

DESARROLLO DE UNA APLICACIÓN EN MATLAB PARA MEDIDAS DE SISTEMAS DE SONIDO CON MÚLTIPLES ENTRADAS Y MÚLTIPLES SALIDAS MEDIANTE SEÑALES SWEEP

Pere Castán Orero

Tutor: José Javier López Monfort

Cotutor: Pablo Gutiérrez Pareira

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2016-17

Valencia, 4 de julio de 2017

Resumen

Los sistemas acústicos y de sonido considerados lineales pueden ser caracterizados completamente por su respuesta al impulso (IR). En los últimos años, la señal de test consistente en un sweep logarítmico se ha demostrado como la más eficaz para la medida de la IR por diferentes ventajas. Existen aplicaciones comerciales de medida disponibles, pero casi ninguna permite de forma flexible la medida de sistemas complejos con múltiples entradas y múltiples salidas (MIMO).

El objetivo de este Trabajo Fin de Grado es el desarrollo de una aplicación en MATLAB que permita la medida de sistemas acústicos MIMO de forma flexible, rápida y fiable. El software desarrollado dispone de interfaz gráfica amigable a la vez que permite utilizar drivers multicanal de cualquier tarjeta de sonido que se conecte al ordenador. Es multiplataforma, ya que funciona por igual en Windows, Apple OSX y Linux (en Windows además permite drivers de baja latencia ASIO).

Además del desarrollo del software, se han realizado unas pruebas de laboratorio con el mismo, a fin de comprobar el correcto funcionamiento del software en condiciones acústicas reales.

Resum

Els sistemes acústics i de so considerats lineals poden ser caracteritzats completament per la seua resposta a l'impuls (IR). En els últims anys, la senyal de test consistent en un sweep logarítmic ha demostrat ser la més eficaç per a la mesura de la IR per diferents avantatges. Existeixen aplicacions comercials de mesura disponibles però casi ninguna permet de forma flexible la mesura de sistemes complexos amb múltiples entrades i múltiples eixides (MIMO).

L'objectiu d'aquest Treball Fi de Grau es el desenvolupament d'una aplicació en MATLAB que permet la mesura de sistemes acústics MIMO de forma flexible, ràpida i fiable. El software desenvolupat disposa d'una interfície amigable a la vegada que permet utilitzar divers multicanal de qualsevols tarjeta de so que es connecte a l'ordinador. És multiplataforma, ja que funciona per igual en Windows, Apple OSX i Linux (en Windows es poden usar els drivers de baixa latència ASIO).

A més del desenvolupament del software, s'han realitzat unes proves de laboratori amb ell, amb la fi de comprovar el correcte funcionament del software en condicions acústiques reals.

Abstract

Acoustic and sound systems considered linear can be characterized completely by their Impulse Response (IR). In recent years, the test signal consisting of a logarithmic sweep has been shown to be the most effective for IR measurements for different advantages. There are commercial measurement applications available, but almost none permits the flexible measurement of complex systems with multiple inputs and multiple outputs (MIMO).

The aim of this Final Thesis is the development of a MATLAB application that allows the measurement of MIMO acoustic systems in a flexible, fast and reliable way. The developed software has a friendly graphical interface and allows the use of multichannel drivers of any sound card that is connected to the computer. It is cross-platform since it works equally on Windows, Apple OSX and Linux (in Windows, ASIO low latency drivers can be used).

In addition to the development of the software, laboratory tests have been made, in order to check the correct operation of the software under real acoustic conditions.

Índice

Capítulo 1. Introducción.....	2
1.1 Motivación.....	2
1.2 Objetivos	3
Capítulo 2. Metodología.....	4
2.1 Metodología del desarrollo de la aplicación.....	4
2.2 Metodología de los ejemplos de aplicación.....	5
Capítulo 3. Conceptos teóricos sobre medidas de sistemas acústicos	6
3.1 Respuesta al impulso y respuesta en frecuencia.....	6
3.2 Señal sweep logarítmica.....	8
Capítulo 4. Desarrollo de la aplicación.....	9
4.1 Audio System Toolbox	9
4.2 Descripción de la aplicación.....	11
4.2.1 Vista general de la aplicación.....	12
4.2.2 Introducción de los datos	14
4.2.3 Generación del sweep logarítmico	16
4.2.4 Captura de las medidas	17
4.2.5 Generación de las gráficas.....	17
4.2.6 Guardado de los datos.....	19
Capítulo 5. Ejemplos de aplicación	21
5.1 Medida de la respuesta al impulso de una sala (una salida y cuatro entradas).....	21
5.2 Medida de HRTF de un maniquí acústico (seis salidas y dos entradas).....	26
Capítulo 6. Conclusiones y propuestas de trabajo futuro.....	33
6.1 Conclusiones.....	33
6.2 Propuestas de trabajo futuro	34
Capítulo 7. Bibliografía.....	35
Capítulo 8. Anexos	36
8.1 Código de la aplicación (interfaz gráfica).....	36
8.2 Código de la función generadora del sweep logarítmico	46

Capítulo 1. Introducción

1.1 Motivación

Dentro de la medición de sistemas de audio lineales, la respuesta al impulso (IR) y la respuesta en frecuencia son dos de las características más importantes que identifican su comportamiento. La obtención de la IR, permite calcular la respuesta en frecuencia del sistema a través de la transformada de Fourier.

Hay diversas formas de realizar la medición de la IR, aunque este trabajo ha escogido el método considerado como más eficaz hoy en día que es el “sweep logarítmico” [1], y que se revisa en la sección 3.2.

Existen variedad de programas capaces de llevar a cabo estas mediciones pero casi ninguna permite de forma flexible la medida de sistemas complejos con múltiples entradas y múltiples salidas (MIMO). Además se ha querido desarrollar una aplicación para la medida de estos parámetros desde cero con MATLAB, que tiene un carácter pedagógico a la vez que multiplataforma, estando muy extendido en los estudios de ingeniería y en la I+D. MATLAB cuenta con muchos recursos y funciones para el uso y tratamiento de audio desde hace bastantes versiones pero no fue hasta la 2016a [2] cuando se introdujo una nueva librería llamada Audio System Toolbox. Esta librería presenta una serie de objetos, algoritmos y funciones destinadas a la gestión y tratamiento de audio [3]. Con este programa se pueden hacer medidas multicanal entre tantas entradas y salidas como se desee de forma simultánea y con total flexibilidad, cosa que no suelen permitir la mayoría de programas.

La incorporación de esta nueva funcionalidad en MATLAB hacía más interesante el desarrollo de la herramienta pues estas prestaciones tienen relativamente poco tiempo y ofrecía una oportunidad para aprender su manejo y su forma de trabajar. También se ha buscado el manejo fácil de la herramienta por lo que ha sido desarrollada una interfaz gráfica con el entorno de programación visual GUIDE de MATLAB. Esto permite un uso más intuitivo de la aplicación a la hora de introducir los datos y ejecutar el programa.

También era interés de este trabajo el desarrollar algunas pruebas sencillas que permitieran, en primer lugar, comprobar el correcto funcionamiento de la aplicación y, a parte, permitir ver las posibilidades que ofrece la herramienta poniendo en práctica los conocimientos adquiridos durante el desarrollo de la misma. Estas pruebas están descritas en su correspondiente capítulo así como los resultados obtenidos.

Se incluyen, más adelante en este documento, una serie de propuestas de trabajo futuras para el programa con la finalidad de enriquecer ciertos aspectos tanto de su funcionamiento como de su interfaz gráfica.

1.2 Objetivos

Los objetivos perseguidos en este trabajo son:

- Adquirir los conocimientos sobre medidas de sistemas de audio y el uso de la señal sweep logarítmica en ellas.
- Aprendizaje del uso de objetos de la librería Audio System Toolbox para el desarrollo de la aplicación.
- Aprendizaje del uso del entorno de programación visual GUIDE para el desarrollo de la interfaz gráfica de la aplicación.
- Uso fácil e intuitivo del programa por parte del usuario
- Desarrollo de pruebas para valorar el correcto funcionamiento de la aplicación.
- Adquirir destreza a la hora de manejar sistemas de audio tales como tarjetas de sonido, altavoces, micrófonos y cableado.

Las características que deberá tener el software:

El programa deberá pedirlos datos que definen el sweep logarítmico, tales como la frecuencia de muestreo, la frecuencia de inicio y la final y su duración. Después el usuario tendrá que seleccionar los drivers con los que quiere trabajar. Una vez hecho esto el programa detectará qué dispositivos de audio soportan los drivers seleccionados y dará a elegir entre ellos. A continuación, aparecerán dos columnas de botones: a la izquierda las entradas y la derecha las salidas, todo ello en función del dispositivo elegido. Es entonces cuando el usuario decidirá por qué entradas va ser capturada cada salida y también qué tipo de representación: respuesta en frecuencia o respuesta al impulso. Después de esto se generará un sweep logarítmico que será lanzado tantas veces como salidas se hayan seleccionado y capturado por las entradas que hayamos vinculado a cada una de ellas.

A continuación, se deberá abrir una ventana por cada medida realizada que contenga la gráfica del tipo de respuesta que se haya seleccionado previamente de cada salida por la que se ha capturado el audio. Cada una de estas gráficas estará identificada con el número de la salida que ha lanzado el sweep y la entrada que lo ha capturado. Si se quiere trabajar más tarde y con más profundidad con estas respuestas capturadas, tras la ejecución del programa se almacenará la respuesta del sistema en un archivo .mat que contendrá cuatro campos: la salida por la que se ha emitido, las entradas por las que se ha grabado, la respuesta de cada salida capturada y la frecuencia de muestreo.

Capítulo 2. Metodología

En este capítulo se describe cual ha sido el proceso seguido para el desarrollo del software y los ejemplos de aplicación llevados a cabo con el mismo.

2.1 Metodología del desarrollo de la aplicación

Primero, una vez asignado el TFG, se debía conocer que enfoque se quería dar al trabajo. Se eligió la opción de desarrollar un programa propio con las herramientas proporcionadas por MATLAB. Esto llevó a plantearse el uso de la librería Audio System Toolbox ya que al ser relativamente nueva ofrecía la posibilidad de aprender su funcionamiento. Este proceso consistió en conocer qué funciones y objetos presentes en esta librería podrían ser de utilidad. Se consultó la documentación de MATLAB [4] y se localizaron dichas funciones. Esta documentación es un listado de las funciones y algoritmos presentes en la librería, donde se explica de forma detallada sus propiedades y se dan ejemplos para su uso, siendo estos de gran ayuda. Una vez hecho esto comenzó el proceso de aprendizaje para ver cómo trabajan, cómo se gestionan y que propiedades tienen para poder usarlas en el programa. Al principio se desarrolla un programa con una funcionalidad básica parecida a la aplicación final pero sin algunas de sus características. Este programa de prueba se desarrolló usando las funciones anteriormente mencionadas pero sin el uso del entorno de programación visual GUIDE. Por tanto, el usuario debía introducir los datos solicitados a través de la ventana de comandos. Se vio que este método, aunque funcional, no era cómodo para el usuario. Como se ha dicho en el capítulo anterior en el apartado de objetivos, una de las finalidades del desarrollo de esta herramienta era un uso fácil e intuitivo. Es en este punto cuando se decide empezar a trabajar con el entorno de programación visual GUIDE.

Con el esqueleto del programa por un parte, se empieza a consultar documentación sobre GUIDE [5]. Donde se presentan las funciones de este entorno de programación y sus propiedades, así como pequeños ejemplos de uso. Gracias a estas consultas se aprende cual es la forma de trabajar de este entorno y se procede al desarrollo de la interfaz gráfica. También se consultaron algunos tutoriales online con ejemplos de programas desarrollados [6]. Aquí surgieron varios imprevistos por la forma de trabajar de cada entorno pues generalmente se había trabajado con la ventana de comandos de MATLAB y había que adaptar este primer programa de prueba a su nuevo “recipiente” que era la interfaz gráfica.

Una vez “adaptado” el programa a la interfaz gráfica se empiezan a introducir funciones para facilitar su uso.

Se implementa una función que permite la creación dinámica de botones para que el programa se adapte al número de salidas y entradas del dispositivo seleccionado evitando así un número estático que, aunque hubiera sido funcional, no sería tan eficiente. También se incluye la

posibilidad de representar la respuesta en frecuencia o la respuesta al impulso, a elección del usuario. Pensando en cómo utilizaría el usuario la aplicación, también se desarrolla la posibilidad de hacer click sobre una gráfica y abrir la misma en una ventana aparte. Se implementa, una vez visto que el programa funciona, el guardado de datos de la respuesta. Se crea una estructura por cada medida en la que hay cuatro campos. Con esto se pretende que el usuario pueda en otro momento consultar las respuestas de los sistemas medidos y trabajar con ellas.

2.2 Metodología de los ejemplos de aplicación

Una vez finalizado el programa, se preparan dos ejemplos de aplicación con el objetivo de demostrar las posibilidades que ofrece trabajando con múltiples salidas y entradas a la vez.

En estas medidas acústicas reales se buscan que éstas sean precisas y para ello los equipos usados han sido los de más calidad posible disponibles en el laboratorio.

Para el lanzamiento de los sweep logarítmicos se utilizaron unos altavoces autoamplificados JBL LSR 305. Estos monitores se adaptan a estos ejemplos ya que su respuesta en frecuencia de 43 Hz a 24 kHz.

En el segundo ejemplo de aplicación también se usan cuatro altavoces Apart SDQ5P con unas respuestas de frecuencia de 45 Hz a 20 kHz.

Los micrófonos usados para la captura han sido micrófonos de medida Earthworks calibrados, con un patrón polar omnidireccional y una respuesta en frecuencia de 5 Hz a 30 kHz. Con estos micrófonos se asegura que están bien calibrados y unas capturas de audio muy precisas.

También se ha usado un maniquí acústico Type 4100 de Brüel & Kjaer, diseñado para la evaluación de la calidad de sonido, lo que proporciona una precisa grabación tridimensional.

La primera prueba consiste en medir la respuesta al impulso del sistema formado por la sala y un altavoz con cuatro micrófonos de medida puestos en fila y equiespaciados delante del altavoz y apuntando al centro del mismo. Con esto se pretende ver cómo cambia la respuesta al impulso en cada punto.

En la segunda prueba aplicaremos la herramienta en otro campo para el cual también puede ser útil: el cálculo de HRTFs. En este caso se usa un maniquí acústico y se miden sus HRTFs con seis altavoces: cuatro delante, uno a la izquierda y otro a la derecha. Aquí se pretende percibir la diferencia en las HRTFs teniendo en cuenta la posición de la fuente.

