CÁTEDRA DE PROYECTO FINAL / INGENIERÍA EN ELECTRÓNICA

EQ EMULATOR

SISTEMA DE PROCESAMIENTO DIGITAL DE SEÑALES EN TIEMPO REAL BASADO EN FILTROS ADAPTATIVOS PARA EL MODELADO DE SISTEMAS ELECTROACÚSTICOS

PROYECTO FINAL

Versión 1.0
9/02/2024

INFORMACIÓN DEL PROYECTO

Autor	
Nombre completo	Francisco Julián Farré
Legajo	42945
e-mail	franjulianfarre@gmail.com

Tutor	Dr. Ing. Rodrigo Gonzalez
Director	Dr. Ing. Rodrigo Gonzalez
Jurado	Ing. Cesar Boschi
Año Académico	2024
Responsable de la cátedra	Ing. Ana Lattuca

Empresa / Cliente / Laboratorio	Laboratorio de Acústica “Mario Guillermo Camín" UTN - FRM
Patrocinador (Sponsor)	CEGA Electrónica

UTN FACULTAD REGIONAL MENDOZA
UNIVERSIDAD TECNOLÓGICA NACIONAL

CÁTEDRA DE PROYECTO FINAL / INGENIERÍA EN ELECTRÓNICA

EQ EMULATOR

SISTEMA DE PROCESAMIENTO DIGITAL DE SEÑALES EN TIEMPO REAL BASADO EN FILTROS ADAPTATIVOS PARA EL MODELADO DE SISTEMAS ELECTROACÚSTICOS

1. RESUMEN DEL PROYECTO

1.1 RESUMEN

El presente proyecto consiste en el diseño, desarrollo e implementación de un dispositivo electrónico de bajo coste cuya aplicación se encuentra en el procesamiento digital de señales de audio, específicamente para la obtención de modelos digitales de sistemas electroacústicos y su consecuente utilización en el contexto del audio profesional.

El avance tecnológico en el procesamiento de señales de audio ha revolucionado la industria musical y de audio, permitiendo la emulación digital de sistemas electroacústicos de alta calidad. Estos emuladores permiten, por ejemplo, reemplazar voluminosos amplificadores de audio por sencillos y portables sistemas digitales, siendo una alternativa altamente competitiva y buscada en el mercado de la audio técnica. No obstante, el alto costo y la complejidad de estas soluciones han limitado su accesibilidad.

Este proyecto responde a esta problemática mediante el desarrollo de un dispositivo modular, portátil y asequible. Este sistema utiliza avanzados algoritmos y técnicas de procesamiento digital en tiempo real para emular sistemas electroacústicos de manera personalizada, sin requerir conocimientos técnicos previos de parte del usuario.

Durante el proceso, se realizó un estudio exhaustivo y simulaciones que proporcionaron una sólida base técnica y algorítmica. El uso de filtros adaptativos demostró ser una innovación clave, abriendo posibilidades en diversas áreas de investigación.

La viabilidad comercial es sobresaliente, ya que el dispositivo ofrece una solución de bajo costo y alto rendimiento en comparación con las alternativas disponibles comercialmente. La personalización se convierte en una ventaja funcional única, permitiendo a los usuarios crear sus propias emulaciones de manera sencilla y novedosa.

Este proyecto ofrece un dispositivo revolucionario que simplifica la emulación de sistemas electroacústicos, y que establece bases para futuras investigaciones y aplicaciones en el campo del procesamiento de señales de audio. Su viabilidad comercial, innovación técnica y perspectivas de mejora hacen de este proyecto un logro significativo en la ingeniería de audio.

1.2 SUMMARY

This project outlines the development of a low-cost electronic device designed for digital audio signal processing. Its primary objective is to create digital models of electroacoustic systems for application in professional audio contexts.

Advancements in audio signal processing have transformed the music and audio industry, enabling high-fidelity digital emulation of electroacoustic systems. These emulators provide a compelling alternative to bulky audio amplifiers, offering simplicity and portability in the technical audio market. Nonetheless, their cost and complexity limit accessibility.

Addressing this challenge, this project presents a modular, portable, and affordable device. Leveraging advanced real-time digital processing algorithms, it customizes the emulation of electroacoustic systems without requiring prior technical expertise from users.

The project involved exhaustive studies and simulations, establishing a robust technical and algorithmic foundation. Notably, the integration of adaptive filters emerged as a pivotal innovation, expanding possibilities across diverse research domains.

The device showcases exceptional commercial viability, providing a cost-effective, high-performance solution compared to existing alternatives. Its unique selling point lies in user-friendly customization, allowing for innovative, personalized emulation creation.

This endeavor introduces a groundbreaking device simplifying electroacoustic system emulation while laying groundwork for future audio signal processing research and applications. With its commercial viability, technical innovation, and potential for enhancements, this project stands as a significant achievement in audio engineering.

1.3 PALABRAS CLAVES

Audio, sonido, emulación, amplificador, procesamiento digital, filtro adaptativo, LMS, ecualización, portabilidad.

2. ÍNDICE

2.1 ÍNDICE DE FIGURAS

2.2 ÍNDICE DE TABLAS

3. INTRODUCCIÓN

3.1 IDEA Y DESCRIPCIÓN DEL PROYECTO

El proyecto realizado abarca la investigación, desarrollo e implementación de un dispositivo de procesamiento digital de señales de audio, diseñado para la creación de modelos digitales de sistemas electroacústicos, como amplificadores y gabinetes acústicos. Estos modelos digitales deben ser capaces de replicar fielmente las características de cualquier sistema electroacústico elegido por el usuario, dentro de los límites de funcionamiento del dispositivo, y permitir el almacenamiento de dichos modelos para su posterior uso en grabaciones o actuaciones en vivo.

Este emulador es un dispositivo de bajo costo y complejidad, lo que le confiere una fuerte ventaja competitiva en comparación con alternativas similares disponibles en el mercado. Además, gracias a su capacidad de emulación de sistemas electroacústicos elegidos por el usuario, ofrece un nivel de personalización que lo distingue de la competencia.

El dispositivo consta de un gabinete compacto y portátil que aloja internamente un procesador digital de señales (DSP) responsable de llevar a cabo el procesamiento requerido. Externamente, dispone de conectores para las señales de entrada y salida de audio, así como una pantalla táctil y luces indicadoras para la interacción con el usuario. A través de la interfaz de usuario, se pueden seleccionar los modos de funcionamiento y supervisar el estado del dispositivo.

La realización de este dispositivo requirió, en primer lugar, una exhaustiva investigación de algoritmos de procesamiento digital de señales que posibilitaran el funcionamiento deseado. Se empleó un algoritmo de filtrado digital adaptativo, para el cual se desarrolló un simulador de alto nivel en el software MATLAB. Este simulador facilitó la evaluación de diversas alternativas algorítmicas, la optimización de parámetros utilizados posteriormente en la implementación y la validación del funcionamiento previo a la etapa de desarrollo.

En segundo lugar, se diseñó una plataforma de hardware en PCB que permitió la interconexión de todos los componentes electrónicos necesarios. Para este propósito, se seleccionó el módulo STM32F4DISCOVERY, que incorpora un microprocesador STM32F407VGT6 encargado de las operaciones DSP, y el módulo ESP32, con el microprocesador ESP-WROOM-32, responsable de la interfaz de usuario. Asimismo, se empleó el módulo Pmod I2S2, equipado con ADCs y DACs que utilizan el protocolo I2S para el procesamiento de señales de audio de alta calidad, además de una pantalla táctil MSP2807.

Además, se desarrolló el firmware correspondiente para ambos microprocesadores. Los algoritmos previamente evaluados en simulación se implementaron en lenguaje C en el DSP, mientras que el control de usuario se programó en C++. Se diseñaron protocolos de comunicación para facilitar la interacción entre todos los subcomponentes del sistema y para el manejo de las señales de audio desde y hacia los convertidores.

De manera concurrente, se diseñó un chasis que permitió el montaje compacto y que presenta una interfaz de usuario atractiva e intuitiva, albergando todos los componentes de hardware y los conectores externos.

Por último, se destacan los aspectos de documentación relacionados con el proyecto, que incluyen un manual de usuario pertinente y un informe técnico exhaustivo que documenta todo el desarrollo del mismo, desde las fases de análisis y simulación hasta el desarrollo de firmware y hardware, así como la implementación y verificación frente a modelos de funcionamiento comprobables.

Como se explicó, el dispositivo permite obtener un modelo digital de sistemas electroacústicos que presenten cierta complejidad y cuyo comportamiento sea desconocido. Luego, hace posible utilizar dicho modelo como emulador del sistema original, para uso en aplicaciones musicales y técnicas. De aquí se desprenden los dos modos principales de funcionamiento: el modo de adaptación y el modo de filtrado. En el primero se realiza la estimación del modelo lineal del sistema desconocido utilizando un filtro adaptativo que se vale del algoritmo NMLS para la afectación adaptativa de los coeficientes de un filtro digital FIR. Luego, en el modo de filtrado, se usa dicho modelo como filtro emulador estático, el cual debe permitir a su vez pasar de la señal original a la señal filtrada. En la Figura 1 y Figura 2 se presentan los diagramas de funcionamiento de cada uno de estos modos.

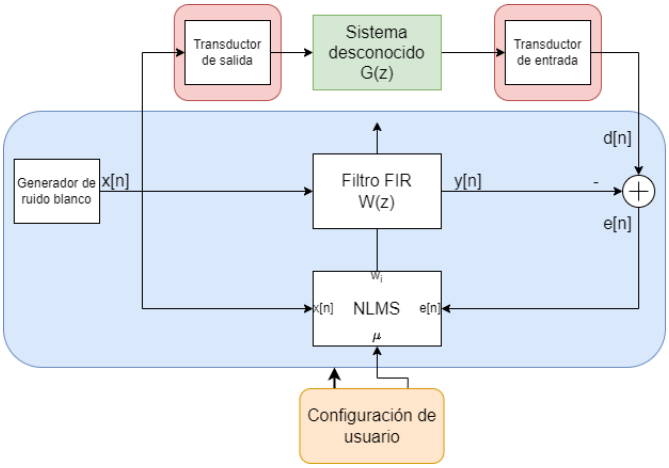


Figura 1 - Modo de adaptación

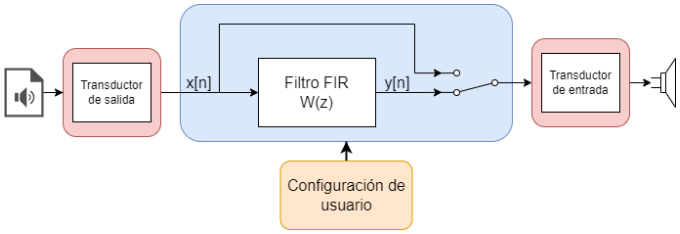


Figura 2 - Modo de filtrado

Finalmente, el diagrama en bloques general del sistema es el que se muestra en la Figura 3, el cual subdivide la arquitectura, como se explicó, en un microcontrolador DSP y uno de control de usuario. Los bloques de color azul representan los elementos del sistema que deben realizar tareas de procesamiento de señales. En el caso de la adaptación, tenemos el generador de ruido blanco, el que genera la señal x[n], y el filtro adaptativo, que recibe dicha señal junto con d[n], proveniente de la salida del sistema desconocido,

y genera las señales y[n] y e[n] (las cuales son transducidas al dominio analógico para verificar la evolución del sistema de adaptación). Para el modo de filtrado, el bloque de procesamiento de señal corresponde al filtro FIR, el cual es un sistema estéreo con dos señales de entrada y dos de salida, una para el canal izquierdo y otra para el derecho en ambos casos.

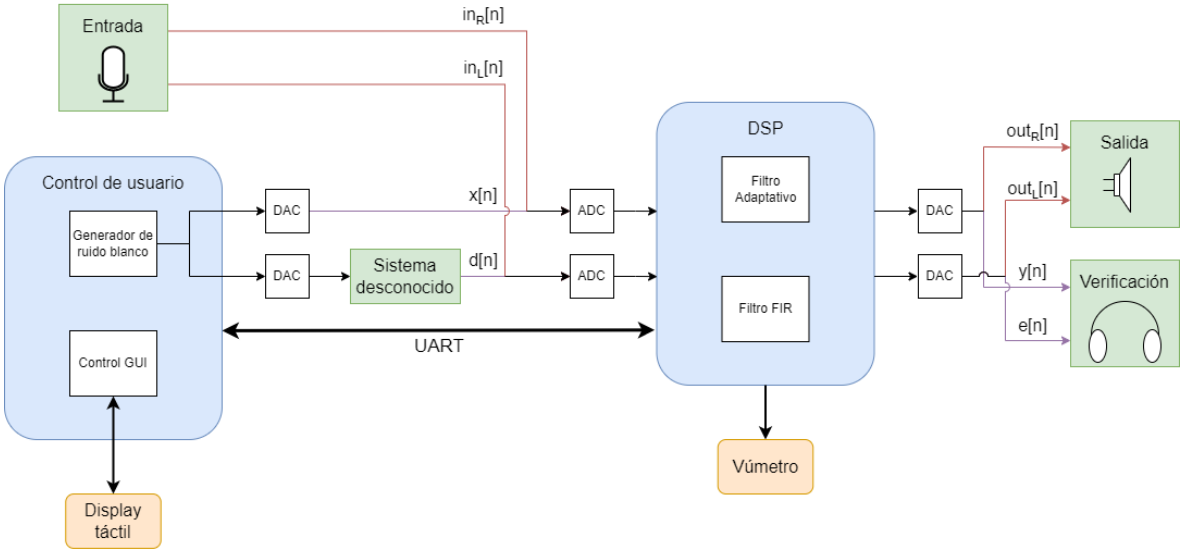


Figura 3 - Diagrama en bloques del sistema

3.1.1 Objetivo general

Con el presente proyecto se busca desarrollar un dispositivo de procesamiento de señales de audio en calidad profesional completamente funcional, que no solo resulte novedoso en cuanto a los algoritmos y técnicas digitales utilizadas desde el punto de vista técnico, sino que además resulte atractivo para productores y músicos en cuanto presente una buena calidad de emulación y cumpla con los estándares esperables para un dispositivo de estas características.

Para ello se pone especial atención en el estudio previo de las características acústicas y electrónicas de los dispositivos que se pretende emular, de los algoritmos que permiten realizar este sistema, de los estándares de la industria del audio y de las soluciones tecnológicas existentes que dan respuesta a cada uno de ellos.

3.1.2 Objetivo particular

- Comprender y modelar el comportamiento de sistemas electroacústicos complejos, a fin de poder extraer sus características importantes y utilizarlas para el desarrollo de la solución propuesta.
- Implementar técnicas de filtrado adaptativo con aplicación en el modelado digital de sistemas electroacústicos.
- Desarrollar un sistema de procesamiento digital de señales en hardware y firmware de forma acabada, haciendo uso de técnicas de simulación, algorítmicas y de programación acordes a los estándares de la audio técnica.
- Generar en base a los conocimientos y técnicas desarrolladas un prototipo de un producto comercial viable que resuelva el problema de una forma sencilla y económica.
- Realizar una documentación exhaustiva de cada una de las etapas del proyecto, de los conocimientos técnicos involucrados en su desarrollo y de las innovaciones y desafíos que se tuvieron en su implementación a fin de que pueda constituirse como fuente de consulta para líneas de trabajo futuro en esta área.

3.2 JUSTIFICACIÓN DEL PROYECTO

3.2.1 Antecedentes del proyecto

La necesidad imperante de perfeccionar y adaptar el sonido en la producción musical y el ámbito del audio ha impulsado el surgimiento de sistemas de emulación digital de gabinetes y amplificadores. Estos sistemas no solo representan un avance tecnológico en materia técnica, sino que también ofrecen ventajas substanciales con respecto a los gabinetes y amplificadores reales. En contraposición, los gabinetes tradicionales a menudo resultan voluminosos y pesados, lo que dificulta su transporte y manejo en entornos de actuación en vivo. Además, su variabilidad tonal limita la capacidad de los músicos y productores para obtener el sonido exacto que desean.

La emulación digital de gabinetes, por otro lado, proporciona una portabilidad inigualable y una versatilidad sobresaliente. Los usuarios pueden contar con un abanico de opciones tonales a su disposición, sin necesidad de cargar con equipos voluminosos. Además, la posibilidad de conectar estos sistemas directamente a sistemas de sonido externos o a interfaces de grabación facilita tanto las actuaciones en vivo como la producción en estudio. La tecnología de respuesta de impulso (IR) permite la captura de las características sonoras de diferentes espacios y gabinetes, lo que añade un nivel de precisión y control inigualable.

Históricamente, replicar con precisión el tono de un gabinete o amplificador tradicional ha sido un desafío, debido a la amplia variabilidad de factores que influyen en el sonido de un sistema de audio. Los avances tecnológicos desempeñan un papel fundamental en el desarrollo de sistemas de emulación digital de gabinetes, ya que la potencia de procesamiento necesaria no estaba disponible en el pasado.

A pesar de su portabilidad y su capacidad para lograr una fidelidad impresionante en la reproducción del sonido de los equipos de audio, los sistemas de emulación disponibles en la actualidad se enfrentan a un desafío significativo: el costo elevado debido a la complejidad inherente al procesamiento en tiempo real de las respuestas impulsivas. Incluso en el caso de sistemas electroacústicos simples, como amplificadores, la generación de una respuesta impulsiva puede resultar en la creación de filtros con miles de coeficientes, lo que requiere el uso de potentes procesadores dedicados para su funcionamiento en situaciones de grabación y producción de audio. A menudo, estas emulaciones se llevan a cabo directamente en computadoras a través de software especializado.

Además, otro inconveniente que suele surgir con estos sistemas es que el proceso de obtención de emulaciones de gabinetes generalmente se restringe al ámbito técnico y de desarrollo, principalmente debido a la complejidad asociada con la captura de respuestas impulsivas. Esto limita la capacidad de los músicos para obtener sonidos personalizados y únicos, obligándolos a depender de emulaciones preexistentes.

En vista de estas consideraciones, el presente proyecto se enfoca en el desarrollo técnico de un dispositivo de emulación de sistemas electroacústicos de bajo costo. Este dispositivo no solo permite la emulación dinámica de sistemas, lo que significa que los usuarios pueden obtener el sonido exacto que desean, sino que también utiliza un método alternativo al de la respuesta impulsiva: un sistema de filtrado digital adaptativo. Esto

conduce a una implementación de complejidad moderada, con el objetivo de reducir los costos de desarrollo y fabricación. Además, se busca ofrecer modelos dinámicos de fácil uso y personalización para músicos y productores, sin necesidad de poseer conocimientos técnicos avanzados.

3.2.2 Estado actual

Actualmente, la emulación de amplificadores y gabinetes acústicos se ha convertido en una técnica ampliamente adoptada por músicos y productores en diversas áreas. Según el contexto de uso, se encuentran disponibles varias soluciones tecnológicas que aprovechan esta técnica. En todos los casos, se emplean técnicas de modelado digital para recrear la respuesta tonal y acústica de altavoces y gabinetes. Estos modelos matemáticos se basan en mediciones precisas de altavoces y gabinetes reales que lo que hacen es captar su respuesta al impulso.

Las respuestas al impulso (IR) son archivos de audio que registran la respuesta acústica de un gabinete y un altavoz específico cuando se exponen a un estímulo, como un impulso sonoro. Los emuladores de gabinetes utilizan IRs para simular el sonido de equipos físicos. Los usuarios pueden cargar IRs de gabinetes y altavoces específicos en sus emuladores para lograr una reproducción precisa de tonos. Sin embargo, estas respuestas impulsivas suelen requerir procesadores dedicados o una capacidad informática significativa para su uso en aplicaciones de audio en tiempo real.

En cuanto a alternativas comerciales que dispongan de estas técnicas podemos clasificarlas en tres tipos: emuladores integrados, pluggins de software y hardware especializado.

Muchos amplificadores y dispositivos de modelado de efectos incorporan emuladores de gabinetes. Estos sistemas integran tanto el modelado de amplificadores como el de gabinetes para ofrecer una experiencia de tono completa. Dado que estos equipos incluyen el costo del amplificador en sí, tienden a ser relativamente costosos y generalmente se utilizan como parte de una cadena de efectos integrada, en lugar de reemplazar la necesidad de un amplificador físico.

Por otro lado, existen numerosos plugins de software diseñados para la emulación de gabinetes de audio. Estos complementos se utilizan en estaciones de trabajo de audio digital (DAW) y proporcionan una amplia gama de opciones para emular distintos gabinetes y altavoces. Aunque estos plugin suelen tener un costo razonable, no superando los $200 USD por unidad, su utilización requiere del uso de ordenadores y de un software DAW, por lo que no brindan portabilidad y su uso puede verse limitado a ciertas aplicaciones.

Finalmente, algunas compañías fabrican hardware específico para la emulación de gabinetes, como cajas de carga reactiva que permiten conectar un amplificador a un dispositivo de modelado de gabinetes sin necesidad de un gabinete físico. Estos dispositivos, a menudo presentados en formato de rack o pedaleras de efectos, son más costosos, generalmente superando los $1000 USD, debido a la complejidad inherente al procesamiento de señales requerido para emular IRs de alta calidad. En la mayoría de los casos, estos dispositivos incluyen emulaciones precargadas de diversos gabinetes y amplificadores, y los usuarios pueden seleccionarlas a través de presets almacenados internamente.

3.2.3 Necesidad del negocio y definición del problema

En función del estado del arte previamente expuesto, nuestro proyecto tiene como objetivo el desarrollo de un dispositivo de hardware especializado destinado a la emulación de sistemas electroacústicos. Este dispositivo responde a dos demandas cruciales, que a su vez representan una valiosa oportunidad de negocio, ya que no se encuentran plenamente satisfechas por las soluciones actualmente disponibles en el mercado.

En primer lugar, buscamos que el dispositivo permita la emulación dinámica y personalizada de amplificadores y gabinetes de acuerdo con las preferencias del usuario. En otras palabras, los usuarios podrán replicar digitalmente sistemas específicos que deseen, de manera directa y sin necesidad de cargar respuestas impulsivas. Esta característica es excepcional, ya que la mayoría de las alternativas comerciales existentes exigen que los usuarios generen y graben sus propias IRs, un proceso que a menudo se encuentra más allá de sus capacidades técnicas y conocimientos. A través del uso de la técnica de filtrado adaptativo, nuestro dispositivo pretende liberar a los usuarios de los complejos detalles involucrados en la obtención de los modelos, haciendo que el proceso de emulación sea mucho más accesible y sencillo.

En segundo lugar, nuestro dispositivo estará disponible a un costo considerablemente inferior. Esto se logrará gracias a las técnicas algorítmicas empleadas, que permiten utilizar procesadores de bajo costo sin sacrificar la calidad y la eficacia. Este enfoque no solo hace que la emulación de sistemas sea más asequible, sino que también facilita la portabilidad del dispositivo y su uso generalizado. De esta manera, brindamos a los usuarios la posibilidad de acceder a una emulación de sistemas de audio de alta calidad con un dispositivo económico y fácil de utilizar.

3.2.4 Beneficios del proyecto

Este proyecto se origina a través del respaldo del Laboratorio de Acústica "Mario Guillermo Camín" de la Universidad Tecnológica Nacional, Facultad Regional Mendoza. Esta colaboración está estrechamente relacionada con la cátedra de Sistemas de Sonido en la carrera de Ingeniería Electrónica. Aunque el patrocinio del proyecto no tiene como objetivo principal la comercialización, es importante resaltar los diversos beneficios que aporta desde una perspectiva de investigación y los posibles beneficios económicos asociados.

En primer lugar, el Laboratorio ampliará su campo de conocimiento hacia áreas previamente no exploradas pero relacionadas con su ámbito de aplicación. Históricamente, tanto el laboratorio como la cátedra de sistemas de sonido se han centrado en temas de acústica arquitectónica, medición de recintos y mitigación del ruido acústico. Sin embargo, existía una falta de investigación y desarrollo en áreas clave, como los sistemas electrónicos de audio, especialmente en lo que respecta al audio digital y las nuevas tecnologías relacionadas. A través de la investigación y el desarrollo práctico de este proyecto y el dispositivo electrónico resultante, se agrega una nueva área de investigación y desarrollo, generando interés tanto a nivel técnico como en la esfera musical. Además, la disponibilidad de modelos digitales de sistemas electroacústicos, un logro posible gracias a este proyecto, se convierte en una herramienta esencial para los

objetivos que el laboratorio y la cátedra ya están persiguiendo, especialmente en el análisis y tratamiento acústico, así como en la mitigación del ruido.

En segundo lugar, el patrocinador se beneficia al obtener un prototipo de dispositivo que abre oportunidades de investigación y posibles desarrollos comerciales en el ámbito de los sistemas de audio digital profesional a bajo costo. La utilización de componentes electrónicos asequibles y de fácil adquisición hace que esta sea una alternativa no solo económica sino también de bajo riesgo en términos de desarrollo. Esto representa una oportunidad para el desarrollo local de sistemas de audio digital, un mercado incipiente y en gran medida inexplorado en la región.

3.3 ALCANCE

3.3.1 Alcance

En referencia al desarrollo del proyecto, se deben aclarar los siguientes alcances y posibilidades brindadas por el mismo. El mismo incluye como entregables un software de simulación en MATLAB, un dispositivo físico que corresponde al emulador con su correspondiente placa electrónica en PCB, su firmware asociado y chasis, junto con los ensayos que aseguren su funcionamiento y la documentación asociada al proyecto. A continuación, se desglosan cada uno de estos ítems:

- **Simulador en MATLAB:** Se desarrolló un simulador de alto nivel en MATLAB con el propósito de probar diversas alternativas de modelado de sistemas electroacústicos. Este simulador se inició caracterizando la respuesta al impulso de sistemas reales utilizando respuestas al impulso previamente existentes, obtenidas de estudios de grabación. Luego, se implementó la simulación del modo de adaptación adaptativo, permitiendo evaluar los sistemas caracterizados y seleccionar parámetros adecuados para el filtro adaptativo. Además, se evaluó la calidad auditiva del modelo generado. Es importante mencionar que el método no es apto para la emulación de recintos reverberantes, lo que queda fuera del alcance del dispositivo final.
- **Desarrollo del Dispositivo Emulador:** Se utilizaron módulos de desarrollo de microprocesadores tanto para el DSP como para el control de usuario. Los módulos STM32F4DISCOVERY y ESP32 se emplearon para el procesamiento y la interfaz de usuario, respectivamente. Ambos módulos proporcionan pines de interconexión, manejo de alimentación y señales de control. Se utilizó electrónica modular para los módulos I2S y la pantalla táctil. El hardware desarrollado comprende el circuito que interconecta estos bloques, junto con los conectores y componentes necesarios para el funcionamiento conjunto del dispositivo.
- **Diseño del PCB:** Se creó un PCB para el dispositivo prototipo, utilizando la técnica de planchado. Cabe destacar que este diseño no es válido ni óptimo para una versión comercial del dispositivo.
- **Firmware:** Se programaron los dos procesadores involucrados (DSP y ESP32) utilizando CubeIDE y ArduinoIDE en lenguajes C y C++, respectivamente. Se emplearon librerías para el manejo de instrucciones DSP de bajo nivel, memorias, periféricos y protocolos de comunicación.
- **Diseño del Chasis:** Se diseñó un chasis con el propósito de alojar la electrónica y proporcionar soporte para los conectores, pantalla e indicadores de la interfaz de usuario. Además de contribuir a un acabado estético adecuado para el prototipo.
- **Verificación Funcional:** Se llevó a cabo un proceso de verificación funcional del dispositivo utilizando el simulador desarrollado como referencia. Se utilizaron equipos reales como dispositivos de prueba y herramientas de caracterización profesional de audio para la validación.
- **Documentación:** Se generó toda la documentación necesaria para el proyecto. Esto incluye un informe técnico que detalla todos los aspectos de la investigación, desarrollo, implementación y verificación del proyecto. Además, se proporcionaron los códigos tanto del simulador como del firmware. También se

creó un manual de usuario que explica la operación del dispositivo para posibles compradores.

3.3.2 Límites o fuera de alcance

El desarrollo del proyecto se encuentra acotado por las siguientes premisas:

- El enfoque principal del proyecto se centra en la creación de un dispositivo capaz de obtener modelos de sistemas electroacústicos. Aunque se contempla una etapa de verificación del funcionamiento, esta no incluirá un número exhaustivo de sistemas, sino que servirá como una prueba para comprobar que el módulo responde de manera correcta y de acuerdo a lo desarrollado en la etapa de simulación. Los alcances y limitaciones del dispositivo en forma general se consideran ya definidos en la etapa de análisis y simulación previa.
- El dispositivo desarrollado se concentra en la emulación a través del modelado por filtros adaptativos y el posterior uso de dichos filtros en modo estático. Dado que se trata de un prototipo, no se incorporarán funcionalidades adicionales, como opciones de ecualización, filtrado adicional o cadenas de efectos, características típicas de los dispositivos comerciales de este tipo. Sin embargo, el sistema establece las bases para posibles desarrollos futuros que podrían incluir estas funcionalidades.
- El desarrollo del hardware del dispositivo se lleva a cabo de manera modular, lo que implica el uso de placas de desarrollo para los microcontroladores asociados. No se diseñarán circuitos de control y alimentación para los microcontroladores. Lo mismo se aplica al controlador de los ADCs y DACs con protocolo I2S y al driver SPI de la pantalla táctil. El PCB creado tiene como propósito servir como soporte para la electrónica en la etapa de prototipado y no está pensado para aplicaciones comerciales, en las que su diseño deberá profesionalizarse aún más.
- Según el método utilizado para la emulación de sistemas electroacústicos, el dispositivo desarrollado es adecuado para la emulación de sistemas con respuestas impulsivas cortas, como gabinetes y amplificadores acústicos. Sin embargo, no es apto para su uso en dispositivos reverberantes o sistemas de gran latencia.

3.3.3 Soluciones y entregables principales

La siguiente tabla muestra un listado de los entregables del proyecto:

Entregable	Descripción
Simulador de alto nivel	Set de programas en MATLAB junto con archivos asociados que permiten simular de forma completa el funcionamiento teórico de los algoritmos implementados, tanto del modo de adaptación como el de filtrado. Además, permite la variación de parámetros de diseño de dichos algoritmos y evaluar la performance de los mismos frente a modelos reales de diferentes sistemas electroacústicos debidamente caracterizados.
EQ Emulator	Dispositivo funcional de emulación obtenido como resultado del desarrollo del proyecto. El mismo consta del hardware utilizado con su correspondiente PCB de soporte, el firmware desarrollado correspondiente a los módulos STM32F4 y ESP32, el chasis que aloja la electrónica junto con los conectores y la interfaz de usuario y la conexión de alimentación correspondiente.
Informe técnico	Documento que explicita todo el desarrollo técnico del proyecto. Esto incluye el marco teórico, investigación y análisis previo realizado, la etapa de simulación, el desarrollo e implementación del hardware y firmware de manera detallada y las pruebas de verificación del dispositivo.
Repositorio del proyecto	Repositorio público en GitHub donde se brinda acceso a bibliografía, archivos de diseño y el código del software y firmware desarrollado en forma completa y libre.
Manual de usuario	Documento destinario al usuario del dispositivo EQ Emulator, el cual explica sus características, los diferentes modos de operación y cómo hacer uso del mismo.

