger el valor del “Check Box” para ver si esta activado o no.

Si esta activado vamos a detener la reproducción y luego invertimos el eje de tiempos y la señal con **flipud**, tras esto reiniciamos el reproductor.

Además del mismo modo hemos calculado el espectrograma correspondiente para que también se invierta en caso de activar la función

Si se desactiva vamos a restaurar la señal mediante los parámetros **app.OriginalSignal** y **app.OriginalTime** y reiniciamos el reproductor.

Texto, Escala de tiempo

Descripción generada automáticamente

*Figura 20: Función Invertir\_Senal*

## Histograma

Con esta función se mostrará una nueva ventana, cuando el usuario accione el botón (ver Figura 21), con el histograma correspondiente a la última señal cargada pueda ser editada o no.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

*Figura 21: Botón histograma*

Para esta función (ver Figura 22) tras comprobar que tenemos un audio cargado creamos la nueva ventana con **uifigure** y el nuevo axes con **uiaxes.**

Para el calculo del histograma realizamos la FFT a través de **fft** y **fftshift** y por último lo graficamos en el nuevo axes que hemos creado.

Texto

Descripción generada automáticamente

*Figura 22: Función Histograma*

## CompresionExpansion

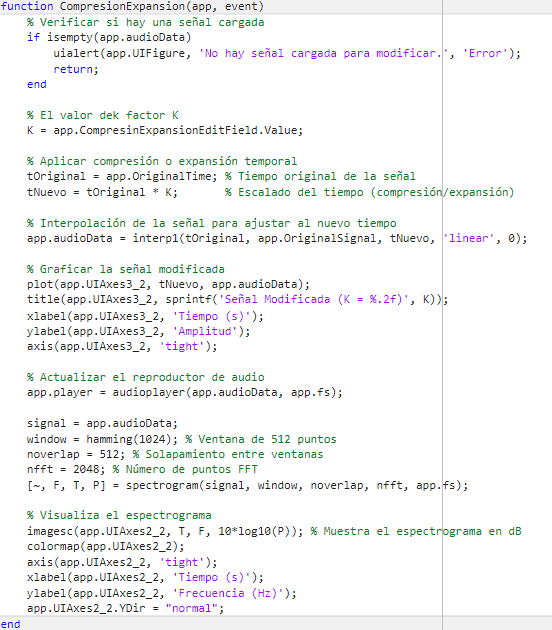
Con esta función tras la introducción de un facto k introducido por el usuario (ver Figura 23) se procede a hacer una expansión o compresión de la señal.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

*Figura 23: Parámetro k*

Dentro de la función (ver Figura 24) tras comprobar que tenemos una función cargada recogemos el valor introducido por el usuario. Luego vamos a multiplicar el tiempo original por el factor introducido y graficamos el nuevo dominio temporal y el nuevo espectograma



*Figura 24: Función CompresionExpansion*

## Efectos

Con esta función el usuario puede elegir entre varios efectos disponibles (ver Figura 25) para aplicar a la señal cargada.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

*Figura 25: Efectos*

Dentro de esta función (ver Figura 26.1) tras comprobar que tenemos alguna señal cargada y recoger los parámetros originales vamos a ver que elección a introducido el usuario mediante un **switch** y dependiendo de la selección vamos a modificar el valor de **app.audioData**.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

*Figura 26.1 Funcion Efectos*

Para el efecto “Eco” hemos introducido un delay y luego se sumará a la señal original atenuada mediante una atenuación que hemos introducido con anterioridad.

Para el efecto “Distorsión” hemos aplicado una saturación mediante la función **tanh** que comprime las amplitudes grandes haciendo una simulación de “saturación”

Para el efecto “Reverberación” se simula un eco prolongado con un filtro se convolución utilizando una respuesta al impulso con retardos y atenuaciones

Para el efecto “Amplificación” se multiplica la señal por un factor constante aumentando la amplitud

Tras todo esto (ver Figura 26.2) procedemos a la creación del nuevo objeto con la modificación necesaria y lo graficamos, como también se le puede introducir efectos a la señal generada debemos comprobar el tamaño de la ventana para ver el espectrograma por si es muy corta evitar errores

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Figura 26.2: Función Efectos*

## Ruido

Con esta función el usuario puede elegir entre varios ruidos disponibles (ver Figura 27) para aplicar a la señal cargada.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

*Figura 27: Ruidos*

Para esta función (ver Figura 28.1) tras comprobar que tenemos alguna señal cargada y recoger los parámetros originales vamos a ver que elección a introducido el usuario mediante un **switch** y dependiendo de la selección creamos un ruido

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

*Figura 28.1: Función Ruido*

Para el ruido “Blanco” hemos generado el ruido con una distribución normal ya que el ruido blanco presenta una potencia uniforme frecuencial.

Para el ruido “Uniforme” hemos generado valores aleatorios uniformemente distribuidos similar a lo realizado en el ruido blanco

Para el ruido “Impulsivo” introducimos picos esporádicos en la señal generando valores aleatorios simulando “frecuencias bruscas”

Para el ruido “Sinusoidal” genera una señal sinusoidal que se sumara a la original introduciendo un tono sinusoidal con frecuencia constante

Por último (ver Figura 26.2) procedemos a la creación del nuevo objeto sumándole al original el ruido y lo graficamos, como también se le puede introducir ruidos a la señal generada debemos comprobar el tamaño de la ventana para ver el espectrograma por si es muy corta evitar errores.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

*Figura 28.2: Función Ruido*

## CambiarPanel

Con esta función el usuario cuando pulse el botón (ver Figura 29) puede visualizar la señal cargada en el panel inferior.



*Figura 29: Botón CambiarPanel*

Para esta función lo que hemos hecho es obtener los datos de los dos gráficos superiores para replicarlos abajo mediante **findobj** debido a que así si se modificase la señal la cual se carga en los gráficos inferiores nos aseguramos de que al pulsar el botón lo que se visualice sea la señal que se ve en los gráficos superiores.

Una vez obtenido los datos los hemos graficado en los “axes” correspondientes.

PEGAR CODIGO