El programa y los ejemplos de aplicación serán descritos más adelante en este documento en sus respectivos apartados.

Capítulo 3. Conceptos teóricos sobre medidas de sistemas acústicos

En este capítulo se cubren los conceptos teóricos necesarios para la comprensión de este trabajo y de las decisiones tomadas durante su desarrollo.

3.1 Respuesta al impulso y respuesta en frecuencia

La respuesta al impulso (IR) y la respuesta en frecuencia son dos de las características más útiles a la hora de describir el comportamiento un sistema lineal [7].

Cada una nos proporciona una información:

- La respuesta al impulso nos dice cómo se comportará el sistema ante un impulso en el dominio del tiempo.
- La respuesta en frecuencia indica cómo afectará el sistema a la señal de entrada en el dominio de la frecuencia.

Para la comprensión de estos dos términos podemos imaginar un sistema (H) con una sola entrada (x(t)) y una sola salida (y(t))

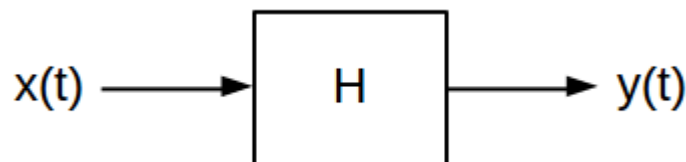


Figura 1. Esquema de un sistema

La caja H aplicará sobre la señal de entrada $x(t)$ una transformación que hará que a la salida obtengamos una señal de salida $y(t)$ que será igual a la convolución entre la señal de entrada y la respuesta al impulso ($h(t)$) del sistema: [1]

$$y(t) = x(t) \otimes h(t) \quad (3.1)$$

En la teoría este sistema es un sistema LTI, es decir, lineal invariante. Si la entrada de un sistema lineal es multiplicada por un factor la salida también será multiplicada por el mismo factor. Además, al combinar dos señales e introducir las en el sistema, la salida es la misma combinación lineal que si las entradas hubieran sido introducidas en el sistema de forma individual

$$H \{a_1 x_1(t) + a_2 x_2(t)\} = a_1 y_1(t) + a_2 y_2(t) \quad (3.2)$$

La propiedad invariante indica que las características del sistema no varían en el tiempo. Si le aplicamos a la señal de entrada un retardo, la señal a la salida del sistema conservará ese mismo retardo. [8]

$$H\{x(t-\tau)\} = y(t-\tau) \quad (3.3)$$

Para calcular una función de transferencia $h(t)$ desconocida es aplicar una señal $x(t)$ conocida a la entrada y medir la respuesta $y(t)$. En los sistemas reales no es posible aplicar un impulso perfecto para este cálculo por lo que las señales más usadas son señales banda ancha, deterministas y periódicas para luego proceder a la deconvolución entre el estímulo y la respuesta del sistema. Estas señales se usan para mejorar la relación señal a ruido (S/N) al coger múltiples promedios de la señal de salida antes de deconvolucionar la respuesta al impulso del sistema [1]. Se emplea la transformada Fourier y la transformada de Fourier inversa para calcular $h(t)$:

$$h(t) = IFFT \left[\frac{FFT(y(t))}{FFT(x(t))} \right] \quad (3.4)$$

Sin embargo, los sistemas reales presentan no linealidades que hacen su estudio mucho más complejo. Con los métodos que usan señales sweep lineales, pulsos periódicos o técnicas MLS la separación entre el comportamiento lineal y no lineal no puede realizarse. [9]

3.2 Señal sweep logarítmica

Tras estas consideraciones la pregunta que surge es cómo podemos calcular estas respuestas. En este trabajo se opta por el uso de una señal sweep logarítmica, donde la frecuencia varía exponencialmente en función del tiempo a lo largo de las frecuencias de interés. Estimulando el sistema con esta señal se obtienen una separación entre la respuesta lineal y la distorsión armónica. [9]

Con el método del sweep logarítmico desarrollado en [1] se consigue que la distorsión mencionada más arriba desaparezca del resultado del proceso de deconvolución. Esto se obtiene con un filtro inversor $f(t)$, que es la propia señal sweep logarítmica invertida temporalmente, que hace que al ser convolucionado con la señal de entrada se obtenga una función $\delta(t)$ retrasada:

$$x(t) \otimes f(t) \Rightarrow \delta(t) \quad (3.5)$$

Por lo que la respuesta al impulso $h(t)$ del sistema se puede obtener convolucionando la señal de salida con el filtro inversor $f(t)$:

$$h(t) = y(t) \otimes f(t) \quad (3.6)$$

Este método de medida consigue datos fiables aunque el sistema presente un comportamiento no lineal [1], además consigue un rendimiento más alto en comparación a otras técnicas en cuanto a la relación señal a ruido [9]. Es por estas razones por las que se ha optado por el sweep logarítmico, ya que solo presenta ventajas respecto a las otras técnicas de medida.

Como estas respuestas describen completamente el comportamiento lineal de cualquier sistema de audio [10], una aplicación práctica consiste en convolucionar una señal de audio con esta respuesta para que la señal de audio aplicada sea escuchada como si se estuviera en esa localización.

En los ejemplos de aplicación que se describen más adelante en los apartados 5.1 y 5.2, aplicaremos esto a las respuestas capturadas, donde se puede apreciar el efecto que tiene sobre la señal.

Capítulo 4. Desarrollo de la aplicación

Este capítulo aborda la descripción de la herramienta desarrollada. Se ha dividido en dos secciones. La primera servirá como breve introducción a la librería Audio System Toolbox y en ella se presentan los dos objetos de ésta librería que usa el programa junto con una descripción de sus propiedades. La segunda sección describirá el programa y su funcionamiento.

El entorno de trabajo ha sido un Windows 7 y la versión MATLAB usada ha sido la versión 2016b. Los esquemas para la descripción de los objetos de Audio System Toolbox se han conseguido directamente de la documentación del mismo [4].

4.1 Audio System Toolbox

Audio System Toolbox incluye librerías de algoritmos, fuentes y medidores que están pensados para el diseño y simulación de sistemas de procesamiento de audio en tiempo real. Dentro de estas librerías encontraremos aplicaciones, funciones y objetos. Fue incluido en Matlab por primera vez en la versión 2016a [2], por lo que es una adición relativamente novedosa como ya se ha comentado y por tanto ofrecía una buena oportunidad para explorar las nuevas funcionalidades y trabajar el procesamiento de audio y el diseño de este programa desde un enfoque distinto al que normalmente ofrece MATLAB.

Los dos principales objetos que se han usado en este programa son `AudioDeviceWriter` y `AudioDeviceReader`.

Ambos objetos gestionan el audio como matrices donde cada columna se corresponde con un canal, siendo la primera columna el primer canal del dispositivo, la segunda el segundo canal y así sucesivamente.

El objeto `AudioDeviceWriter` escribe muestras de audio en un dispositivo de salida de audio [4]. Es decir, el dispositivo encaminará el audio hacia una salida, como por ejemplo un altavoz, como se ve en la figura 2. Para poder hacerlo debemos indicar las características que va a tener:

- **Driver:** definimos que driver queremos usar para acceder al dispositivo. En el programa desarrollado se pueden elegir entre los que haya instalados en el ordenador pero es recomendable el uso de los drivers ASIO por su facilidad para trabajar con múltiples canales. En Windows los drivers ASIO no vienen preinstalados y por tanto habrá que instalarlos fuera de MATLAB. Para ello se descargaron los drivers de baja latencia ASIO independientes de hardware ASIO4ALL [11]. Estos drivers son gratuitos, siendo su última versión la 2.14.

- **Device:** dispositivo usado para reproducir el audio. El programa, una vez detectados los drivers, nos dará a elegir entre los distintos dispositivos que los soportan.
- **SampleRate:** frecuencia de muestreo de la señal enviada al dispositivo. Por defecto coge el valor de 44100 Hz pero por norma general se escogerán 48000Hz.
- **ChannelMappingSource:** por defecto, mapeará los canales haciendo corresponder cada columna de la matriz con un canal de salida. El problema se da cuando el número de la columna no se corresponde con el número de canal o el número de columnas y canales no coincide. En estos casos se deben mapear los canales de forma manual. Este campo se inicializará a 'Property' para poder editar el campo ChannelMapping, que es el que permite hacer dicho mapeo.
- **ChannelMapping:** mapeo personalizado de canales entre la matriz de entrada y canales del dispositivo de salida. Permitirá ver cómo está mapeada la matriz de entrada y se podrá indicar cómo se quiere hacer. Para ello se especifica un escalar o un vector con índices válidos para el dispositivo que se está usando. Es decir, si se tiene un dispositivo de ocho canales no se podrá indicar que se mapeé el canal nueve ya que éste no existe y por tanto devolverá un error. Si se le introduce un vector el orden de los índices es importante. Al trabajar de la forma comentada, el número de columna se corresponde con su índice en este vector y por tanto cada columna de la matriz de entrada se escribe en la posición correspondiente a su índice. Por ejemplo, si se introduce el vector [2 1] de un archivo de audio estéreo, los canales estarían invertidos ya que la columna número dos se está escribiendo en la primera posición del vector y la primera columna, en la segunda.

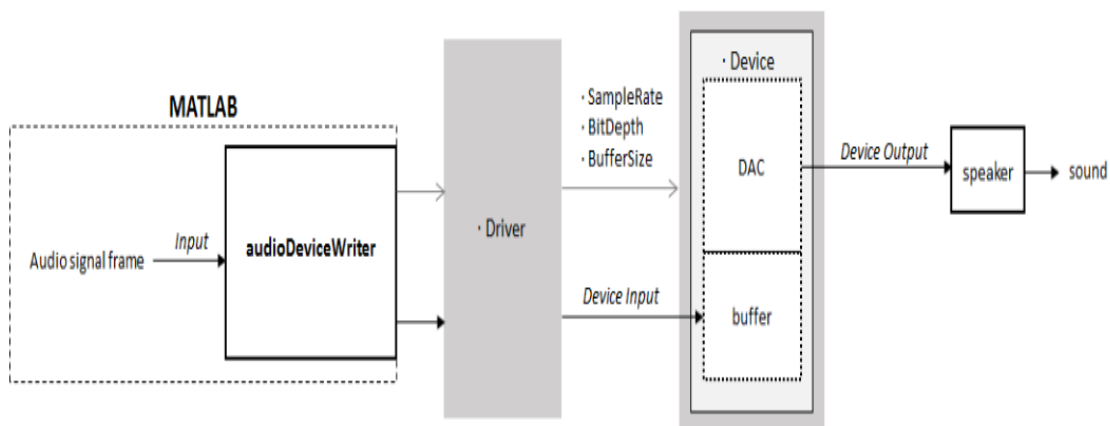


Figura 2. Interacción del objeto AudioDeviceWriter

El objeto AudioDeviceReader lee muestras de audio usando el dispositivo de audio de entrada [4], como por ejemplo un micrófono, como se aprecia en la figura 3. Es un objeto muy parecido al AudioDeviceWriter y tiene básicamente las mismas propiedades pero invertidas, es decir, las

que allí sirven para reproducir audio aquí se usan para capturarlo. Es importante destacar que el objeto `AudioDeviceReader` lee, por defecto, frames de audio de 1024 muestras. En el caso de este programa se ha optado por adaptarse a esta característica y no editar el campo `SamplesPerFrame`.

En este objeto los canales se gestionan de la misma forma que en el `AudioDeviceWriter`, es decir, cada columna de la matriz se corresponde con un canal de entrada. También se debe poner el campo de `ChannelMappingSource` a 'Property' para poder editar el campo `ChannelMapping`.

En el campo `ChannelMapping` se especifica el mapeo con un escalar o un vector con índices válidos para el dispositivo. Se puede especificar un mapeo alternativo entre los canales y las columnas de salida pero en el caso de este programa esta posibilidad es irrelevante ya que se busca que los canales se correspondan con sus respectivas salidas.

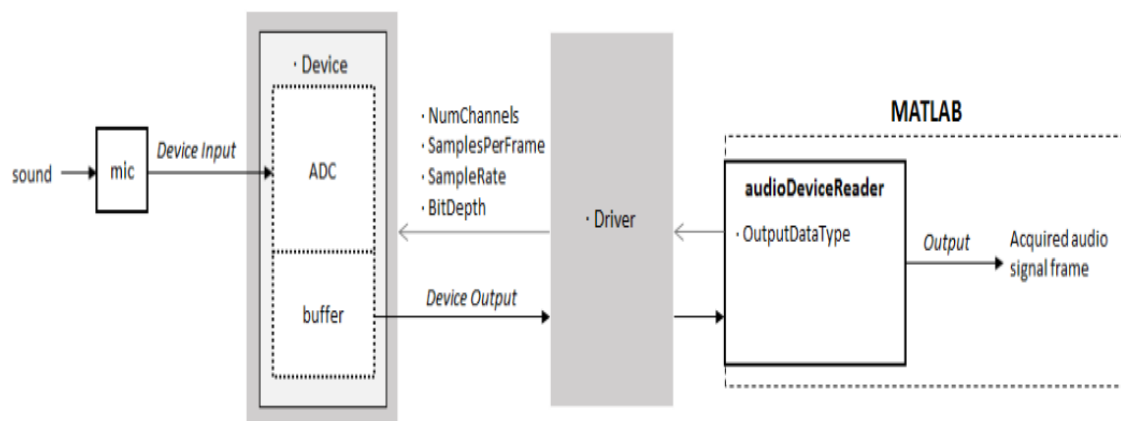


Figura 3. Interacción del objeto `AudioDeviceReader`

4.2 Descripción de la aplicación

La aplicación ha sido desarrollada con la facilidad de uso en mente. También se han incluido algunas funcionalidades para poder trabajar mejor con los datos obtenidos como poder abrir cada gráfica en una ventana a parte para su mejor visualización y el guardado de datos una vez finalizadas las medidas. Para el desarrollo de la parte gráfica con el entorno de programación visual GUIDE se consultó la documentación de MATLAB [5] y tutoriales en línea [6].

4.2.1 Vista general de la aplicación

La pantalla principal de la aplicación es la mostrada en la figura 4. Está dividida en tres partes:

- Parte izquierda: aquí se hace la selección de parámetros del sweep, la selección de drivers y dispositivos y la selección del tipo de gráficas
- Parte centro: aquí aparecen las entradas y salidas del dispositivo
- Parte derecha: donde está el botón generador del sweep

Esta ventana puede maximizarse para que ocupe toda la pantalla si se desea trabajar a pantalla completa.