Tabla 1 – Soluciones y entregables principales

3.4 PLANIFICACIÓN DEL PROYECTO

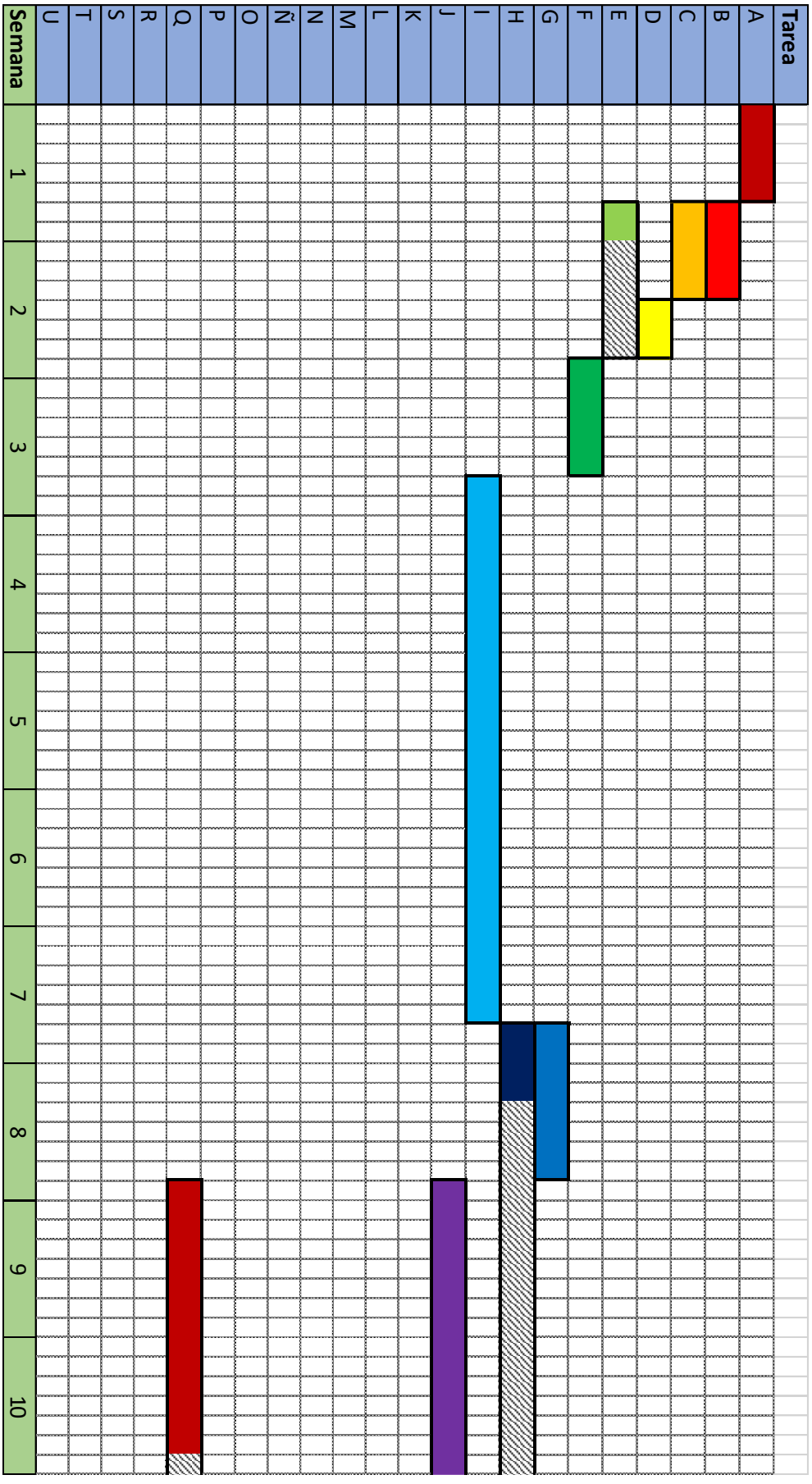
3.4.1 Cronograma

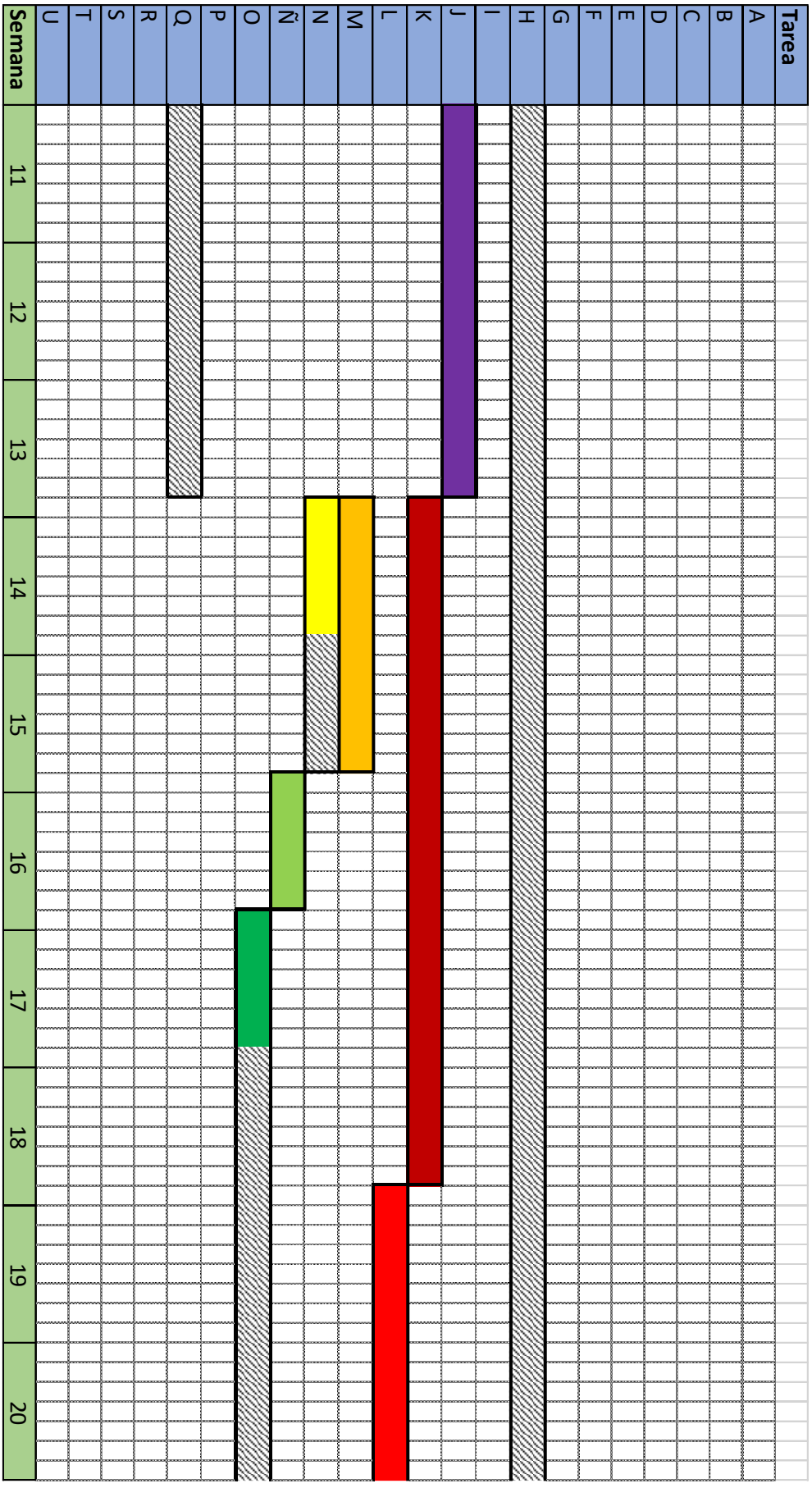
Previo a armar el diagrama de Gantt del proyecto, organizaremos el proyecto en tareas que deberán cumplirse y las dependencias entre ellas. Se han tomado jornadas de trabajo de 4 horas, las cuales equivalen a un día.

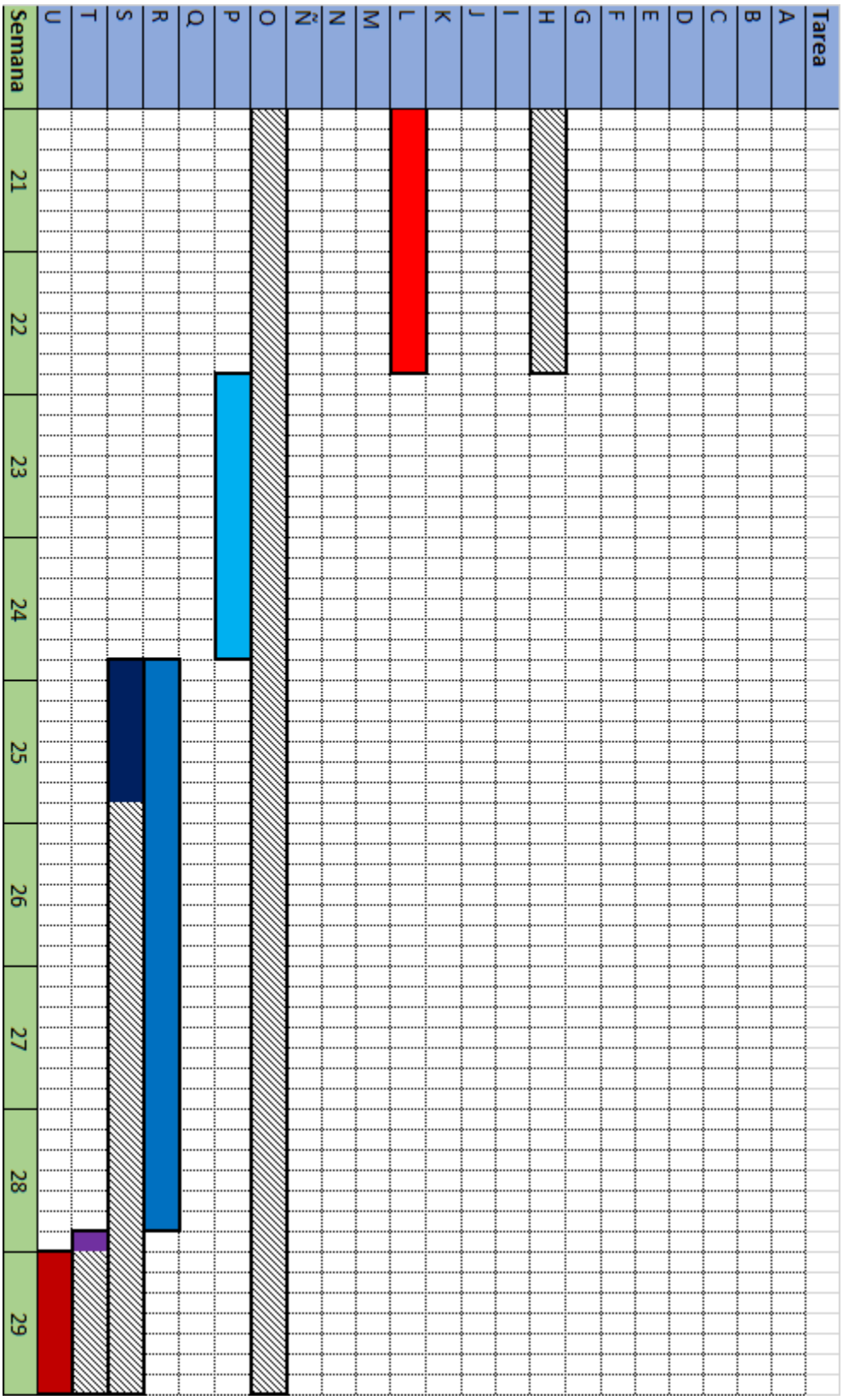
Tarea	Referencia	Precedentes	Duración (días)	Duración (horas)
Análisis preliminar	A	Inicio	5	20
Estado del arte de la tecnología	B	A	5	20
Caracterización de sistemas objetivo	C	A	5	20
Requerimientos y especificaciones	D	B, C	3	12
Planificación	E	A	2	8
Diseño funcional	F	D, E	6	24
Diseño técnico	G	I	8	32
Diseño de verificación	H	I	4	16
Desarrollo de simulador (software)	I	F	28	112
Desarrollo de hardware	J	G	35	140
Desarrollo de firmware	K	F, J, Q	35	140
Pruebas de funcionamiento	L	J, K	28	112
Diseño de interfaz de usuario	M	J	14	56
Diseño de PCB	N	J	7	28
Diseño de chasis	Ñ	M, N	7	28
Ensamblado	O	Ñ	7	28
Pruebas de verificación	P	H, L, Ñ	14	56
Capacitación	Q	G	14	56
Documentación técnica	R	P	28	112

Manual de usuario	S	P	7	28
Repositorio de proyecto	T	R	1	4
Documentación de proyecto	U	P	7	28

Tabla 2 – Tareas del proyecto


Tarea
A
B
C
D
E
F
G
H
I
J
K
L
M
N
Ñ
O
P
Q
R
S
T
U
Semana
1
2
3
4
5
6
7
8
9
10


Semana
11
12
13
14
15
16
17
18
19
20
Tarea
A
B
C
D
E
F
G
H
I
J
K
L
M
N
Ñ
O
P
Q
R
S
T
U


Tarea
A
B
C
D
E
F
G
H
I
J
K
L
M
N
Ñ
O
P
Q
R
S
T
U
Semana
21
22
23
24
25
26
27
28
29

Las celdas rayadas representan los tiempos de holgura de aquellas tareas que la poseen.

Al realizar el análisis de los tiempos de desarrollo del proyecto, se estimó que el tiempo de realización optimista era de 25 semanas, el tiempo normal de desarrollo era de 29 semanas y el tiempo pesimista de 33 semanas. Teniendo en cuenta que los tiempos de realización especificados para cada tarea han sido estimados, se concluye que el tiempo total de desarrollo del proyecto se encuentra entre 29 y 31 semanas. Por lo tanto, el proyecto se ha realizado en un tiempo estimado normal, con inclinaciones a tiempos superiores, pero dentro de los márgenes especificados.

3.4.2 Hitos

Se dividió el proyecto en etapas, cada una de las cuales presenta una serie de hitos.

Etapa	Hitos	Fecha de finalización
Etapa 1: Documentación preliminar	Presentación de documentos relacionados con la planificación previa: acta de inicio, EDT, anteproyecto.	Semana 2
Etapa 2: Análisis y simulación	Caracterización de sistemas objetivo y finalización del simulador de alto nivel.	Semana 7
Etapa 3: Desarrollo de hardware - firmware	Presentación de diagrama esquemático del circuito final, código completo de firmware y prueba de funcionamiento.	Semana 22
Etapa 4: Implementación física	Ensamblado del dispositivo final (PCB, chasis e interfaz de usuario)	Semana 23
Etapa 5: Verificación	Puesta en funcionamiento del dispositivo de contrastación y presentación de resultados de verificación	Semana 24
Etapa 6: Documentación técnica	Presentación de informe técnico y manual de usuario.	Semana 28
Etapa 7: Documentación de proyecto	Publicación de repositorio del proyecto e informe de proyecto.	Semana 29

Tabla 3 – Etapas del proyecto

3.4.3 Riesgo

Identificador	Riesgo	Impacto	Probabilidad	Puntaje
A	Identificación de fallos algorítmicos en el simulador que requieran una redefinición de la arquitectura	**2**	**3**	**6**
B	Información disponible limitada o de difícil acceso	**2**	**4**	**8**
C	Dificultad en la adquisición de componentes	**4**	**4**	**16**
D	Demora en los tiempos de envío de electrónica	**3**	**3**	**9**
E	Tiempo excesivo de configuración de entorno de desarrollo de firmware	**1**	**4**	**4**
F	Fallo en el diseño/implementación de PCB	**3**	**3**	**9**
G	Reemplazo de componentes por destrucción/fallo	**4**	**2**	**8**
H	Aumento de precios	**2**	**5**	**10**
I	Mala elección de framework de trabajo (procesador + IDE)	**5**	**2**	**10**
J	Avance de la competencia en tecnologías similares	**4**	**2**	**8**
K	Falta de feedback/interés de parte de los stakeholders	**4**	**5**	**20**

Tabla 4 – Riesgos

		Impacto				
		Mínimo	Moderado	Serio	Elevado	Grave
Probabilidad		1	2	3	4	5
Frecuente	5	5	10 **H**	15	20 **K**	25
Recurrente	4	4 **E**	8 **B**	12	16 C	20
Posible	3	3	6 **A**	9 **D, F**	12	15
Inusual	2	2	4	6	8 **G, J**	10 **I**
Remota	1	1	2	3	4	5

Tabla 5 – Matriz de riesgo

Identificador	Riesgo	Solución posible
A	Identificación de fallos algorítmicos en el simulador que requieran una redefinición de la arquitectura	Realizar un estudio previo exhaustivo y pormenorizado de los métodos de emulación existentes e incluirlos en la etapa de simulación. Evaluar diferentes alternativas de emulación y no una exclusivamente.
B	Información disponible limitada o de difícil acceso	Establecer como parte del cronograma un periodo de capacitación que se anticipe a los problemas que pudiera generar una curva de aprendizaje lenta de las herramientas de firmware y de los algoritmos matemáticos utilizados.
C	Dificultad en la adquisición de componentes	Evaluar diferentes alternativas de hardware para cada uno de los bloques componentes. Hacer uso de la sociedad comercial con CEGA Electrónica para importar componentes de forma eficiente.

D	Demora en los tiempos de envío de electrónica	Anticiparse a los tiempos en el cronograma a fin de que tiempo estimado de envío del componente se dé antes del inicio de su uso.
E	Tiempo excesivo de configuración de entorno de desarrollo de firmware	Establecer como parte del cronograma un periodo de capacitación que se anticipe a los problemas que pudiera generar una curva de aprendizaje lenta de las herramientas de firmware y de los algoritmos matemáticos utilizados.
F	Fallo en el diseño/implementación de PCB	Consultar externamente a profesionales en estas áreas para el diseño y verificación de estas etapas.
G	Reemplazo de componentes por destrucción/fallo	Verificación exhaustiva de cada componente y de la circuitería a fin de evitar fallos. Asegurarse tiempos mínimos de reemplazo de forma anticipada.
H	Aumento de precios	Realizar la compra de componentes de forma anticipada al cronograma.
I	Mala elección de framework de trabajo (procesador + IDE)	Realizar un estudio pormenorizado de las herramientas integradas en auge, que dispongan de buen soporte técnico y que no se encuentren desactualizadas al momento de desarrollo.
J	Avance de la competencia en tecnologías similares	Planteo de una arquitectura y tecnología flexibles a nuevas tendencias y cambios que pudieran surgir y que impliquen un

		cambio/mejora en las especificaciones del proyecto.
K	Falta de feedback/interés de parte de los stakeholders	Agendar reuniones periódicas donde se presenten los avances del proyecto y se hagan demostraciones parciales de sus potencialidades.

Tabla 6 – Mitigación de riesgos

4. DESARROLLO DEL PROYECTO

4.1 DESARROLLO TÉCNICO

4.1.1 Análisis y simulación

4.1.1.1 Generalidades sobre los sistemas acústicos, electroacústicos y de sonido

Puesto que en el desarrollo que aquí se detalla se tratará con señales y sistemas que forman parte del estudio de la acústica y los sistemas de sonido, resulta importante definir algunas generalidades y conceptos que serán útiles para el análisis y comprensión del mismo.

4.1.1.1.1 Principios de acústica

Denominamos Acústica a la rama de la Física que estudia la producción, transmisión, almacenamiento, percepción y reproducción del sonido. El sonido consiste en una variación en la presión de un medio elástico, como el aire o el agua, que se propaga a través de la materia, ya sea en estado gaseoso, líquido o sólido, en pequeñas fluctuaciones rápidas llamadas ondas sonoras.

La velocidad de propagación del sonido depende de las características del medio en el que se realiza dicha propagación y no de las características de la onda o de la fuerza que la genera. El sonido en el aire se genera al crearse una variación o perturbación que establece una serie de ondas de presión (ondas sonoras) que fluctúan por encima y por debajo de la presión del aire en el equilibrio (la atmosférica) y que, en general, se propagan en todas las direcciones desde la fuente sonora. Nuestro oído es sensible a estas fluctuaciones de presión y las convierte en impulsos eléctricos que se transmiten al cerebro para su interpretación. A 20° y a presión atmosférica la velocidad de propagación del sonido en el aire es de 342.2 m/s.

Al analizar el sonido existen tres elementos a considerar: la fuente emisora, que puede ser bien deseable o indeseable; el medio a través del que se produce la transmisión del sonido y finalmente el receptor. Cuando se desea escuchar el sonido (p.ej. palabra o música) es necesario optimizar las condiciones de producción, transmisión y recepción, mientras que si lo que se desea es no recibir el sonido habrá que hacer justo lo contrario.

Análisis espectral del sonido

El sonido que nos llega es, en general, superposición de ondas sonoras de distintas frecuencias. Denominamos banda de audiofrecuencias a la gama de frecuencias audibles. El oído humano puede escuchar frecuencias entre 20 Hz y 20 kHz, aunque es más sensible en el intervalo entre 1 y 5 kHz. Por otra parte, en música generalmente se emplean sonidos comprendidos en el rango de 30 Hz a 12 kHz.

El caso más sencillo de sonido corresponde a un tono puro, en el que la onda de sonido que se propaga puede representarse por una función armónica, que contiene una única frecuencia. Cualquier sonido periódico puede representarse por una superposición discreta de tonos puros, de acuerdo con el teorema de Fourier, cada uno de ellos con su

correspondiente intensidad. Asimismo, cualquier función describiendo un sonido complejo puede representarse mediante una integral de Fourier en la que puede aparecer un continuo de frecuencias en vez de la serie de frecuencias discretas que teníamos para una función periódica.

Cualquier sonido sencillo, como es el caso de una nota musical, puede describirse mediante tres parámetros: la intensidad, el tono y el timbre, que corresponden a tres características físicas: la amplitud, la frecuencia y la composición espectral (o en frecuencias), dada por la forma de la onda.

La intensidad, como veremos más adelante, está asociada a la cantidad de energía que lleva la onda por unidad de tiempo y de superficie en la dirección de propagación. El tono está asociado con la frecuencia de la componente principal del sonido (o armónico fundamental) y el timbre con las intensidades relativas de otras frecuencias además de la frecuencia fundamental.

Se denomina ruido a un sonido que contiene una combinación aleatoria de frecuencias, aunque con frecuencia se utiliza la palabra ruido para todo sonido que distrae, incomoda o daña al receptor humano y perturba sus actividades cotidianas (trabajo, descanso, entretenimiento, estudio, salud, etc.) en un momento dado. A su vez, también puede definirse ruido como toda señal espuria aleatoria que coexiste en el dominio de un conjunto de señales que sí son de interés para la aplicación en cuestión.

Ruido

Como dijimos, el ruido se define como toda señal espuria y aleatoria que por lo general interfiere con una señal de interés. Resulta interesante y útil poder llegar a una comprensión y clasificación del mismo, la cual puede llevarse a cabo según su composición espectral

Ruido blanco

El ruido blanco consiste en una señal de banda ancha que contiene todas las frecuencias del espectro con distribución aleatoria de amplitud que da una densidad espectral independiente de la frecuencia. En la práctica, su rango a efectos de medidas experimentales va de los 20 Hz a los 20 kHz.

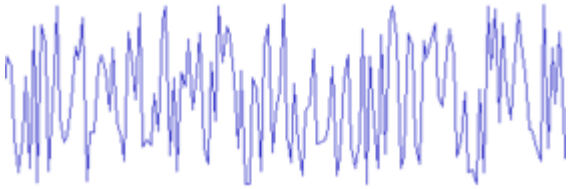


Figura 4 - Ruido blanco en el dominio del tiempo

Si se representa la densidad de energía en función de bandas de octava en vez de linealmente frente a la frecuencia, se obtiene una recta ascendente de pendiente 3

dB/octava ya que en cada banda hay el doble de frecuencias que en la anterior. El nombre proviene de la luz blanca, que es una mezcla de todas las frecuencias.

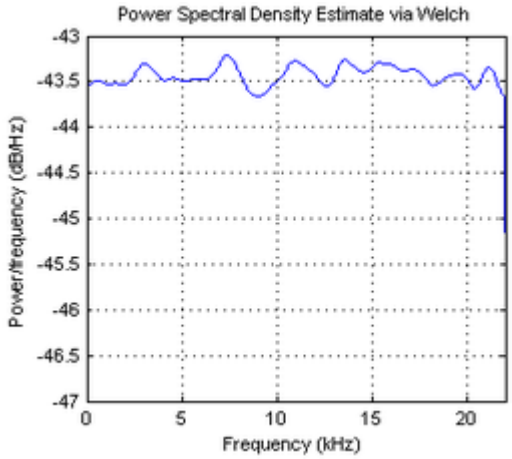


Figura 5 - Ruido blanco en el dominio de la frecuencia

El sonido producido por el agua corriente se ajusta bastante bien al ruido blanco. La imagen de un televisor analógico cuando no está sintonizado ningún canal también es ruido esencialmente blanco. El ruido blanco se utiliza para la calibración de la respuesta en frecuencia de equipamientos electrónicos que trabajan con sonido. Cabe destacar que en el presente trabajo la utilización del ruido blanco es esencial, por lo que resulta muy importante su caracterización.

Ruido rosa

El ruido rosa o ruido 1/ν es un ruido con una distribución de frecuencias tal que su densidad espectral de potencia es proporcional a la inversa de la frecuencia. Esto implica que su nivel, por bandas de tercio de octava, es constante. Por el contrario, el nivel del ruido rosa por intervalo fijo de frecuencias va decayendo a razón de 3 dB/octava, de forma no lineal, aunque las frecuencias se representen en un eje lineal.

El nombre "ruido rosa" obedece a una analogía con la luz blanca (que es una mezcla de todos los colores) que, después de ser coloreada de forma que se atenúen las frecuencias más altas (los azules y violetas) resulta un predominio de las frecuencias bajas (los rojos). Así pues, el ruido rosa es ruido blanco coloreado de manera que es más pobre en frecuencias altas, esto es: en agudos. Mientras que el ruido blanco es más "silbante", el ruido rosa es más "apagado".

Ruido marrón

Su densidad espectral de energía es proporcional a $1/\nu^2$ y, por tanto, tiene más energía a bajas frecuencias que el ruido rosa.

4.1.1.1.2 Electroacústica y sistemas electrónicos de sonido

En numerosas aplicaciones prácticas sucede que nos interesa captar y trasmitir sonido en forma electrónica ya sea para su procesamiento, amplificación, grabación y/o reproducción. Es por ello que resulta importante hablar también de la electroacústica, los diferentes medios de transducción del sonido, sus etapas típicas de procesamiento y aplicaciones tecnológicas comunes de esta disciplina.

Transductores electroacústicos

Los transductores son dispositivos capaces de transformar energía de una naturaleza en energía de otra naturaleza. En electroacústica nos interesamos por aquellos capaces de realizar esa transformación entre energía de naturaleza acústica y energía de naturaleza eléctrica y viceversa.

Al aplicar una determinada magnitud a la entrada de un transductor, la cual denominamos excitación, se obtiene una magnitud a la salida, llamada respuesta. Excitación y respuesta se relacionan a través de la función de transferencia que es característica de cada transductor. Dado que nuestra aplicación se realiza en un ancho de banda grande esta función de transferencia puede venirnos dada a través del Diagrama de Bode, donde es posible la observación de la respuesta en amplitud y fase en función de la frecuencia.

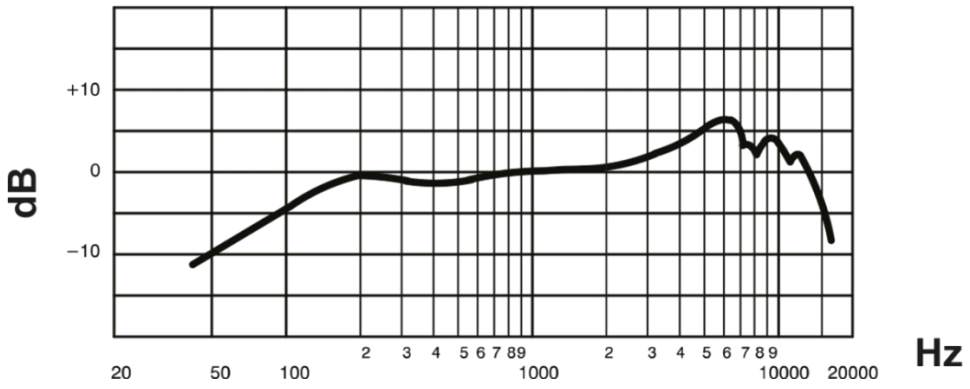


Figura 6- -Diagrama de Bode de amplitud de un micrófono de uso general

La función de transferencia depende de las condiciones de excitación y de las condiciones de medida. Por ejemplo, la presión acústica obtenida de un altavoz depende de la excitación, su potencia (Tensión y corriente eléctrica aplicadas), de la posición de medida (distancia y dirección), de las condiciones de montaje (sobre pantalla, libre, en caja acústica, etc.), del medioambiente acústico (campo libre, presencia de un sólido reflectante, sala anecoica, etc.), de la temperatura y la presión atmosféricas. En resumen, de las condiciones eléctricas, mecánicas y acústicas.

Procesamiento del sonido

En este punto se ejemplifica de manera muy general la cadena de procesamiento de audio de un sistema de sonido típico. Como ya fue explicado anteriormente, se considera un sistema que debe convertir una señal del dominio acústico, procesarla convenientemente

en el dominio electrónico (puede ser de forma analógica o digital o una combinación de ambas) y devolver a su salida nuevamente la señal procesada al dominio acústico.

En general este tipo de sistemas presentan características que los distinguen de otros tipos de sistemas de procesamiento de señales. En primer lugar, se tienen en cuenta las características de las señales de sonido en el rango audible, por lo que este tipo de sistemas en la mayoría de los casos presentan un ancho de banda de 20kHz, correspondientes al rango de frecuencias captables por el oído humano. A su vez, pueden seguir un esquema monoaural (1 señal única) o estéreo (canal izquierdo y derecho).

Simplificadamente, un sistema de audio se puede representar por el siguiente diagrama en bloques:

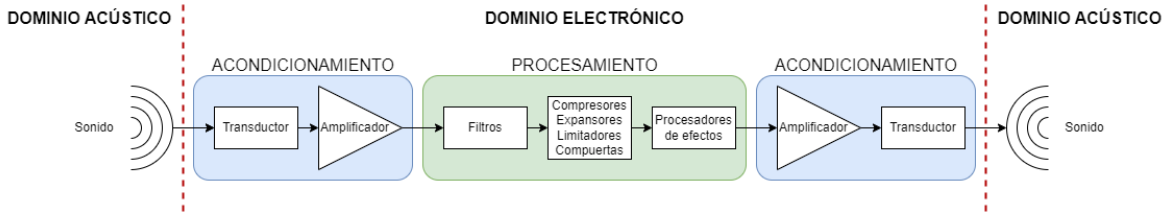


Figura 7 - Cadena de procesamiento de audio

En el esquema se identifican tanto el dominio acústico como el electrónico. Dentro de la cadena existen etapas de acondicionamiento, que adecuan la señal proveniente de un medio al siguiente, y la etapa de procesamiento, que realiza cambios en la señal entre su entrada y su salida.

Audio digital

Hasta ahora hemos hablado del sonido en términos analógicos, puesto que la percepción de nuestros oídos necesita de la vibración del aire para poder percibir el sonido. Sin embargo, en la actualidad se suele recurrir a la digitalización de la señal, como mínimo en algún momento de los distintos procesos que supone la producción de una grabación sonora, cuando no en todo el proceso.

La grabación digital se basa en el principio del muestreo, es decir, se toman datos a intervalos regulares de la señal sonora a los que se les da un valor numérico. Este valor es convertido en datos binarios que se van almacenando mediante lo que se denomina conversión analógico-digital sobre el soporte correspondiente. Durante la reproducción, esos datos son leídos secuencialmente para reconstruir la onda original a partir de un proceso inverso al de la codificación, denominado conversión digital-analógica.

Hay que tener en cuenta que nuestro oído escucha analógicamente, es decir, que el final del proceso nos encontraremos siempre con una vibración en el aire que es analógica. Precisamente por ello, se deben tener en cuenta diversos principios relacionados con la escucha en el momento de realizar una grabación digital, lo que determina el uso de determinados parámetros a la hora de efectuar los procesos de digitalización y reconstrucción de la señal sonora.

La frecuencia de muestreo hace referencia a los intervalos regulares en los que se obtienen muestras, es decir, datos medidos, de la señal analógica original. Esta frecuencia debe ser elevada para que la reconstrucción de la señal sea lo más fiel posible a la original.

Shannon y Nyquist propusieron en 1948 los principales teoremas en los que se basa la digitalización del sonido: el teorema del muestreo dice que deben tomarse al menos dos muestras por cada ciclo de la onda sonora para obtener una información suficiente de ella. Si se toman menos muestras, no es posible registrar correctamente el espectro de frecuencias de la señal. Por otra parte, y como consecuencia del principio anterior, para poder registrar cierta frecuencia de sonido es necesario que el sistema tenga como frecuencia de muestreo al menos el doble de la frecuencia a registrar. Así pues, para registrar frecuencias de hasta 20 KHz, que es la más alta que puede oír un humano, es necesario que el sistema tenga como frecuencia de muestreo mínima 40 KHz.

Hay un problema muy importante derivado del uso de los sistemas digitales, relacionado con la frecuencia de muestreo: en el mundo físico existen, como hemos visto anteriormente, frecuencias que están por encima de los límites de percepción, llamadas ultrasonidos. Los sistemas analógicos, a la hora de registrar el sonido, no pueden captar estas frecuencias. Sin embargo, los digitales, al tratar de registrar frecuencias que estén por encima del límite del sistema (20 KHz en el caso de este sistema ideal que tiene como frecuencia de muestreo 40 KHz), codifican estas como ruidos que pueden afectar a la calidad del sonido registrado. En la Figura 8, puede verse esto, debido a que se utiliza una frecuencia de muestreo ligeramente superior a la frecuencia a digitalizar, sin llegar al doble de esta, como debería ser. En línea continua, se observa la onda original mientras que, en línea discontinua, la onda resultante de la conversión errónea. A este problema se le llama aliasing. Para evitarlo, todo sistema digital está provisto de un filtro que elimina cualquier frecuencia por encima de la máxima registrable. A este filtro se le llama antialiasing.

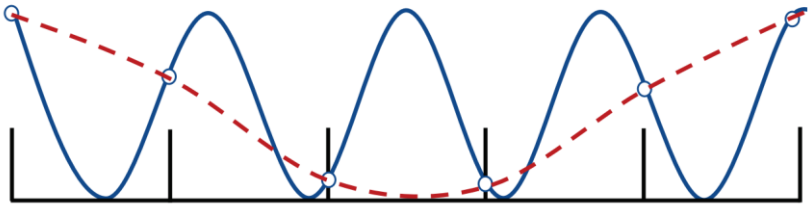


Figura 8 - Frecuencia de muestreo no adecuada

Las frecuencias de muestreo estándar más difundidas en los sistemas digitales de sonido se corresponden con 44100Hz y 48000Hz.

Una vez efectuada la lectura de la señal a intervalos regulares, se efectúa la cuantización, es decir, se le da un valor a cada uno de los datos obtenidos. Este valor debe estar en relación con la frecuencia y el nivel de presión sonora de la señal. Puesto que estamos en un entorno digital, este valor debe tener una expresión numérica concreta, de entre un conjunto de valores prefijados. Cuantos más valores tengamos para escoger, mucho más fiel será la señal digital a la original.

En un sistema digital binario, el número de niveles que tenemos está directamente relacionado con el número de bits del sistema. Así, un sistema de 1 bit podrá representar

solo dos valores de volumen, uno de 2 bits representará cuatro valores, etc. En el estándar de CD de audio, con 16 bits de resolución, se puede representar 65536 niveles de señal distintos, lo que ofrece una calidad bastante aceptable. Actualmente, los sistemas digitales permiten efectuar grabaciones con 24 bits de resolución y se trabaja para conseguir superar este número.

La señal convertida en un flujo de datos binarios debe ser codificada para ser colocada sobre un soporte. Para ello se utiliza un código de canal que aprovecha las posibilidades del soporte para registrar los datos binarios en forma de transiciones entre dos estados, que representarían los 1 y los 0.

4.1.1.2 Identificación de sistemas mediante la respuesta al impulso

Como fue planteado al comienzo de este informe, el objetivo del proyecto es la obtención de un modelo de un sistema electroacústico para su posterior utilización como emulador digital de dicho sistema. En este sentido, uno de los métodos más difundidos en el marco del análisis de sistemas en general, y en los sistemas de sonido en particular, es el método de la respuesta al impulso. Analizaremos este procedimiento con la intención de obtener importantes conclusiones para nuestro desarrollo.

La respuesta al impulso (RI) se puede definir como la respuesta en el dominio del tiempo (tiempo vs. amplitud) del sistema que estamos analizando bajo un estímulo impulsivo o de corta duración.

Matemáticamente la respuesta impulsional de un sistema es la que se presenta en la salida cuando en la entrada se introduce un impulso. Un impulso es el caso límite de un pulso infinitamente corto en el tiempo pero que mantiene su área o integral (por lo cual tiene un pico de amplitud infinitamente alto). Aunque es imposible obtener amplitud infinita en un intervalo infinitamente corto en cualquier sistema real, es un concepto útil como idealización, debido principalmente a la simplicidad de su uso en la integración.

Matemáticamente, un impulso se representa por una función Delta de Dirac. Cuando se trabaja con sistemas discretos el impulso se aproxima por medio de un pulso que tiene área unidad y de ancho tiene el periodo de tiempo entre dos muestras. Si el tiempo entre dos muestras consecutivas x[n] y x[n+1] lo tomamos como τ, entonces el valor del impulso será el inverso de 1/τ, de modo que el área, que es su producto, valga la unidad.

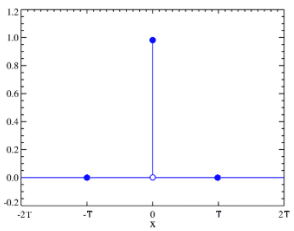


Figura 9 - Función impulso para el caso discreto

Supongamos que T es un sistema discreto, es decir, que toma una entrada x[n] y produce una salida y[n].

$$y[n] = T\{x[n]\} \quad (1)$$

Se demuestra que:

$$y[n] = \sum_{k} x[k]\, h[n-k] \tag{2}$$

La sucesión h[n] es la respuesta a impulso del sistema representado por T.