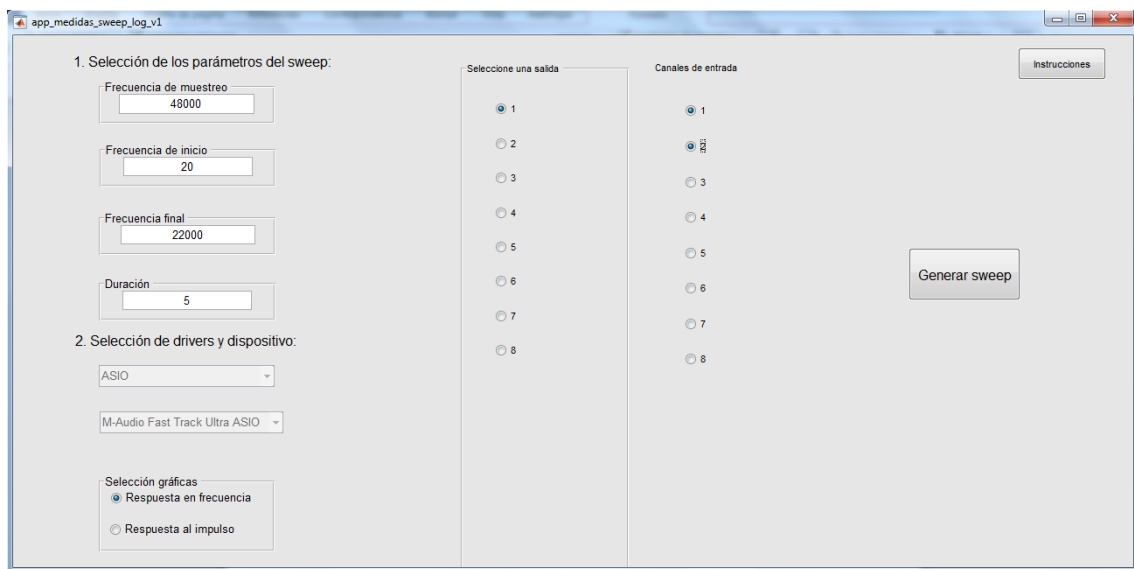


Figura 4. Vista general de la aplicación

En la parte izquierda lo primero que se ve es la sección de selección de parámetros del sweep. Sirve para introducir las características que va a tener el sweep logarítmico. En primer lugar está la frecuencia de muestreo, que es la frecuencia a la que se muestrea la señal que grabaremos y la del sweep. Después están la frecuencia inicial y final, la frecuencia inicial marca la frecuencia a la que empieza el sweep y la final a la que acaba. Esto delimitará el rango que recorre el sweep. Para aplicaciones de audio el rango más normal es de 20 Hz a 20000 Hz, que es el rango de las frecuencias audibles. Por último, seleccionaremos la duración en segundos del sweep. Este parámetro dependerá principalmente de la longitud de la respuesta del sistema bajo prueba. Cuanto más larga sea esta respuesta (una sala con mucha reverberación, por ejemplo) más duración debe tener el sweep para que la respuesta obtenida sea fiable.

Más abajo, se encuentran los menús de selección de drivers y dispositivos. Son dos menús desplegables. El primero corresponde al menú de drivers. Al iniciar el programa, éste obtendrá los drivers instalados en el ordenador y a través de este menú los mostrará y el usuario

seleccionará uno. Después se activa el menú de selección de dispositivos. Funciona igual que el anterior menú y mostrará los dispositivos instalados, el usuario debe elegir uno.

Por último, en esta parte izquierda de la pantalla está el menú de selección de gráficas. Aquí el usuario le dirá al programa qué tipo de representación gráfica quiere para las respuestas obtenidas más adelante.

En la parte central están las salidas y las entradas del dispositivo. Aparecen al seleccionar el dispositivo. La columna de la izquierda son las salidas y por ellas se lanza el sweep. La columna de la derecha son las entradas y por ellas se captura el audio. Estos botones están generados dinámicamente por lo que dependiendo del dispositivo seleccionado habrá más o menos. En la figura 4 se tiene conectada al ordenador una tarjeta de ocho canales de entrada y salida. El usuario seleccionará por qué salidas se lanza el sweep y por qué entradas se captura.

En la parte derecha está el botón para generar el sweep. Cuando se pulse se llamará a una función que recogerá los datos introducidos en la sección de selección de parámetros del sweep y generará un sweep logarítmico con esas características. Esta señal será lanzada por las salidas seleccionadas anteriormente y capturada por las entradas relacionadas con ella.

En la esquina superior derecha hay un botón donde al pulsar aparecerá la ventana de la figura 5 con unas pequeñas instrucciones sobre el uso del programa. Al pulsar el botón OK se cerrará.

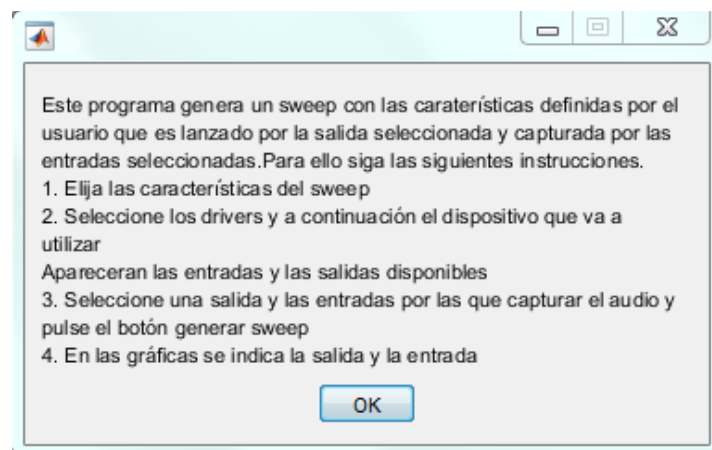


Figura 5. Ventana emergente de instrucciones

4.2.2 Introducción de los datos

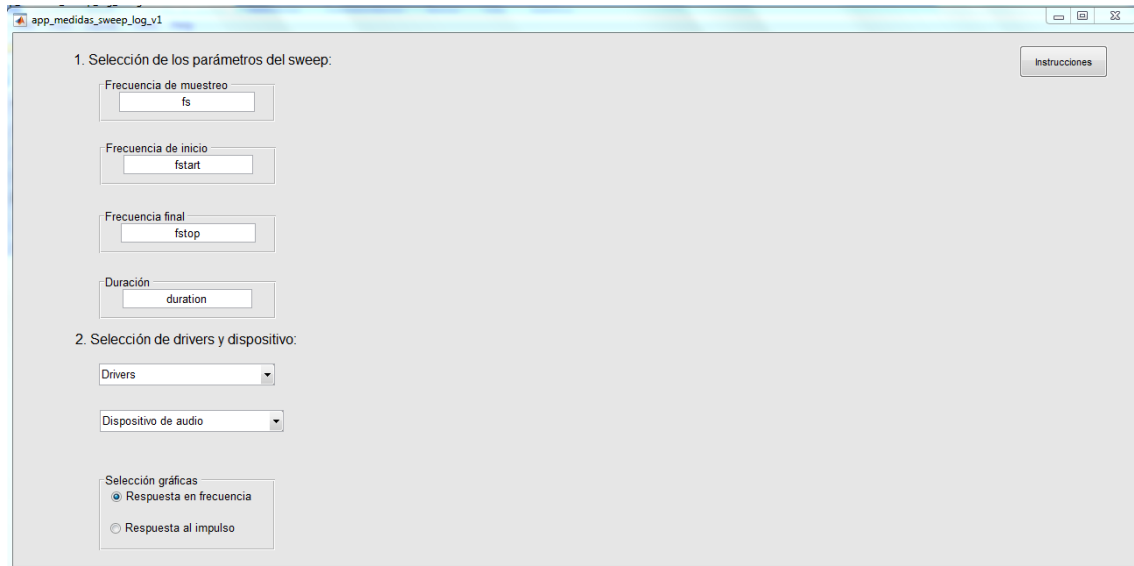


Figura 6. Inicio de la aplicación

Al iniciar la aplicación aparecerá la pantalla de la figura 6. Como aún no se ha seleccionado el dispositivo, los botones de salidas y entradas no aparecen la parte central.

Primero se introducen los parámetros del sweep. Aparecen cuatro cuadros para introducirlos. Se ha configurado el programa para que el tabulador pase de uno a otro en orden descendente. Estos datos serán capturados por la aplicación en el momento de generar el sweep, por lo que se pueden cambiar en cualquier instante anterior a la generación del sweep. Esto es debido a que hasta que no se llame a la función que crea el sweep estos datos no influyen en el comportamiento de la aplicación.

Después se seleccionan los drivers. Al iniciar el programa la función encargada muestra los drivers instalados en el ordenador y se seleccionan a través de un menú desplegable, el de la figura 7. Cuando se seleccionen los drivers, el campo Driver del `AudioDeviceWriter` y del `AudioDeviceReader` adoptan ese valor y este menú es deshabilitado (quedará en gris) para no poder ser modificado. Es solo tras esta selección cuando se activa la función dedicada a mostrar los dispositivos, como se ve en la figura 8. La razón de elegir primero los drivers es que los nombres de los dispositivos pueden cambiar dependiendo del driver. Este menú de dispositivos detectará los dispositivos de audio instalados en el ordenador y los mostrará en el menú desplegable. Como ocurría con el menú de drivers, éste también será deshabilitado al haber sido seleccionada una opción. El campo Device del `AudioDeviceWriter` y del `AudioDeviceReader` cogen el valor seleccionado. En el ejemplo de la figura 8 se tiene conectada una tarjeta de audio M-Audio Fast Track Ultra con ocho salidas y ocho entradas.

Con estas dos selecciones se activan las funciones encargadas de generar gráficamente las salidas y las entradas.

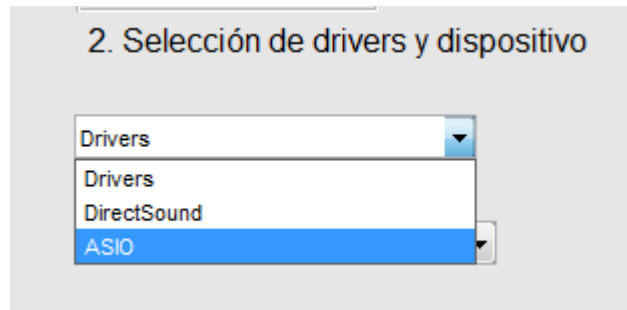


Figura 7. Selección de drivers

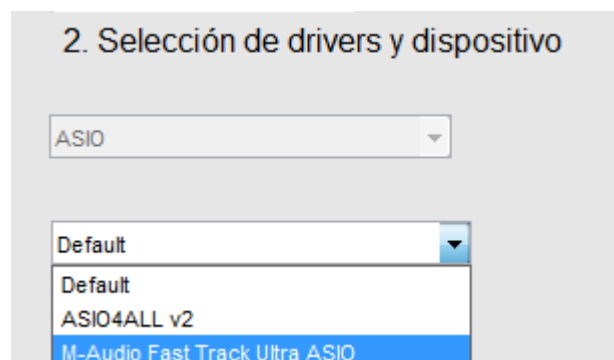


Figura 8. Selección de dispositivos

En el menú de selección de gráficas que se observa en la figura 9 se escoge qué tipo de representación se quiere para las gráficas que se generarán. Esta selección también puede cambiarse hasta antes de generar el sweep por lo que no es definitiva hasta ese momento, solo cuando la función que genera las gráficas es llamada se tiene en cuenta la opción seleccionada.

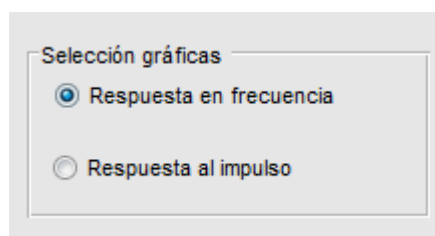


Figura 9. Selección de gráficas

Tras seleccionar el dispositivo, el menú desplegable se deshabilitará y aparecerán en la parte central las entradas y salidas disponibles, representadas por botones. Estos botones se generan de forma dinámica. Se opta por representar las salidas y entradas con botones por permitir que la selección de canales sea más gráfica e intuitiva que introducir el número de canal.

A la izquierda aparecen las salidas y forman parte de un grupo de botones. Un grupo de botones es la forma que tiene el entorno de programación visual GUIDE de agrupar botones. Dentro de un grupo de botones solo puede haber una opción seleccionada a la vez. Cuando se seleccione otra salida, el botón seleccionado anteriormente se desmarca para representar visualmente que ya no está seleccionado. La función está pensada para relacionar cada una de las salidas con una o varias entradas.

La columna derecha son las entradas. Para relacionar las salidas con sus entradas se selecciona la salida que se quiere capturar y a continuación las entradas que se quiere que la capturen. El programa genera una estructura por cada salida seleccionada. Estas estructuras reciben el nombre de *infocanales*. En ellas hay dos campos: el primero es la salida y el segundo, cada una de las entradas que se encuentren seleccionadas. Estas estructuras son las que se encargan de mantener relacionadas las salidas y las entradas. La función que las genera se activa cada vez que se haga una nueva selección en el grupo de botones con las salidas, por tanto se puede modificar la selección de entradas que tiene asignadas una salida volviendo a pulsar en ella.

Cuando se seleccione una nueva salida, los botones que representan las entradas se vaciarán para indicar que se ha seleccionado otra salida y quede claro de forma gráfica que es otra medida diferente.

4.2.3 Generación del sweep logarítmico

El sweep se genera al pulsar el botón generar sweep de la figura 4. Será lanzado tantas veces como salidas se hayan seleccionado y capturado por las entradas relacionadas con cada una de ellas.

La aplicación llama a una función que recoge los datos introducidos en los cuadros que definen los parámetros del sweep explicados en el apartado 4.2.2. Esta función solo es llamada una vez y, por tanto, el sweep solo es generado una sola vez. Es decir, si se han seleccionado varias salidas el sweep lanzado por ellas es exactamente el mismo. Esta función también genera el sweep invertido necesario para la convolución con la señal de salida capturada.

Aparecerá una pequeña ventana emergente como la de la figura 10 informando al usuario de que el proceso de lanzamiento del sweep se ha iniciado.

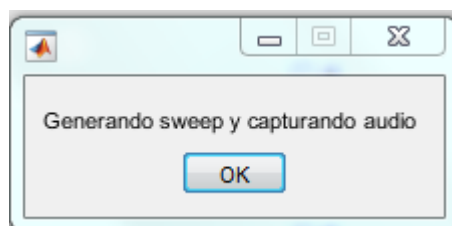


Figura 10. Mensaje de aviso

4.2.4 Captura de las medidas

En este punto el programa lee la información contenida en los campos de cada una de las estructuras anteriormente mencionadas, llamadas *infocanales*.

El bucle recorre las estructuras y a cada iteración leerá el campo con la salida, que será el valor dado a la propiedad ChannelMapping del AudioDeviceWriter y el campo con las entradas, que será el valor de la propiedad ChannelMapping del AudioDeviceReader. De este modo se mapean los canales de salida y entrada del dispositivo.

Tras el mapeo se lanza el sweep por la salida asignada y se captura por las entradas relacionadas.

Una vez la señal ha sido capturada, se convoluciona con el sweep invertido usando una función de convolución rápida que permite realizar esta operación en menor tiempo, con esto se obtiene la respuesta del sistema. Tras esto, la función realiza las operaciones necesarias con cada respuesta para su representación gráfica. Aquí se modifica la estructura *infocanales* que está leyendo el bucle. Se crea en ella un nuevo campo para que guarde la respuesta obtenida por cada entrada. Esta respuesta será una matriz que tendrá tantas columnas como entradas hayan capturado la salida. Esto se ve más en profundidad en el apartado 4.2.6.

4.2.5 Generación de las gráficas

Tras el lanzamiento y captura de los sweeps aparecen tantas ventanas emergentes como medidas se han hecho. En cada una de ellas habrá tantas gráficas como entradas hayan capturado esa salida. En el título de cada ventana aparece el número de medida y encima de cada gráfica la salida y la entrada, como se ve en la figura 11.

Se ha implementado la opción de poder abrir una gráfica en una ventana aparte para poder verla en más detalle si así se quiere. Esto es útil, sobre todo, cuando se captura una salida con muchas entradas y los subplots son pequeños. Para poder hacer esto solo hay que hacer click sobre la gráfica que se quiera ver mejor y ésta se abrirá en una ventana aparte, como se ve en la figura 12.

En función del texto de la opción seleccionada en el recuadro de selección de gráficas de la figura 9, la aplicación hará la representación apropiada.

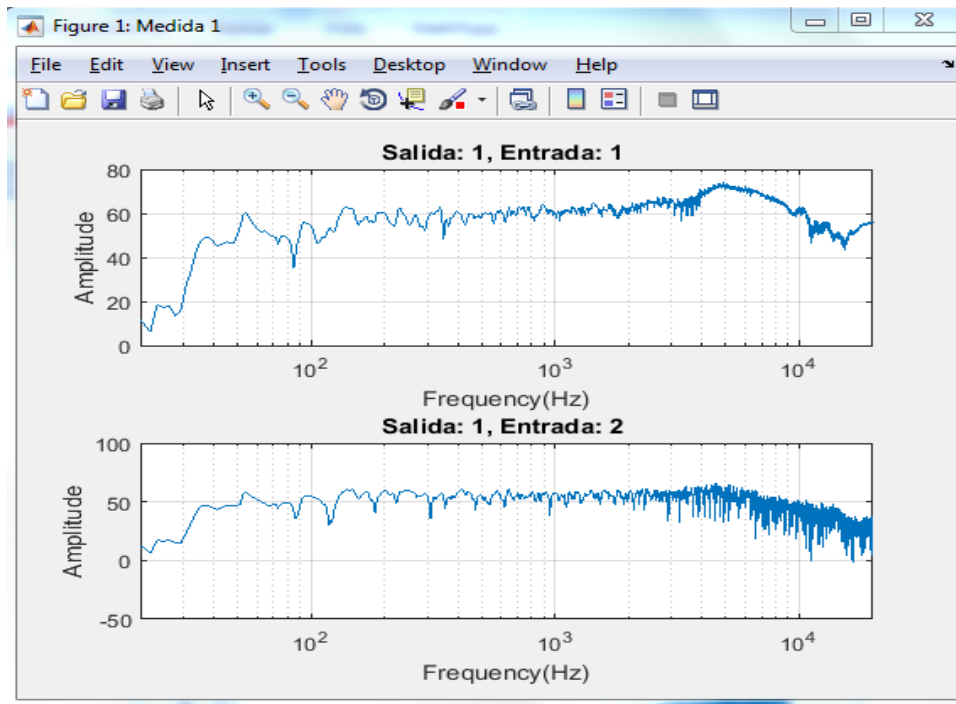


Figura 11. Gráfica de la respuesta en frecuencia con las medidas

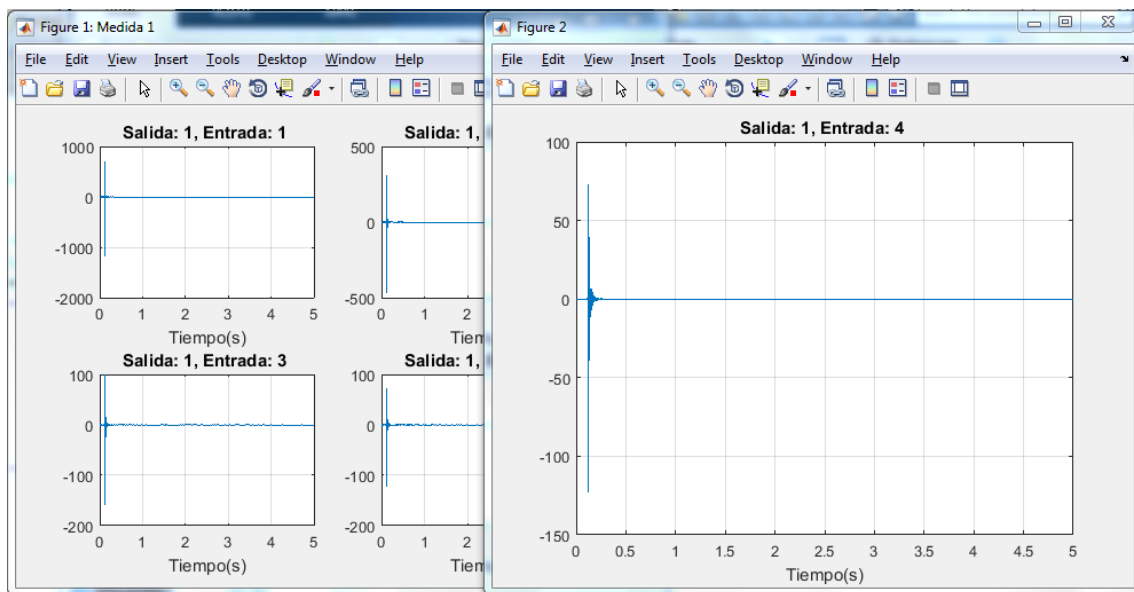
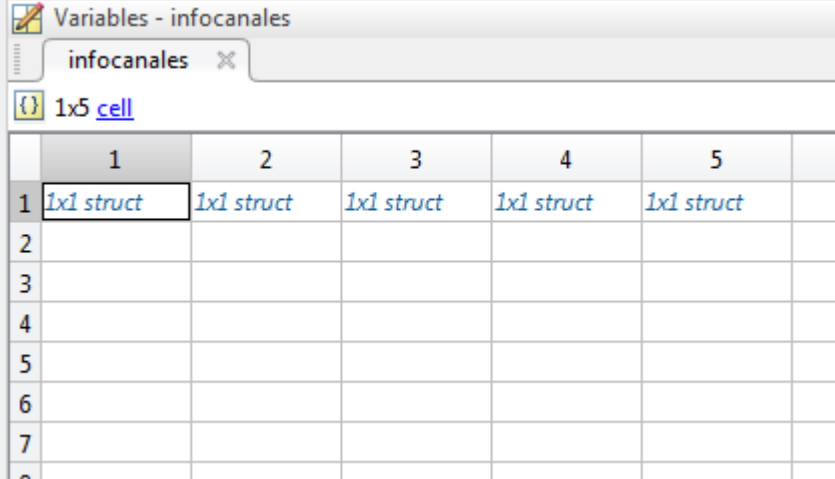


Figura 12. Gráfica de la respuesta al impulso abierta en una ventana aparte

4.2.6 Guardado de los datos

Una vez finalizada la ejecución del programa las respuestas son guardadas en un archivo .mat como el de la figura 13 que se salvará en el directorio en el que se esté trabajando. Este archivo contendrá tantas estructuras, las mismas mencionadas en los apartados 4.2.2 y 4.2.4 con el nombre *infocanales*, como sweeps se hayan lanzado. En el ejemplo de la figura 13 se hicieron cinco medidas, por lo que se generaron cinco estructuras.



	1	2	3	4	5
1	1x1 struct	1x1 struct	1x1 struct	1x1 struct	1x1 struct
2					
3					
4					
5					
6					
7					

Figura 13. Estructuras en el archivo .mat

Cada estructura tendrá los cuatro campos que se ven en la figura 14:

- ADW: Abreviatura de AudioDeviceWriter. Indica el número de canal de salida por el cual se ha lanzado el sweep.
- ADR: Abreviatura de AudioDeviceReader. Muestra los canales de entrada por los que se ha capturado la salida.
- data: Es la respuesta capturada, en forma de matriz, por las entradas que indica el anterior campo. Cada columna es la respuesta capturada por una entrada, por lo que el número de columnas de la matriz se corresponde con el número de entradas.
- fs: Frecuencia de muestreo de la señal. Es de utilidad si más tarde se quiere trabajar con las respuestas.

Field ▲	Value
ADW	1
ADR	[1 2]
data	240640x2 double
fs	48000

Figura 14. Contenido de las estructuras

Capítulo 5. Ejemplos de aplicación

En este capítulo se describen los dos ejemplos de aplicación llevados a cabo con el programa.

El objetivo de estas pruebas es mostrar las posibilidades que ofrece la aplicación. Para demostrarlo se han llevado a cabo dos medidas con distintas configuraciones de entradas y salidas.

La primera es una medida de la respuesta al impulso de una sala. Aquí se usa una configuración con una única salida y múltiples entradas con la finalidad de ver como varía la respuesta en función de la posición en la sala.

La segunda medida es la medida de las HRTFs de un maniquí acústico. Aquí se ve como la aplicación maneja múltiples salidas y múltiples entradas. Además, brinda la posibilidad de, una vez calculada dichas respuestas, convolucionar el resultado con una pista de audio y obtener la auralización binaural.

El material usado es el que se encuentra en el laboratorio del GTAC. En las dos se trabajó con un Windows 7 y el dispositivo de audio fue una tarjeta de audio M-Audio Fast Track Ultra [12] con ocho canales de entrada y ocho de salida.

5.1 Medida de la respuesta al impulso de una sala (una salida y cuatro entradas)

La primera de las aplicaciones prácticas que se llevó a cabo fue el cálculo de la respuesta al impulso del laboratorio de audio del edificio 8G. La sala, como se ve en las figuras 15 y 16, es pequeña y no habrá demasiadas reflexiones. Para realizarla se usaron cuatro micrófonos de medida Earthworks y un altavoz autoamplificado JBL LSR305. Se dispusieron estos elementos de la siguiente forma: se colocó el altavoz sobre un soporte (marcado con el círculo amarillo en la figura 15), situando su centro a una altura de 1,40 metros. Los micrófonos se colocaron en fila, uno detrás de otro, sobre sendos soportes de micrófono a una altura de 1,40 metros apuntando al centro del altavoz. La distancia entre el micrófono más cercano y el altavoz era de 80 centímetros, así como la distancia entre los micrófonos.



Figura 15. Disposición de los micrófonos y altavoz (amarillo)

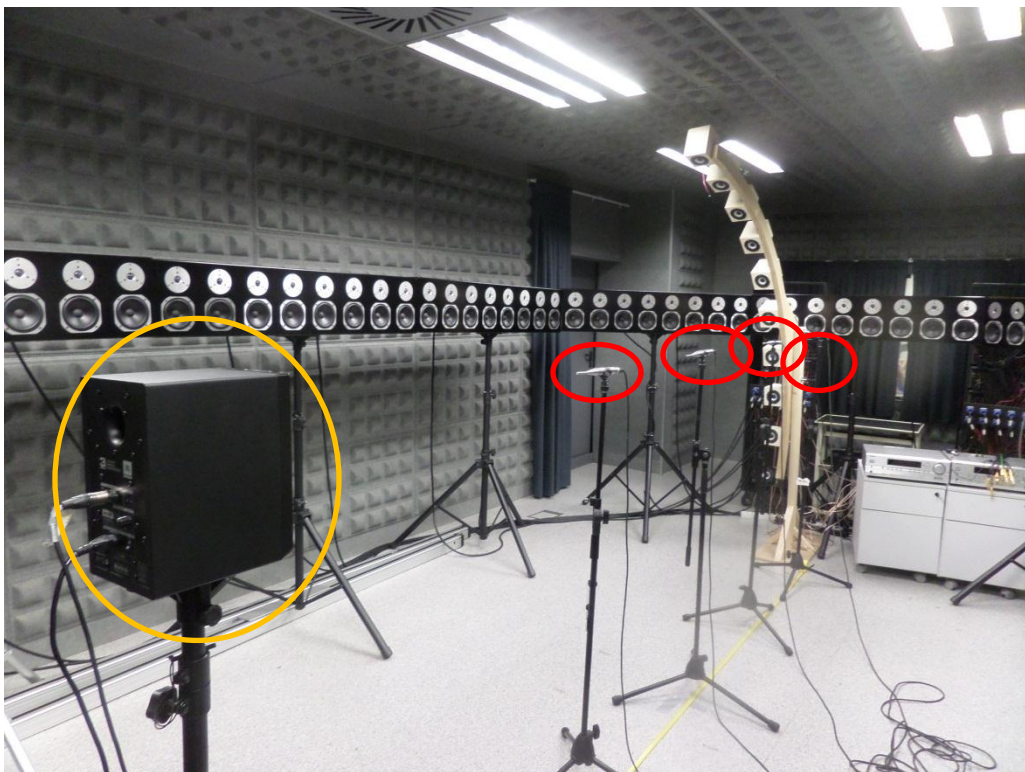


Figura 16. Disposición de los micrófonos (rojo) y altavoz (amarillo)

Una vez hecho el montaje se conectaron los dispositivos a la tarjeta de audio del siguiente modo: el altavoz se conectó a la primera salida y los micrófonos en las entradas de la uno a la cuatro según la distancia, siendo la primera entrada para el micrófono más cercano y la cuarta para el más alejado. Tras esto, se conectó la tarjeta al ordenador y se realizó una comprobación para ver que los micrófonos no recibían demasiado nivel y saturaban. Después de ajustar los niveles y la ganancia de los micrófonos se puso en marcha la aplicación. Se configuró un sweep con una frecuencia de muestreo de 48000 Hz que recorriera las frecuencias de 20 a 20000 Hz con una duración de 5 segundos. La representación elegida fue de respuesta al impulso.