Se observa que la operación definida en la ecuación (2) representa la convolución en el tiempo entre la respuesta al impulso del sistema y la señal de entrada al mismo, lo que nos permite obtener la señal de salida del sistema.

Por esto es que decimos que la h[n] o h[t] para el tiempo continuo, reúne todas las características del sistema, y su transformada de Fourier se corresponde con la característica en frecuencia del sistema en cuestión.

En los sistemas electroacústicos reales no es posible generar un impulso perfecto para aplicar como prueba en ninguna entrada. Por lo tanto, se utilizan diferentes aproximaciones de pulsos muy breves. Debido a que el pulso es suficientemente corto comparado a la respuesta a impulso, el resultado obtenido será bastante cercano a la respuesta a impulso teórica. Por otro lado, es posible obtener la respuesta al impulso de un sistema utilizando métodos indirectos de procesamiento de señales, como ser la aplicación de un estímulo conocido y luego proceder la deconvolución entre este y la respuesta del sistema bajo estudio. Como ejemplos de este método se desatacan la utilización de barridos senoidales y las secuencias MLS.

El sistema a analizar puede ser desde un micrófono o un altavoz, a un dispositivo electrónico como un ecualizador. O incluso un punto en una habitación con una determinada fuente sonora en ella.

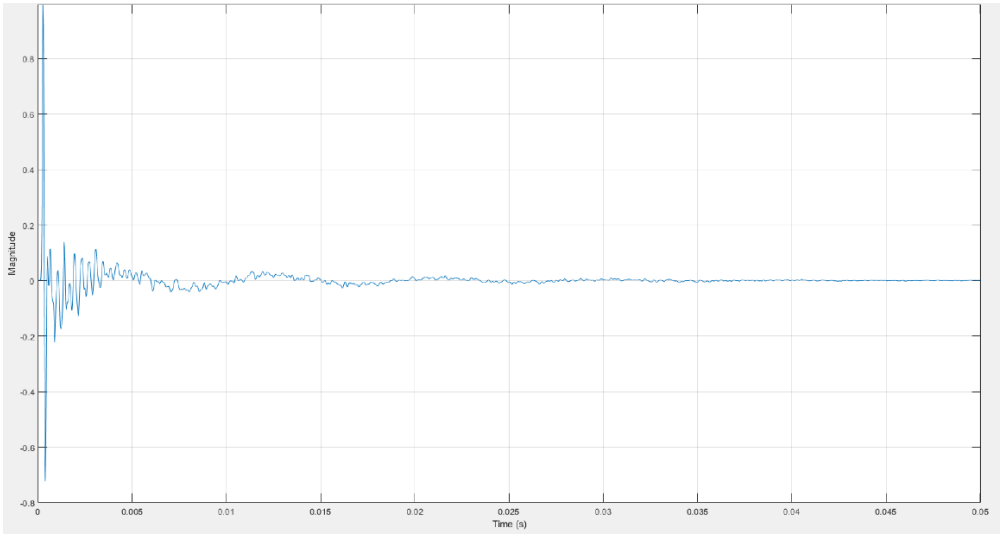


Figura 10 - Respuesta al impulso de un sistema desconocido en el dominio del tiempo

Si hablamos de análisis acústico, podemos entender la respuesta al impulso como la "firma acústica" del sistema que analizamos, en el sentido de que cada elemento que midamos tendrá una respuesta de impulso distinta.

A nivel práctico, para obtener la respuesta al impulso de un sistema de audio, se puede enviar un impulso unitario al sistema y medir la señal de salida resultante. Por ejemplo, si se tratara de un sistema parlante-gabinete esta medición se realiza con un micrófono colocado a una distancia adecuada del altavoz. La respuesta al impulso obtenida muestra cómo el sistema de audio responde a señales de entrada muy cortas en función del tiempo.

La respuesta al impulso de un sistema de audio se puede utilizar para determinar diversas características del sistema, como la respuesta de frecuencia, la distorsión, la resonancia y la directividad. A su vez, también es útil para diseñar filtros digitales y optimizar el rendimiento de los sistemas de audio.

Por otro lado, resulta necesario saber qué información podemos obtener de un impulso en el dominio de la frecuencia. Aquí entra en juego la transformada de Fourier, que nos permite relacionar el dominio frecuencial con el dominio del tiempo. Una respuesta al impulso pertenece al dominio temporal, pero se puede convertir al dominio frecuencial mediante la transformada de Fourier (y viceversa).

Dependiendo el tipo de sistema analizado es si nos interesará más evaluar la respuesta temporal o la respuesta frecuencial. Por ejemplo, para el caso de un recinto reverberante, resulta mucho más útil evaluar las características temporales, puesto que aportan mucha información respecto a los diferentes retardos que sufre la señal acústica en su propagación. Por el contrario, en el ejemplo del altavoz, la respuesta en frecuencia nos indica mucho mejor cómo se comporta el sistema ante diferentes frecuencias de entrada, su ganancia y ancho de banda. Es en este sentido que, si bien una respuesta al impulso es esencialmente una señal en el dominio del tiempo, resulta importante definir una relación, al menos intuitiva y general de cómo se relaciona con el dominio de la frecuencia.

Comencemos analizando la forma de onda típica de una RI en el dominio del tiempo, la cual se muestra en la Figura 10. Esta forma de onda pertenece a la respuesta al impulso de un sistema gabinete-altavoz. Para diferentes sistemas de este tipo, esta respuesta puede variar dependiendo de diversos factores, como la geometría y las propiedades acústicas del altavoz, el tipo de filtro utilizado en el circuito de cruce, y la ubicación del micrófono de medición. Sin embargo, la respuesta al impulso típica de un sistema de audio puede tener una forma de onda que se asemeja a una función de pulso gaussiano o a una función de pulso exponencial.

Un pulso gaussiano es una función de onda que tiene una forma de campana y se describe por una ecuación matemática que involucra una función exponencial y una función gaussiana. Esta forma de onda se caracteriza por una amplitud máxima en su centro y una rápida caída a medida que se aleja del centro.

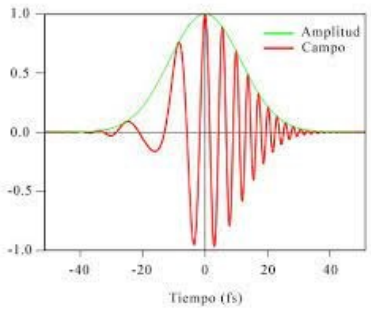


Figura 11 - Pulso gaussiano

Por otro lado, un pulso exponencial tiene una forma de onda que aumenta rápidamente a un valor máximo y luego se desvanece exponencialmente con el tiempo. Esta forma de onda se describe por una ecuación matemática que involucra una función exponencial.

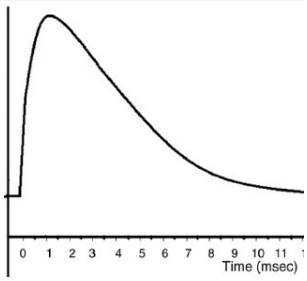


Figura 12 - Pulso exponencial

Estas formas de onda no pretenden ser un modelo exhaustivo de una RI de un sistema de audio, sino más bien funcionar como una aproximación muy simplificada que nos permita deducir algunas características en frecuencia de este tipo de sistemas.

La forma de onda de la respuesta al impulso afecta directamente la respuesta en frecuencia del sistema. Si la respuesta al impulso tiene una caída rápida después de su pico máximo, esto indica que el sistema de audio tiene una respuesta en frecuencia que cae rápidamente a medida que aumenta la frecuencia de la señal de entrada. Por otro lado, si la respuesta al impulso tiene una caída más lenta después de su pico máximo, esto indica que el sistema de audio tiene una respuesta en frecuencia que decae más lentamente a medida que aumenta la frecuencia de la señal de entrada.

En general, los primeros picos en la respuesta al impulso están más relacionados con las frecuencias medias, mientras que los picos posteriores y la caída de la respuesta al impulso están más relacionados con las frecuencias altas. Los picos iniciales de la respuesta al impulso pueden ser el resultado de la resonancia en el sistema de audio. Por ejemplo, en un altavoz, el primer pico puede ser causado por la resonancia de la caja del altavoz, mientras que el segundo pico puede estar relacionado con la resonancia de la membrana del altavoz. Estas resonancias pueden estar más relacionadas con las frecuencias medias, ya que las frecuencias bajas y altas pueden no tener suficiente energía para producir una resonancia fuerte. Por último, si la RI presenta oscilaciones de periodo largo, estas oscilaciones corresponden a la respuesta en bajas frecuencias, y pueden coincidir con la resonancia conjunta del sistema si existiese.

En la Figura 13 se representa la respuesta en frecuencia del sistema cuya RI es la correspondiente a la Figura 10. Se puede observar que se trata de un sistema complejo y que la energía decae a medida que se avanza hacia frecuencias más elevadas.

En la Figura 14 se muestra dicha respuesta separada en las diferentes áreas del espectro: frecuencias bajas, hasta los 500Hz, frecuencias medias, de 500 a 4000Hz y frecuencias altas, de 4000 a 20000Hz. El eje de frecuencias se representa en escala logarítmica para una mejor visualización de estas bandas.

Por último, en la Figura 15 se muestra como cada una de estas bandas de frecuencia contribuye en el efecto de la respuesta al impulso. Como fue explicado, se puede notar el efecto de las bajas frecuencias en la oscilación de menor periodo de la respuesta total, y la contribución de las medias y altas frecuencias en los picos de mayor tamaño de la señal. Además, se puede apreciar la relación que existe entre el decaimiento temporal de la respuesta al impulso y la caída de energía de su respuesta en frecuencia. Vemos que el sistema presenta una importante atenuación en altas frecuencias que se corresponde con el decaimiento de la señal en el tiempo para esta banda y, consecuentemente, para la respuesta total.

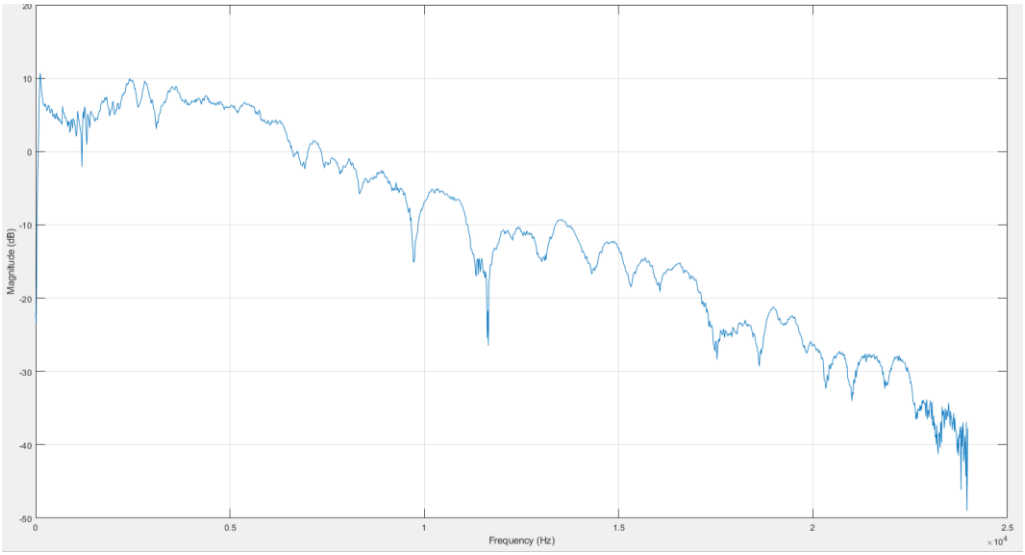


Figura 13 - Respuesta en frecuencia del sistema analizado

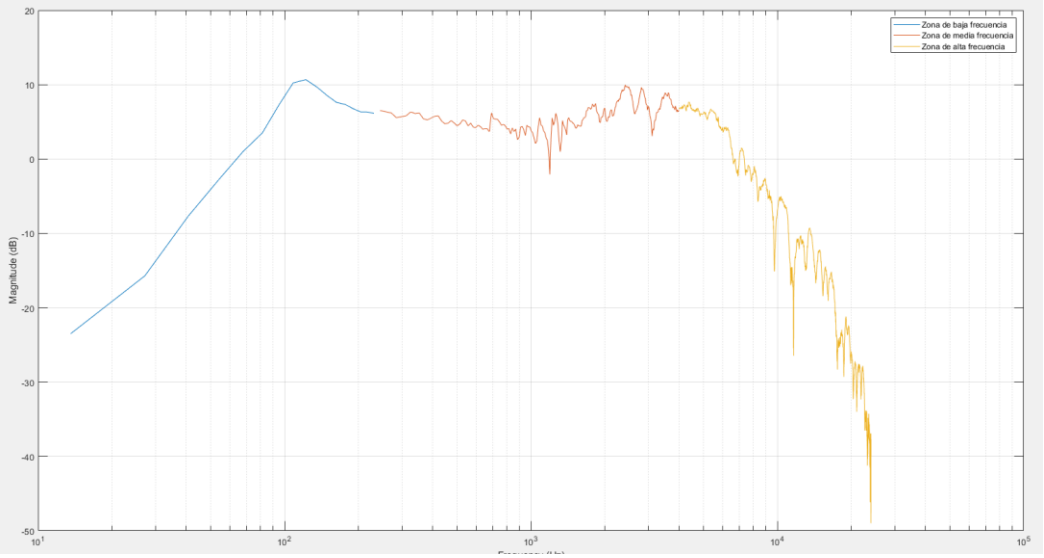


Figura 14 - Respuesta en las diferentes bandas de frecuencia

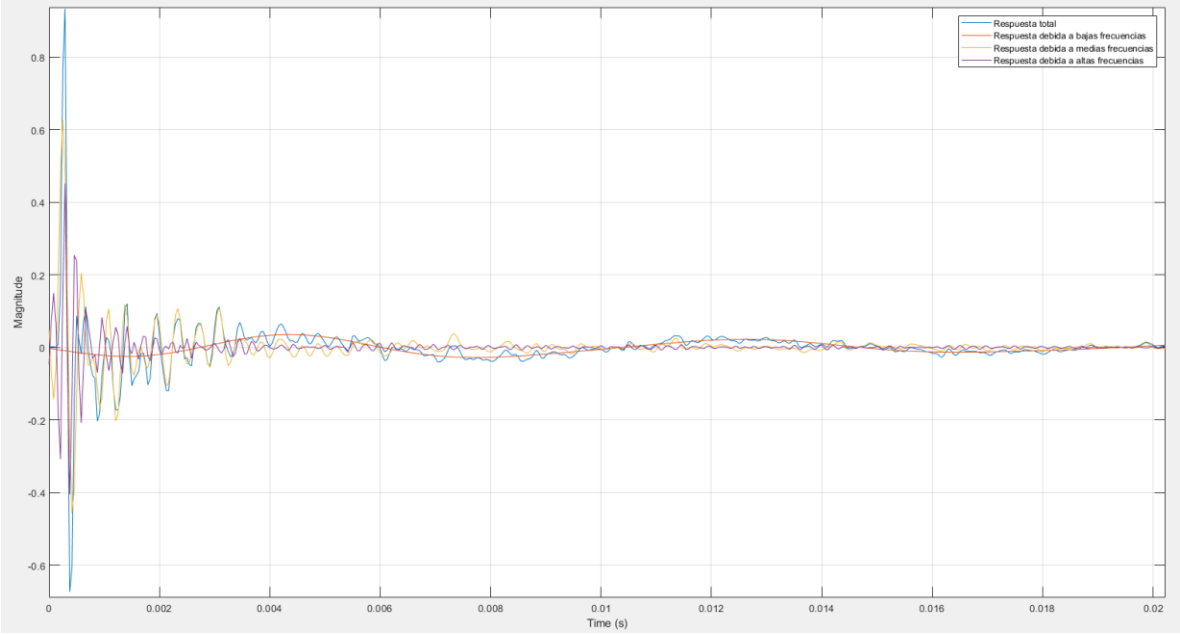


Figura 15 - Contribución de las diferentes bandas de frecuencia en la respuesta al impulso

4.1.1.2.1 Conclusiones del método

Vemos que analizando una RI típica capturamos todas las características en el dominio del tiempo del sistema analizado. En este sentido podemos decir que lo que limita una mejor o peor caracterización es la cantidad de muestras que seamos capaces de tomar para obtener la respuesta al impulso. Por ejemplo, en caso de un espacio muy reverberante, para no perder esa cola de reverberación deberíamos ser capaces de

almacenar tantas muestras equivalentes al tiempo en que dicho efecto se extinga o sea lo suficientemente pequeño para ser despreciable.

Para ejemplificar tomemos un recinto cuya curva de decaimiento tenga una pendiente de 30dB/s, es decir que la energía de reverberación demora 1 segundo en decaer 30dB, o lo que es equivalente, que posee un TR30=1s. Si deseamos captar hasta ese nivel de reverberación, suponiendo una frecuencia de muestreo estándar de 48kHz tendremos que almacenar un total de 48000 muestras de la señal de respuesta. Además, si tomamos una resolución de audio profesional de 24 bits, esto equivale a un espacio de memoria de 144000 bytes.

Aquí subyace una de las principales complicaciones de este método. En ocasiones, en búsqueda de una buena caracterización de un sistema acústico complejo, como un recinto muy reverberante, se obtienen RI's de varios segundos, lo cual puede implicar una secuencia de mucho tamaño y que puede resultar impráctica tanto en su almacenamiento como en su utilización posterior.

Este último punto queda más claro si consideramos la naturaleza digital de una secuencia RI. En esencia, una RI es una serie de coeficientes de un filtro digital de tipo FIR (finite impulse response). Un filtro FIR, en su forma más simple, se representa mediante una convolución entre una señal de entrada x[n] y el kernel del filtro h, que en nuestro caso está representado por la secuencia finita RI:

$$y[n] = \sum_{k=0}^{M-1} h_k \, x[n-k] \tag{3}$$

Donde M es el número de muestras que posee h.

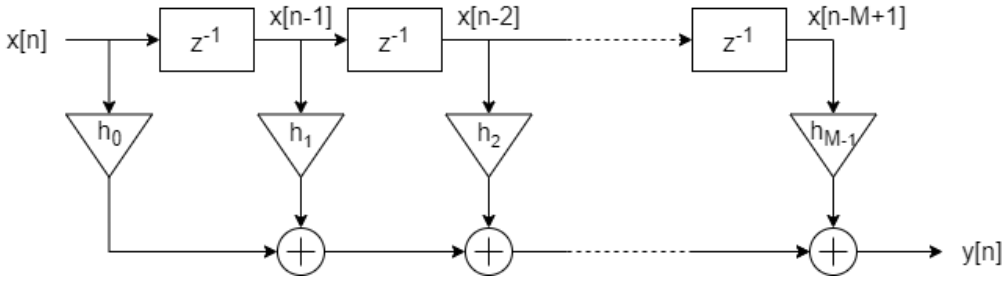


Figura 16 - Implementación de un filtro FIR

En la Figura 16 se tiene una implementación mediante sumas acumuladas de la operación descripta en la ecuación (3). Cada uno de los bloques z^{-1} representa una operación de retardo de un intervalo de muestreo. Puede demostrarse que, mediante esta implementación, un filtro FIR de M muestras presenta un retardo global en su salida de M/2 muestras. Es decir que la salida del filtro actualiza su salida en un tiempo $\tau = \frac{M}{2} T_s$ tras una entrada determinada, donde $T_s = \frac{1}{f_s}$ y F_s es la frecuencia de muestreo digital.

Volviendo al ejemplo presentado anteriormente, para la RI de 1 segundo de duración, para la que habíamos considerado $F_s = 48kHz$ tenemos:

$$M = 48000 \tag{4}$$

$$\tau = \frac{M}{2}\, T_s = \frac{M}{2\, f_s} = 0.5\ s \tag{5}$$

Una latencia de 0.5 segundos para un filtro de audio en tiempo real es catastrófica, puesto que en este orden de magnitud nuestro oído percibiría la señal de salida como un eco de la señal de entrada. Como referencia, el oído humano puede distinguir una señal retardada respecto de otra a partir de los 50 milisegundos, por lo que este tiempo es la latencia máxima admisible por un sistema digital de audio en tiempo real.

Este problema es menos notorio en RI's que caracterizan altavoces o sistemas electrónicos de audio, donde la duración de estas secuencias es mucho menor. Existen también implementaciones de convolución rápida que permiten salvar este problema para secuencias de mayor tamaño, pero que aumentan la complejidad de procesamiento para la operación descripta en (3).

Otro problema que presenta el método descripto para la obtención de la RI es que se encuentra limitada por el ruido de fondo. Si el ruido fuera excesivo, se perdería resolución en aquellas muestras cuyo valor sea excedido por este, por lo que se demanda un nivel elevado de SNR. En este punto, veremos que el método presentado en este informe aporta una ventaja respecto al de la RI.

Dado que para sistemas LTI la h[n] es una señal de energía finita, podemos considerar sin cometer un error elevado que la energía de la RI de la mayoría de los sistemas que analizaremos se encuentra contenida en un tiempo pequeño luego del pico máximo de la señal. La elección de cuán pequeño es este tiempo en los sistemas prácticos determinará tanto la fidelidad del modelo obtenido como así también la complejidad del mismo (su tamaño en muestras).

Por ejemplo, en la Figura 17 se representa la RI de un sistema electroacústico compuesto por un altavoz en un gabinete, a la izquierda en el dominio del tiempo y a la derecha, luego de realizar la transformada de Fourier, en el dominio de la frecuencia. Vemos que, en el dominio temporal, luego del pico, el nivel de señal se extingue rápidamente y se vuelve cercano a 0 cerca de los 6 milisegundos. Esto, a una frecuencia de muestreo de 44.1kHz equivaldría aproximadamente a 250 muestras, lo cual es un valor razonable para un sistema práctico. En el dominio de la frecuencia se puede observar una buena caracterización, con suficiente resolución para captar picos y valles de decenas de Hertz de diferencia y un ancho de banda mayor a 15kHz.

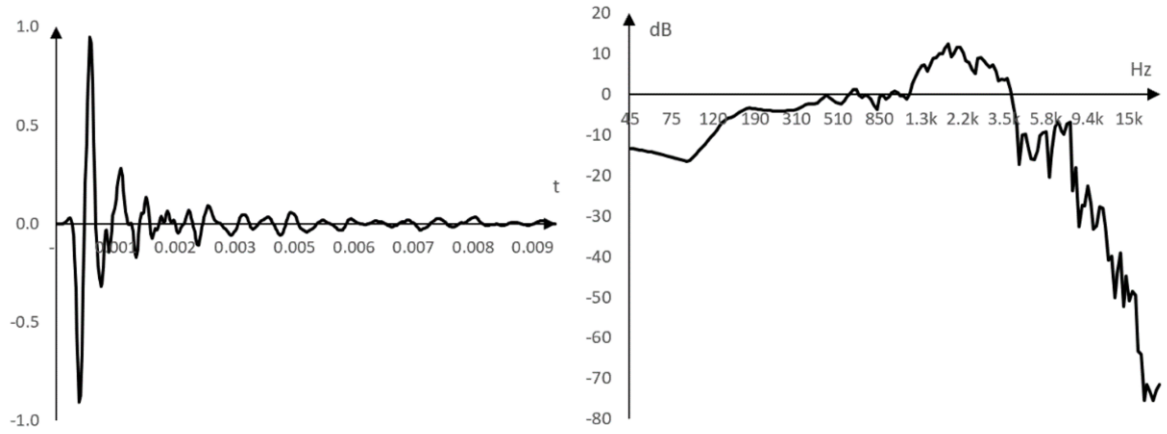


Figura 17 - RI de un sistema altavoz + gabinete en el dominio del tiempo y la frecuencia

El método de caracterización y emulación por RI es ya prácticamente un estándar de la audiotécnica y el procesamiento digital de señales de audio. Se han analizado sus ventajas y potencialidades como así también sus desventajas y dificultades para su implementación. Estas dificultades, si bien ya han sido sorteadas en dispositivos de alta complejidad como pueden ser sistemas DAW (digital audio workstation) que operan en PC's o en procesadores DSP de alto costo, suponen un salto tecnológico para sistemas de bajo costo y que no disponen de alta velocidad y/o capacidad de cómputo.

Es en este sentido que, tomando como punto de partida la caracterización de sistemas electroacústicos por RI, se propondrá una alternativa a ella que sea implementable y permita obtener resultados aplicables en dispositivos sencillos.

4.1.1.3 Filtros adaptativos

El concepto de filtro adaptativo sugiere el de un dispositivo que intenta modelizar la relación entre señales en tiempo real de forma iterativa. Se diferencia de los filtros digitales en que éstos últimos tienen coeficientes invariantes en el tiempo, mientras que un filtro adaptativo puede cambiar su forma de comportarse, es decir, pueden cambiar sus coeficientes de acuerdo con un algoritmo adaptivo. De hecho, no se conocen los coeficientes del filtro cuando se diseña, debido a que estos coeficientes son calculados cuando el filtro se implementa y se reajustan automáticamente en cada iteración mientras dura su fase de aprendizaje.

En el caso de un filtro FIR como el de la Figura 16, esto equivale a decir que los coeficientes h_1, h_2, … h_{M-1} no tienen un valor constante, sino que evolucionan temporalmente, dando lugar a una serie de coeficientes para cada instante de tiempo n: $h_1[n]$, $h_2[n]$, … $h_{M-1}[n]$.

4.1.1.3.1 Filtro de Wiener

Para un sistema discreto como el que se muestra en la Figura 18 la salida y[n] puede expresarse como:

$$y[n] = \sum_{k=0}^{M-1} w_k[n]\, x[n-k] \tag{6}$$

O:

$$y[n] = \bar{w}[n]^T\, \bar{x}[n] \tag{7}$$

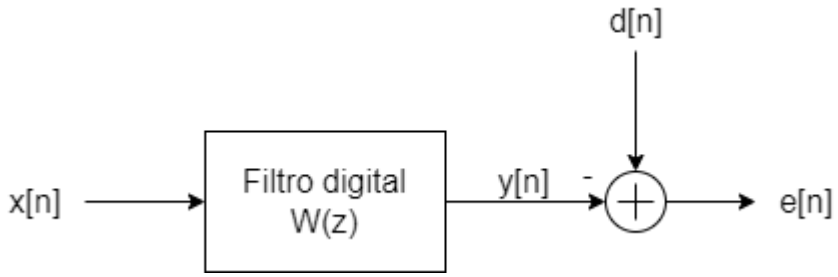


Figura 18 - Sistema discreto de Wiener

$\bar{x}[n]$ es el vector de la señal de entrada y $\bar{w}[n]$ el vector de coeficientes del filtro digital W.

$$\bar{x}[n] = \begin{bmatrix} x[n] & x[n-1] & \cdots & x[n-M+1] \end{bmatrix} \tag{8}$$

$$\bar{w}[n] = \begin{bmatrix} w_0[n] & w_1[n] & \cdots & w_{M-1}[n] \end{bmatrix} \tag{9}$$

Se tiene una señal objetivo o señal deseada d[n] y una señal error e[n] que representa la diferencia entre d[n] e y[n].

$$e[n] = d[n] - y[n] \tag{10}$$

El objetivo es encontrar los valores para cada uno de los coeficientes del filtro que minimicen la función de costo cuadrática dada por el error cuadrático medio:

$$J = E[e^2[n]] \tag{11}$$

Donde E denota el valor esperado o la esperanza de la variable aleatoria discreta en cuestión, en este caso el cuadrado de la señal de error. En este procedimiento consideraremos que todas las señales presentes son estacionarias y ergódicas.

Se dice que un proceso aleatorio es estacionario cuando las funciones de densidad que lo describen son invariantes en el tiempo; y que es ergódico si el promedio en el tiempo es equivalente al promedio de un conjunto de muestras. Por tanto, es factible decir que la esperanza del error cuadrático será invariante en el tiempo y puede ser calculada como el promedio de un conjunto de muestras.

Combinando las ecuaciones (7), (10) y (11) podemos escribir:

$$J = w^T A\, w - 2\, w^T B + C \tag{12}$$

Donde:

$$A = E[x[n]x[n]^T] \tag{13}$$

$$B = E[x[n]\, d[n]] \tag{14}$$

$$C = E[d^2[n]] \tag{15}$$

Lo importante a destacar de esto, es que al escribir la función de costo con la forma que se da en la ecuación (12), queda claro que el error cuadrático medio es una función cuadrática de cada uno de los coeficientes del filtro FIR. Cuando se hace una gráfica de la función de costo contra dos coeficientes cualquiera, se obtiene una superficie parabólica con forma de cuenco, que es una superficie de error, tal como se muestra en la Figura 19.

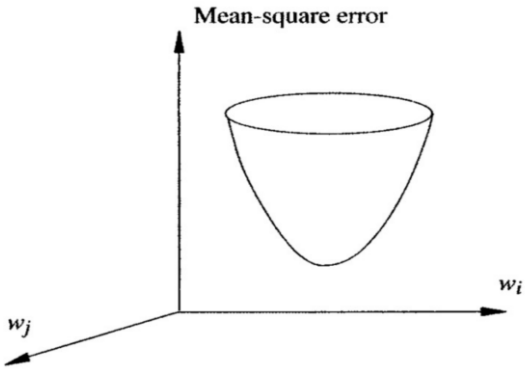


Figura 19 - Superficie de error cuadrático

Entonces, un enfoque para determinar los coeficientes de un filtro que minimicen el error cuadrático medio del problema modelo de la Figura 18 es usar un filtro que sea adaptativo. Los datos se usan de forma secuencial para ajustar los coeficientes del filtro de forma tal que evolucionen en la dirección en la que se minimice el error cuadrático medio. Esto nos conduce al llamado algoritmo de gradiente descendente.

El algoritmo de gradiente descendente está basado en el hecho de que, como ya hemos visto, la superficie de error tiene forma cuadrática. Esto sugiere que, si un coeficiente del filtro se ajusta por una cantidad pequeña, que es proporcional al negativo del gradiente local de la función de costo con respecto a ese coeficiente del filtro, entonces el coeficiente está obligado a moverse hacia el mínimo global de la superficie de error. Si todos los coeficientes del filtro son ajustados usando este algoritmo, la adaptación para el vector de coeficientes del filtro se puede escribir como:

$$w_i[n+1] = w_i[n] - \frac{\mu}{2}\frac{\partial J}{\partial w} \tag{16}$$

Donde μ es un factor de convergencia y:

$$\frac{\partial J}{\partial w} = \left[\frac{\partial J}{\partial w_1} \quad \frac{\partial J}{\partial w_2} \quad \cdots \quad \frac{\partial J}{\partial w_{M-1}}\right]^T \tag{17}$$

Derivando la ecuación (12):

$$\frac{\partial J}{\partial w} = 2\,[A\,w[n] - B] \tag{18}$$

$$\frac{\partial J}{\partial w} = 2\,E[x[n]\,x[n]^T\,w[n] - x[n]\,d[n]] \tag{19}$$

La señal de error se puede escribir como:

$$e[n] = d[n] - x[n]^T\,w[n] \tag{20}$$

Por lo tanto, la ecuación (19) puede escribirse:

$$\frac{\partial J}{\partial w} = -2\,E[x[n]\,e[n]] \tag{21}$$

Para encontrar el mínimo de la función de costo igualamos su derivada $\frac{\partial J}{\partial w}$ a cero, lo que implica:

$$E\big[x[n]\,e_{min}[n]\big] = 0 \tag{22}$$

Donde e_{min} denota el mínimo error. Aplicando la misma condición, pero en la ecuación (19):

$${w_{opt}}^T\,E[x[n]^T\,x[n]\,] = E[x[n]\,d[n]] \tag{23}$$

Donde w_{opt} representa el vector óptimo de coeficientes del filtro, que minimizan la función coste. La ecuación (23) puede reescribirse como:

$$R(x)\,w_{opt} = p(x,d) \tag{24}$$

Donde R(x) es la matriz de autocorrelación de la señal de entrada x, y p(x,d) es el vector de correlación cruzada entre la señal de entrada x y la señal deseada d. La ecuación (24) es conocida como la ecuación de Wiener-Hopf.

Para un valor subóptimo de los coeficientes del filtro se tiene entonces:

$$\frac{\partial J}{\partial w} = 2\,[R(x)\,w[n] - p(x,d)] \tag{25}$$

Sustituyendo en la ecuación (16) obtenemos la forma general de actualización de coeficientes del filtro para el algoritmo de gradiente descendente:

$$w_i[n+1] = w_i[n] - \mu\,[R(x)\,w[n] - p(x,d)] \tag{26}$$

4.1.1.3.2 Algoritmo LMS

El cómputo de la matriz de autocorrelación R(x) y de la correlación cruzada p(x,d) involucra el conocimiento de información del proceso que en la mayoría de los casos no se encuentra disponible directamente, por lo que existen diversos métodos para poder estimar sus valores y así realizar correctamente la evolución de los coeficientes.

Uno de ellos es el algoritmo LMS, el cual consiste en una idea muy sencilla que no es más que aproximar el cálculo del vector gradiente mediante un estimador que tome sólo las muestras actuales. Haciendo uso de la ecuación (21):

$$\frac{\partial J}{\partial w} = -2\,E\big[x[n]\,e[n]\big] \approx -2\,x[n]\,e[n] \tag{27}$$

Podemos pensar que esta aproximación es muy pobre para obtener un algoritmo con buenos resultados, pero se puede demostrar matemáticamente que el comportamiento estadístico del algoritmo es exactamente el mismo que el del gradiente descendente. La principal diferencia es que la evolución del mismo se realizará de forma más ruidosa, es decir, con más varianza que la que tendríamos en el método del gradiente, pero que finalmente los dos algoritmos nos llevarían a la misma solución.

Habiendo hecha la estimación, la ecuación de adaptación queda de la forma:

$$w_i[n+1] = w_i[n] + \mu\,x[n]\,e[n] \tag{28}$$

En este caso el valor de μ cumple el rol de una constante de adaptación. Con esta aproximación se puede demostrar que la varianza de la solución óptima será:

$$\sigma^2{}_{opt} = \mu\,J_{min} \tag{29}$$

No entraremos en los detalles del cálculo de la misma, pero es importante que nos quedemos con la idea, obvia por otra parte, que cuanto mayor sea el valor de la constante de adaptación, más varianza presentará la solución del sistema, es decir, más le costará

al sistema estabilizarse en la solución correcta sin sufrir pequeñas variaciones alrededor de la misma.

El algoritmo LMS, representado por la ecuación (28), implica entonces actualizar los M coeficientes w_i del filtro haciendo uso de los valores instantáneos de la señal de referencia y la señal de error, x[n] y e[n], y utiliza como único parámetro el factor μ. Posee como fortalezas que resulta sencillo de implementar, es numéricamente robusto y ha sido usado ampliamente en una variedad de aplicaciones prácticas, como por ejemplo en cancelación adaptativa de ruido eléctrico y en modelado adaptativo. En la Figura 20 se muestra la operación del algoritmo LMS, en el cual las señales de error y de referencia son multiplicadas y su producto es usado para adaptar los coeficientes del filtro.