1. Selección de los parámetros del sweep:

Frecuencia de muestreo: 48000

Frecuencia de inicio: 20

Frecuencia final: 20000

Duración: 5

2. Selección de drivers y dispositivo:

ASIO

M-Audio Fast Track Ultra ASIO

Selección gráficas:

Respuesta en frecuencia

Respuesta al impulso

Seleccione una salida:

1

2

3

4

5

6

7

8

Canales de entrada:

1

2

3

4

5

6

7

8

Generar sweep

Instrucciones

Figura 17. Parámetros de la aplicación en la medida

Tras esto, se obtienen las gráficas y se guardan los resultados automáticamente como se ha explicado más arriba en el capítulo 4, apartado 2.6. Con esto ya se podía analizar los resultados y comprobar si eran correctos.

Como se ha dicho más arriba, la respuesta al impulso nos indica cómo se comporta un sistema en el dominio del tiempo. Por tanto, lo primero que esperamos ver en las gráficas es que la respuesta en cada uno de los puntos estará desplazada en el tiempo. La señal emitida llega a cada entrada en un instante diferente al estar cada micrófono a una distancia distinta del altavoz. Además, por esta misma razón las respuestas deberían tener menos amplitud y deberían verse más reflexiones a medida que se alejan de la fuente. En el caso de esta sala en particular, las reflexiones no deberían ser muchas. Para ilustrar los resultados, la figura 18 ayuda a entender lo que se espera de estas medidas.

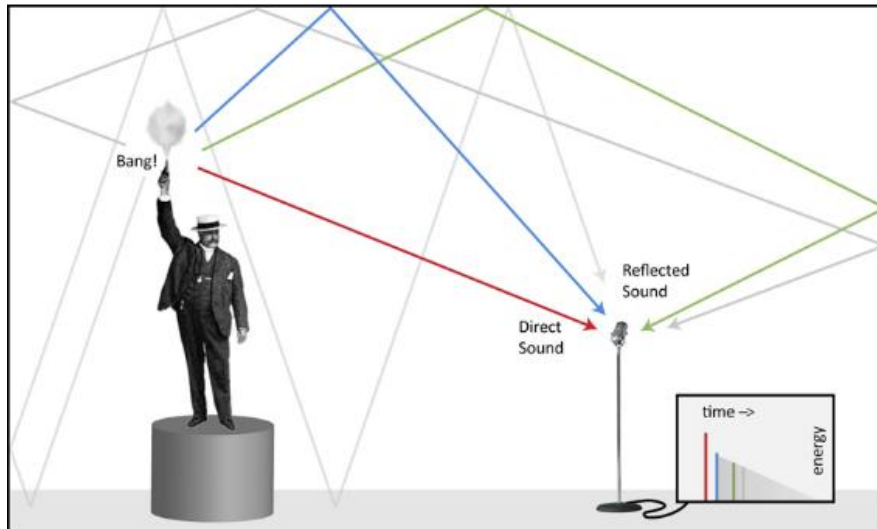


Figura 18. Ilustración respuesta al impulso

Por tanto, en las gráficas se aprecia cómo el sonido directo es el primero en llegar y que es el que tiene más nivel. Después estarán las reflexiones, que al tomar caminos más largos para alcanzar el micrófono, llegarán más tarde. Además, a causa de los rebotes con los distintos elementos de la sala y las paredes, estas reflexiones habrán perdido energía y llegarán con un nivel menor [13]. Como se ve en las figuras 15 y 16, la sala medida se usa para medidas acústicas y sus paredes están recubiertas para evitar reflexiones. Además, los distintos elementos presentes en la sala también actuarán como absorbentes acústicos haciendo que en estas medidas las reflexiones sean pequeñas. En las gráficas el eje horizontal está en el dominio del tiempo y el vertical es la energía.

Con los datos guardados por la aplicación se procede a representar gráficamente los resultados.

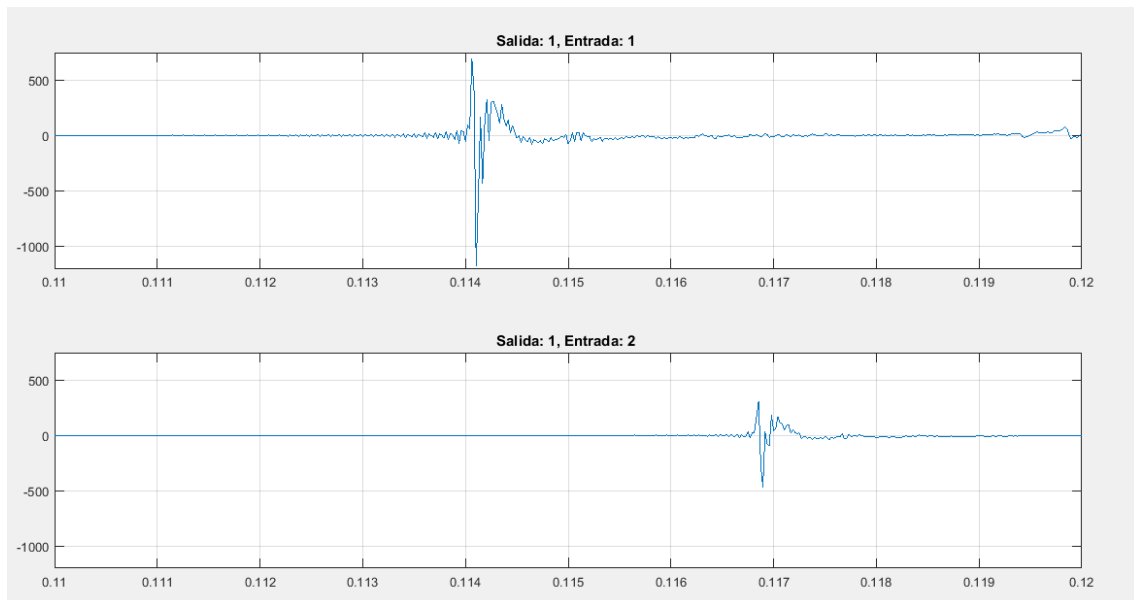


Figura 19. Respuesta al impulso en las entradas 1 y 2

La figura 19 corresponde a las respuestas de los dos micrófonos más cercanos. Los resultados son los esperados pues se aprecia lo dicho anteriormente. El sonido directo en la primera llega antes en el tiempo y tiene un nivel mayor pues corresponde al micrófono más cercano. La gráfica correspondiente a la segunda entrada tiene un nivel menor y el sonido directo llega con cierto retraso respecto a la primera. A continuación se representan las respuestas de las entradas tres y cuatro, es decir, las dos más alejadas. Es de esperar que sus niveles sean más bajos de forma progresiva y que la llegada del sonido directo sea más tardía. También se aprecian en las gráficas de la figura 20 las reflexiones del suelo, que se pueden ver a continuación del sonido directo.

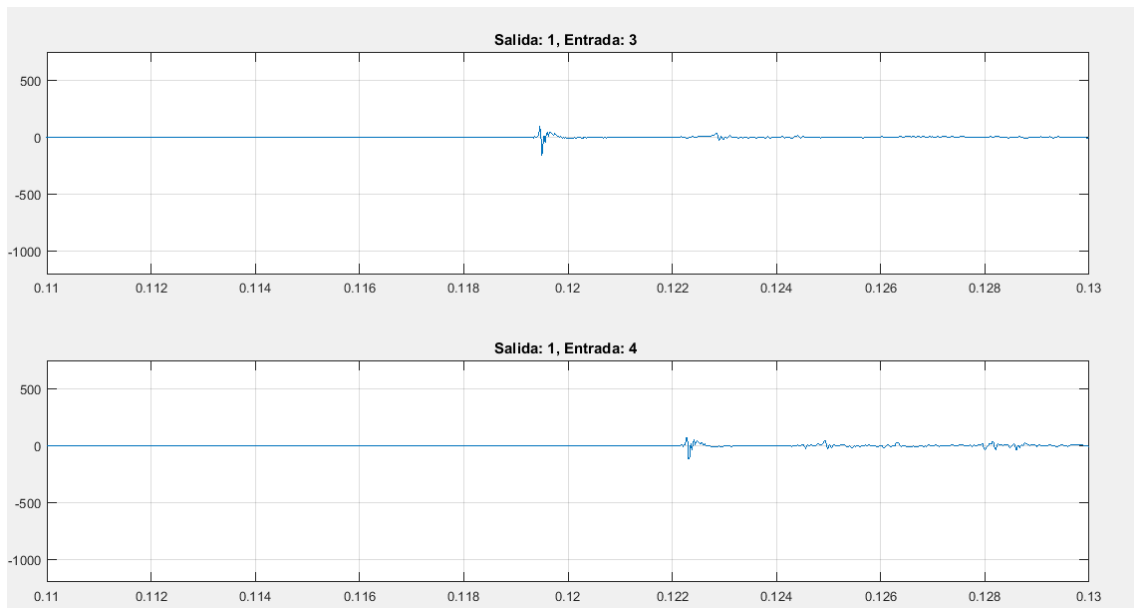


Figura 20. Respuesta al impulso en las entradas 3 y 4

Se aprecia en las gráficas los resultados esperados: la llegada del sonido directo está desplazada en cada una de las representaciones y su nivel va decreciendo. Nótese que en la figura 19 el eje X (tiempo) no es el mismo que en la figura 20. Esto se debe a que al estar desplazadas no aparecerían en la gráfica si se toman los mismos valores temporales.

La sala está adaptada para una escucha neutra ya que su principal función es la de tomar medidas acústicas y, por tanto, no se aprecian en las gráficas ni reverberaciones ni reflexiones que destaquen.

Tras la realización de estas medidas, vemos, siguiendo el método de este ejemplo de aplicación, que ofrece la posibilidad de medir la respuesta al impulso de una sala con una acústica característica, como una sala de conciertos, y a continuación convolucionar dicha respuesta con una pista de audio para así escucharla como si se estuviera presente en esa misma sala.

Se ha convolucionado una señal de audio con las respuestas obtenidas y, como era de esperar, no se parecía ningún efecto destacable en la señal de salida por las características acústicas de la sala.

Estas convoluciones han sido guardadas en formato .wav para poder ser escuchadas en un reproductor multimedia.

5.2 Medida de HRTF de un maniquí acústico (seis salidas y dos entradas)

El siguiente ejemplo de aplicación consiste en la medición de las HRTFs de un maniquí acústico. Para ello se utilizó un maniquí acústico Type 4100 de Brüel & Kjaer, dos altavoces autoamplificados JBL LSR305 y cuatro altavoces Apart SDQ5P.

El sistema auditivo usa la diferencia de nivel interaural (Interaural Level Differences, ILD) y la diferencia de tiempos interaural (Interaural Time Differences, ITD) para localizar un sonido en el espacio. Dependiendo de la dirección del sonido el sistema auditivo filtra la señal y ayuda a determinar la posición y dirección de la misma. Esta señal última estará caracterizada por la función de transferencia del sistema auditivo, esto es la HRTF. La HRTF es la función de transferencia relativa a la cabeza (Head Related Transfer Function) y se miden por parejas, una para cada oído. Estas respuestas, como su nombre indica, vendrán determinadas por la cabeza, el torso, los hombros, la forma de las orejas e incluso por el propio canal auditivo [14].

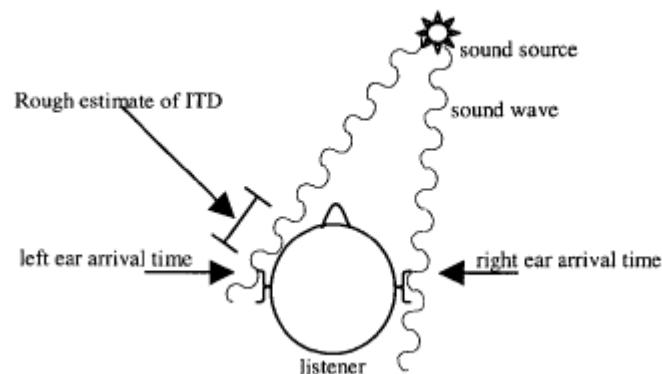


Figura 21. Esquema ILD e ITD

Para la realización de estas medidas pueden usarse maniqués acústicos. El maniquí usado para este ejemplo de aplicación, en la figura 22, está compuesto por un torso y una cabeza. Para que las medidas sean lo más parecidas a las tomadas a una persona real, estos maniqués suelen tener nariz y orejas con las partes del pabellón auditivo definidas. Donde debería estar el canal auditivo cuenta con dos micrófonos, que serán las dos entradas usadas. La entrada uno será el micrófono izquierdo y la entrada dos, el derecho.

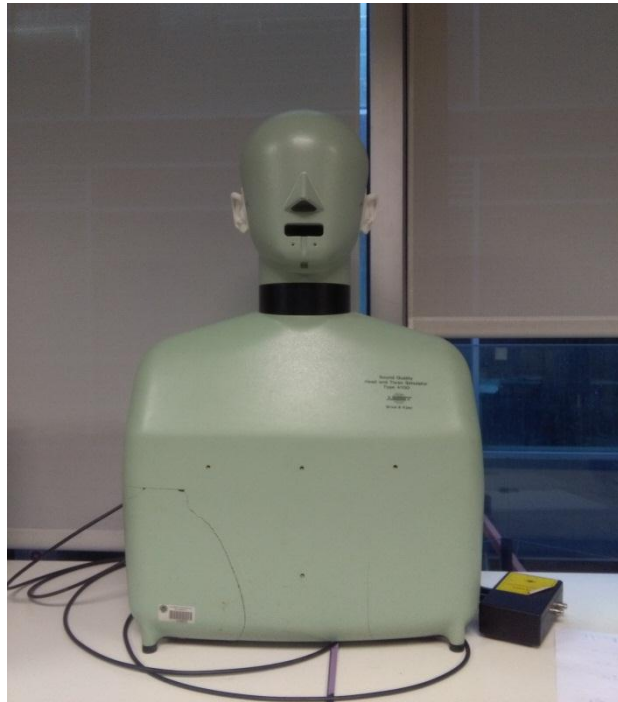


Figura 22. Maniquí acústico Brüel & Kjaer Type 4100

En el montaje de la prueba los elementos se dispusieron como se ve en las figuras 23 y 24

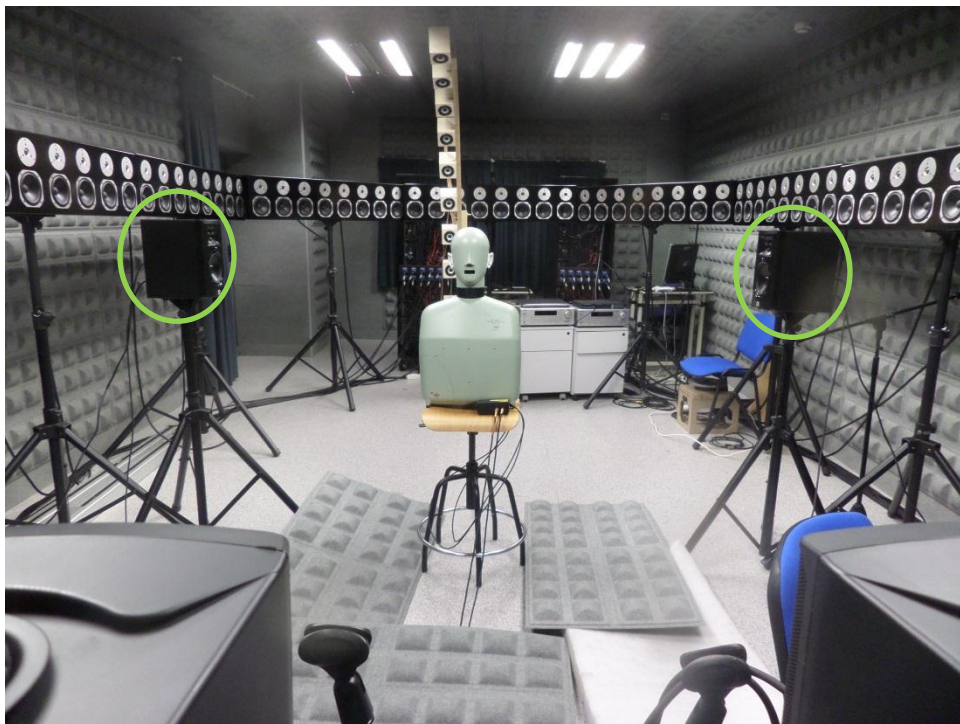


Figura 23. Disposición de los elementos. Altavoces a -90° y 90° en verde

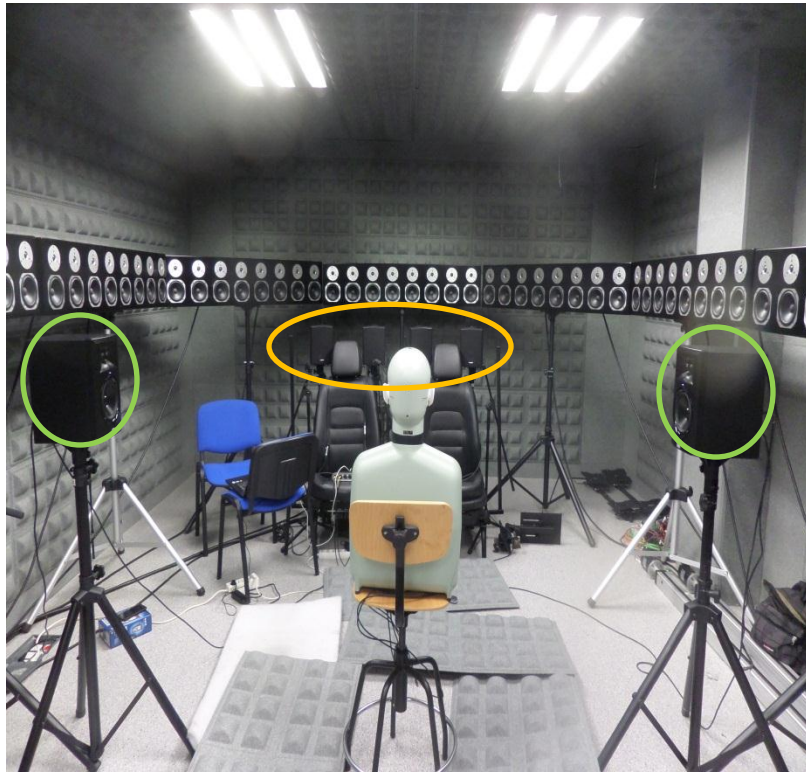


Figura 24. Disposición de los elementos. Altavoces a -90° y 90° (verde) Cuatro altavoces delanteros (amarillo)

El maniquí se colocó en el centro de la sala encima de un taburete para que estuviera a la misma altura que los altavoces. Los dos altavoces autoamplificados JBL LSR305 se colocaron uno a la izquierda y otro a la derecha del maniquí con un azimut de -90° (lado izquierdo del maniquí) y 90° (lado derecho) y a una distancia de 1.10 metros respecto a él. Los cuatro altavoces Apart SDQ5P (marcados en amarillo en la figura 24) se colocaron enfrente del maniquí a una distancia de 2.5 metros uno al lado del otro. En el suelo se dispusieron unas láminas de material absorbente para mitigar todo lo posible las reflexiones de la sala capturadas por los micrófonos.

En este ejemplo de aplicación el número de micrófonos que capturan es dos y están situados en el canal auditivo del maniquí. El oído izquierdo se corresponde con el canal de entrada uno y el derecho con el canal de entrada dos.

Los altavoces se conectaron a la tarjeta de sonido siendo el altavoz colocado a la izquierda del maniquí la salida uno, los cuatro altavoces delanteros (de izquierda a derecha según la figura 24) a las salidas dos, tres, cuatro y cinco y el altavoz a la derecha del maniquí a la salida seis.

Tras establecer las conexiones se pasó a ver con qué nivel llegaba a señal de las distintas fuentes a los micrófonos. Se buscaba que este nivel fuese uniforme y que todas llegaran con el mismo. También se comprobó que en ningún momento los micrófonos saturasen pues la medida hubiera sido incorrecta.

Una vez hecho esto, se procede al lanzamiento de los sweeps. Se configura un sweep de 20 a 20000 Hz con una frecuencia de muestre de 48000 Hz y una duración de 5 segundos. Cada salida se asocia a las entradas uno y dos y se lanzan seis sweeps seguidos, cada uno por la salida correspondiente. Tras esto, el programa guarda los datos y se procede a trabajar con ellos.

Para la representación se han usado ejes logarítmicos tanto para la frecuencia, en Hz, como para la magnitud, en dB.

Los resultados obtenidos son los que se muestran a continuación.

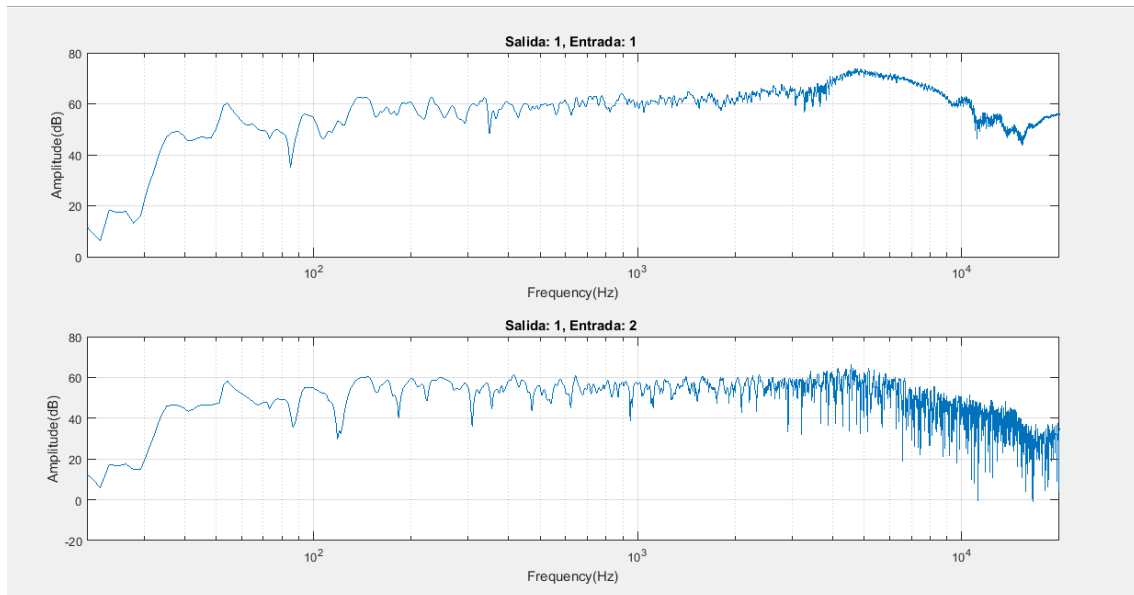


Figura 25. HRTF salida 1 (-90°)

La figura 25 son las medidas de la salida uno, es decir, la situada a -90° . Se aprecia claramente la diferencia de niveles (ILD) ya que la gráfica correspondiente al oído izquierdo tiene un nivel mayor que la del oído derecho, que además recibe más reflexiones. Esto se debe a que la oreja situada enfrente del altavoz es la que recibe de forma directa la señal mientras que la opuesta está en la parte tapada por la cabeza (zona de sombra) [14].

En la figura 26 se ve el mismo resultado pero invertido. Ahora es evidente que la oreja en la zona de sombra es la izquierda, pues el altavoz situado a 90° (salida 6) radia la señal directamente sobre el oído derecho.

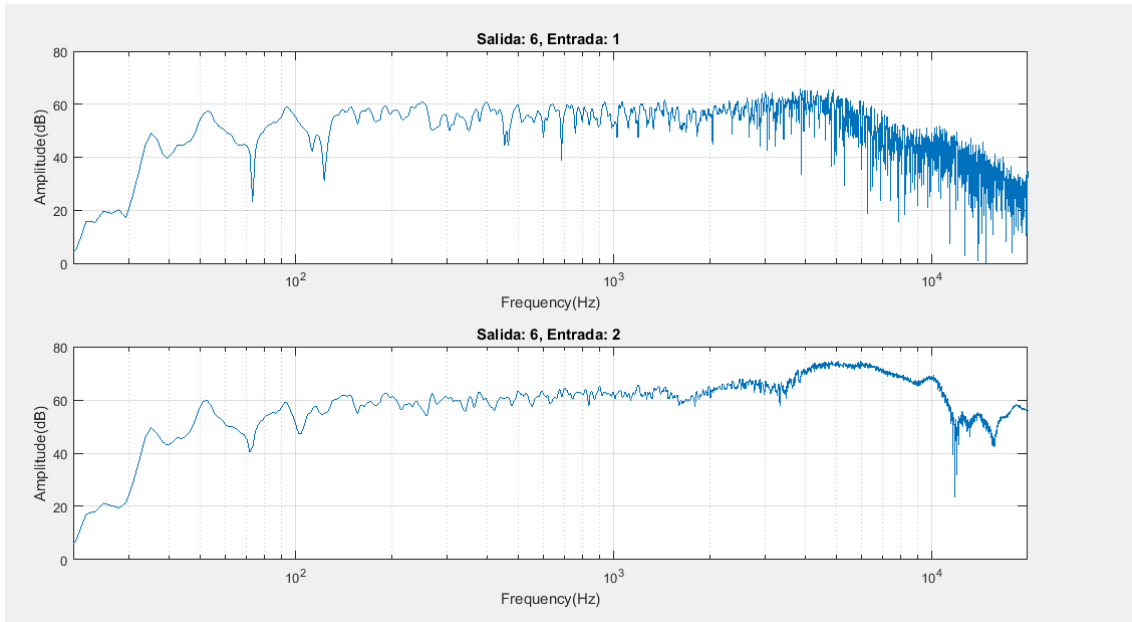


Figura 26. HRTF salida 6 (90°)

Los resultados obtenidos para la medida de la salida dos se muestran en la figura 27.

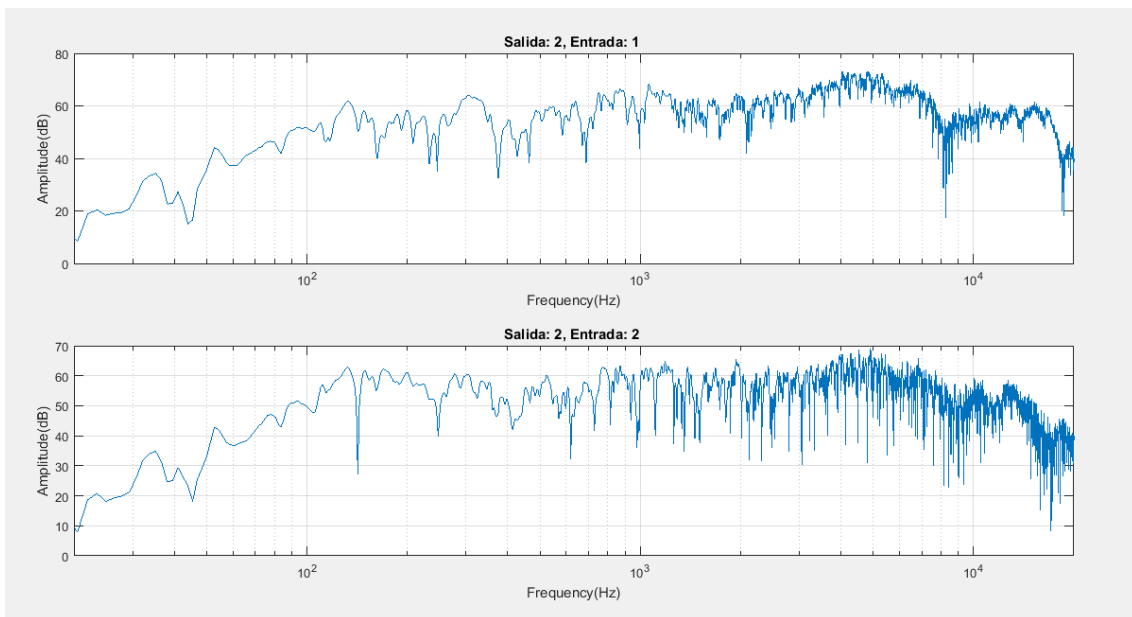


Figura 27. HRTF salida 2 (-25°)

Este es el altavoz frontal situado más a la izquierda en la figura 24 (rodeados en amarillo). El ángulo respecto al maniquí es de -25° por lo que el oído izquierdo sigue recibiendo algo más de nivel que el derecho y, por tanto, su espectro tiene menos reflexiones

Las figuras 28 y 29 muestran las medidas de las salidas tres y cuatro, respectivamente.