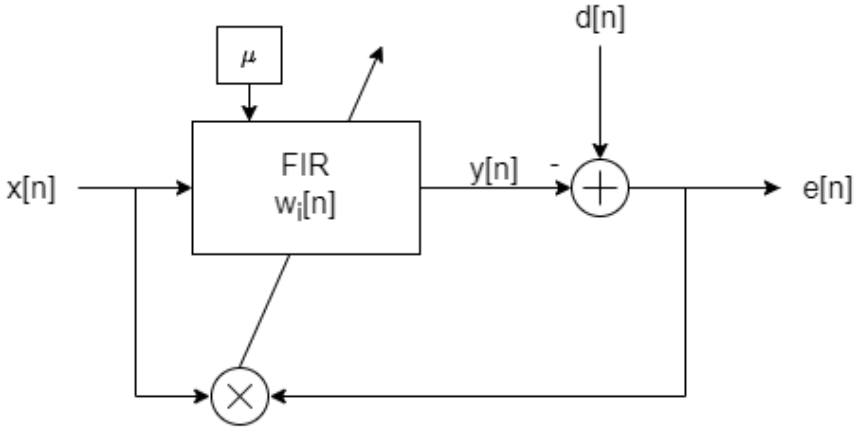


Figura 20 - Algoritmo LMS

El parámetro μ es el paso de adaptación, y gobierna la velocidad de convergencia del algoritmo y el desajuste final del error cuadrático con respecto a su valor mínimo. De forma que, cuando se escoge un factor grande se llega rápidamente a las inmediaciones del error cuadrático mínimo, oscilando alrededor del mismo una determinada magnitud. Sin embargo, si se elige un paso de adaptación pequeño la velocidad de convergencia hacia el valor mínimo del error cuadrático es lenta, aunque el desajuste final es pequeño. Los límites en los que se mueve el paso de adaptación son:

$$0 < \mu < \frac{1}{M\ \sigma^2} \tag{30}$$

Donde M es el número de coeficientes del filtro FIR y σ^2 es la potencia de la señal de entrada x(n). Si el valor del paso de adaptación está fuera de estos límites el algoritmo diverge y no es capaz de encontrar los valores de los coeficientes que hacen mínimo el error cuadrático instantáneo.

4.1.1.3.3 Algoritmo NLMS

Tal como fue presentado, el algoritmo LMS definido en la ecuación (28) es fácil de implementar y también proporciona buenos resultados en el ajuste de coeficientes. Sin embargo, la tasa de convergencia de dichos coeficientes puede ser lenta para algunas

aplicaciones. Para superar este problema, se puede utilizar el algoritmo LMS normalizado o NMLS.

Este método difiere del LMS en que tiene un coeficiente de normalización adicional en el tamaño del paso de adaptación del algoritmo, el cual tiene en cuenta un estimador de la potencia de la señal de entrada. Como resultado, la actualización de coeficientes se convierte en:

$$w_i[n+1] = w_i[n] + \frac{\mu}{\|\bar{x}[n]\|^2}\, x[n]\, e[n] \tag{31}$$

En caso de que en un momento particular de la evolución del sistema la señal x[n] sea cero, $\|\bar{x}[n]\|^2$ también lo será y el término $\frac{\mu}{\|\bar{x}[n]\|^2}$ tendería a infinito, lo que haría desestabilizar al sistema. Por ello se aplica la siguiente corrección a la actualización de coeficientes:

$$w_i[n+1] = w_i[n] + \frac{\mu}{\epsilon + \|x[n]\|^2}\, e[n]\, x[n] \tag{32}$$

ϵ es un número positivo constante lo suficientemente pequeño para no dividir por cero cuando $\|x[n]\|^2$ es cero o cercano a cero.

El algoritmo NLMS posee una mejor convergencia que el LMS y dicha convergencia se da cuando se cumple:

$$0 < \mu < 2 \tag{33}$$

Para calcular el término $\|\bar{x}[n]\|^2$ se necesitan N muestras de la señal x[n] (precedentes al instante n) y se obtiene de la forma:

$$\|\bar{x}[n]\|^2 = \sum_{k=0}^{N-1} x^2[n] \tag{34}$$

4.1.1.3.4 Aplicación del filtro adaptativo

Dado un sistema desconocido, deseamos encontrar un modelo lineal que estime el comportamiento del mismo, a partir de un filtro lineal. En este caso se utiliza un algoritmo adaptativo siguiendo la arquitectura presentada en la Figura 21.

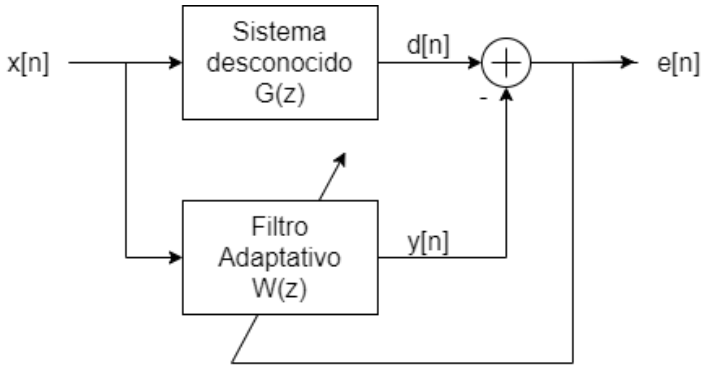


Figura 21 - Filtro adaptativo para identificación de sistemas

4.1.1.4 Diagrama en bloques del sistema

Ya fueron presentados los conceptos teóricos de base que son necesarios para abordar el planteamiento del problema. En este punto determinaremos formalmente el objetivo de la aplicación que se desarrollará y propondremos una arquitectura de bloques para realizar simulaciones y evaluar su desempeño en una etapa previa a su implementación.

El objetivo es desarrollar una arquitectura sencilla y de bajo coste computacional para la obtención de un modelo digital lineal de sistemas electroacústicos que presenten cierta complejidad y cuyo comportamiento sea desconocido. Es decir, nos interesa encontrar un método que obtenga un modelo sencillo y luego verificaremos qué tan bien estima ese modelo el comportamiento real del sistema. Si la reducción de complejidad no implica una pérdida de fidelidad respecto al modelo más preciso, entonces estaremos en condiciones de decir que la propuesta es superadora en el sentido que llega a buenos resultados haciendo uso de muchos menos recursos, lo que en una etapa de implementación puede significar una baja considerable en el coste, el desarrollo y/o la producción.

Por otro lado, el segundo objetivo es utilizar el modelo obtenido como emulador del sistema original, para uso en aplicaciones musicales y técnicas. Con esto lo que se pretende es que, al generar un modelo electrónico, el mismo pueda reemplazar un equipo o sistema original que muchas veces puede resultar impráctico utilizar en ciertas aplicaciones. Un ejemplo de esto puede ser el reemplazo de un gabinete acústico, los cuales en la práctica resultan incómodos en tamaño para su transporte, por un sistema electrónico pequeño que copia su comportamiento acústico y puede ser conectado a un sistema de amplificación sencillo. Esto aporta al proyecto un interés desde el punto de vista técnico que puede ser explotado comercialmente.

Dicho esto, se desprenden entonces dos modos de funcionamiento que nuestro sistema debe cumplir mínimamente:

1. Adaptación: estimación del modelo lineal del sistema desconocido.
2. Filtrado: uso del modelo como filtro emulador.

4.1.1.4.1 Modo de adaptación

Para este modo consideraremos la arquitectura de un filtro adaptativo utilizado en la configuración ya presentada para identificación de sistemas (Figura 21). Este esquema necesita ser definido de forma más específica. Primeramente, en nuestro caso, debemos diferenciar los dominios electroacústico y electrónico. El primero queda configurado por el sistema desconocido G(z), el cual puede ser de naturaleza puramente acústica, puramente eléctrica o una combinación de ambas, por lo que deben considerarse también los dispositivos transductores para hacer las conversiones de señal pertinentes. El resto de las partes del sistema corresponden al sistema electrónico que es el que nos disponemos a desarrollar.

En segundo lugar, debemos definir la señal de excitación x[n], la cual nos permite "entrenar" el filtro para que obtenga las características de G(z) lo mejor posible. Recordando que el sistema desconocido se trata de un sistema de audio cualquiera, nos interesa evaluar su funcionamiento en todo su ancho de banda, por lo que necesitamos excitar al mismo en dicho rango de frecuencias, que se corresponde con 20kHz para este tipo de sistemas:

$$BW_{x[n]} = 20kHz \tag{35}$$

Inútil sería, por ejemplo, que x[n] fuera una señal monotonal, ya que entonces los coeficientes de W(z) se ajustarían exclusivamente para esa frecuencia, ignorando el resto del espectro.

Por otro lado, es importante que x[n] presente la misma amplitud en todo el ancho de banda. De no ser así se estaría reforzando algunas frecuencias más que otras, sesgando el modelo resultante.

De estas características se desprende que la mejor elección para x[n] es que la misma sea ruido blanco, que es una señal que presenta todos los rasgos necesarios. Este ruido debe ser generado por nuestro sistema, asegurando que cumpla lo mejor posible con su definición teórica.

En la simulación, siguiendo las posibilidades que habrá en la futura implementación, debe ser posible variar el sistema a evaluar, como así también los parámetros de W(z), a saber, su factor de adaptación y su número de coeficientes. Esto implica necesariamente un método de configuración, como así también un método de validación de funcionamiento, lo cual se puede sintetizar mediante un bloque genérico de configuración de usuario.

El diagrama en bloques para este modo queda representado en la Figura 22. En azul se engloban todos los bloques que forman parte del sistema electrónico a implementar. En naranja se representa la configuración de parámetros por parte del usuario. En rojo, las etapas de transducción de un dominio a otro (lo cual en la simulación no será necesario incluir, pero sí en la implementación) y en verde, el sistema a evaluar.

Se representan por separado el filtro FIR de su algoritmo NLMS de actualización de coeficientes. Esto es así ya que en el modo de filtrado veremos que solo debemos remover el segundo bloque mientras que el primero queda fijado.

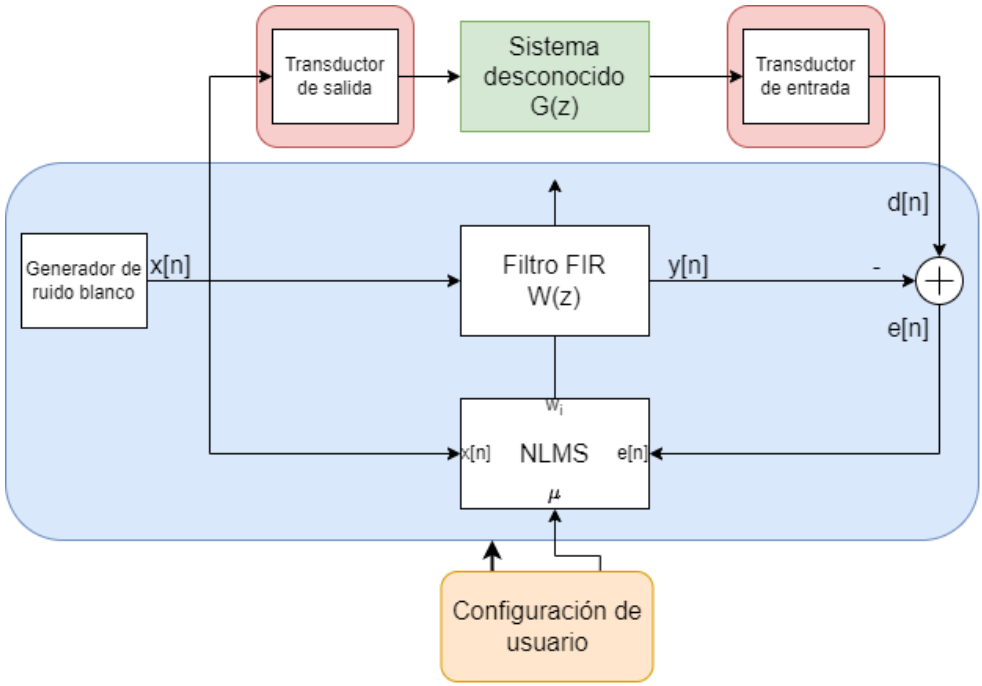


Figura 22 - Diagrama en bloques para el modo de adaptación

4.1.1.4.2 Modo de filtrado

En este modo consideramos que ya se efectuó la etapa de adaptación, por lo que los coeficientes del filtro FIR han sido ajustados de acuerdo con la solución que minimiza el valor del error cuadrático esperado y han sido guardados convenientemente. En este sentido, nuestro diagrama conceptual se convierte en un sistema de filtrado digital simple como el que se muestra en la Figura 23.

Figura 23 - Arquitectura conceptual para el modo de filtrado

En este caso la señal de entrada x[n] es cualquier señal de entrada que el usuario desee procesar mediante el filtro obtenido. Podría ser una señal de voz, música, etc. El sistema debe proveer los dispositivos transductores a la entrada y la salida, que dependen del usuario y donde sea aplicado el filtro obtenido.

Por último, se debe también considerar la posibilidad de activar y desactivar el filtro según un comando de usuario. En este último caso se debe proveer una línea de bypass entre la entrada y la salida. Otro control adicional puede ser la variación del volumen de salida, pero esos detalles no los tendremos en cuenta en esta etapa.

El diagrama en bloques para este modo se muestra en la Figura 24.

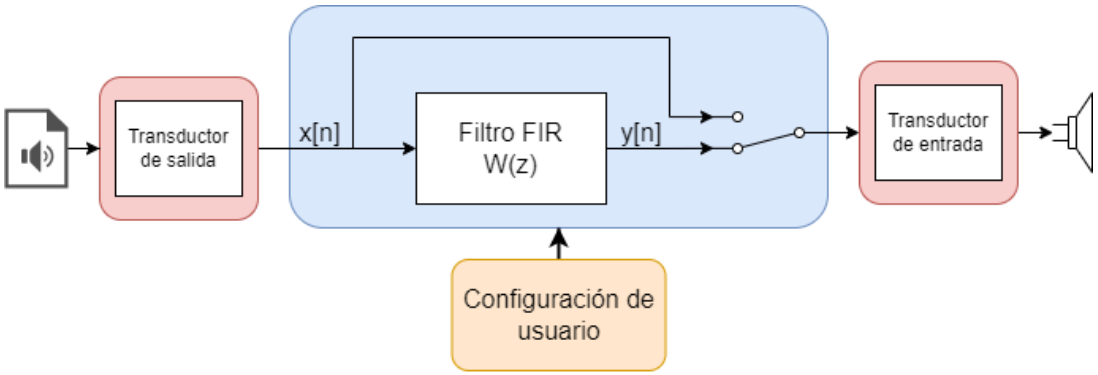


Figura 24 - Diagrama en bloques para el modo de adaptación

4.1.1.5 Modelado del sistema en MATLAB

Previo a la implementación tecnológica del proyecto, resulta necesario evaluar su performance y sus capacidades mediante una simulación. De esta manera se pueden depurar los algoritmos sin depender de otros factores subyacentes a la implementación de los mismos en un entorno más limitado y con condiciones más exigentes como las de un sistema de tiempo real. En este punto es donde se propone la utilización del software MATLAB debido a sus capacidades de prototipado y evaluación de algoritmos numéricos de forma muy precisa y sencilla.

Se hará uso de librerías de filtrado, procesamiento de señales y herramientas de gráficos que simplifican la evaluación de la arquitectura propuesta. Todo el código desarrollado se detalla en el Anexo A.

4.1.1.5.1 Modelo de adaptación

Primero desarrollaremos un modelo que nos permita definir y depurar el algoritmo correspondiente al modo de adaptación, donde pretendemos encontrar un modelo lineal de un sistema electroacústico a través de un filtro FIR adaptativo, utilizando el algoritmo NLMS. Evaluaremos los parámetros y características de convergencia de este algoritmo como así también la comparativa entre el modelo obtenido y el sistema real.

Modelado del sistema desconocido

En primer lugar, debemos definir un modelo que nos permita caracterizar el sistema "desconocido" a evaluar. Para ello tenemos varias alternativas que nos provee el software como sistemas LTI a evaluar:

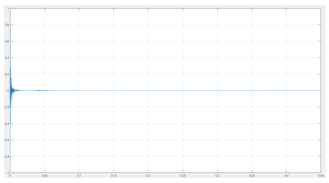
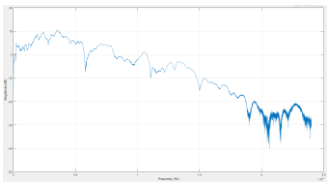
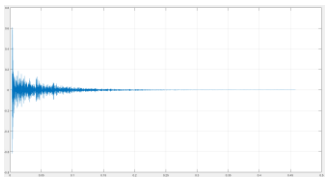
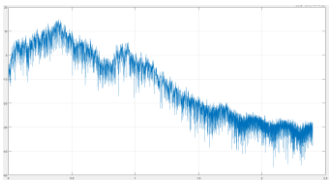
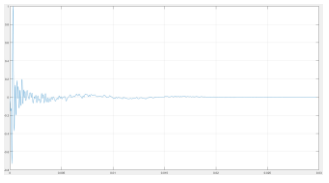
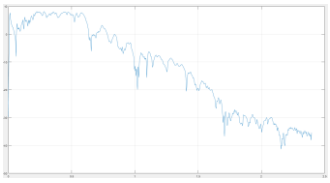
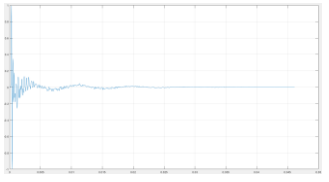
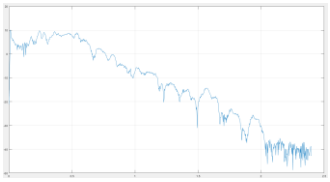
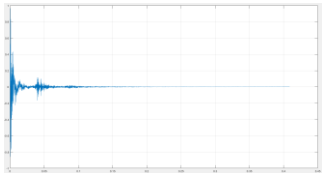
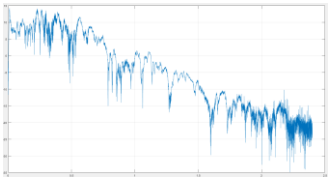
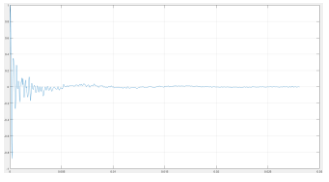
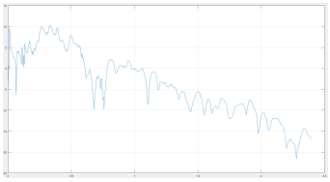
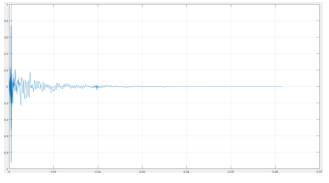
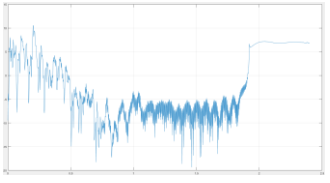
1. Filtro digital FIR (finite impulse response) parametrizable: se propone un filtro de orden configurable, el cual puede ser diseñado por la herramienta, que emule la respuesta en frecuencia deseada. Se implementa, por ejemplo, mediante la línea "numerator=fir1(n,Wn,'low');", perteneciente a la librería "dsp" de MATLAB. En este ejemplo generamos un filtro pasabajos de orden n y frecuencia angular de corte normalizada Wn. El filtro, al ser de tipo FIR, queda caracterizado por un único vector

de coeficientes, que es equivalente al numerador de su función de transferencia digital H(z).

2. Filtro digital IIR (infinite impulse response) parametrizable: esta alternativa es similar a la anterior, con la diferencia que el filtro es de tipo IIR, es decir que el mismo se caracteriza por una función de transferencia digital H(z) que posee polos, o sea que para describirlo necesitamos una secuencia de coeficientes para su numerador y su denominador. Estos filtros pueden diseñarse, por ejemplo, mediante la instrucción "[numerator,denominator] = butter(n,Wn);" donde "butter" significa que el diseño se basa en un filtro de Butterworth, pudiéndose utilizar, claro está, cualquier método de diseño disponible.
3. Filtro analógico: MATLAB también permite el diseño e implementación de filtros descriptos en el dominio analógico, mediante una función de transferencia H(s) donde s representa la variable compleja de Laplace. También se describen a través de vectores de coeficientes para su numerador y denominador y se diseñan de forma similar a los de la opción 2, pero especificando que se utiliza la variable "s". Ejemplo de esto es la línea "[numerator,denominator] = butter(n,Wc,'s');".
4. Secuencia de respuesta al impulso de sistemas electroacústicos reales: Se toma la grabación digital de RI's de altavoces, salas o amplificadores reales, las cuales generalmente se encuentran en formato de audio '.wav' y se utiliza esa secuencia temporal como coeficientes de un filtro FIR. Esto permite obtener una versión digital del sistema en cuestión. MATLAB permite importar estos archivos fácilmente y obtener la respuesta de estos sistemas.

Todas estas alternativas fueron implementadas y se encuentran documentadas en el Anexo A. Sin embargo, presentaremos únicamente los resultados que corresponden con la opción 4, ya que son los que más se asemejan a la operación real del diseño final, y específicamente de los sistemas analizados en la realidad.

Se tomaron diferentes respuestas al impulso de diversos dispositivos y sistemas de audio comerciales, grabadas de forma profesional. En la Tabla 7 se presentan todas las respuestas al impulso evaluadas, junto con algunas de sus características y los tipos de dispositivos reales que representan.

Referencia	Tipo	Duración [s]	Respuesta al impulso	Respuesta en frecuencia
Vox AC30	Amplificador de guitarra	0.4442		
Roland Jazz Chorus	Amplificador de guitarra	0.4582		
Hiwatt DR103	Cabezal de guitarra	0.0298		
Mesa Boogie Rectifier	Cabezal de guitarra	0.0462		
London City	Cabezal de guitarra	0.4088		
Ampeg V-4B	Cabezal de bajo	0.0281		
Gibson Jumbo J200	Guitarra acústica	0.0616		

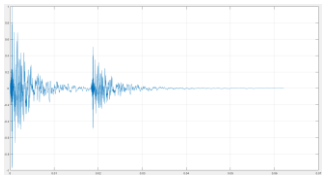
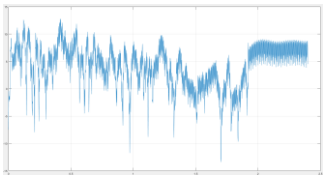
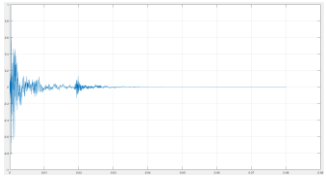
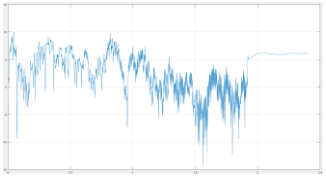
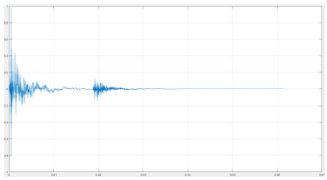
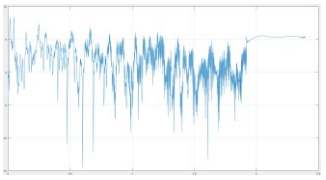
Samick JZ4	Guitarra acústica	0.0621		
Gitane DG-300	Guitarra acústica	0.0801		
Lava Nylon	Guitarra acústica	0.0613		

Tabla 7 - RI's de sistemas electroacústicos

Generación de señales

Una vez determinado el modelo que se utilizará para caracterizar el sistema a identificar, se deben generar las señales correspondientes al diagrama en bloques (representado en la Figura 22) que permitirán que el filtro adaptativo realice su labor.

En primer lugar, debemos fijar la frecuencia de muestreo digital. En el modelo la misma será configurable, pero es evidente que debería ser alguna de las frecuencias de muestreo estándares de los sistemas digitales de audio. En adelante para todas las simulaciones que aquí se muestren tomaremos:

$$F_s = 48000Hz \tag{36}$$

Esta frecuencia rige a su vez la escala de tiempo, ya que entre muestra y muestra tomada en cada una de las señales se tiene un salto temporal de:

$$\Delta t = \frac{1}{F_s} \approx 0.02083\ ms \tag{37}$$

A su vez, debido al teorema del muestreo de Nyquist el valor máximo de frecuencia que podremos representar es:

$$F_{max} = \frac{F_s}{2} = 24000Hz \tag{38}$$

El cual está por encima del rango de frecuencias de interés, por lo que es más que adecuado.

De acuerdo con la arquitectura propuesta del sistema, como se muestra en la Figura 22, se debe generar la señal x, que se corresponde con una señal de ruido blanco que excita tanto al sistema a evaluar como a la entrada del filtro FIR adaptativo. En la implementación real, está previsto que esta señal sea generada sin un límite temporal, sino hasta que el usuario decida interrumpir el proceso de adaptación en base a algún criterio de parada. Sin embargo, en la simulación es conveniente definir un tiempo de señal, en el cual estimamos que el filtro ajuste sus coeficientes de forma óptima. Propondremos entonces:

$$T_{adaptacion} = 100\ s \tag{39}$$

Debemos generar una señal de ruido blanco, con una frecuencia de muestreo Fs y una duración de 100 segundos, lo que equivale a la generación de 4800000 muestras. Esto lo conseguimos en MATLAB mediante la instrucción "x = randn(trainSeconds*Fs,1);", la cual genera un vector de ruido gaussiano con $\mu = 0$ y $\sigma^2 = 1$. En la práctica la distribución de probabilidad del ruido blanco no altera su característica en frecuencia, siempre y cuando se mantenga la misma frecuencia de muestreo y la aleatoriedad de este proceso estocástico. Se puede demostrar que si el ruido blanco tuviera una distribución de probabilidad uniforme no habría grandes variaciones en los resultados obtenidos.

Seguidamente, siguiendo la arquitectura del modelo, debemos generar la señal d, que es la señal de salida del sistema a evaluar. Para obtenerla debemos filtrar la señal x con el filtro que hemos propuesto como modelo del sistema a modelar. Esto se realiza mediante la línea: "d = filter(numerator,denominator,x);", donde numerator y denominator son los coeficientes del numerador y denominador de la función de transferencia del modelo del sistema desconocido (en el caso de utilizar una RI, numerator=RI y denominator=1), y x es la señal de ruido blanco.

De este modo, las entradas al bloque del filtro LMS son las señales x y d.

Filtro LMS

En MATLAB, la librería "dsp" incluye la clase "LMSFilter", la cual permite crear una instancia de un filtro LMS con parámetros ajustables y que sigue un esquema como el de la Figura 25.

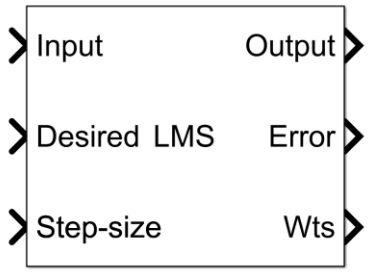


Figura 25 - Bloque LMS en MATLAB

En este bloque se deben especificar las señales de entrada (x) y deseada (d) como así también el parámetro de adaptación (μ). El bloque devuelve como salida las señales de salida (y) y de error (e), como así también los coeficientes del filtro FIR (w) para cada instante de tiempo. Como configuración interna el bloque requiere que fijemos el orden del filtro, como así también el tipo de algoritmo utilizado.

En nuestro caso, debido a sus características de mejor convergencia utilizaremos el algoritmo NLMS. El orden del filtro en la simulación es configurable, pero lo setearemos a un valor razonable de:

$$N = 128 \tag{40}$$

Esto lo haremos así para asegurar que el modelo resultante sea significativamente menor al obtenido mediante una RI. Recordemos, por ejemplo, que para una RI de 0.02 segundos, a una Fs=48000 se necesitan almacenar 960 coeficientes, lo que puede volver a su implementación en tiempo real un verdadero reto debido a su complejidad.

El parámetro μ también lo haremos ajustable y variaremos su valor en la simulación para verificar la dependencia de la tasa de convergencia del algoritmo con su valor.

Una vez configurados los parámetros del filtro adaptativo, efectuamos el proceso de adaptación mediante la instrucción "[y,e,w] = lms(x,d);". Vemos que la misma nos devuelve los vectores y, e y w que corresponden a las señales de salida, de error y a los coeficientes del filtro respectivamente.

Tomemos, por ejemplo, el sistema desconocido correspondiente al cabezal de bajo Ampeg V-4B, cuyas respuestas en el dominio del tiempo y la frecuencia se muestran en la Figura 26 y Figura 27 respectivamente. Aplicamos el filtrado NLMS en esta configuración con un número de coeficientes $N = 128$ y un factor de convergencia $\mu = 0.0005$.

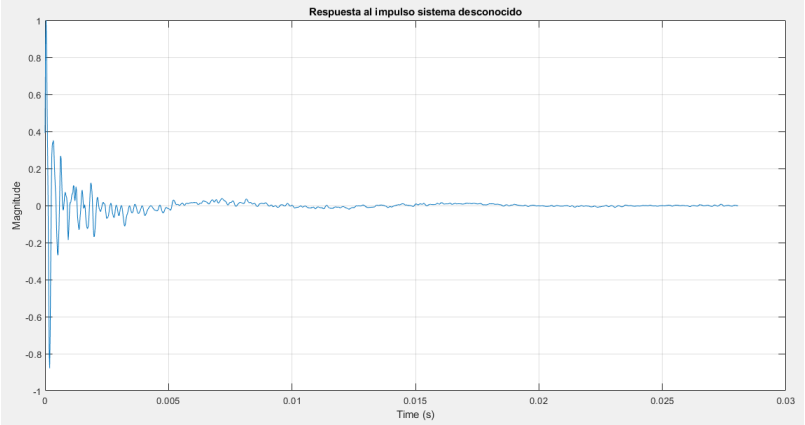


Figura 26 - Respuesta temporal Ampeg V-4B

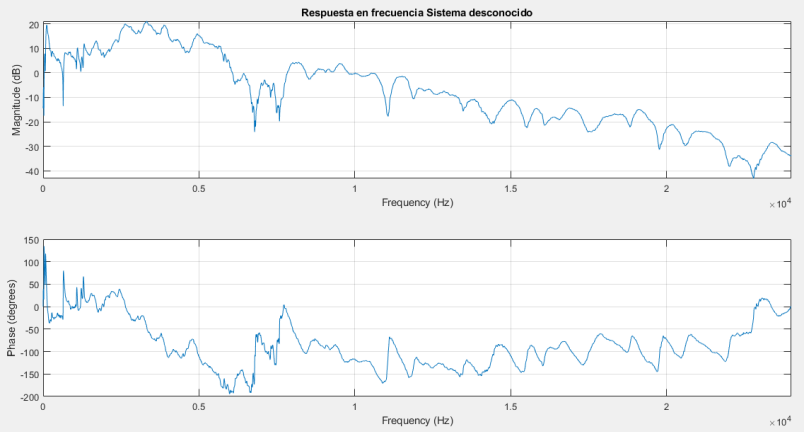


Figura 27 - Respuesta en frecuencia Ampeg V-4B

En la Figura 28 se pueden observar las señales involucradas en el proceso de adaptación. En un comienzo todos los coeficientes del filtro FIR tienen un valor de cero, la señal de salida es mínima y por tanto la señal de error comienza en su valor máximo. A medida que la fase de adaptación transcurre, se van ajustando los coeficientes de tal manera que la señal de salida comienza a seguir a la señal deseada de entrada, y la señal de error disminuye hasta alcanzar un valor mínimo, el cual en la gráfica, para este valor particular del coeficiente μ, se da aproximadamente a los 20 segundos.

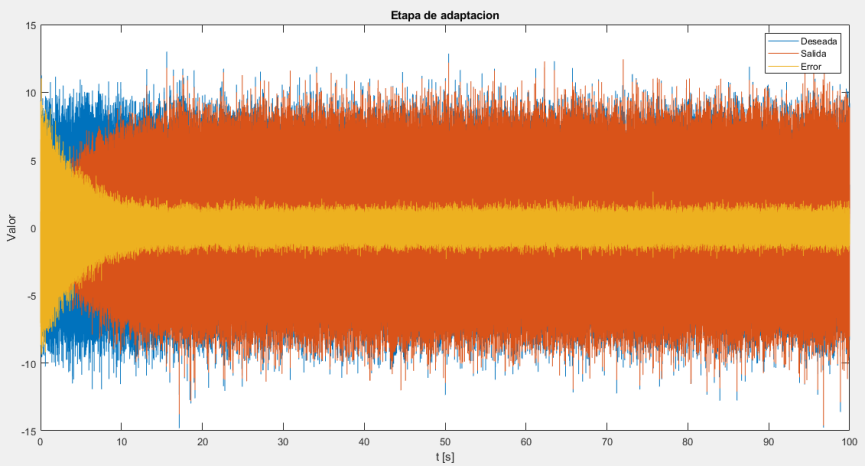


Figura 28 - Evolución de señales en la etapa de adaptación

En la Figura 29 se representan los 128 coeficientes obtenidos por el modelo, o lo que es equivalente, su respuesta al impulso expresada en muestras. La respuesta en frecuencia del modelo que describen estos coeficientes se muestra en la Figura 30, el cual se puede observar que tiende a igualar la respuesta del sistema desconocido.

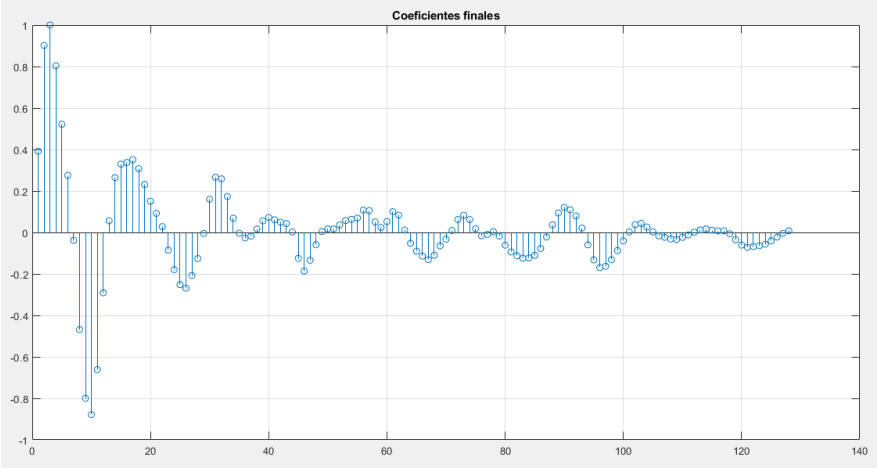


Figura 29 - Coeficientes obtenidos como modelo del sistema Ampeg V-4B

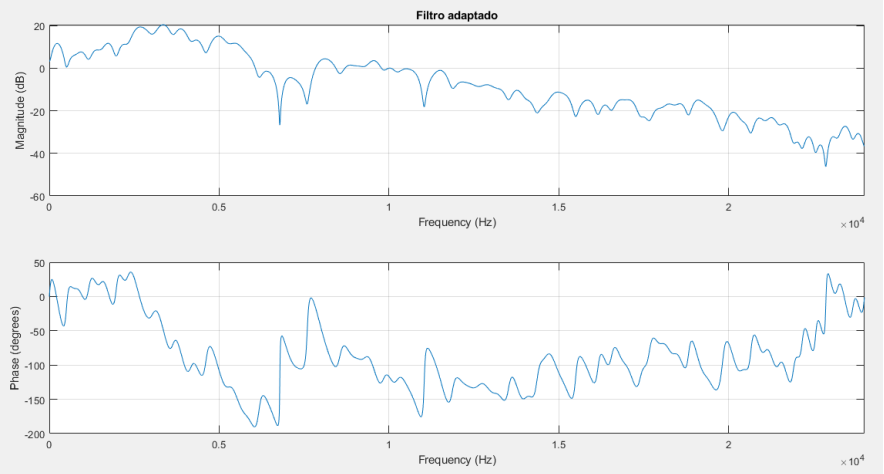


Figura 30 - Respuesta en frecuencia del modelo obtenido para el sistema Ampeg V-4B

MSE

Como fue explicado en la sección 4.1.1.3, el principio de adaptación de estos sistemas se basa en minimizar una función de coste dada por el valor esperado de la señal de error al cuadrado, como indica la ecuación (11). Resulta útil entonces definir una expresión formal de una métrica que aporte información sobre la evolución del filtro adaptativo en función de la señal de error. La esperanza de la variable de error al cuadrado, en un sistema discreto como el que se ha planteado, podemos estimarla como la media de dicha variable. De esta manera, obtenemos que la métrica que representa la evolución del algoritmo está dada por el error cuadrático medio o Mean Square Error (MSE), cuya fórmula se expresa en la ecuación (41).

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(d[i] - y[i])^2 = \frac{1}{n}\sum_{i=0}^{n} e[i]^2 \tag{41}$$

En esta ecuación, n representa la cantidad de muestras totales de la señal de error en el instante de evaluación de la métrica, por lo que calcular el MSE equivale a obtener la media total de la señal de error elevada al cuadrado. Dado que la señal de error evoluciona constantemente, por lo que el valor de n también lo hace, se puede obtener una versión iterativa de la ecuación (41), la cual permita obtener la estimación del MSE para el instante n:

$$MSE[n] = MSE[n-1] + \frac{1}{n}(e[n]^2 - MSE[n-1]) \tag{42}$$

Aplicando la ecuación (42) para la evolución del sistema de adaptación presentado en el ejemplo previo obtenemos la Figura 31. Podemos observar que el valor del MSE decrece a medida que el sistema evoluciona en su proceso de adaptación y converge a un valor mínimo.

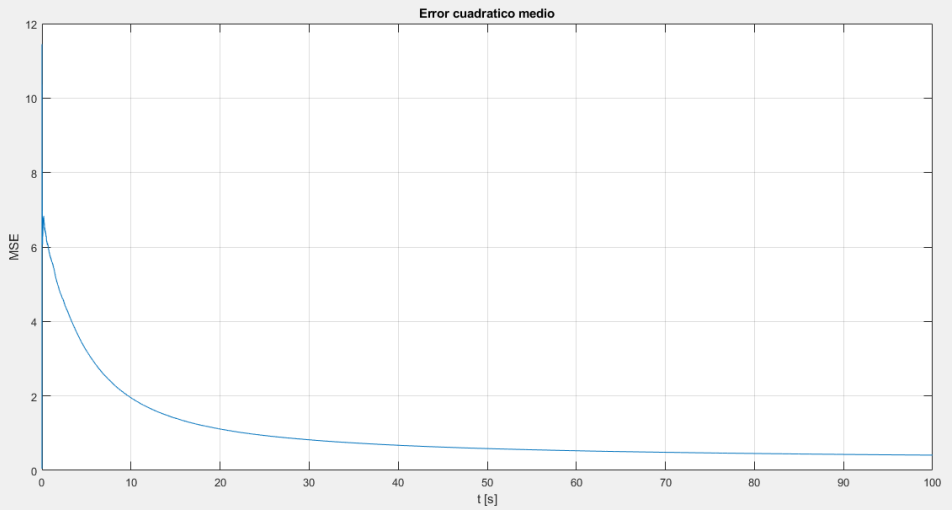


Figura 31 - Evolución del MSE para la etapa de adaptación del sistema Ampeg V-4B

En la práctica, el cálculo del MSE utilizando la ecuación (42) presenta un problema ya que la señal evoluciona en el tiempo infinitamente, por lo que el valor de n crece indefinidamente dando lugar a inestabilidades numéricas y overflow en la estimación de este parámetro. Para resolver este problema se utiliza una estimación por bloque, donde se define un bloque de muestras de tamaño fijo de la señal de error, se calcula el valor de MSE del mismo utilizando la ecuación (41) y este es el valor de MSE utilizado como promedio en el tiempo de bloque. De esta manera el valor de n corresponde con el tamaño de bloque utilizado.

En la Figura 32 se muestra la gráfica de evolución del MSE calculado en bloque para diferentes tamaños de bloque. Vemos que esta estimación sigue capturando la tendencia descendiente de la métrica en función del proceso de adaptación, por lo que resulta una forma válida de evaluar la velocidad y la precisión del algoritmo de adaptación. Se puede observar que el hecho de disminuir el tamaño de bloque no modifica la forma general de la curva sino la varianza de la misma, por lo que la elección de n estará dada en función del tiempo que se disponga de procesamiento para el cálculo, de la precisión deseada en la estimación y de la representación numérica de las variables involucradas, debiendo optarse por un valor de compromiso que equilibre estos aspectos.

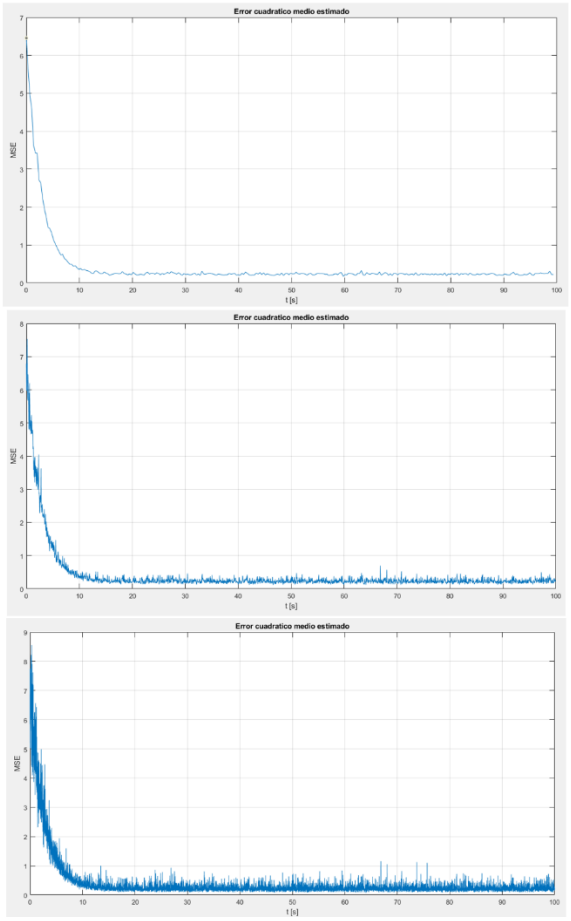


Figura 32 - MSE en bloque a) n=16384 b) n=2048 c) n=512

Efecto del coeficiente de adaptación μ

Tal como fue expuesto en la sección 4.1.1.3, el valor del factor μ permite fijar la tasa de ajuste de los coeficientes del filtro. Un valor pequeño hace que la granularidad en los valores que toman dichos coeficientes sea menor, dando mayor precisión en el resultado, pero a costa de que se necesite mayor tiempo de convergencia para arribar a valores definitivos. Por el contrario, un valor elevado de μ, permite que el sistema converja más rápidamente, a expensas de una representación con mayor incertidumbre de los coeficientes ideales del filtro.

Esto se muestra en la Figura 33, donde se representa la evolución del MSE para diferentes valores de μ. Se puede observar que para valores pequeños el algoritmo se vuelve sumamente lento, incluso presentando dificultad en llegar a la convergencia, como así también exhibiendo un comportamiento oscilatorio en el comienzo de la adaptación. Por el contrario, en el caso de valores elevados ($\mu = 1$) vemos que el valor de convergencia es subóptimo, puesto que, si bien es el primer caso en converger, no llega al mínimo valor de MSE.

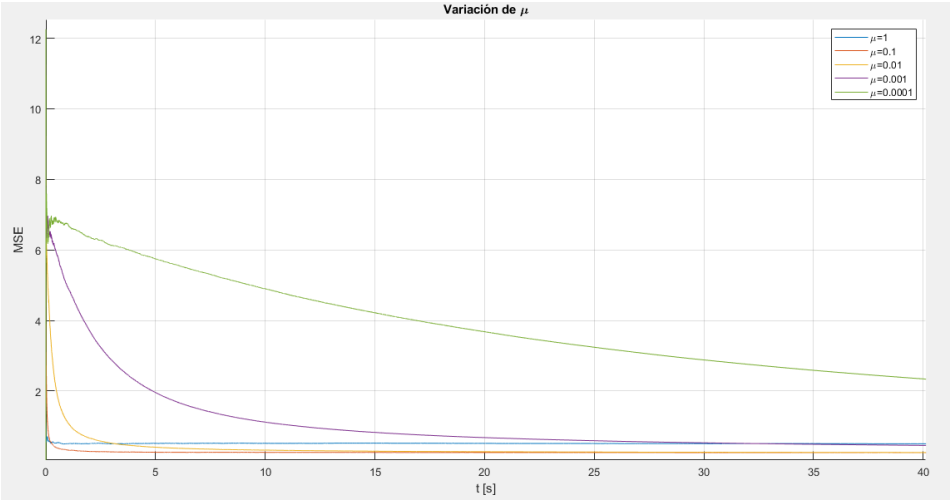


Figura 33 - Variación del coeficiente de adaptación

Por este motivo queda claro que los valores óptimos de este coeficiente se encuentran aproximadamente en el rango:

$$0.1 \leq \mu \leq 0.001 \tag{43}$$

La elección final del valor del parámetro quedará determinada de forma práctica, según el tiempo de convergencia que la aplicación pueda soportar en funcionamiento y los resultados que exhiba el sistema, pero en todos los casos dentro de este entorno se observa que el sistema logra llegar a un valor mínimo del MSE en mayor o menor tiempo.

Resultados del algoritmo

Como fue planteado, el objetivo de este sistema es obtener un modelo simplificado de un sistema electroacústico desconocido. Por esto, los resultados del algoritmo se corresponden con la evaluación de qué tan bien aproxima al sistema desconocido la respuesta obtenida. En este apartado expondremos las fortalezas y debilidades del mismo en los diferentes casos evaluados y determinaremos en cuales de ellos se cumple con el objetivo inicial.

Para ello analizaremos las similitudes y diferencias que existen entre las respuestas en el tiempo y la frecuencia de diferentes sistemas y sus respectivos modelos obtenidos. Por simplicidad y a fin de tener una misma base de comparación, en todos los casos se trabajó con los siguientes parámetros:

$$T_{adaptacion} = 100\ s$$

$$N = 128$$

$$\mu = 0.0005$$

Tomaremos 4 casos de evaluación para presentar en este informe, uno por cada tipo de sistema de los presentados en la Tabla 7, pero los resultados obtenidos pueden

extrapolarse a cualquier otro sistema sin perder generalidad en la valoración de la performance del algoritmo.

Amplificador de guitarra

El caso de estudio elegido para evaluar el algoritmo en amplificadores de guitarra es el Vox AC30.

En la Figura 34 se muestra la estimación del MSE en la etapa de adaptación. Vemos que el mismo logra converger a un valor bajo en menos de 20 segundos, lo que nos indica que la adaptación es lo suficientemente buena. En la Figura 35 se representan los N coeficientes del filtro modelo obtenidos como resultado de este proceso.

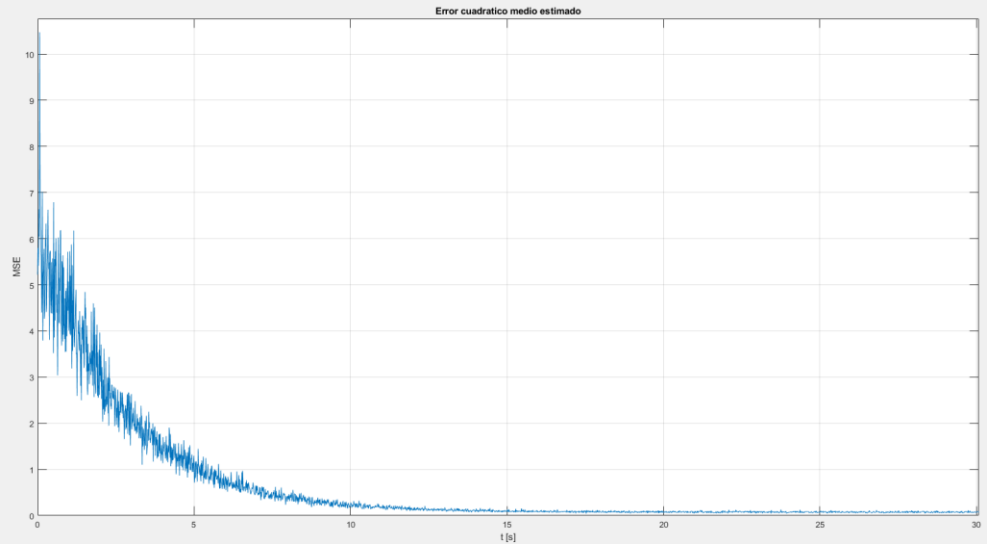


Figura 34 – Estimación del MSE en etapa de adaptación del sistema Vox AC30

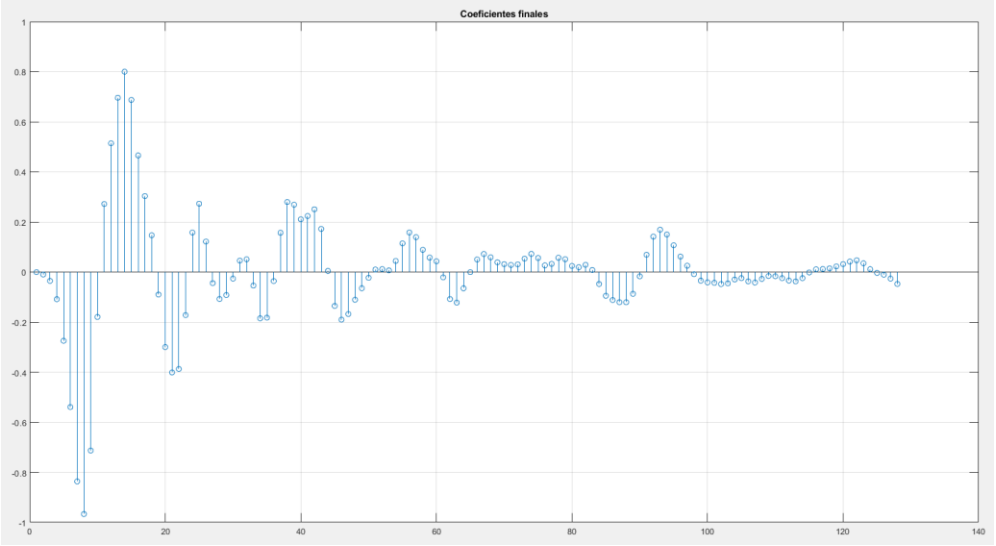


Figura 35 - Coeficientes del modelo para el sistema Vox AC30

En la Figura 36 se muestran simultáneamente la respuesta al impulso del sistema original y la respuesta al impulso del modelo obtenido. Esta última coincide con los N coeficientes del modelo representados en orden distanciados uno de otro según el periodo de muestreo. Se puede observar que la respuesta del modelo es exactamente igual que la del sistema en el rango $0 \leq t \leq N\,\Delta t$, mientras que el modelo al no poseer más coeficientes no puede "continuar" siguiendo la respuesta original más allá de dicho intervalo y se corta abruptamente.

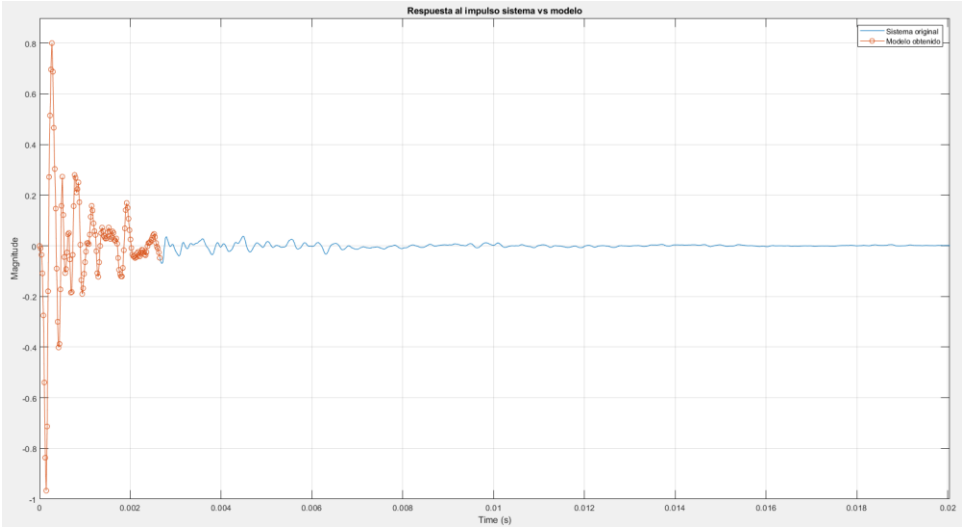


Figura 36 - Comparativa sistema vs modelo en la respuesta temporal (Vox AC30)

En la Figura 37 se observa la respuesta en frecuencia del sistema contra la respuesta en frecuencia del modelo. Se aprecia que esta última sigue muy bien la respuesta original, con alta precisión en el rango de frecuencias medias. En el rango de graves, donde la respuesta original presenta pequeños picos y valles muy cercanos entre sí, el modelo capta la envolvente en frecuencia, pero no llega a igualar perfectamente la forma. Por último, vemos que en la banda de agudos se tiene un piso de ruido que impide al modelo alcanzar la atenuación que se tiene en dicho rango.

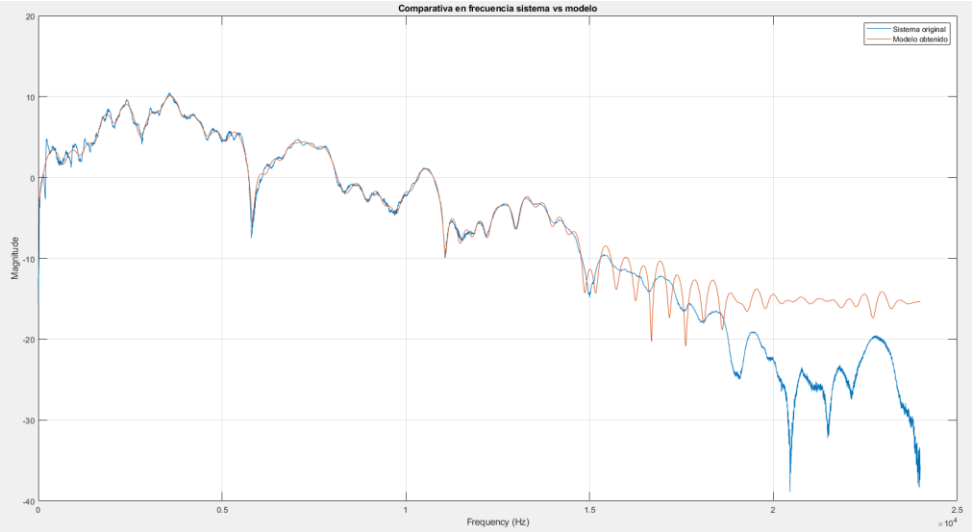


Figura 37 - Comparativa sistema vs modelo en la respuesta en frecuencia (Vox AC30)

Este último problema, aunque imperceptible en la evaluación auditiva, resulta muy evidente en la comparativa gráfica. Si bien no se implementó en la práctica, se desarrolló un método de modificación al algoritmo que permite atenuar este problema, el cual se describe en la siguiente sección.

Algoritmo de ventaneo

El problema observado en la respuesta anterior, donde el modelo no puede seguir correctamente la respuesta en altas frecuencias del sistema deseado, se asemeja a un problema similar que existe en el diseño de filtros FIR.

Cuando se limita el número de coeficientes de un determinado filtro a un valor máximo, en su respuesta en frecuencia real se observa una particularidad denominada fenómeno de Gibbs, la cual tiene su origen en la truncación de la serie de Fourier en la que se fundamenta el diseño de este tipo de filtros. El fenómeno de Gibbs consiste en que se tiene una oscilación o ripple alrededor de un determinado “borde” de la respuesta resultante, como se muestra en la Figura 38.d.

Lo que se realiza en estos casos es multiplicar punto a punto los coeficientes del filtro por una función ventana. Una función ventana es una función que evita las discontinuidades que presentan los coeficientes en sus extremos. El filtro resultante de esta transformación elimina el fenómeno de Gibbs, a costa de bordes más redondeados en el dominio de la frecuencia.

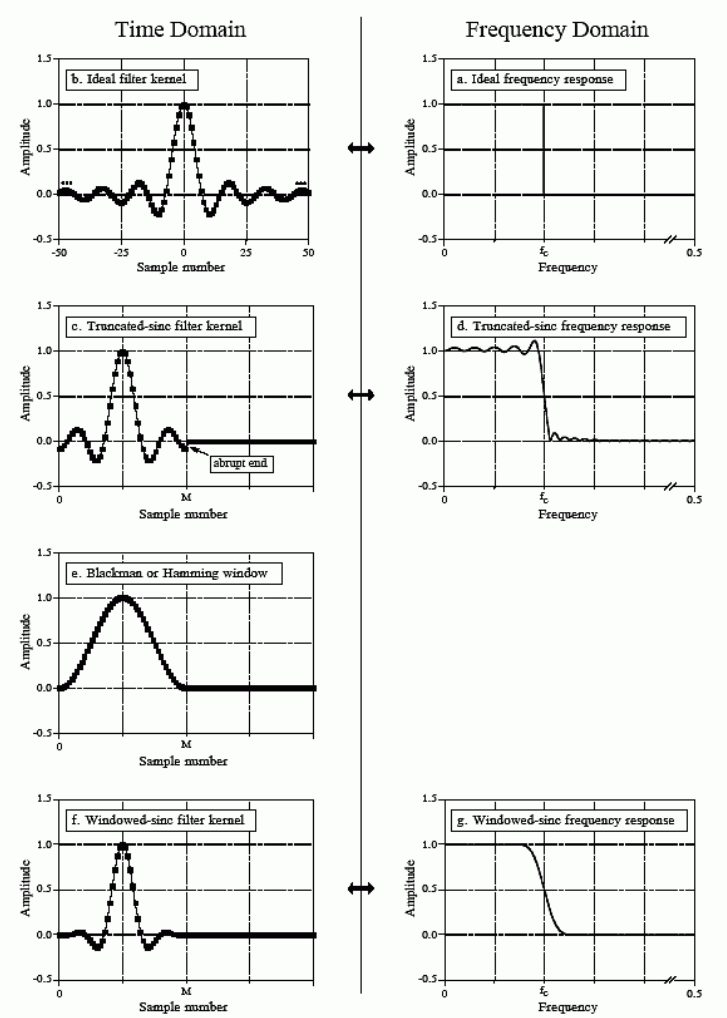


Figura 38 - Metodo del ventaneo en filtros FIR

Utilizando esta analogía del diseño de filtros FIR, la idea detrás de este algoritmo es aplicar una ventana a los coeficientes obtenidos del modelo para suavizar su corte abrupto y mejorar su respuesta en alta frecuencia.

Como se pudo observar, el algoritmo tal cual fue implementado hasta ahora logra igualar la respuesta temporal hasta N muestras, cortando abruptamente a partir de la muestra N+1. Se debe aplicar una ventana que modifique lo menos posible esta similitud pero que elimine la discontinuidad que aparece al final de la respuesta en el tiempo. A su vez, si recordamos de la sección 4.1.1.2, una respuesta impulsiva típica consta de una señal que se puede asemejar a un pulso gaussiano, el cual posee un pico de amplitud máxima que luego decae. En base a esto, la propuesta es aplicar una ventana asimétrica que valga 1 hasta el valor de pico absoluto de los coeficientes obtenidos y que luego decaiga lentamente hasta hacerse cercana a 0 en el coeficiente N+1, tal como se muestra en la Figura 39.

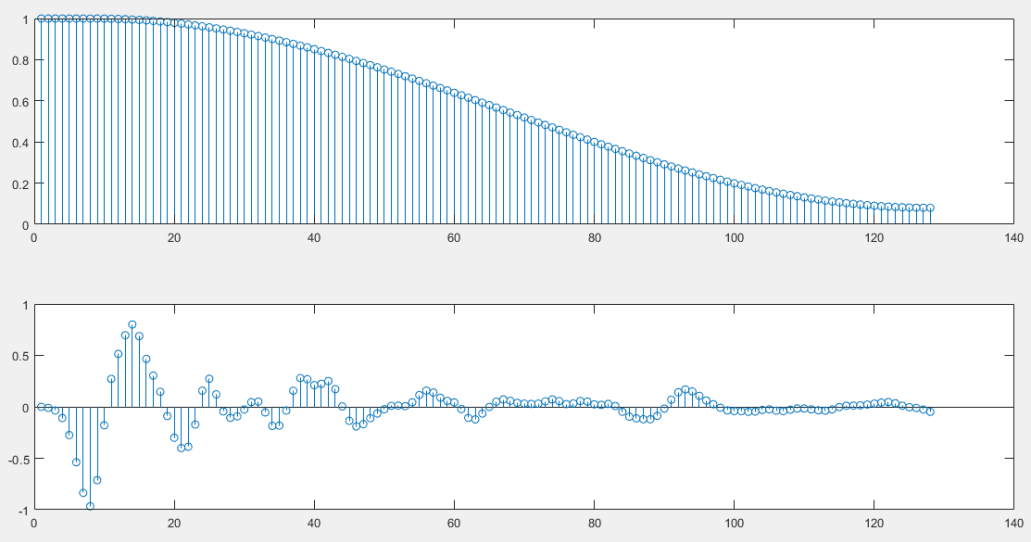


Figura 39 - Ventana asimétrica aplicada

La aplicación de esta ventana da una versión "suavizada" de los coeficientes del modelo y de la respuesta temporal del mismo, como se observa en la Figura 40.

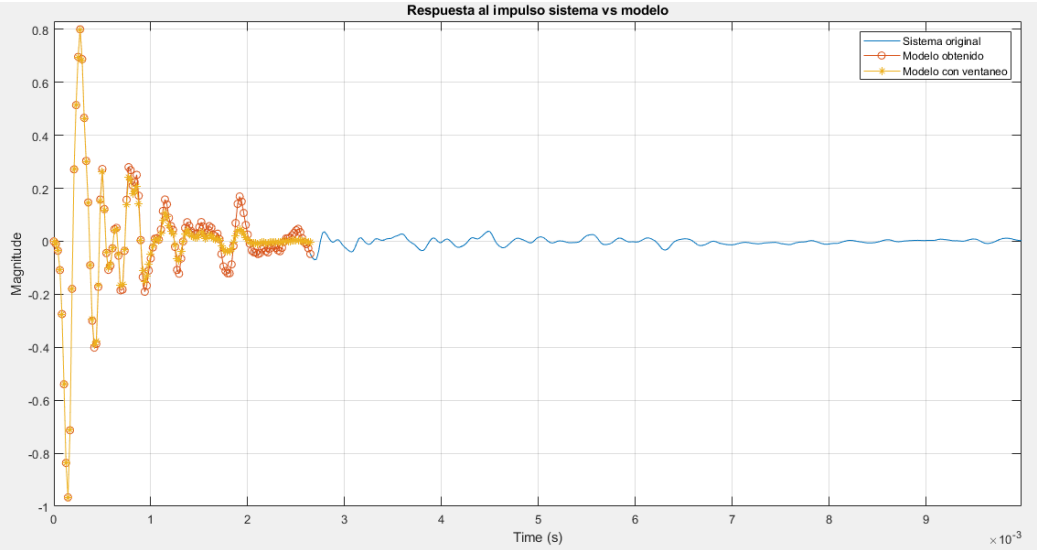


Figura 40 - Respuesta temporal del algoritmo de ventaneo

Por último, la respuesta en frecuencia resultante se muestra en la Figura 41. Se puede apreciar que la aplicación de la ventana permite alcanzar una respuesta en altas frecuencias más semejante al sistema original a costa de una pequeña reducción en la precisión con respecto a la curva original.

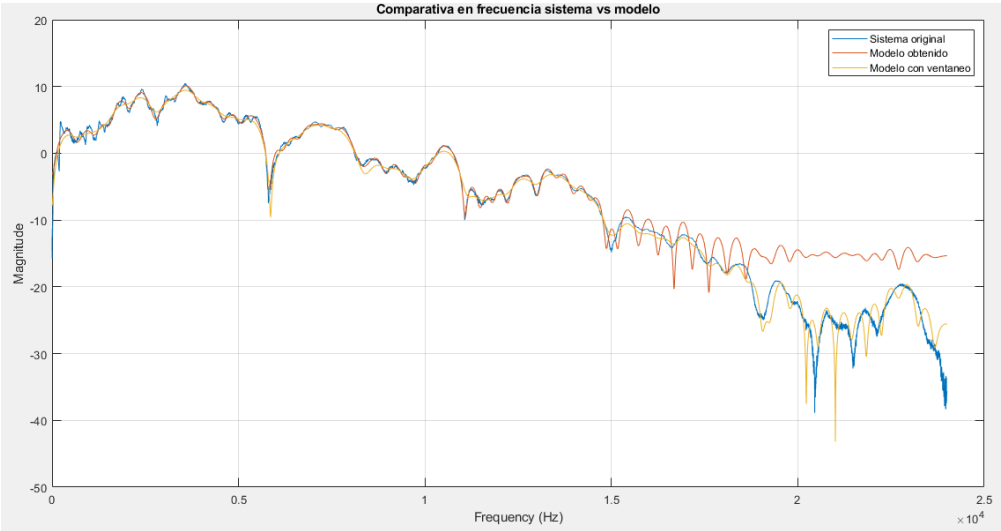


Figura 41 - Respuesta en frecuencia del algoritmo de ventaneo

Esta transformación no fue aplicada en el sistema de tiempo real resultante debido a que agregaba más complejidad al mismo y a que sus resultados a nivel auditivo no presentaban mucha diferencia respecto a los obtenidos sin ventaneo. A pesar de eso, resulta interesante incluirlo como parte del modelo ya que en otro tipo de aplicaciones podría resultar útil y representa una solución simple y que permite llegar a resultados muy satisfactorios sin añadir más coeficientes al modelo.

Guitarra acústica

El caso de estudio elegido para evaluar el algoritmo en guitarras acústicas es la guitarra Gitane DG-300.

En la Figura 42 se muestra la estimación del MSE en la etapa de adaptación. Vemos que el mismo logra converger a un valor bajo en menos de 20 segundos, lo que nos indica que la adaptación es lo suficientemente buena. Sin embargo, el valor final es mayor que en el resto de los sistemas estudiados, lo que nos denota cierto error en el modelo. En la Figura 43 se representan los N coeficientes del filtro modelo obtenidos como resultado de este proceso.

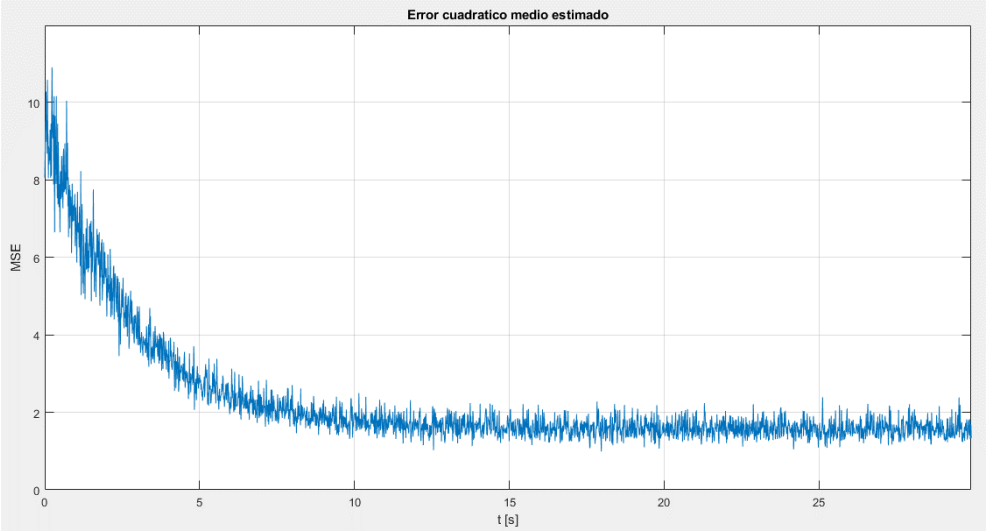


Figura 42 – Estimación del MSE en etapa de adaptación del sistema Gitane DG-300

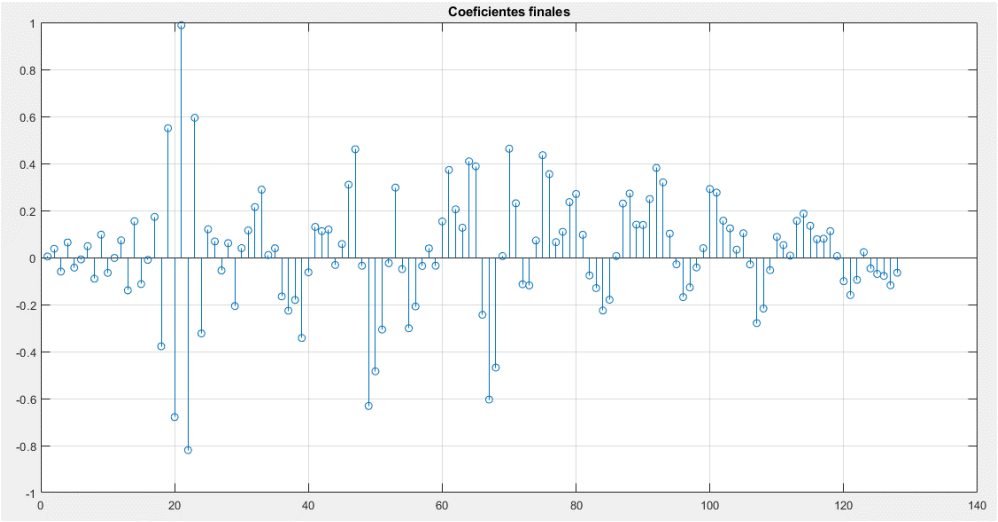


Figura 43 - Coeficientes del modelo para el sistema Gitane DG-300

En la Figura 44 se muestran simultáneamente la respuesta al impulso del sistema original y la respuesta al impulso del modelo obtenido, tanto en su versión original como con su modelo ventaneado. Por tratarse de un sistema puramente acústico como lo es una guitarra de este tipo, se observa cerca de los 20ms un pico correspondiente a una reflexión del sonido. Vemos que, por la naturaleza acortada del modelo, este efecto no es captado por el mismo. Tampoco es capturada la energía de señal que existe luego de los 5ms, la cual no es tan insignificante como en el resto de los sistemas previamente analizados. Esto permite explicar la diferencia en el valor al que converge el MSE.