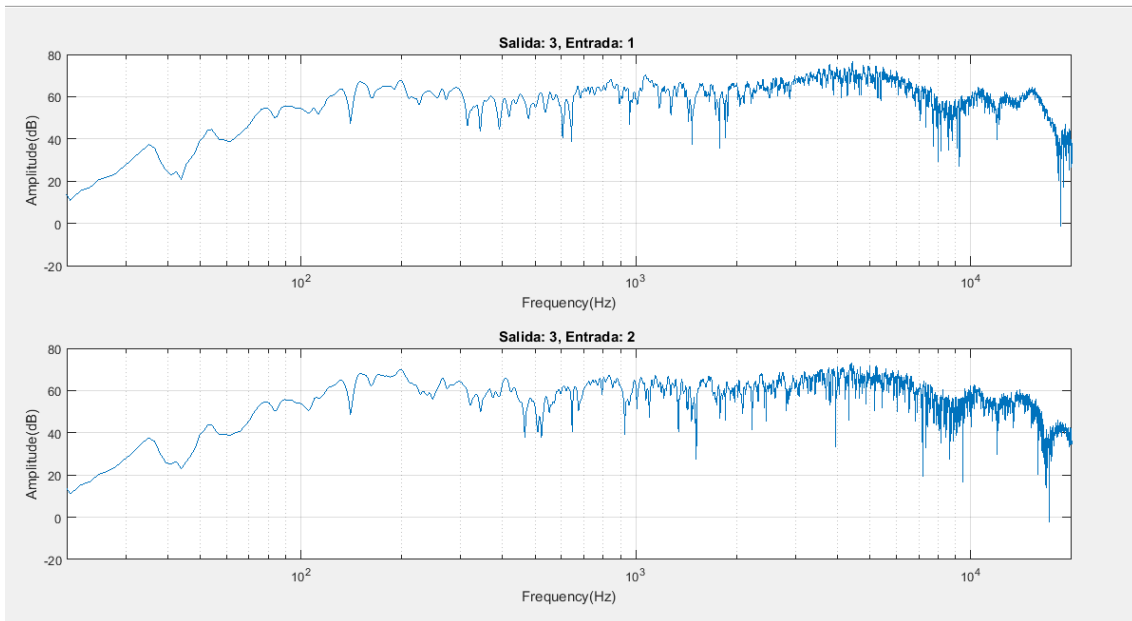


Figura 28. HRTF salida 3 (0° aproximadamente)

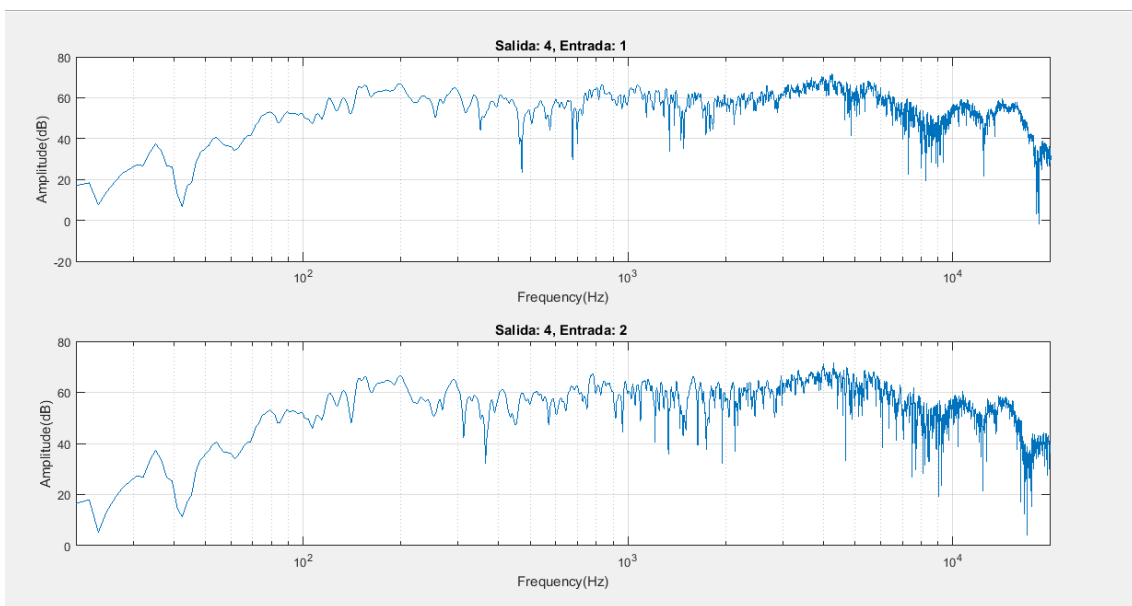


Figura 29. HRTF salida 4 (0° aproximadamente)

Estos dos altavoces, que son los dos más al centro de los rodeados en amarillo en la figura 24, están a aproximadamente 0°. Por tanto, el nivel que llega a los dos oídos es prácticamente el mismo. Esto se debe a que el camino que hay entre el altavoz y cada uno de los micrófonos es aproximadamente igual.

Por último queda ver la medida de la salida cinco, que es el altavoz delantero situado más a la derecha en la figura 24 (rodeados en amarillo).

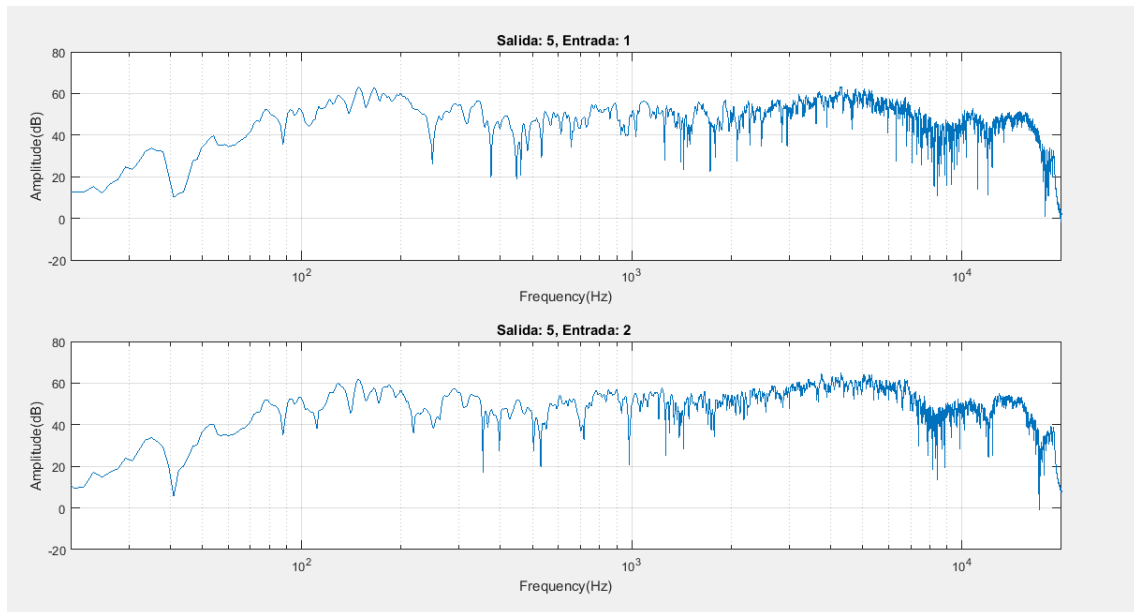


Figura 30. HRTF salida 5 (25°)

Este altavoz está a 25° respecto al maniquí por lo que la señal llega de forma más directa al oído derecho. En la figura 30 se aprecia como la gráfica correspondiente al oído izquierdo tiene más reflexiones que el derecho.

Para mostrar las posibilidades que ofrece la aplicación, aparte de la realización de las medidas anteriores, se han convolucionado las respuestas obtenidas para cada medida con una señal de audio. De esta forma, se pueden escuchar los resultados obtenidos y su correspondencia con las gráficas generadas.

Con esto se consigue una sensación de sonido espacial muy espectacular al ser escuchados. Por tanto, en la grabación binaural de cada salida se percibe como la señal proviene de la posición en la que se encontraba el altavoz. Si se escuchan seguidas y en orden, se aprecia claramente como la señal va “desplazándose” dando la sensación de estar en el mismo entorno acústico que el maniquí.

Estas convoluciones han sido guardadas en formato .wav para poder ser escuchadas en un reproductor multimedia.

Capítulo 6. Conclusiones y propuestas de trabajo futuro

6.1 Conclusiones

En este Trabajo Fin de Grado se ha desarrollado una aplicación capaz de medir sistemas de audio y se puede concluir que los objetivos marcados al inicio del mismo han sido alcanzados.

La interfaz gráfica desarrollada para el programa tiene un uso intuitivo y cómodo para el usuario y facilita la medida de sistemas de audio.

El usuario tan solo debe introducir los datos del sweep logarítmico, que son la frecuencia de muestreo, la frecuencia inicial del sweep, la frecuencia final del sweep y la duración de éste. A continuación, se seleccionan los drivers y el dispositivo. En función de esta selección aparecerán un número de botones generados dinámicamente que representan las salidas y entradas del dispositivo. Después seleccionará las salidas por las que se lanzará el sweep y las entradas que capturarán la señal emitida. También se elige qué tipo de representación gráfica se quiere para la respuesta que se obtendrá. Al pulsar en el botón que genera el sweep logarítmico empezarán a realizarse las medidas. Tras esto se podrá ver de forma inmediata la representación gráfica de la respuesta obtenida. Si se desea ver en más detalle alguna de las gráficas, solo hay que pulsar sobre ella para que se abra en una ventana independiente.

Los datos obtenidos son guardados en el directorio de trabajo en un archivo .mat que permite que sean usados más adelante. Como se ve con las respuestas obtenidas en los ejemplos de aplicación, que han sido convolucionadas para poder ser escuchadas después.

Estos dos ejemplos de aplicación llevados a cabo demuestran la utilidad y versatilidad de la aplicación desarrollada.

A nivel personal este trabajo ha supuesto la adquisición de nuevos conocimientos sobre las medidas de sistemas de audio y la ampliación de los ya obtenidos durante los estudios. También ha ayudado conocer más a fondo MATLAB y su entorno. El desarrollo de una aplicación desde cero es un reto y más teniendo en cuenta que tanto la librería Audio System Toolbox como el entorno de programación gráfica GUIDE eran completamente nuevos para mí. Por esta razón considero que el trabajo me ha resultado de gran utilidad no solo para adquirir los conocimientos necesarios para el desarrollo de la aplicación, sino también para potenciar la capacidad de trabajo independiente y la búsqueda de información desde diversas fuentes. La realización de este trabajo me ha hecho incrementar mi capacidad para trabajar de forma organizada, tanto en el puesto de trabajo como a nivel de planificación temporal. Esto se ha visto acentuado en el desarrollo de los dos ejemplos de aplicación explicados en su respectivo capítulo. Seguir una metodología y organizarse es casi la mitad del trabajo pues cuando se trabaja con diversos sistemas de audio y componentes no resulta difícil confundirse cuando se

presenta algún inconveniente y perder tiempo intentando averiguar qué puede estar saliendo mal.

En resumen, este Trabajo Fin de Grado ha supuesto un crecimiento tanto a nivel de académico como a nivel personal.

6.2 Propuestas de trabajo futuro

En esta sección se listan una serie de propuestas de trabajo futuro que pueden servir como guía para continuar con el desarrollo de la aplicación:

- Diseño gráfico de la aplicación más atractivo. Aunque la interfaz desarrollada es lo suficientemente intuitiva y fácil de manejar, el entorno de programación visual GUIDE cuenta con una serie de funcionalidades que no se han explorado en este trabajo y que harían que la interfaz gráfica del programa tuviese un aspecto más llamativo para el usuario.
- Convertir la aplicación en independiente de MATLAB. MATLAB cuenta con un compilador que convierte el programa a archivo .exe que puede ejecutarse en cualquier ordenador sin necesidad de tener MATLAB instalado. Con esto se mejoraría la portabilidad del programa al facilitar su uso independientemente de MATLAB.
- Realización de pruebas adicionales para ampliar las capacidades de la aplicación. Como su uso para el cálculo de parámetros acústicos de una sala como el RT_{60} .
- Crear una funcionalidad de guardado de archivos en la que el usuario decida cómo y dónde guardar los datos obtenidos desde la misma interfaz gráfica.

Capítulo 7. Bibliografía

- [1] Angelo Farina, "Simultaneous Measurement of Impulse Response and Distortion with a Swept-Sine Technique," pp. 1-25, Febrero 2000.
- [2] (2017) Audio System Toolbox™ Release Notes. [Online]. https://es.mathworks.com/help/pdf_doc/audio/rn.pdf
- [3] Audio System Toolbox Documentation. [Online]. <https://es.mathworks.com/help/audio/index.html>
- [4] (2017) Audio System Toolbox™ Reference. [Online]. https://es.mathworks.com/help/pdf_doc/audio/audio_ref.pdf
- [5] (2017) MATLAB® App Building. [Online]. https://www.mathworks.com/help/pdf_doc/matlab/buildgui.pdf
- [6] Diego Orlando Barragán Guerrero. [Online]. https://www.dsplace.espol.edu.ec/bitstream/123456789/10740/11/MATLAB_GUIDE.pdf
- [7] Mark Kahrs and Karlheinz Branderburg, Eds., *Applications of digital signal processing to audio and acoustics.*: Kluwer Academic Publishers, 2002.
- [8] Stack Overflow. (2017, Junio) Signal Processing Stack Exchange. [Online]. <https://dsp.stackexchange.com/questions/536/what-is-meant-by-a-systems-impulse-response-and-frequency-response>
- [9] D. Sen, S. Wang y L. Hayes Q. Meng, "Impulse response measurement with sine sweeps and amplitude modulation schemes," *2nd International Conference on Signal Processing and Communication Systems*, pp. 1-5, Diciembre 2008.
- [10] José Javier López Monfort, *Tratamiento digital de audio. Tema 2. Filtros digitales de audio.*, Curso 2015-2016.
- [11] ASIO4ALL - Universal ASIO Driver For WDM Audio. [Online]. <http://www.asio4all.com/>
- [12] M-Audio, *Fast Track Ultra Manual del usuario.*
- [13] ProSoundWeb. [Online]. http://www.prosoundweb.com/channels/live-sound/what_is_an_impulse_response/#
- [14] György Wersényi, "Representations of HRTFs using MATLAB: 2D and 3D," in *20th International Congress on Acoustics*, Sydney, 2010, pp. 1-9.

Capítulo 8. Anexos

En este apartado se han adjuntado los códigos comentados del programa desarrollado. También se han adjuntado, como documentación electrónica, las convoluciones de una señal de audio con las respuestas obtenidas en el ejemplo de aplicación 5.1 y 5.2.