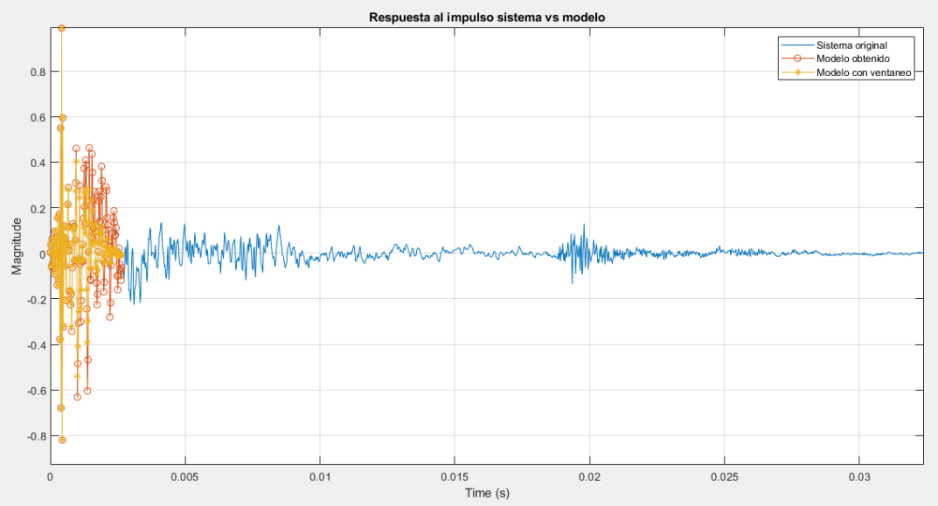


Figura 44 - Comparativa sistema vs modelo en la respuesta temporal (Gitane DG-300)

En la Figura 45 se observa la respuesta en frecuencia del sistema contra la respuesta en frecuencia del modelo. En este caso nuevamente vemos que el modelo sin ventaneo aproxima correctamente la región de altas frecuencias sin generar piso de ruido, incluso se puede apreciar que la estimación del modelo con ventaneo es de inferior calidad. Esto nos da otro motivo por el cual, si bien en ciertos casos este método soluciona el problema de altas frecuencias, no significa que sea aplicable de forma general a todas las situaciones. Sí es posible observar nuevamente la dificultad que tiene el sistema en obtener picos altos en baja frecuencia.

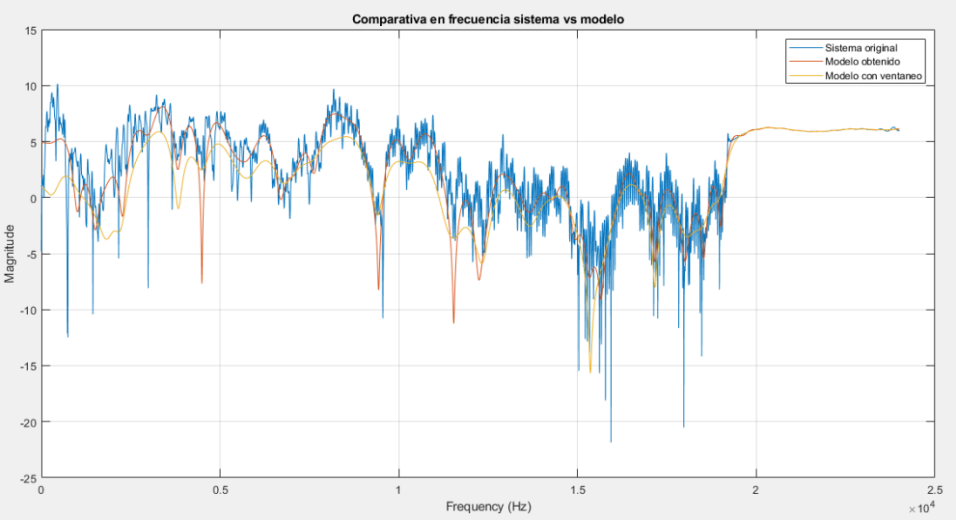


Figura 45 - Comparativa sistema vs modelo en la respuesta en frecuencia (Gitane DG-300)

Conclusiones del algoritmo

En base a los resultados expuestos, podemos concluir que el filtro adaptado obtenido, siendo un modelo muy simple, de apenas 128 coeficientes en el caso analizado, logra una aproximación muy buena de sistemas muy complejos.

En todos los casos el algoritmo permite llegar a un valor de MSE bajo, que asegura una convergencia al mínimo de la función de error. El estimador del MSE por bloque presenta una buena aproximación a la fórmula de cálculo precisa, incluso para tamaños de bloque pequeños (512 muestras).

Observando las simulaciones de la respuesta en frecuencia del sistema original y del modelo obtenido podemos decir que el algoritmo permite aproximar muy satisfactoriamente el comportamiento de la gran mayoría de sistemas. Su mayor fortaleza se encuentra en el rango de las frecuencias medias, donde no solamente es capaz de imitar la forma de la respuesta original sino también ciertos picos y valles angostos que la misma pueda exhibir. En frecuencias graves aproxima muy bien la tendencia del sistema real, aunque es factible de perder los picos máximos si los hubiere. En frecuencias muy altas, si el sistema original presenta alto rango dinámico en frecuencias, el modelo puede exhibir un cierto piso de ruido pero que a los fines prácticos es imperceptible al oído humano.

Con respecto a la respuesta en el dominio del tiempo, siendo un modelo de duración acortada a $N\,\Delta t$ segundos, el cual para los parámetros de simulación se traduce en aproximadamente 3ms, vemos que resulta un modelo de corta duración, por lo que no podrá reproducir correctamente recintos reverberantes o sistemas con una caída muy larga. Esto impone la condición de que para que el modelo sea válido, la mayor parte de su energía debe extinguirse antes de ese tiempo.

En resumen, podemos decir que el sistema caracteriza muy bien la respuesta en frecuencia de los dispositivos analizados, pero no es bueno aproximando su respuesta temporal. De todas maneras, en la mayoría de los casos esto resulta muy útil para muchos sistemas donde lo que verdaderamente interesa es la coloración en frecuencia y se desean eliminar ciertas partes de la respuesta como los ecos y la reverberación.

4.1.1.5.2 Modelo de filtrado

En este caso, el modelo implementado en MATLAB es sumamente sencillo ya que se tienen los coeficientes del modelo obtenido, que constituyen un filtro FIR. Se ensayará el funcionamiento de este filtro con diferentes señales de prueba, que pueden ser señales musicales o pertenecientes a algún instrumento. A su vez también se procesa la misma señal con el modelo del sistema original, a fin de verificar acústicamente que el filtro permite arribar a resultados que lo asemejan al sistema ensayado.

Para ello se hace uso de la función “salidaModelo = filter(w,1,audio);”, donde w son los coeficientes obtenidos, el segundo argumento representa el denominador del filtro digital, el cual es 1 por tratarse de un filtro FIR, y audio son las muestras de la señal de prueba.

Finalmente se utiliza la función “sound(salidaModeloA,Fs);” que permite reproducir la señal obtenida (la cual debe ser previamente normalizada) en el sistema de audio de la PC a una frecuencia de muestreo Fs.

4.1.1.6 Procesamiento de señales en tiempo real

Hasta ahora, no hemos establecido una limitación temporal en el procesamiento de señales. Seguimos deliberadamente este enfoque para explicar los conceptos y algoritmos de procesamiento de señales con detalle. Sin embargo, las aplicaciones de DSP en tiempo real como la que pretendemos realizar siempre tendrán una limitación de tiempo. En otras palabras, habrá un tiempo limitado para procesar datos. Además, los datos que se deben procesar se transmitirán de manera continua y, por lo tanto, no tendrán un final (al menos desde nuestra perspectiva). Por otra parte, sabemos que el sistema digital siempre tendrá recursos limitados. En esta sección se explicarán los conceptos necesarios para pasar del entorno de simulación a la etapa de implementación del sistema en tiempo real teniendo en cuenta estas consideraciones.

4.1.1.6.1 Esquemas de procesamiento

Para una comprensión clara de las operaciones a realizar, primero definimos el procesamiento de señales en tiempo real. Dos métodos fundamentales de implementación de algoritmos que también se pueden utilizar en el procesamiento de señales en tiempo real son el procesamiento basado en muestras y el basado en bloques. En el procesamiento de señales basado en muestras, las señales de entrada se adquieren una por una, mientras que, en el procesamiento basado en bloques, las señales de entrada se almacenan en un búfer (espacio de memoria limitado). Luego, se procesan y se alimentan a la salida en forma de bloque.

Procesamiento basado en muestras

El primer método de implementación que se puede utilizar en el procesamiento de señales digitales en tiempo real se basa en el enfoque basado en muestras. Definamos lo que entendemos por “tiempo real" para este tipo de procesamiento. Describimos un sistema basado en muestras como en tiempo real si la muestra adquirida se procesa y se genera la salida correspondiente antes de que llegue la siguiente muestra. En otras palabras, en un sistema de tiempo real basado en muestras, los datos no se acumulan.

Procesamiento basado en bloques

El segundo método de implementación que se puede utilizar en el procesamiento de señales digitales en tiempo real es el enfoque basado en bloques. Las muestras de entrada son almacenadas en un búfer de tamaño fijo. Luego, el búfer (bloque) se procesa y se envía a la salida. Podemos actualizar la definición de procesamiento basado en bloques de la siguiente manera: describimos un sistema basado en bloques como de tiempo real si el búfer de entrada se procesa y se alimenta a la salida antes de que se llene el siguiente búfer de entrada.

La latencia dentro de los sistemas basados en bloques siempre es mayor que el tiempo necesario para llenar el búfer de entrada. Por lo tanto, para un sistema con un tamaño de búfer L y un período de muestreo $T_{muestreo}$, la latencia será al menos $L \times T_{muestreo}$. El usuario debe tener esto en cuenta en la implementación.

Diferencias entre procesamiento basado en muestras y en bloques

El procesamiento basado en muestras y el basado en bloques se pueden comparar según tres propiedades principales: uso de CPU, uso de memoria y latencia. En el procesamiento basado en muestras, la CPU debe invocarse para cada muestra adquirida por el módulo ADC. Si esta operación se realiza mediante una interrupción, su sobrecarga será en términos de ciclos de reloj utilizados en la operación.

Sin embargo, en el procesamiento basado en bloques, el módulo ADC puede funcionar sin interrumpir la CPU. Por lo tanto, la sobrecarga neta será mínima. Esto también permite el procesamiento paralelo de las operaciones de muestreo y CPU. Además, si se utiliza acceso directo a memoria (DMA), la ventaja del procesamiento paralelo se volverá más dominante. El procesamiento basado en bloques requiere espacio de memoria adicional para su operación. Sin embargo, los microcontroladores recientes pueden manejar tales requisitos de memoria. Si el espacio de memoria es limitado, entonces la única opción que queda es utilizar el procesamiento basado en muestras.

Finalmente, la latencia para el procesamiento basado en bloques es directamente proporcional al tamaño del búfer. Esto puede afectar negativamente la operación del sistema. Si la baja latencia es de suma importancia, entonces se debe utilizar el procesamiento basado en muestras. De lo contrario, se debe establecer un equilibrio entre una latencia aceptable y el tamaño del búfer y se debe utilizar el procesamiento basado en bloques.

4.1.1.6.2 Doble Buffer

En el procesamiento basado en bloques, se requiere más de un buffer para las operaciones del ADC, procesamiento y DAC. La idea aquí es establecer un marco para aplicar todas estas operaciones de manera paralela, lo que permite acelerar el procesamiento y disminuir la latencia del sistema. En este sentido, se propone la utilización del llamado “doble buffer” o ”buffer ping-pong”.

El doble buffer es un método eficiente de procesamiento para DSP. Se basa en la utilización de la técnica de acceso directo a memoria (DMA), la cual permite que los datos que ingresan del ADC del sistema como aquellos que salen al DAC sean escritos en memoria sin involucrar a la CPU en el proceso, permitiendo que la misma pueda realizar procesamiento en paralelo. El funcionamiento del doble buffer se esquematiza en la Figura 46.

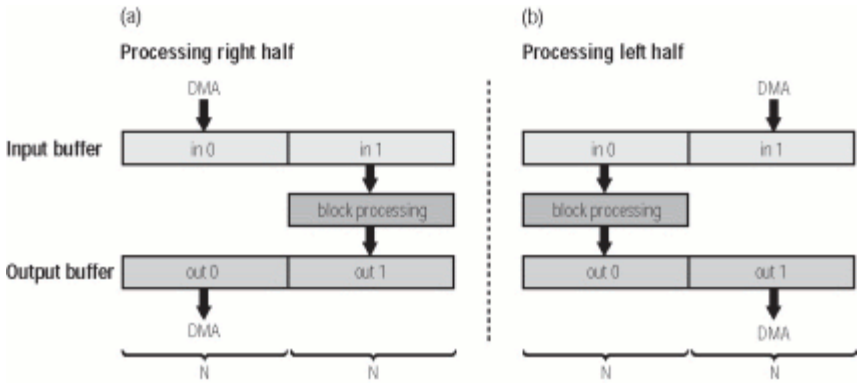


Figura 46 - Doble buffer

Decimos que el buffer es doble porque si nuestro esquema de recepción o trasmisión de datos opera en un bloque de N muestras, el buffer será de tamaño 2N. Consideremos el caso de recepción de datos, aunque para el caso de trasmisión es completamente análogo. El sistema de DMA se encarga de leer el bloque de datos del ADC y provee al procesador de una interrupción (ISR) cuando ha completado una escritura de N muestras, o lo que es equivalente, cuando llena la mitad del buffer.

El ciclo comienza con el DMA llenando la primera mitad del buffer y cuando finaliza este proceso, dispara una ISR que indica a la CPU que dicha mitad está lista para ser procesada. Mientras esta última realiza el procesamiento de ese bloque, el DMA continúa llenando la segunda mitad del buffer. Como este ciclo se realiza en forma circular, cuando el DMA finaliza de llenar la segunda mitad, informa a la CPU mediante la ISR y procede nuevamente a llenar la primera mitad del buffer, permitiendo que se realice el procesamiento de la segunda mitad de los datos.

En un sistema que posee tanto entrada como salida de datos estos procesos pueden estar coordinados, como se indica en la Figura 46. El bloque de entrada es procesado y esta información se escribe en la mitad del buffer de salida que no está siendo utilizada por el DMA. De esta forma, ambos procesos funcionan en paralelo y se hace más eficiente el procesamiento y gestión de los datos.

Debe notarse que quien comanda la latencia general del proceso es la escritura de los N datos en la mitad del buffer, por lo que esto impone la restricción de que el tiempo para el procesamiento de los datos debe ser menor al ocupado en la adquisición y escritura de los mismos:

$$N\,\Delta T_{ADC} + T_{DMA} \geq T_{PROCESAMIENTO} \tag{44}$$

Esto permite definir adecuadamente el tamaño de bloque necesario para el sistema de procesamiento. Entendiendo que ΔT_{ADC} es el periodo de muestreo del conversor, que es fijo y en nuestro caso, dependiente de la frecuencia de muestreo de audio estándar utilizada, y que el DMA y la CPU están configurados para operar a la mayor velocidad permisible, podemos ver que la constante a afectar es el valor de N.

Si hacemos que N sea muy elevado, podremos cumplir holgadamente con la restricción temporal de la ecuación (44) pero introduciremos una latencia muy grande al sistema,

corriendo el riesgo de que deje de percibirse como “de tiempo real”. Para simplificar consideremos que los tiempos de DMA y de CPU son absurdamente pequeños. El tiempo que tarda una muestra en ser leída, procesada y devuelta a la salida del DAC, en definitiva, su latencia, está dada por un proceso de escritura de bloque en el buffer de entrada y un proceso de escritura de bloque en el buffer de salida. Esto es:

$$Latencia > 2\ N\ \Delta T_{ADC} = \frac{2\ N}{F_s} \tag{45}$$

$$N < \frac{L\ F_s}{2} \tag{46}$$

Por ejemplo, tomemos un sistema de audio, donde sabemos que la latencia máxima admisible para que el oído humano no perciba como diferentes dos sonidos separados en el tiempo, es de aproximadamente 50ms. Además, consideremos una frecuencia de muestreo estándar de 48000Hz. Obtenemos así la restricción de tamaño para el tamaño de bloque:

$$N < 1200 \tag{47}$$

La cual estará restringida además por el tiempo de procesamiento, de escritura en memoria y detalles de implementación, como el espacio de memoria disponible.

4.1.1.6.3 Representación en punto fijo

Los números racionales pueden ser representados en un sistema digital ya sea como punto fijo o punto flotante. Hasta ahora, hemos utilizado números de punto flotante en todos los algoritmos desarrollados. Estos algoritmos también se pueden implementar utilizando números de punto fijo. Esto puede ser ventajoso en términos de uso de memoria y velocidad, e incluso en muchos casos de implementación nos encontramos restringidos a utilizar este tipo de representación. Por ello, expondremos las características principales de los números de punto fijo, algunos detalles de implementación en firmware, y la conversión de y a punto flotante.

En esta representación, el número de bits asignados a la partes entera y decimal están fijos. Un número de punto fijo se representa como Qp.q, donde Q es el signo, y p y q son la parte entera y decimal del número, respectivamente. En la Tabla 8 se muestran algunos formatos estándar de punto fijo utilizadas en implementaciones de firmware. Como se puede observar, se asume que los números de punto fijo están normalizados entre -1 y 1.

Formato	Mínimo	Máximo	Resolución	P bits	Q bits	Bits totales
Q1.7	-1	$1 - 2^{-7}$	2^{-7}	1	7	8
Q1.15	-1	$1 - 2^{-15}$	2^{-15}	1	15	16
Q1.31	-1	$1 - 2^{-31}$	2^{-31}	1	31	32
Q1.63	-1	$1 - 2^{-63}$	2^{-63}	1	63	64

Tabla 8 - Representaciones en punto fijo

El lenguaje C no tiene tipos de datos básicos para almacenar los números de punto fijo resumidos en la Tabla 8. En su lugar, podemos representar un número de punto fijo utilizando los tipos de datos básicos de C con signo char, short, int o long long. El archivo de encabezado stdint.h redefine estos para facilitar su uso. Además, existen también las versiones sin signo de cada uno de estos formatos.

Formato	C data type	stdint.h
Q1.7	signed char	int8_t
UQ1.7	unsigned char	uint8_t
Q1.15	short	int16_t
UQ1.15	unsigned short	uint16_t
Q1.31	int	int32_t
UQ1.31	unsigned int	uint32_t
Q1.63	long long	int64_t
UQ1.63	unsigned long long	uint64_t

Tabla 9 - Variables para punto fijo en C

4.1.2 Desarrollo

Anteriormente fueron analizados todos los conceptos teóricos concernientes al desarrollo del proyecto. También se presentaron las simulaciones llevadas a cabo para evaluar su funcionamiento, las cuales, al estar aisladas de los detalles típicos de la implementación, permitieron depurar los algoritmos y experimentar controladamente las condiciones en las que operará el sistema.

En esta sección se detallará el desarrollo propiamente dicho del mismo, particularmente su plataforma de hardware y firmware sobre la que finalmente funcionarán los algoritmos y métodos presentados, como así también la interacción con el usuario final.

4.1.2.1 Hardware

4.1.2.1.1 Diagrama en bloques

Para el diseño en bloques del sistema partimos de las dos funcionalidades básicas del mismo, las cuales son el modo de adaptación y el modo de filtrado. En la Figura 47 y Figura 48 se muestra la arquitectura necesaria para cada uno de ellos.

Los bloques de color azul representan los elementos del sistema que deben realizar tareas de procesamiento de señales. En el caso de la adaptación, tenemos el generador de ruido blanco, el que genera la señal x[n], y el filtro adaptativo, que recibe dicha señal junto con d[n], proveniente de la salida del sistema desconocido, y genera las señales y[n] y e[n] (las cuales son transducidas al dominio analógico para verificar la evolución del sistema de adaptación). Para el modo de filtrado, el bloque de procesamiento de señal corresponde al filtro FIR, el cual será un sistema estéreo con dos señales de entrada y dos de salida, una para el canal izquierdo y otra para el derecho en ambos casos.

Los bloques rojos corresponden a las etapas de transducción, en este caso del dominio analógico al digital y viceversa. Vemos que el generador de ruido blanco en el modo de adaptación exige a su salida un bloque de transducción extra respecto al modo de filtrado.

El bloque anaranjado engloba todas las funcionalidades que están relacionadas con la interacción con el usuario. Se incluye todo lo referido a la configuración de parámetros como así también la visualización de modos y lo concerniente a la experiencia de usuario al utilizar el sistema.

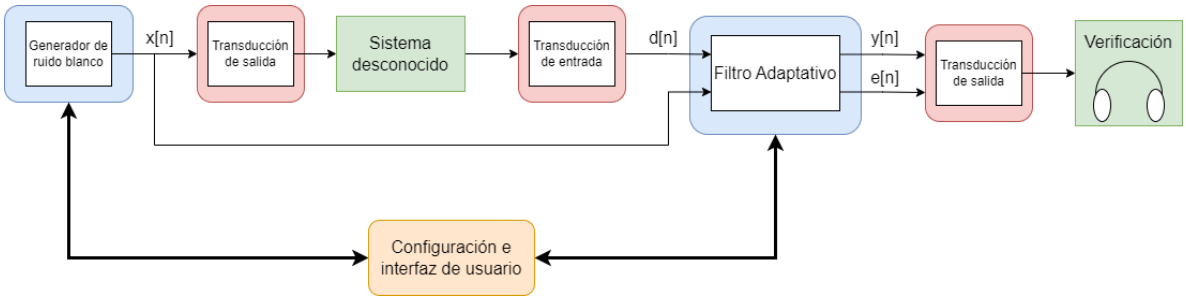


Figura 47 - Arquitectura para el modo de adaptación

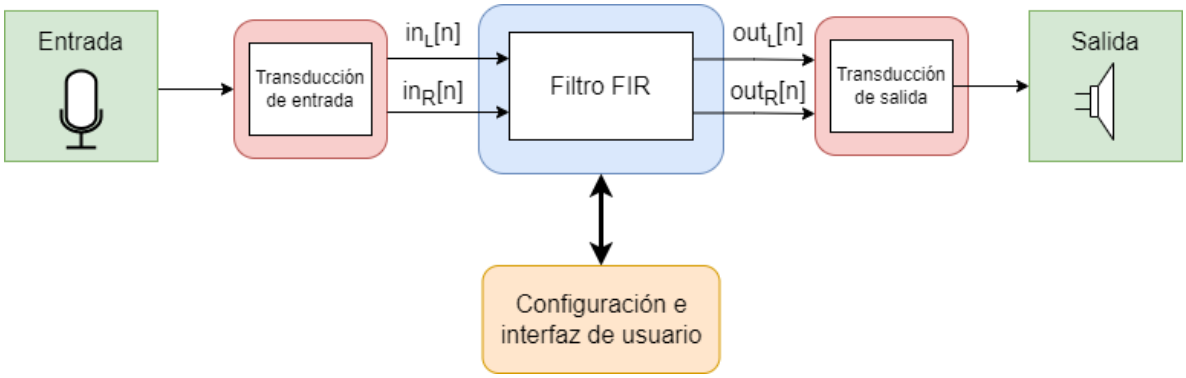


Figura 48 - Arquitectura para el modo de filtrado

Se propone una arquitectura que fraccione estos bloques fundamentales intentando lograr un equilibrio que no sature la capacidad de cada uno de los componentes del sistema. En la Figura 49 se representa el diagrama en bloques propuesto para el sistema.

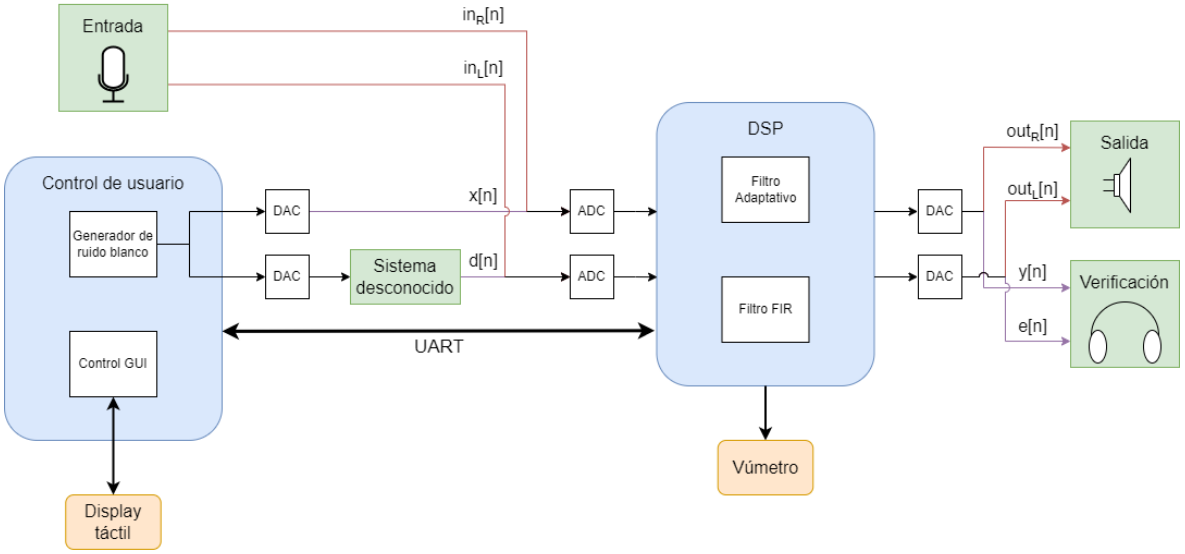


Figura 49 - Diagrama en bloques

Se ha dividido el procesamiento en dos bloques principales: un sistema de procesamiento de señales principal (DSP), encargado de la implementación del filtrado adaptativo y del filtro FIR resultante; y un sistema de control de usuario, el que generará el ruido blanco y llevará la gestión de la interfaz de usuario. Ambos bloques se sincronizarán y compartirán información mediante una interfaz de comunicación UART.

El sistema de configuración e interfaz de usuario fue subdividido entre ambos bloques. La interfaz, compuesta por un display táctil, será manejada por el sistema de control de usuario. A través de la misma se podrá conmutar los diferentes modos de operación del sistema, como así también visualizar y configurar los parámetros de operación. Por otro lado, el indicador de nivel de las señales de entrada, realizado mediante un vúmetro, será implementado en el sistema DSP.

La interconexión de señales externas según el modo de operación quedará en manos del usuario. En color violeta se indica la interconexión de señales para el modo de adaptación y en color rojo, para el modo de filtrado. Esto se resume en la Tabla 10.

Conexión en control de usuario (solo modo adaptación)	Conexión en DSP	Señal en modo adaptación	Señal en modo filtro
DAC L	ADC L	d[n]	in L [n]
DAC R →Sistema desconocido	ADC R	x[n]	in R [n]
-	DAC L	e[n]	out L [n]
-	DAC R	y[n]	out R [n]

Tabla 10 - Señales de audio del sistema

La arquitectura planteada nos permite modularizar cada uno de los bloques expuestos. Se opta por implementar cada bloque en una pieza de hardware por separado, para luego realizar una placa general que interconecte cada uno de dichos componentes. En las siguientes secciones se explicará la electrónica concreta para cada bloque.

4.1.2.1.2 DSP

El bloque de procesamiento digital de señales principal está encargado del manejo de 4 señales de audio, dos como entrada y dos como salida, en un sistema de tiempo real de baja latencia. Para ello deberá contar con una interfaz de comunicación con los ADC's y DAC's correspondientes.

En él se implementará el filtrado adaptativo NLMS utilizando los algoritmos y técnicas ya explicados en la primera sección de este informe. A su vez, deberá ser capaz de almacenar los coeficientes de este filtro en una memoria no volátil para luego utilizarlos como coeficientes fijos del filtro FIR. Este último será capaz de procesar las dos señales de entrada y obtener las dos salidas correspondientes. El bloque, a su vez, deberá permitir la generación de un loopback entre la entrada y la salida.

Por otro lado, mantendrá una comunicación UART con el bloque de control de csuario, que le permitirá enviar y recibir información de control y de funcionamiento. Los modos de operación del sistema serán conmutados a partir de la información recibida.

Por último, utilizando GPIO's de salida, implementará un sistema indicador del nivel de las dos señales presentes en su entrada, a fin de que el usuario posea un feedback respecto al volumen de señal mínimo y máximo admisible para la correcta operación del sistema.

En la Figura 50 se esquematizan de forma simplificada las operaciones de este bloque tal como fueron descriptas. Esto nos permite definir los requerimientos de hardware de la electrónica que implemente este sistema. Debe poseer una interfaz de manejo de las señales provenientes del ADC y de salida al DAC, con operación con DMA para implementar las técnicas de procesamiento digital de señales analizadas en la sección 4.1.1.6. Además, debe contar con la velocidad de procesamiento suficiente para implementar los algoritmos de filtrado y una memoria no volátil para escribir y leer de ella los coeficientes y parámetros almacenados. Finalmente, debe poseer una interfaz UART que posibilite la comunicación mencionada y GPIO's para controlar los LEDs del vúmetro.

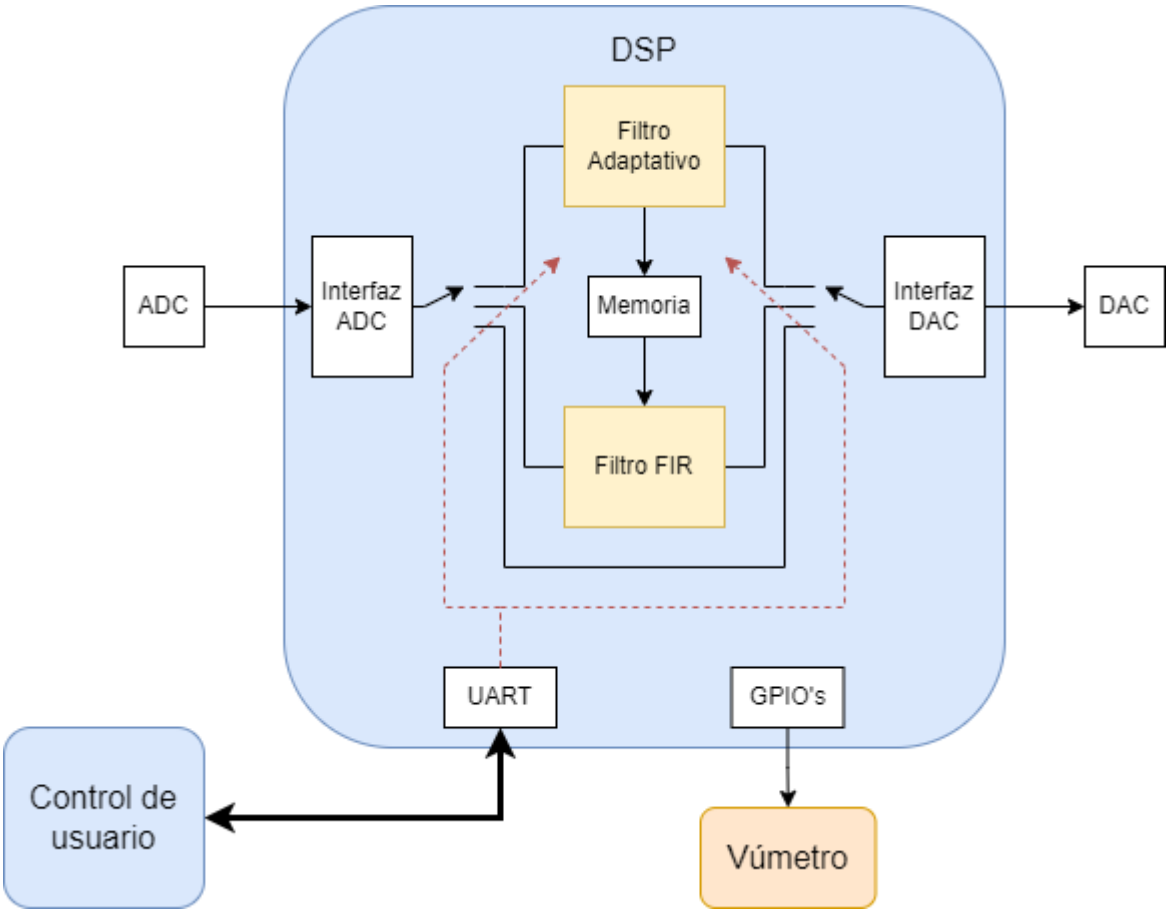


Figura 50 - Diagrama del bloque DSP

Para implementar este bloque se utilizó la placa de desarrollo STM32F4DISCOVERY. Esta placa ofrece las siguientes características:

STM32F407VG	
Categoría	**Características**
Núcleo	CPU Arm® Cortex®-M4 de 32 bits con FPU, acelerador de tiempo real adaptable (ART Accelerator) que permite la ejecución sin esperas desde la memoria Flash, frecuencia de hasta 168 MHz, unidad de protección de memoria, 210 DMIPS / 1.25 DMIPS / MHz (Dhrystone 2.1) e instrucciones DSP.
Memorias	Hasta 1 Mbyte de memoria Flash Hasta 192+4 Kbytes de SRAM incluyendo 64-Kbyte de RAM de datos CCM (core coupled memory) 512 bytes de memoria OTP Controlador de memoria estática flexible que admite memorias Compact Flash, SRAM, PSRAM, NOR y NAND
Interfaz display	Interfaz paralela LCD, modos 8080/6800
Gestión de clock, reset y manejo de alimentación	Suministro de aplicaciones e I / O de 1,8 V a 3,6 V POR, PDR, PVD y BOR Oscilador de cristal de 4 a 26 MHz RC interno de 16 MHz calibrado en fábrica (precisión del 1%) Oscilador de 32 kHz para RTC con calibración RC interno de 32 kHz con calibración
Operación de bajo consumo	Modos Sleep, Stop y Standby Suministro V_{BAT} para RTC, 20 × 32 registros de respaldo + 4 KB de RAM de respaldo opcional
ADC's y DAC's	3 × convertidores A / D de 12 bits y 2.4 MSPS: hasta 24 canales y 7.2 MSPS en modo triple entrelazado 2 × convertidores D / A de 12 bits
DMA	DMA de propósito general: controlador DMA de 16 streams con FIFO y soporte de ráfaga
Timers	Hasta 17 temporizadores: hasta doce temporizadores de 16 bits y dos de 32 bits hasta 168 MHz, cada uno con hasta 4 entradas IC / OC / PWM o contador de pulsos y codificador cuadrático (incremental)
Debug	Interfaces de depuración de serie por cable (SWD) y JTAG Cortex-M4 Embedded Trace Macrocell
Puertos I/O	Hasta 140 puertos E / S con capacidad de interrupción

	Hasta 136 E / S rápidas de hasta 84 MHz Hasta 138 E / S tolerantes a 5 V
Comunicación	Hasta 15 interfaces de comunicación Hasta 3 × interfaces I2C (SMBus / PMBus) Hasta 4 USART / 2 UART (10.5 Mbit / s, interfaz ISO 7816, LIN, IrDA, control de módem) Hasta 3 SPI (42 Mbits / s), 2 con I2S de full dúplex multiplexado para lograr una precisión de audio mediante PLL de audio interno o reloj externo 2 × interfaces CAN (2.0B Activo) Interfaz SDIO
Conectividad avanzada	Controlador USB 2.0 de device/host/OTG de velocidad completa con PHY incorporado Controlador USB 2.0 de device/host /OTG de alta velocidad/velocidad completa con DMA dedicado, PHY de velocidad completa incorporado y ULPI MAC Ethernet 10/100 con DMA dedicado: soporta hardware IEEE 1588v2, MII/RMII
Interfaz cámara	Interfaz de cámara paralela de 8 a 14 bits de hasta 54 Mbytes/s
RNG	Generador de números aleatorios verdadero
CRC	Unidad de cálculo CRC
ID	ID único de 96 bits
RTC	RTC: precisión de subsegundo, calendario de hardware

Tabla 11 - Especificaciones del STM32F407VG

Vemos que esta unidad cuenta con interfaces I2S full dúplex, ideales para recibir datos de audio codificados aprovechando la característica de DMA. Por este motivo, y sumado a que los ADC's y DAC's integrados no son de suficiente resolución, es que optamos por utilizar conversores externos, lo que simplifica la obtención de las señales de audio en el dominio digital.

Como memoria no volátil utilizaremos la memoria FLASH embebida, cuyo tamaño de 1Mbyte es más que suficiente para nuestro propósito. Además, se puede observar que el procesador cuenta con una interfaz UART de buena velocidad y con la cual también podemos aprovechar la funcionalidad de DMA.

Por último, podemos destacar la unidad de punto flotante e instrucciones optimizadas para DSP, las cuales serán de gran utilidad para la implementación del sistema en tiempo real.

4.1.2.1.3 Control de usuario

El bloque de control de usuario tiene como tarea principal manejar los modos de operación del sistema en su conjunto. Esto lo realizará mediante una interfaz de usuario compuesta por una pantalla táctil, la cual deberá controlarse mediante algún protocolo de comunicación estándar. El control de dicha interfaz implica gestionar el menú de opciones y pantallas a través de las cuales el usuario seleccionará los modos de operación, dar la oportunidad de que el mismo fije los parámetros del sistema y la visualización de métricas e indicadores del estado de funcionamiento.

Por otro lado, cuando se seleccione el modo de adaptación, deberá ser capaz de generar una señal limpia de ruido blanco del ancho de banda y características especificadas en la sección 4.1.1.4.1. Para esto, es necesario que cuente con un procesamiento acorde a esta especificación, y una interfaz de salida a un DAC con dos salidas (una para conectar a la entrada x[n] del bloque DSP y otra como entrada al sistema a modelar).

Por último, debe mantener una comunicación UART con el bloque DSP, a través de la cual enviará comandos con los estados y parámetros del sistema, y recibirá señales de control y de funcionamiento del DSP.

Las funcionalidades del bloque se esquematizan en la Figura 51. Electrónicamente el módulo debe contar con una interfaz UART para la comunicación con el bloque DSP, y también disponer del protocolo de comunicación que posea el display táctil. En la mayoría de los casos estos protocolos son I2C o SPI, por lo que es necesario que cuente con alguno de ellos, preferiblemente ambos. Debe proveer suficiente velocidad de procesamiento para actualizar las pantallas y gráficos de la interfaz de usuario, como también para generar la señal de ruido blanco y realizar otras tareas de procesamiento menor. Por último, debe contar con una interfaz de comunicación para el DAC, el cual, por simplicidad del diseño y disponibilidad de componentes, definimos que sea el mismo que el utilizado en el bloque DSP.

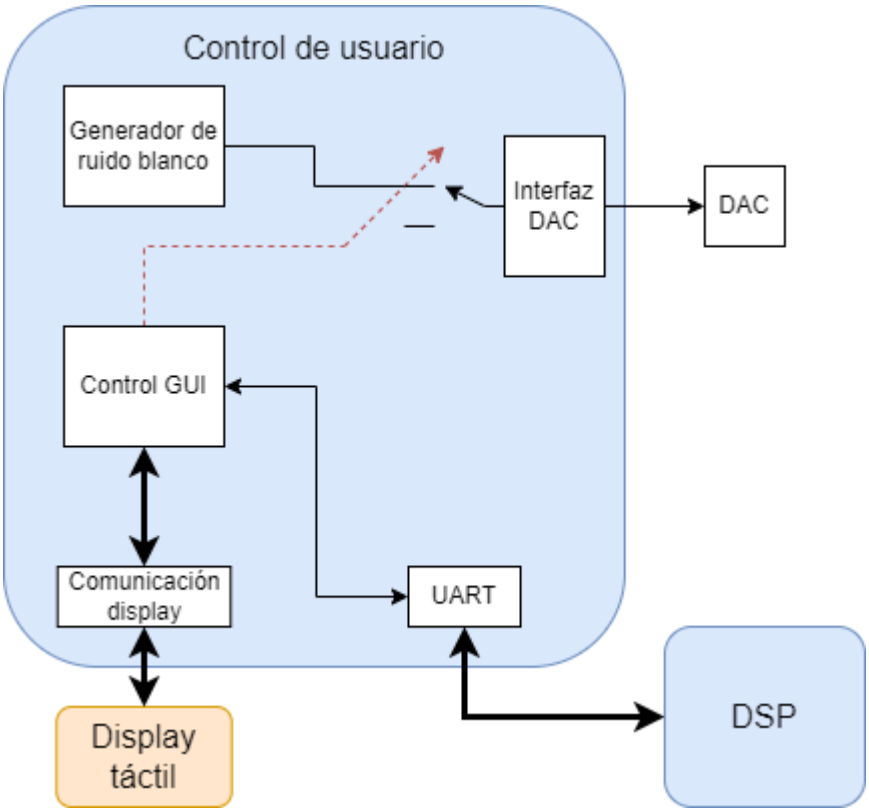


Figura 51 - Diagrama del bloque control de usuario

Para implementar este bloque se utilizó la placa de desarrollo ESP32-DevkitC. Esta es una placa de desarrollo basada en ESP32 de tamaño reducido producida por Espressif. La mayoría de los pines de entrada/salida (I/O) están expuestos en los encabezados de pines en ambos lados para una interfaz fácil.

Su parte central es el ESP-WROOM-32. Este es un potente módulo que integra WiFi y Bluetooth, ideal para desarrollar productos de IoT.

ESP-WROOM-32	
Categoría	**Características**
Voltaje de Alimentación	3.3V DC (2.7~ 3.6V)
Voltaje lógico entradas/salidas (GPIO)	3.3V
Corriente de Operación	~80mA (fuente superior a 500mA)
CPU	Dual core Tensilica Xtensa LX6 (32 bit)
Frecuencia de Reloj	240MHz
Memorias	Memoria SRAM: 520KB Memoria FLASH Externa: 4MB
GPIOS's	Pines Digitales GPIO: 34 (incluyendo todos los periféricos)
Comunicación	UART: 2 SPI: 3 I2C: 2
Sensores e interfaces	Capacitive touch sensors: 10 Interfaz SD
Timers	3 (16-bit) PWM Led: 16 canales independientes (16-bits)
ADC's y DAC's	ADC: 2 (12-bit) DAC: 2 (8-bit)
Comunicación avanzada	Wi-Fi, Protocolo 802.11 b/g/n/e/i (802.11n up to 150 Mbps) Wi-Fi, certificación RF: FCC/CE/IC/TELEC/KCC/SRRC/NCC Wi-Fi, rango de Frecuencia: 2.4 ~ 2.5 GHz Wi-Fi mode Station/SoftAP/SoftAP+Station/P2P

	Wi-Fi Security WPA/WPA2/WPA2-Enterprise/WPS Network protocols IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT Bluetooth, Protocolos: V4.2 BR/EDR and BLE specification Bluetooth, Radios: NZIF receiver with -97 dBm sensitivity, Class-1, class-2 and class-3 transmitter, AFH Bluetooth, Audio: CVSD and SBC Stack de Protocolo TCP/IP integrado en hardware

Tabla 12 - Especificaciones del ESP-WROOM-32

4.1.2.1.4 Conversores ADC/DAC

Debido a que la aplicación de audio requiere de conversiones entre el dominio analógico y digital de alta velocidad y precisión, es que se optó por utilizar conversores externos a los módulos de procesamiento. En particular, el bloque DSP requiere un conversor ADC estéreo y un conversor DAC estéreo, mientras que el bloque de control de usuario, un conversor DAC también estéreo.

Para aplicaciones de audio digital, es muy frecuente utilizar un protocolo específico para comunicar las señales del DAC y el ADC en formato estéreo al módulo de procesamiento. Este protocolo es el I2S, el cual será el utilizado por los conversores elegidos y los microprocesadores.

Pmod I2S2

Para la implementación de la conversión analógica a digital y digital a analógica se utilizará el módulo Pmod I2S2.

El módulo Digilent Pmod I2S2 cuenta con un conversor estéreo de audio A/D Cirrus CS5343 y un conversor estéreo D/A Cirrus CS4344, cada uno conectado a uno de dos jacks de audio. Estos circuitos permiten que una placa del sistema transmita y reciba señales de audio estéreo a través del protocolo I2S. El Pmod I2S2 admite una resolución de 24 bits por canal en tasas de muestra de entrada de hasta 108 kHz y tasas de muestra de salida de hasta 200 kHz.

El módulo cuenta con las siguientes características:

- Convertidores estéreo A/D y D/A de 24 bits para entrada y salida de audio I2S.
- Jacks de audio estéreo estándar de 1/8 pulgadas (3.5 mm).
- Generación automática opcional de reloj serie para entrada de audio.
- Puerto Pmod de 12 pines con dos interfaces I2S.

En la Tabla 13 se indican las especificaciones de alimentación y frecuencia del módulo. En la Tabla 14 se detalla el pinout del mismo, donde se observa que se cuenta con la posibilidad de utilización de la señal de clock maestro (MCLK) y de controlar las señales de control del DAC y del ADC por separado.

Parámetro	Mínimo	Típico	Máximo	Unidad
Voltaje de alimentación	3.0	3.3	5.25	V
Frecuencia de muestreo de entrada (Single-Speed Mode)	4	-	54	kHz
Frecuencia de muestreo de entrada (Double-Speed Mode)	86	-	108	kHz
Frecuencia de muestreo de salida	2	-	200	kHz

Tabla 13 - Especificaciones del Pmod I2S

Pin	Señal	Descripción
1	D/A MCLK	Master clock I2S del DAC
2	D/A WS	Selector de palabra I2S del DAC
3	D/A SCK	Clock I2S del DAC
4	D/A SD	Datos estéreo I2S del DAC
5	GND	Masa de la señal de la alimentación
6	VCC	Voltaje de alimentación (3.3V)
7	A/D MCLK	Master clock I2S del ADC
8	A/D WS	Selector de palabra I2S del ADC
9	A/D SCK	Clock I2S del ADC
10	A/D SD	Datos estéreo I2S del ADC
11	GND	Masa de la señal de la alimentación
12	VCC	Voltaje de alimentación (3.3V)

Tabla 14 - Pinout Pmod I2S2

Los dos circuitos integrados principales del Pmod I2S2 se comunican con la placa base anfitriona a través del protocolo GPIO. Como cada CI utiliza el protocolo I2S, se requieren varias líneas de reloj diferentes, como se describe a continuación.

El CS4344 y CS5343 (de aquí en adelante referidos como DAC y ADC, respectivamente) están conectados a la placa base anfitriona a través de su propia interfaz I2S. Como se ve en la Tabla 14, la interfaz I2S del conversor de salida de línea está conectada a la interfaz de la fila superior del conector de pines, mientras que la interfaz I2S del conversor de entrada de línea está conectada a la fila inferior.

Cualquier alimentación externa aplicada al Pmod I2S debe estar dentro de 3 V y 5,25 V; sin embargo, se recomienda que el Pmod se opere a 3,3 V. Los niveles lógicos digitales deben corresponder al voltaje de la fuente de alimentación.

En nuestro desarrollo utilizaremos dos módulos Pmod I2S2, uno conectado al bloque DSP, el cual proveerá las dos señales estéreo de entrada y salida; y otro asociado al bloque control de usuario, que será encargado de convertir la señal estéreo de salida del mismo.

4.1.2.1.5 Interfaz gráfica

Como elemento de interacción principal con el usuario nos valdremos de una pantalla táctil, controlada por el bloque de control de usuario. En dicha pantalla se mostrará un menú con opciones a través de las cuales el usuario podrá ir operando los modos de funcionamiento del sistema y visualizando parámetros y opciones del mismo. Se necesita que sea un módulo cuya implementación sea sencilla, tanto en hardware como en firmware, lo cual implica que posea un controlador específico que limite las señales de control necesarias para su funcionamiento, como así también de recursos para acelerar su programación, como un set de librerías.

Se optó por utilizar una pantalla LCD con tecnología TFT. El módulo en cuestión es el MSP2807, que cuenta con un display de 2.8 pulgadas con una resolución de 320x240 pixeles y cuenta con un driver SPI ILI9341 y control táctil también por SPI.

En la Tabla 15 se describen los parámetros principales del módulo y en la Tabla 16 se muestra su pinout.

Parámetro	Descripción
Color	RGB 65K colores
SKU	MSP2807
Tamaño de pantalla	2.8 pulgadas
Tipo	TFT
Driver	ILI9341
Resolución	320x240 pixeles
Interfaz del módulo	SPI 4 conexiones
Área activa	43.2x57.6 (mm)
Tamaño de PCB	50.0x86.0 (mm)
Ángulo de visión	>60°
Voltaje de alimentación	3.3V / 5V
Corriente de operación	90mA

Tabla 15 - Parámetros del MSP2807

Pin	Señal	Descripción
1	VCC	Voltaje de alimentación
2	GND	Masa de la señal de alimentación
3	CS	Chip select del LCD
4	RESET	Señal de reset del LCD
5	DC/RS	Register/Data select del LCD
6	SDI (MOSI)	Bus de escritura SPI del LCD
7	SCK	Clock SPI del LCD
8	LED	Señal de backlight del LCD (si no requiere control, conectar a 3.3V)
9	SDO (MISO)	Bus de lectura SPI del LCD
10	T_CLK	Clock SPI del Touch
11	T_CS	Chip select del Touch
12	T_DIN	Bus de escritura SPI del Touch
13	T_DO	Bus de lectura del SPI del Touch
14	T_IRQ	Detección de interrupciones del Touch

Tabla 16 - Pinout MSP2807

Como se puede apreciar, el control del LCD y de la pantalla táctil se lleva a cabo mediante dos interfaces SPI diferenciados, tratando a cada uno de ellos como un dispositivo diferente.

4.1.2.1.6 Vúmetro

En la entrada del bloque DSP, tanto en el modo de adaptación como en los modos de filtrado y loopback, se tienen señales cuyo nivel puede ser desconocido aún para el mismo usuario. Por ejemplo, para la señal d[n] se desconoce su nivel máximo, el cual dependerá de la afectación que sufra de parte del sistema desconocido al ser este excitado por ruido blanco. La operación del sistema puede ver degradada su performance si las

señales x[n] y d[n] poseen niveles de amplitud muy dispares, por lo que el usuario deberá regular el volumen de ambas a fin de nivelarlas.

Esto exige una indicación visual que dé al usuario feedback acerca del nivel de ambas señales de entrada al bloque. La propuesta para ello es hacerlo mediante un vúmetro.

Un vúmetro es un dispositivo indicador en equipos de audio para mostrar el nivel de señal en unidades de volumen. Generalmente esta indicación puede realizarse mediante un módulo con una aguja, o mediante señales lumínicas como LEDs.

A su vez, puede ser de tipo analógico o digital. En el primer caso se debe tomar la señal de tensión antes de los ADC's, rectificar su valor y obtener el valor RMS de dicha señal, lo cual puede ser un verdadero desafío dada la complejidad de las señales de audio analizadas. Luego se debe concatenar un circuito que encienda o apague los LED's según el valor pico de esta señal analógica RMS.

Por otro lado, un vúmetro digital toma el valor digitalizado de la señal, calcula su valor absoluto o de potencia de señal (o RMS para un tamaño de bloque prefijado) y aplica una lógica de encendido y apagado de los LED's utilizando los puertos de salida del microcontrolador. Puesto que agrega mucha menos complejidad circuital, y que las señales ya se encuentran digitalizadas en el bloque DSP, utilizaremos esta alternativa.

En la aplicación que operará, no es necesaria una indicación precisa del nivel de volumen de la señal, sino determinar márgenes de operación seguros para ambas señales de entrada. Definiremos entonces tres niveles de cuantificación para el valor de señal: un nivel bajo, donde el usuario deberá interpretar que el volumen a la entrada no es suficiente; un nivel óptimo, donde el volumen es adecuado; y un nivel de saturación, donde el volumen es excesivamente alto y se encuentra cercano a la saturación.

Para cada señal utilizaremos entonces 3 LED's de 3mm conectados cada uno a resistencias de 100Ω. Este valor se obtuvo teniendo en cuenta que la tensión de salida en los pines del STM32F4DISCOVERY es de 3.3V.

La lógica de encendido está basada en tres umbrales para el nivel de la señal: si no se tiene señal presente a la entrada, no se enciende ningún LED. Si se supera el primer umbral, pero la señal no es lo suficientemente grande, se enciende un solo LED. Si la señal supera el segundo umbral y posee un nivel aceptable, se encienden 2 LEDs. Por último, si el valor supera el tercer umbral, el cual se fija lo más cercano a la saturación posible, se encienden los tres LEDs, indicando que el nivel es excesivo.

4.1.2.1.7 Conexionado

Dado que los bloques DSP, implementado en el módulo ST32F4DISCOVERY, y el Control de usuario, llevado a cabo en el módulo ESP32, son los componentes centrales del sistema, presentaremos el conexionado lógico de todo el hardware del sistema en base a ellos. Esto queda expuesto en la Tabla 17 y la Tabla 18.

DSP					
Componente	**STM32F4DISCOVERY**	**Pmod I2S2 (1)**	**Vúmetro R**	**Vúmetro L**	**ESP32**
Pin	PC6	MCLK	-	-	-
	PB10	SCK	-	-	-
	PB12	WS	-	-	-
	PC2	DATA_IN (ADC)	-	-	-
	PC3	DATA_OUT (DAC)	-	-	-
	PD0	-	R_LED_LOW	-	-
	PD2	-	R_LED_MED	-	-
	PD4	-	R_LED_HIGH	-	-
	PC7	-	-	L_LED_LOW	-
	PC9	-	-	L_LED_MED	-
	PC11	-	-	L_LED_HIGH	-
	PD6 (UART Rx)	-	-	-	23 (UART Tx)
	PD5 (UART Tx)	-	-	-	22 (UART Rx)

Tabla 17 - Conexionado bloque DSP

Control de usuario				
Componente	**ESP32**	**MSP2807**	**Pmod I2S2 (2)**	**STM32F4DISCOVERY**
Pin	32	CS	-	-
	33	RESET	-	-
	25	DC/RS	-	-
	26	SDI (MOSI) / T_DIN	-	-
	27	SCK / T_CLK	-	-
	14	SDO (MISO) / T_DO	-	-
	12	T_CS	-	-
	0	-	MCLK	-
	16	-	SCK	-
	4	-	WS	-
	5	-	DATA_IN (ADC)	-
	17	-	DATA_OUT (DAC)	-
	22 (UART Rx)	-	-	PD5 (UART Tx)
	23 (UART Tx)	-	-	PD6 (UART Rx)

Tabla 18 - Conexionado bloque Control de usuario

4.1.2.2 Firmware

4.1.2.2.1 Sistema de procesamiento de audio en tiempo real (DSP)

Describiremos el firmware desarrollado en el bloque DSP, implementado en el bloque STM32F4DISCOVERY sobre un procesador STM32F407VG.

Entorno de programación

El software que utilizaremos para la programación es el STM Cube Ide, un software gratuito provisto por la misma empresa ST Microelectronics, con el que programaremos el dispositivo utilizando lenguaje C.

El Cube Ide es un software muy completo y a la vez complejo, que agrupa a varios programas. El entorno de desarrollo propiamente dicho es el popular Eclipse que integra al compilador GCC (en realidad GCC es un conjunto de herramientas además del compilador, lo que se conoce como "toolchain"), un generador gráfico de código que nos permite inicializar todos los periféricos y bloques del micro de una manera visual (Cube MX, que también puede usarse por separado), un debugger que nos permite ejecutar el programa por pasos, inspeccionar variables y establecer breakpoints entre otras funciones y una herramienta de selección de microcontroladores y documentación (MCU Finder) que nos permite elegir el MCU apropiado para nuestro proyecto, acceder a toda la documentación (hojas de datos, notas de aplicación), así como ejemplos, comunidades de usuarios, etc.

Primeramente, utilizando Cube MX, se configuran todos los periféricos del microcontrolador, sus parámetros, los pines de entrada y salida y su configuración de reloj. Una vez hecho este proceso, la herramienta genera el código resultante que implementa dicha configuración. De esta manera solo debemos preocuparnos por la funcionalidad de nuestro código.

La configuración de periféricos la explicaremos en secciones posteriores. Sin embargo, procederemos a mostrar la configuración de clock del proyecto. Ingresando en la pestaña Clock configuration, accedemos a un diagrama esquemático de las señales de clock del microcontrolador, como el que vemos en la Figura 52.

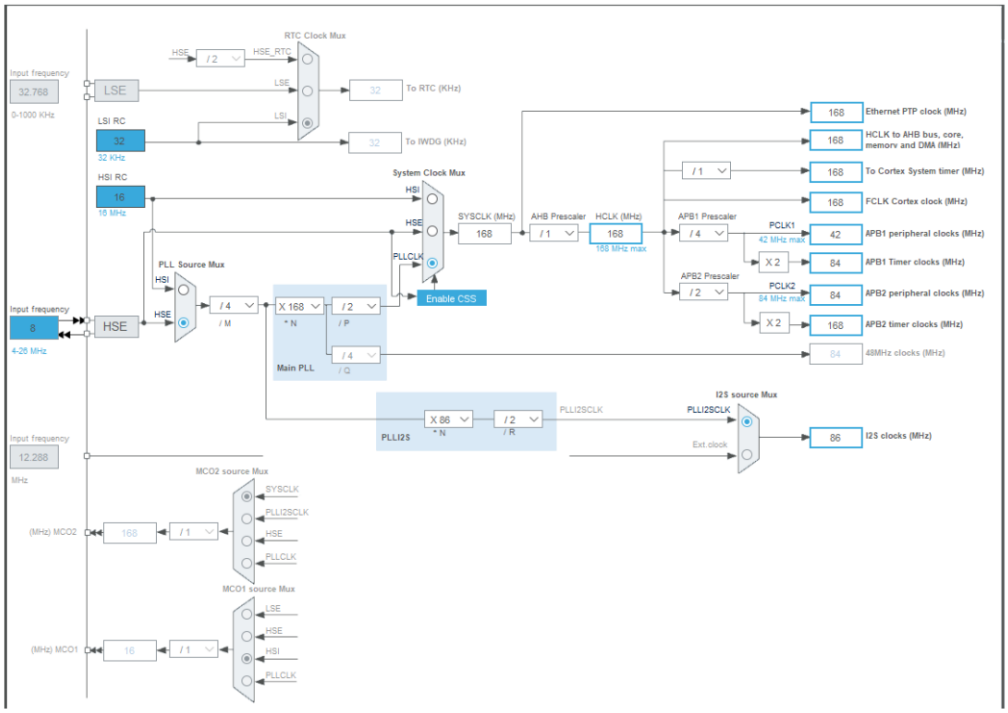


Figura 52 - Configuración de clock

Del lado izquierdo seleccionamos el reloj de entrada. En nuestro caso, se corresponde con el cristal de 8 MHz que posee la placa de desarrollo. Lo importante es completar los recuadros azules que fijan la frecuencia del reloj general elevada por el PLL interno (que en nuestro caso seteamos a 168 MHz) y las velocidades de clock de los periféricos. Como ejemplo, podemos ver que se configuró la velocidad de clock de I2S en 86 MHz. Una vez llenados estos campos, la herramienta configura el PLL y los diferentes divisores de frecuencia para ajustar el árbol de clock a los parámetros fijados por el usuario.

La configuración del pinout se realiza en otra sección de la interfaz gráfica donde se tiene una vista del microcontrolador con todos sus pines, y los mismos se van asignando a cada periférico del MCU según se van fijando los parámetros de las distintas interfaces. La GUI permite, además, asignar un nombre personalizado a cada pin, con el que será referenciado dentro del código. No todos los pines son intercambiables entre sí: cada periférico admite el uso de ciertos pines, y hay algunas funcionalidades que son más flexibles que otras. Como la configuración de periféricos e interfaces la veremos en detalle secciones más adelante, no explicaremos aquí la configuración de pines. Sin embargo, en la Figura 53 se muestra el pinout resultante en esta vista, una vez configuradas todas las interfaces necesarias para el proyecto.

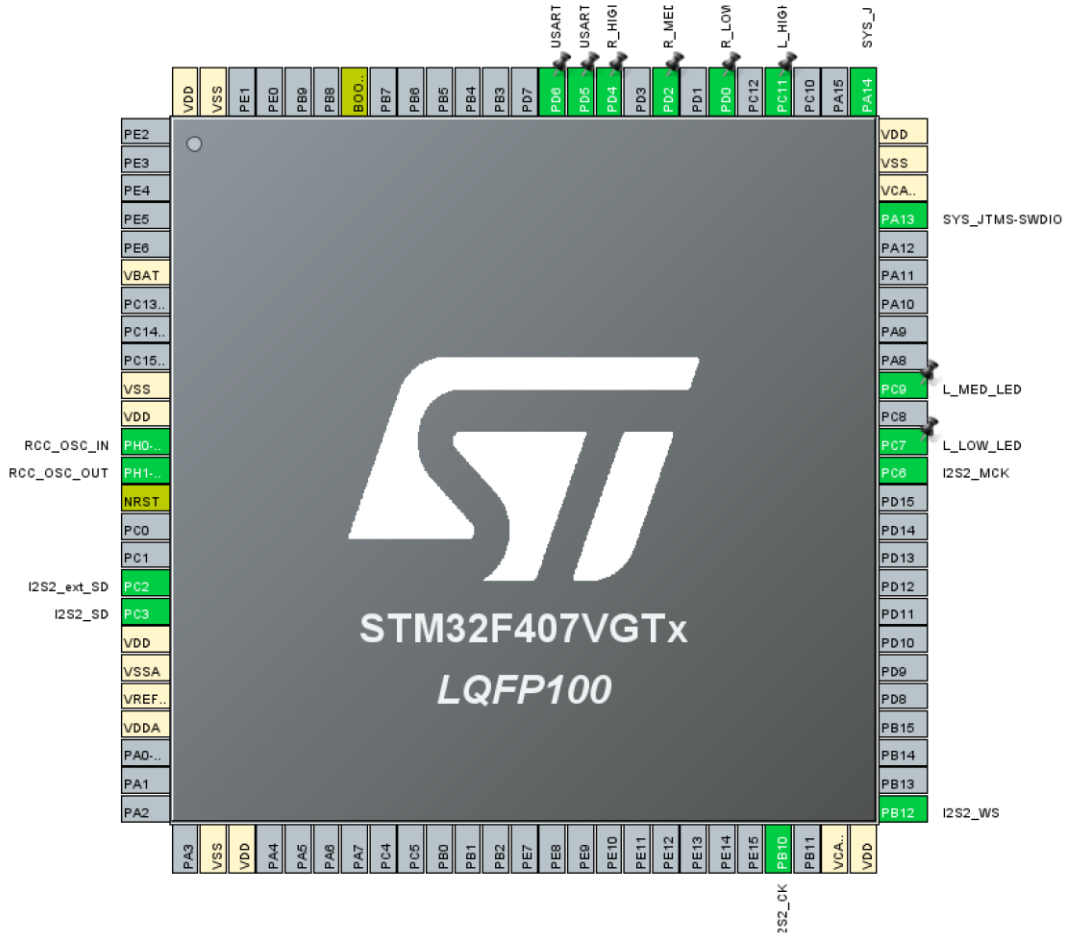


Figura 53 - Configuración de pines

Diagrama de estados

El bloque DSP posee 4 estados de funcionamiento: STOP, ADAPT, FILTER y LOOPBACK. Estos, junto con sus posibles transiciones, se representan en el diagrama de estados de la Figura 54. Cada estado dentro del firmware se simboliza con un número del 0 al 3. Los cambios de un estado a otro están gobernados por los comandos que llegan del bloque control de usuario a través de la interfaz UART.

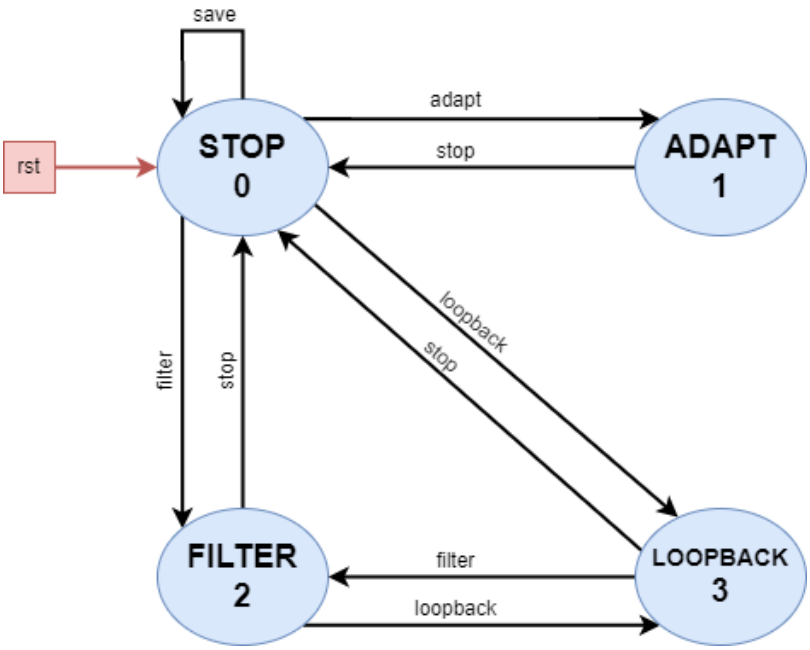


Figura 54 - Diagrama de estados DSP

La adquisición de los datos de audio de entrada y su escritura en la salida, así como el de la interfaz UART, se realiza en segundo plano mediante interrupciones y DMA. Por este motivo, se puede pensar en la escritura de estos registros como un proceso "cuasi-paralelo" al diagrama de estados presentado. Desde este punto de vista, la función de la máquina de estados es actualizar el valor de los registros de las señales de audio de salida en función de la lectura de los registros de entrada, leer o escribir los registros correspondientes al UART y escribir los valores de salida de los GPIO's del vúmetro. A continuación, se explica en detalle cada uno de los estados.

Estado STOP (0)

En este estado comienza la operación del sistema al producirse el encendido o reset. Al comienzo, se setean todos los parámetros de operación, la interfaz de audio, la comunicación UART y los puertos de salida del vúmetro.

Estando en este estado, los registros de salida de audio son puestos en cero, a fin de que no haya señal de audio en la salida. Sin embargo, se siguen monitoreando las señales de entrada, a fin de indicar mediante el vúmetro su presencia.

Cuando llega la orden de "adapt", el sistema debe pasar al estado ADAPT. Antes de ello, se fijan los parámetros del filtro adaptativo, tales como el valor del coeficiente de adaptación μ y el estado inicial de los coeficientes del modelo. De la misma forma, cuando el usuario detiene el proceso de adaptación, siempre se retorna al estado STOP, donde ya se poseen los valores finales de los coeficientes. En este punto, llega el comando "save", que no implica un cambio de estado sino la orden de que el sistema almacene dichos coeficientes en la memoria no volátil del sistema.

Al recibir la instrucción “filter”, se pasa al estado FILTER, donde previamente se debe inicializar el filtro FIR con los coeficientes almacenados en la memoria. El paso al estado LOOPBACK es completamente análogo, con la salvedad que no debe configurarse ningún parámetro antes de avanzar a dicho estado.

Estado ADAPT (1)

En este estado se pone en funcionamiento el algoritmo adaptativo NMLS, el cual actualiza sus coeficientes según la evolución de las señales de entrada x[n] y d[n], y genera las señales de salida y[n] y e[n]. El procesamiento de audio en esta etapa implica leer las señales de entrada del registro correspondiente, actualizar el valor de coeficientes del filtro y las señales de salida, y escribir el valor de estas últimas en el registro de audio de salida.

Por otro lado, se debe calcular la métrica de MSE y escribir su valor cada cierto tiempo en el registro de salida UART, a través del cual llegará al bloque de control de usuario para que este genere en la GUI una gráfica que indique la evolución del proceso de adaptación.

Para salir de este estado, debe llegar el comando “stop”, que devuelve al sistema al estado STOP.

Estado FILTER (2)

En el estado FILTER se lee el registro de las señales de entrada y se procesa su información mediante el filtro FIR de coeficientes fijos almacenado en la memoria del sistema para luego escribir dicho valor en los registros de salida.

A este estado se puede pasar desde el estado STOP, o bien desde el estado de LOOPBACK. Lo mismo sucede para salir del mismo.

Estado LOOPBACK (3)

En el estado de LOOPBACK la información leída del registro de audio de entrada es copiada directamente en el registro de salida, generando así una conexión directa entre la señal de audio a la entrada y el dispositivo de salida.

Al igual que ocurre con el estado FILTER, se pasa a este estado o bien desde STOP o desde FILTER. La salida de este estado ocurre de igual forma, con la única diferencia que al pasar al estado FILTER se debe inicializar el estado de sus coeficientes.

DMA

Dado que tratamos con señales de audio, las cuales son un flujo continuo de datos, resulta necesaria la aplicación de métodos para procesarlas en tiempo real, de tal forma que consuman la menor cantidad de ciclos del procesador y que no haya “cortes” en la recepción o trasmisión de las mismas. La solución a esto se encuentra en el uso de técnicas de procesamiento como las que se explicaron en la sección 4.1.1.6. En este apartado nos detendremos en explicar el funcionamiento del periférico DMA y de su uso en el firmware, ya que el mismo es el que nos permite aplicar los métodos vistos.