8.1 Código de la aplicación (interfaz gráfica)

```
function varargout = app_medidas_sweep_log_v2(varargin)

% Autor: Pere Castán Orero

% Última versión: Junio 2017

% Descripción: El siguiente código MATLAB permite la medida de la
% respuesta
% de sistemas de audio y acústicos. Al ejecutarlo se abrirá una
% interfaz
% gráfica que pedirá insertar los datos para la generación de un sweep
% logarítmico que excitará el sistema bajo prueba.
% Permite elegir los drivers y dispositivos que estén conectados al
% ordenador y qué tipo de representación gráfica se quiere.
% Se elegirán los canales de salida y entrada del dispositivo y se
% lanzará
% el sweep.
% La respuesta a cada sweep será capturada, representada y almacenada.

% APP_MEDIDAS_SWEEP_LOG_V2 MATLAB code for
app_medidas_sweep_log_v2.fig
%     APP_MEDIDAS_SWEEP_LOG_V2, by itself, creates a new
APP_MEDIDAS_SWEEP_LOG_V2 or raises the existing
%     singleton*.
%
%     H = APP_MEDIDAS_SWEEP_LOG_V2 returns the handle to a new
APP_MEDIDAS_SWEEP_LOG_V2 or the handle to
%     the existing singleton*.
%
%
APP_MEDIDAS_SWEEP_LOG_V2('CALLBACK', hObject, eventData, handles,...)
calls the local
%     function named CALLBACK in APP_MEDIDAS_SWEEP_LOG_V2.M with the
given input arguments.
%
```

```

% APP_MEDIDAS_SWEEP_LOG_V2('Property','Value',...) creates a new
APP_MEDIDAS_SWEEP_LOG_V2 or raises the
% existing singleton*. Starting from the left, property value
pairs are
% applied to the GUI before app_medidas_sweep_log_v2_OpeningFcn
gets called. An
% unrecognized property name or invalid value makes property
application
% stop. All inputs are passed to
app_medidas_sweep_log_v2_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows
only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
app_medidas_sweep_log_v2

% Last Modified by GUIDE v2.5 22-Jun-2017 21:46:37

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', ...
@app_medidas_sweep_log_v2_OpeningFcn, ...
                  'gui_OutputFcn', ...
@app_medidas_sweep_log_v2_OutputFcn, ...
                  'gui_LayoutFcn', [] , ...
                  'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before app_medidas_sweep_log_v2 is made visible.
function app_medidas_sweep_log_v2_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to app_medidas_sweep_log_v2 (see
VARARGIN)

%Declaramos variables globales para que todas las funciones puedan
acceder
global q z in_call
q = 0; % Contador
z = 0; % Contador
in_call = 0; %Contador

```

```

% Choose default command line output for app_medidas_sweep_log_v2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes app_medidas_sweep_log_v2 wait for user response (see
UIRESUME)
% uiwait(handles.background);

% --- Outputs from this function are returned to the command line.
function varargout = app_medidas_sweep_log_v2_OutputFcn(hObject,
eventdata, handles)
% varargout    cell array for returning output args (see VARARGOUT);
% hObject     handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function fs_Callback(hObject, eventdata, handles)

global fs

fs = str2double(get(hObject,'String'));

% --- Executes during object creation, after setting all properties.
function fs_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function fstop_Callback(hObject, eventdata, handles)

global fstop

fstop = str2double(get(hObject,'String'));

function fstop_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function duration_Callback(hObject, eventdata, handles)

global duration

```



```

duration = str2double(get(hObject,'String'));

% --- Executes during object creation, after setting all properties.
function duration_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function fstart_Callback(hObject, eventdata, handles)

global fstart

fstart = str2double(get(hObject,'String'));

function fstart_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% Función que selecciona el dispositivo de audio
function pumadw_Callback(hObject, eventdata, handles)

get(handles.pumadw,'Value');
global ADW ADR inputs

allAdwDrivers = get(handles.pumdriver,'String')
selectedDriver = get(handles.pumdriver,'Value');
ADW.Driver = allAdwDrivers{selectedDriver};
allAdwDevices = get(handles.pumadw,'String')
selectedDevice = get(handles.pumadw,'Value');
ADW.Device = allAdwDevices{selectedDevice};
ADR.Device = ADW.Device;

% Deshabilita el menú de dispositivo
hObject.Enable = 'Off';

ADW_Channel

function pumadw_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

%% Función que establece el driver que va a usar el dispositivo de
audio
function pumdriver_Callback(hObject, eventdata, handles)

global ADW ADR

ADW=audioDeviceWriter;
ADR=audioDeviceReader;
allAdwDrivers = get(handles.pumdriver,'String')
selectedDriver = get(handles.pumdriver,'Value');
ADW.Driver = allAdwDrivers{selectedDriver};
ADR.Driver = ADW.Driver
dev=getAudioDevices (ADW);

set(handles.pumadw,'String',dev);

%Deshabilita el menú de driver
hObject.Enable = 'Off';

% --- Executes during object creation, after setting all properties.
function pumdriver_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in pumadr.
function pumadr_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function pumadr_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%% Función para seleccionar la salida. Crea un grupo de botones de
forma dinámica
function ADW_Channel
global ADW ADW_out

bg = uibuttongroup('Title', 'Seleccione una salida',...
    'Units', 'normalized',...
    'Position',[0.4 -0.05 .15 1],...
    'BackgroundColor', [0.902, 0.902, 0.902], ...
    'SelectionChangedFcn',@bselection);

for i=1:length(ADW.ChannelMapping)
    rbutton=uicontrol(bg, 'Style', 'radiobutton', ...
        'units', 'normalized', ...
        'Position', [0.2 0.95-(i/15) 0.5 0.1], ...
        'BackgroundColor', [0.902, 0.902, 0.902],
    ...
        'String', i);

```

```

        if (i==1)
            rbutton.Value = 0;
        end

    end

    bg.Visible = 'on';

    entradasmicro

    %% Función para seleccionar las entradas. Genera los botones de forma
    dinámica
    function entradasmicro
    global ADR
    textbox = uicontrol ('Style', 'Text', 'String', 'Canales de entrada',
    'Units', 'normalized',...
        'Position', [0.56 0.90 0.1 0.05],
    'BackgroundColor', [0.902, 0.902, 0.902]);

    %%Crea botones de forma dinámica.
    for i=1:length(ADR.ChannelMapping)
    handles.entrada=uicontrol('Style', 'radiobutton', ...
        'Callback', @entradas_Callback, ...
        'Units', 'normalized', ...
        'Position', [0.6 0.90-(i/15) 0.05 0.05],
    ...
        'BackgroundColor', [0.902, 0.902, 0.902],
    ...
        'String', i, ...
        'Value', 0);

    end

    function entradas_Callback(hObject, eventdata, handles)

    global in_call

    in_call = in_call + 1;

    %%Push button para generar el sweep
    if (in_call == 1)
        generasweep = uicontrol ('Style', 'pushbutton', 'String', 'Generar
    sweep',...
            'FontSize', 12,'Callback',
    @gensweep_Callback, 'Units', 'normalized', ...
            'Position', [0.8 0.5 0.1 0.1]);
    end

    %% Se ejecuta cada vez que se cambia la entrada.
    function bselection(bg,event)

    global ADR_in z a infocanales new_value

    z = z + 1;

    %% Sirve para coger el valor del último canal de salida que
    seleccionemos, lo relacionará

```

```

% con las entradas al darle al botón gensweep (sólo sucede en el
último)
new_value = str2double(event.NewValue.String);

% Entrará a partir de la segunda entrada seleccionada
%Cada vez que se selecciona un nuevo canal coge el valor del anterior
Y
% lo guarda en una estructura junto con las entradas correspondientes

if (z ~= 1)
    ADR_in = findobj('Callback', @entradas_Callback,'Value', 1);
    in=zeros(1,length(ADR_in));
    for i=1:length(ADR_in)
        in(i)=str2double(ADR_in(i).String);
    end
    in=fliplr(in);
    a = str2double(event.OldValue.String);
    infocanales{a} = struct('ADW', a, 'ADR', in);
end

for i=1:length(ADR_in)
    ADR_in(i).Value = 0;
end

%% Función que recopila la información introducida, genera el sweep y
dibuja los resultados
function gensweep_Callback(hObject, eventdata, handles)

%Al ser globales y activar esta función después de seleccionar todos
los parámetros, coge los valores seleccionados
global ADW ADR fs fstart fstop duration y q infocanales new_value
graficas

% Variables para guardar el valor mayor y menor de la respuesta
h_M = 0;
h_m = 0;

ADW.ChannelMappingSource='Property';

%Guarda la información de la última salida seleccionada
%en una estructura junto con las entradas correspondientes
ADR_in = findobj('Callback', @entradas_Callback,'Value', 1);
in=zeros(1,length(ADR_in));
for i=1:length(ADR_in)
    in(i)=str2double(ADR_in(i).String);
end
in=fliplr(in);
a = new_value;
infocanales{a} = struct('ADW', a, 'ADR', in);

msgbox('Generando sweep y capturando audio');

ADW.SampleRate=fs;
ADR.SampleRate=fs;
ADR.ChannelMappingSource='Property';

[sweep, isweep] = sweep_isweep(fstart,fstop,fs,duration);
sweep = sweep'/32767;

```

```

isweep = isweep';
%Reordena el sweep en una matriz de 1024 (framelength) filas
x=buffer(sweep,1024);

% Bucle que va leyendo las estructuras y dando un valor al canal
del ADW y a
% los canales ADR a cada iteración. Si la estructura está vacía,
pasa a la siguiente
    for i=1:length(infocanales)
        if (isempty(infocanales{i}) == 1)
            continue;
        end
        ADW.ChannelMapping = infocanales{i}.ADW;
        ADR.ChannelMapping = infocanales{i}.ADR;
        y=zeros(length(sweep),length(ADR.ChannelMapping));

%Bucle que lanza el sweep y lo captura

    for j=0:size(x,2)-1
        xv=x(:,j+1);
        xv= repmat(xv,1,length(ADW.ChannelMapping));
        micro=ADR();
        ADW(xv);
        y(j*1024+1:(j+1)*1024,:)=micro;
    end

    release(ADW);
    release(ADR);

% Contador para titular las ventanas con el número correspondiente
q = q + 1;
figure('Name', ['Medida ' num2str(q) '']);

%Número de puntos para la fft
n = 32768;

for p=1:length(ADR.ChannelMapping)
    h=fconv(isweep,y(:,p));
    ll = length(sweep);
    %Quita el retardo de la convolución
    h = h(ll:end);
    ir(:,p) = h;
    fh=fft(h,n);
    t = (0:length(h)-1)/fs;
    fh=20*log10(abs(fh));

%Normaliza el vector de frecuencias
f=fs*(0:n-1)/n;

%Crea un campo nuevo en la estructura para guardar la respuesta
impulso
    infocanales{i} = setfield(infocanales{i}, 'data', ir);
    M = max(h);
    m = min(h);

```

```

    if (M > h_M)
        h_M = M;
    end
    if (m < h_m)
        h_m = m;
    end
end
for p=1:length(ADR.ChannelMapping)
    subplot(ceil(length(ADR.ChannelMapping)),
            ceil(length(ADR.ChannelMapping)/2) , p);
    if (strcmp(graficas,'Respuesta al impulso') == 0)
        semilogx(f,fh);
        xlim([20,20000]);
        hold on
        grid on
        xlabel('Frequency(Hz)');
        ylabel('Amplitude');
    end
    if (strcmp(graficas,'Respuesta al impulso') == 1)
        plot(t,h);
        hold on
        grid on
        xlim([0,duration]);
        ylim([m,M]);
        xlabel ('Tiempo(s)');
    end

    %Nos permitirá abrir la gráfica en una nueva ventana al hacer
click
    set(gca,'ButtonDownFcn',@createnew_fig);
    set(gca,'XTickMode', 'auto');

    % Titula cada subplot con la salida y entrada correspondientes
    title( sprintf( 'Salida: %d, Entrada: %d ', ADW.ChannelMapping,
ADR.ChannelMapping(p) ));

    %Guarda en la estructura el valor la frecuencia de muestreo
    infocanales{i} = setfield(infocanales{i}, 'fs', fs);
end
clear ir;
end

for i=1:length(infocanales)
    if(isempty(infocanales{i}) == 1)
        continue;
    end
    %Guarda las estructuras con la información
    save('Datos_guardados.mat','infocanales');
end

```

```

function instrucciones_Callback(hObject, eventdata, handles)

```

msgbox ('Este programa genera un sweep con las características definidas por el usuario que es lanzado por la salida seleccionada y capturada por las entradas seleccionadas. Para ello siga las siguientes instrucciones.

1. Elija las características del sweep
2. Seleccione los drivers y a continuación el dispositivo que va a utilizar
- Aparecerán las entradas y las salidas disponibles
3. Seleccione una salida y las entradas por las que capturar el audio

y pulse el botón generar sweep

4. En las gráficas se indica la salida y la entrada');

```
%% Permite seleccionar el tipo de gráfica que queremos representar
function tipo_grafica_SelectionChangedFcn(hObject, eventdata, handles)

global graficas

graficas = eventdata.NewValue.String;

%% Función que permite abrir una ventana con la gráfica seleccionada
function createnew_fig(cb,evendata)

%cb es el handle de la gráfica seleccionada
%Hacer click en los espacios en blanco de la figura, no en el plot

hh = copyobj(cb,figure);

%Para la nueva figura, vacía el ButtonDownFcn
set(hh, 'ButtonDownFcn', []);

%resize a los ejes
set(hh, 'Position', get(0, 'DefaultAxesPosition'));
```

8.2 Código de la función generadora del sweep logarítmico

```
function [sweep,isweep] = sweep_isweep(fstart,fstop,fs,duration)

%Esta función genera un sweep logarítmico y su inverso con los datos
%introducidos por el usuario

%fstart = frecuencia inicial
%fstop = frecuencia final
%fs = frecuencia de muestreo
%duration = duración en segundos
%sweep = sweep logarítmico
%isweep = señal para deconvolucionar

Slong=fs*duration;
TT=Slong-1;

w1=2*pi*fstart/fs;    %radianes
w2=2*pi*fstop/fs;
sweep=zeros(1,Slong);
isweep=sweep;

%Parámetros del sweep
k=w1*TT/log(w2/w1);
l=log(w2/w1)/TT;

for t=0:Slong-1
    v=sin(k*(exp(l*t)-1));
    isweep(Slong-1-t+1)=v*exp(-(Slong-1-t)*l);
    % Trabaja con 16 bits, 32767 para que ocupe todo el rango dinámico
    sweep(t+1)=round(32767*v);
end;
end
```