Una unidad de acceso directo a memoria (DMA, por sus siglas en inglés) es un elemento de lógica digital en la arquitectura de computadoras que se puede utilizar junto con el microprocesador principal en el mismo chip para facilitar las operaciones de transferencia de memoria. Esto reduce significativamente la carga de la CPU, ya que el controlador DMA puede realizar transferencias de datos de memoria a memoria, así como transferencias de datos de periféricos a memoria o viceversa. La existencia de DMA junto con una CPU puede acelerar su rendimiento en varios órdenes de magnitud.

DMA en STM32

Los microcontroladores STM32 poseen 2 controladores DMA y 16 "streams" DMA. Los streams son vías por donde puede conectarse la memoria, y cada controlador tiene 8 con los que trabajar.

El controlador DMA puede operar de las siguientes maneras:

- Modo periférico a memoria (DMA1, DMA2)
- Modo memoria a periférico (DMA1, DMA2)
- Modo memoria a memoria (sólo DMA2)

Cada controlador tiene 8 streams, que soportan de manera independiente:

- Disparo por hardware o software.
- Buffer doble o circular.
- Incremento de dirección de memoria en direcciones de origen y destino.
- FIFO de 4 x 32 bits que puede activarse o desactivarse.
- Banderas de interrupción para medio-transmisión, transferencia completa, error, etc.
- La prioridad de cada stream puede ser configurada por software, pero si dos streams tienen igual prioridad de software, el que tenga mayor prioridad de hardware comandará la operación.

Canales DMA

Los controladores DMA tienen acceso a un bus especial denominado puerto periférico AHB, que les proporciona un acceso rápido a los periféricos seleccionados. Esto es útil ya que permite eludir la matriz de bus estándar y reducir significativamente la latencia.

Cada flujo tiene 8 canales, que se pueden pensar como un multiplexor para el bus periférico. El canal selecciona un disparador DMA o solicitud, que se puede utilizar para iniciar el proceso de transferencia DMA en función del estado y la configuración de un periférico. Los canales se suman con un disparador de software que puede invocar manualmente el proceso de transferencia DMA.

Direccionamiento DMA

El flujo de datos puede acceder a los datos como variables char, half-word o word (int_8, int_16 o int_32). El tamaño de los datos de origen y destino se configura de forma independiente. Cada flujo tiene un puntero de dirección de origen y otro de destino.

En el modo Memoria a Periférico, la dirección de origen puede ser una dirección SRAM, una dirección FLASH o cualquier otra dirección a la que pueda estar conectado el puerto DMA en la matriz de bus del MCU.

En el modo Periférico a Memoria, se aplica lo inverso a lo anterior.

En el modo Memoria a Memoria, tanto la dirección de origen como la de destino pueden ser cualquier dirección en la matriz de bus a la que el controlador DMA tenga acceso. Esto puede incluir direcciones de periféricos, que son manejadas por el administrador de memoria de la matriz de bus.

Tanto el puntero de origen como el de destino tienen una función de incremento que se puede habilitar o deshabilitar. Después de cada escritura exitosa, el puntero de dirección se incrementará una unidad de datos. Como las unidades de datos se configuran de forma independiente para origen y destino, es posible, por ejemplo, leer unidades de 16 bits y, sin embargo, incrementar el destino en 32 bits a la vez.

El tamaño máximo de transferencia del búfer de origen es programable. También se admite el modo circular, donde el puntero de dirección se recarga automáticamente después de que se haya vaciado el búfer de origen y comience nuevamente el proceso de transferencia DMA. Esto puede ser útil para la transmisión continua de datos. El modo circular también tiene un modo de doble búfer, donde en lugar de recargar la dirección del puntero inicial, se recarga una segunda dirección de puntero, de modo que el búfer pueda ser escrito por un programa secundario sin ningún conflicto. Esta funcionalidad nos será útil para implementar la técnica de doble buffer explicada en el apartado 4.1.1.6.2.

Interrupciones y callbacks

Los streams del controlador DMA son capaces de generar interrupciones, las cuales pueden ser utilizadas para, por ejemplo, cargar más datos en el búfer de origen cuando la transferencia ha llegado a la mitad de su concreción, o cuando la misma ha sido completada totalmente.

A nivel de firmware, estas interrupciones disparan funciones asíncronas denominadas “callbacks”, que se comportan de igual manera que las rutinas de interrupción y deben

utilizarse para saber cuándo se ha completado una transacción DMA y utilizar la información disponible.

La librería HAL nos provee de dichos callbacks, en todas las direcciones del bus DMA (entre periféricos y memoria, o de memoria a memoria) y para transferencias de buffer simples o dobles. En este último caso tenemos un callback para cuando se completó la mitad del buffer, y otro para cuando se llenó totalmente.

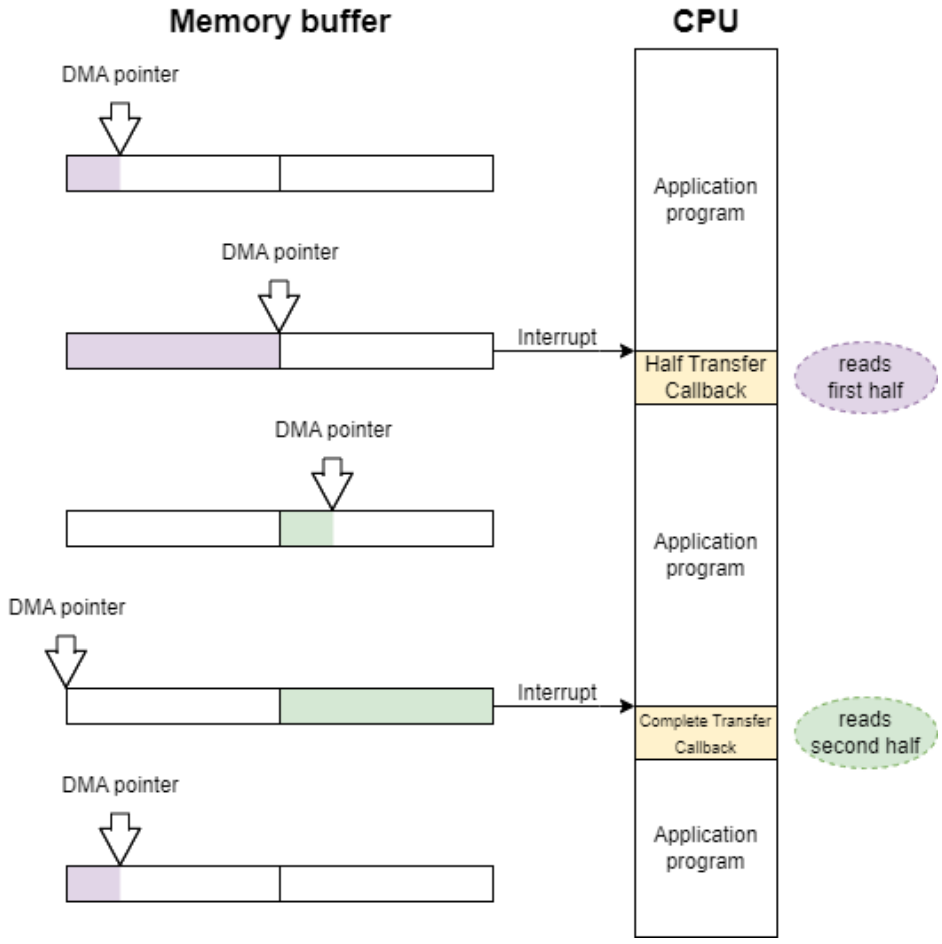


Figura 55 - Buffer doble con DMA y callbacks

En la Figura 55 se representa la implementación de un esquema de adquisición de datos de un periférico mediante la técnica de buffer doble o circular utilizando el DMA y los callbacks que provee la librería HAL. El puntero DMA comienza llenando la primera mitad del buffer al mismo momento que la CPU realiza tareas de procesamiento de la aplicación. Cuando se llena la mitad del buffer, se genera una interrupción y se dispara un callback “Half Transfer” que es atendido por la CPU, la cual lee la información contenida en dicha mitad. Sin embargo, este proceso no interrumpe al DMA, quien continúa llenando la segunda mitad del buffer. Cuando se completa la transferencia de esta mitad, se vuelve a generar una interrupción y un callback “Complete Transfer”, que indica a la CPU que ya puede leer la segunda mitad del buffer, mientras el DMA, por estar configurado de modo circular, vuelve a posicionarse al comienzo del buffer y comienza el ciclo nuevamente.

Si las tareas de transferencia o procesamiento de la información no son tan críticas o cuentan con un tiempo holgado, puede utilizarse un esquema simple donde solo se lea todo el buffer una vez que el mismo se complete. En dicho caso solo se debe hacer uso del callback “Complete Transfer”.

Se ejemplificó el funcionamiento de los callbacks para una transferencia de entrada (periférico a memoria), pero el proceso es completamente análogo si se tratase de una transferencia de salida (memoria a periférico). En dicho caso los callbacks se disparan cuando la transferencia del buffer de memoria al periférico ha sido completada, indicando a la CPU que puede copiar nuevos datos en la memoria.

Para periféricos que cuentan tanto con entradas y salidas, en el caso que las mismas deban ser sincronizadas (como es el caso de I2S), la librería HAL nos provee de callbacks que simplifican esta tarea. Estos callbacks se disparan una sola vez cuando se completaron las transferencias del dispositivo de entrada a memoria y la de la memoria de salida al dispositivo. De esta manera la CPU recibe la información de entrada, la procesa y la devuelve a la salida del periférico manteniendo un periodo fijo y síncrono.

Configuración de periféricos y librerías

En esta sección comentaremos la configuración de parámetros y pines para las interfaces I2S y UART, como así también la inclusión de las librerías para el manejo de la memoria FLASH y CMSIS para las instrucciones DSP.

Configuración de periférico I2S

Para la configuración de I2S tendremos en cuenta las especificaciones del módulo Pmod I2S2, en particular de sus conversores ADC y DAC internos.

En este caso el protocolo debe ser configurado para operar con señal de MCLK, y con un formato para los datos que sigue el estándar I2S Philips, con 24 bits útiles en un frame de 32 bits. Esto implica que la palabra dato es un entero de 32 bits, pero sus 8 bits más significativos son 0.

En la Figura 56 se observa la configuración de estos parámetros en Cube Mx. Vemos que se ha habilitado la comunicación full dúplex con generación del clock Master para comunicar tanto el ADC como el DAC. Además, se ha elegido una frecuencia de muestreo de 48000 Hz, la cual en la práctica es levemente menor debido a la precisión del PLL del microcontrolador, el cual fue ajustado como se indicó en la sección 0 para que el error en dicha frecuencia sea el menor posible, en este caso de 0.01%.

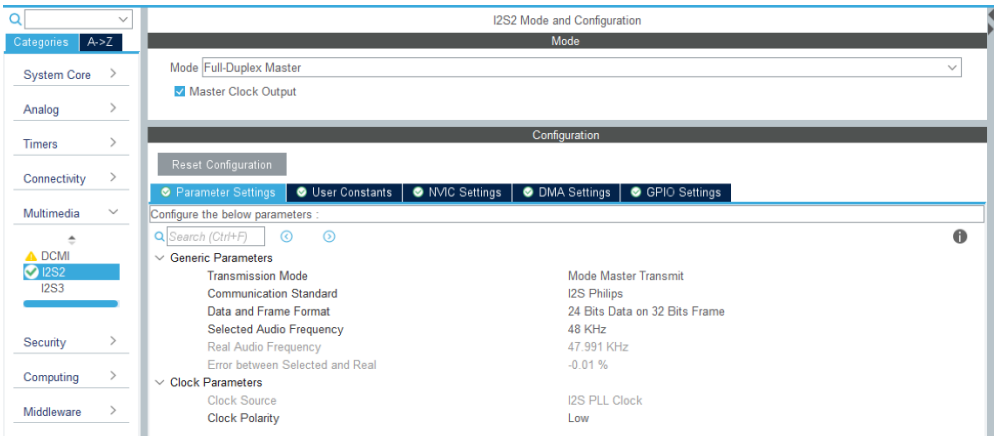


Figura 56 - Configuración de parámetros I2S

En la pestaña DMA Settings debemos configurar el DMA para que opere en este periférico. En la Figura 57 vemos que se establecieron dos streams de DMA, uno de periférico a memoria para los datos de entrada, y otro de memoria a periférico para los datos de salida. En ambos casos, se estableció un buffer circular para poder aplicar la técnica de doble buffer.

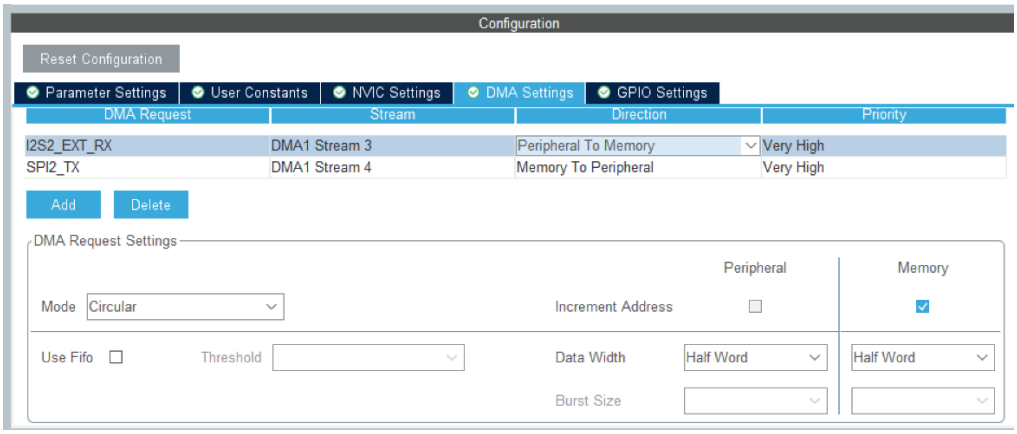


Figura 57 - Configuración de DMA e I2S

Si bien los datos se encuentran en un frame de 32 bits, el DMA para I2S exige trasmitir con un tamaño de dato de “Half Word” o 16 bits lo que significa que los datos serán fraccionados cada 16 bits. Esto implica que al recibir un bloque de 2N datos (N datos para cada canal), el bloque de memoria será de 16 bits de tamaño de palabra y de 4N posiciones. Por otro lado, recordemos que el protocolo I2S transmite datos de los canales izquierdo y derecho de forma alternada, por lo que los datos se ubicarán en posiciones alternadas de memoria.

Para obtener las muestras pertenecientes a los canales izquierdo y derecho se deberá tomar esos datos, combinar sus MSB y LSB y separar las muestras de cada canal, obteniendo así bloques de memoria de 32 bits y tamaño N. Esto se representa en la Figura 58. Para el flujo de datos de salida, la operación se debe realizar a la inversa.

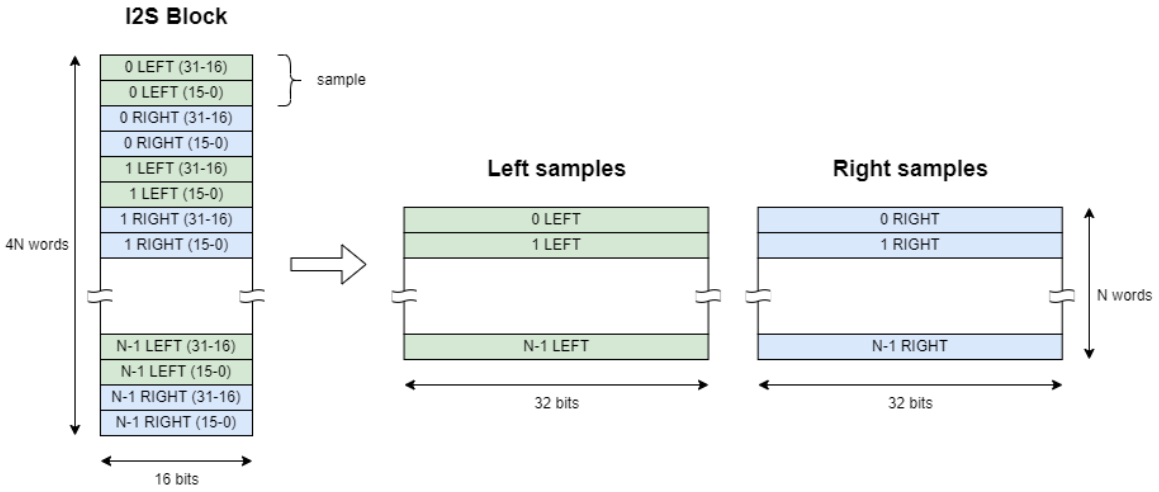


Figura 58 - Recepción de datos I2S

Por último, en la pestaña "GPIO Settings" quedan configurados los puertos de entrada y salida de la interfaz, a través de los cuales nos comunicaremos con el módulo Pmod I2S2.

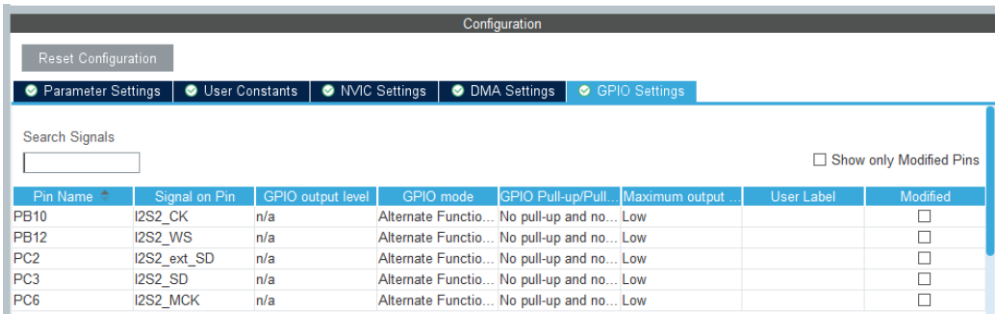


Figura 59 - Configuración de GPIO's I2S

Configuración del periférico UART

Para la configuración de UART tendremos en cuenta los casos de recepción y de trasmisión de datos por este medio.

En el caso de la recepción, es a través de la interfaz UART que se recibirán los comandos que manejan la máquina de estados del sistema. Estos datos pueden arribar en cualquier momento de operación del sistema, y si bien no son datos de gran tamaño ni que se den en ráfagas muy veloces, sí es de vital importancia que el bloque no los pierda ni degrade. Por este motivo, utilizaremos DMA para manejar la interfaz Rx de UART.

Por otro lado, los datos de trasmisión se corresponden con el valor de la métrica MSE que el bloque envía durante el estado de adaptación. Este dato se calcula para cada bloque que se recibe, por lo que se actualiza muy velozmente. A pesar de esto, no es crítico que se trasmita cada vez que se calcula su valor, sino cada cierto tiempo, el cual tampoco es necesario que sea periódico, sino lo suficientemente rápido para que el bloque de control de usuario pueda actualizar su información y generar un feedback al usuario. Por este

motivo, no utilizaremos DMA en la trasmisión de datos UART, sino que en cambio tomaremos un enfoque basado en interrupciones.

La forma de operar la trasmisión UART mediante interrupciones es menos precisa que con DMA, pero demanda menos recursos del MCU, por lo que resulta más conveniente para esta tarea que no es tan prioritaria. Cada vez que se inicia una trasmisión UART, el periférico demora cierto tiempo en completarla, el cual al ser una comunicación asíncrona puede que no sea un tiempo fijo. Cuando la trasmisión ha finalizado, la interfaz UART genera una interrupción a la CPU, que a nivel de firmware se traduce en una función callback, al igual que con DMA. Cuando esto sucede, para seguir trasmitiendo datos es tarea del programador ejecutar dicha instrucción. En esencia, el proceso es muy similar al realizado con DMA, con la diferencia que en este caso la CPU es la responsable de restablecer la trasmisión de datos.

La configuración de parámetros de UART en Cube Mx se muestra en la Figura 60. Se estableció una velocidad de comunicación de 115200 baudios, un tamaño de palabra de 8 bits sin paridad y un bit de parada.

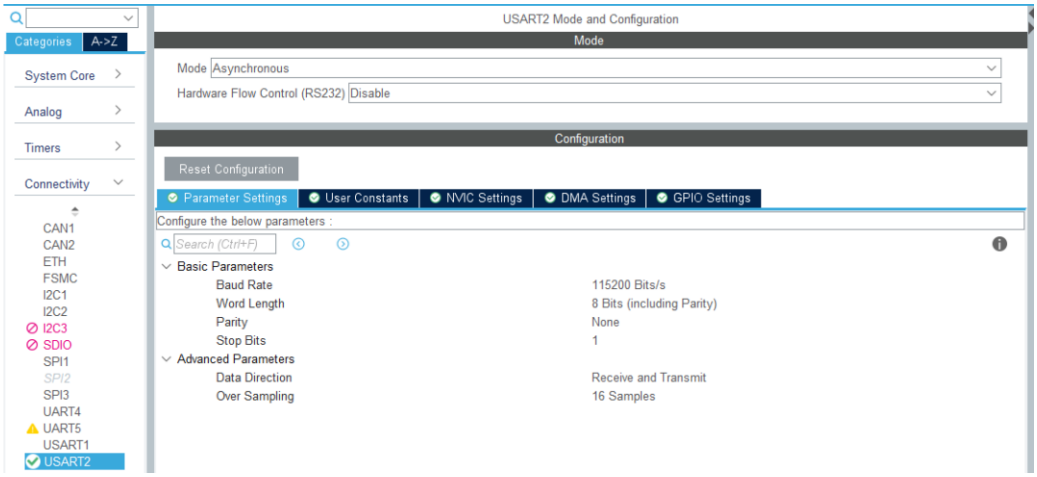


Figura 60 - Configuración de parámetros UART

En la pestaña de NVIC ("Nested Vector Interrupt Controller") activamos las interrupciones para la interfaz de trasmisión como se indica en la Figura 61.

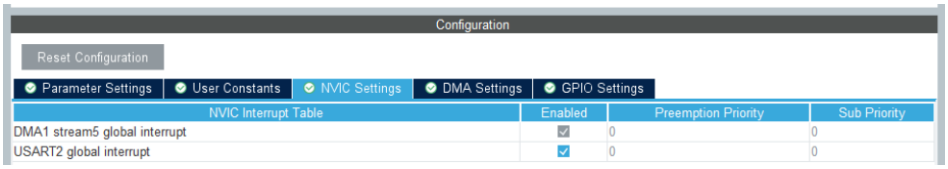


Figura 61 - Configuración de interrupciones UART

Por otro lado, en la pestaña de DMA configuramos el stream para la recepción, como se indica en la Figura 62. El tamaño de palabra se setea en "Byte", ya que recibiremos hasta 8 bits.

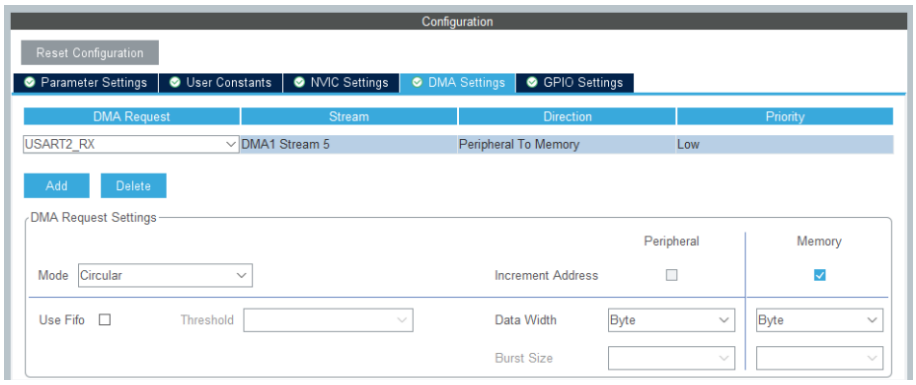


Figura 62 - Configuración de DMA y UART

Finalmente, en la Figura 63 se listan los puertos del MCU utilizados para esta interfaz.

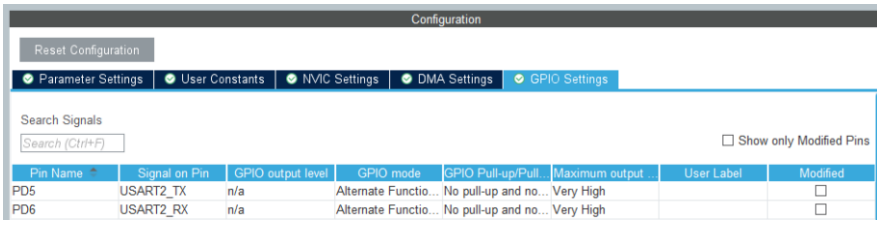


Figura 63 - Configuración de GPIOs UART

Configuración de puertos de salida

A nivel de configuración de periféricos, solo resta configurar los pines de salida para controlar el vúmetro. Esto se realiza muy sencillamente con la interfaz gráfica del microcontrolador de Cube Mx.

Al hacer click en cualquiera de los pines se listan las opciones disponibles. En nuestro caso, seleccionamos GPIO_Output para los 6 pines de salida seleccionados, como se indica en la Figura 64.

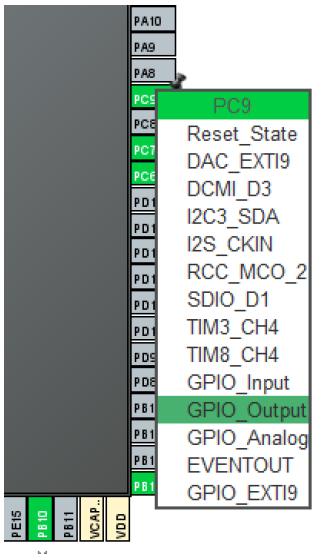
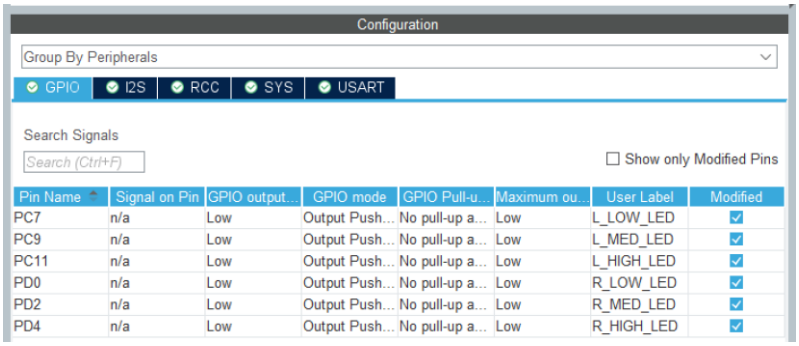


Figura 64 - Selección de GPIO

Seguidamente, en la pestaña de configuración de GPIOs se listan todos los pines seleccionados (incluso también los ya configurados de otros periféricos). En esta vista podemos establecer su estado inicial al reiniciar el chip, su circuitería de salida y además podemos asignar una etiqueta para nombrarlos.



Pin Name	Signal on Pin	GPIO output...	GPIO mode	GPIO Pull-u...	Maximum ou...	User Label	Modified
PC7	n/a	Low	Output Push...	No pull-up a...	Low	L_LOW_LED	☑
PC9	n/a	Low	Output Push...	No pull-up a...	Low	L_MED_LED	☑
PC11	n/a	Low	Output Push...	No pull-up a...	Low	L_HIGH_LED	☑
PD0	n/a	Low	Output Push...	No pull-up a...	Low	R_LOW_LED	☑
PD2	n/a	Low	Output Push...	No pull-up a...	Low	R_MED_LED	☑
PD4	n/a	Low	Output Push...	No pull-up a...	Low	R_HIGH_LED	☑

Figura 65 - Parámetros de GPIOs

Librería CMSIS-DSP

Para la operación del sistema, se deben realizar numerosas operaciones de procesamiento digital de señales, tales como filtros FIR, el algoritmo adaptativo NLMS, cálculo de valores RMS, entre otros. Si bien estas operaciones se pueden ejecutar utilizando operaciones comunes del lenguaje de programación, el procesador STM32F407VG proporciona instrucciones para el procesamiento de señales y admite instrucciones avanzadas SIMD (Single Instruction Multi Data) y MAC (Multiply and Accumulate) de un solo ciclo. El uso de este tipo de instrucciones en una aplicación DSP en tiempo real no solo reduce los costos de procesamiento, sino que también disminuye el consumo de energía total.

Por este motivo, se propone la utilización de librerías específicas de DSP que aprovechen esta funcionalidad del microprocesador. En este sentido, la librería CMSIS (Common Microcontroller Software Interface Standard) resulta altamente beneficiosa.

En particular, la librería de software CMSIS DSP es una colección de funciones comunes de procesamiento de señales para su uso en dispositivos basados en los procesadores Cortex-M y Cortex-A. Se divide en varias funciones que cubren categorías específicas:

- Funciones matemáticas básicas.
- Funciones matemáticas rápidas.
- Funciones matemáticas complejas.
- Funciones de filtrado.
- Funciones de matrices.
- Funciones de transformación.
- Funciones de control de motores.
- Funciones estadísticas.
- Funciones de soporte.
- Funciones de interpolación.
- Funciones de Máquina de Vectores de Soporte (SVM).
- Funciones de clasificación de Bayes.
- Funciones de distancia.
- Funciones de cuaterniones.

La biblioteca tiene funciones separadas para operar en valores de enteros de 8 bits, enteros de 16 bits, enteros de 32 bits y valores de punto flotante de 32 bits. En nuestra aplicación utilizaremos 3 funcionalidades de esta librería, las cuales operan con datos de tipo float:

Filtrado FIR

Utilizaremos dos funciones básicas:

Función de inicialización:

*void arm_fir_init_f32(arm_fir_instance_f32 * S,uint16_t numTaps,const float32_t *pCoeffs,float32_t *pState,uint32_tblockSize);*

Parámetros:

- [in,out] S: puntero a una instancia de la estructura FIR filter, definida también por la librería.
- [in] numTaps: número de coeficientes del filtro.
- [in] pCoeffs: puntero al buffer de coeficientes.
- [in] pState: puntero al buffer de estado del filtro.
- [in] blockSize: número de muestras del bloque a procesar.

pCoeffs es un puntero al vector de coeficientes del filtro almacenados en orden temporal inverso:

$$\{b[numTaps-1], \quad b[numTaps-2], \quad \cdots \quad , \quad b[1], \quad b[0]\} \tag{48}$$

pState apunta a un arreglo de variables de estado definidos por la estructura S. pState tiene un tamaño de numTaps+blockSize-1 muestras, donde blockSize es el número de muestras del bloque de entrada en cada llamada a la función de procesamiento arm_fir_f32().

Es importante tener en cuenta que la longitud del búfer de estado supera la longitud del arreglo de coeficientes en blockSize-1. El aumento en la longitud del búfer de estado permite evitar el direccionamiento circular, que se utiliza tradicionalmente en los filtros FIR, y proporciona una mejora significativa en la velocidad. Las variables de estado se actualizan después de procesar cada bloque de datos, mientras que los coeficientes permanecen sin cambios.

Función de procesamiento:

*void arm_fir_f32(const arm_fir_instance_f32 * S, const float32_t * pSrc,float32_t * pDst,uint32_t blockSize);*

Parámetros:

- [in,out] S: puntero a una instancia de la estructura FIR filter, definida también por la librería.
- [in] pSrc: puntero al bloque de datos de entrada.
- [out] pDst: puntero al bloque de datos de salida.
- [in] blockSize: número de muestras del bloque a procesar.

La función opera en bloques de datos de entrada y salida, y cada llamada a la función procesa blockSize muestras a través del filtro. pSrc y pDst apuntan a matrices de entrada y salida que contienen blockSize valores.

El algoritmo del filtro FIR se basa en una secuencia de operaciones de multiplicación y acumulación (MAC). Cada coeficiente del filtro b[n] se multiplica por una variable de estado que es igual a una muestra de entrada anterior x[n].

$$y[n] = b[0]\,x[n] + b[1]\,x[n-1] + \cdots + b[numTaps-1]\,x[n-numTaps-1] \tag{49}$$

Filtrado adaptativo NMLS

La estructura de los datos e implementación es similar a las del filtrado FIR. También se utilizan dos funciones:

Función de inicialización:

*void arm_lms_norm_init_f32(arm_lms_norm_instance_f32 *S,uint16_t numTaps,float32_t *pCoeffs,float32_t *pState,float32_t mu, uint32_t blockSize);*

Parámetros:

- [in] S: puntero a una instancia de la estructura LMS filter, definida también por la librería.
- [in] numTaps: número de coeficientes del filtro.
- [in] pCoeffs: puntero al buffer de coeficientes.
- [in] pState: puntero al buffer de estado.
- [in] mu: coeficiente de adaptación que controla la actualización de los coeficientes del filtro.
- [in] blockSize: número de muestras del bloque a procesar.

El valor inicial de los coeficientes del filtro debe ser ajustado para que sirva de estado inicial al proceso de adaptación.

Función de procesamiento:

*void arm_lms_norm_f32(arm_lms_norm_instance_f32 * S,const float32_t *pSrc,float32_t *pRef,float32_t *pOut,float32_t *pErr,uint32_t blockSize);*

Parámetros

- [in] S: puntero a una instancia de la estructura LMS filter, definida también por la librería.
- [in] pSrc: puntero al bloque de datos de la señal de entrada x.
- [in] pRef: puntero al bloque de datos de la señal de referencia o deseada d.
- [out] pOut: puntero al bloque de datos de la señal de salida y.
- [out] pErr: puntero al bloque de datos de la señal de error e.
- [in] blockSize: número de muestras del bloque a procesar.

Esta función implementa un filtro FIR adaptativo NMLS idéntico al descripto en la sección 4.1.1.3.3. La función incluye tanto el algoritmo de adaptación de coeficientes como el filtro FIR. El filtro NLMS tiene dos señales de entrada. La señal de entrada alimenta el filtro FIR, mientras que la señal de referencia corresponde a la salida deseada del filtro. Es decir, los coeficientes se actualizan para que la salida coincida con la entrada de referencia. El mecanismo de actualización de coeficientes se basa en la diferencia entre la salida del filtro y la entrada de referencia. Esta señal de error tiende a cero a medida que el filtro se adapta. Las funciones de procesamiento NLMS aceptan las señales de entrada y de referencia y generan la salida del filtro y la señal de error.

Al igual que para el filtro FIR, esta función opera en bloques de datos y cada llamada a la función procesa blockSize muestras a través del filtro. pSrc apunta a la señal de entrada, pRef apunta a la señal de referencia, pOut apunta a la señal de salida y pErr apunta a la señal de error. Todos los arreglos contienen blockSize valores. A pesar de que la función opera en base a bloques, internamente los coeficientes del filtro se actualizan muestra a muestra.

RMS

Para el funcionamiento del vúmetro debemos obtener el valor RMS de las señales de entrada. La librería CMSIS nos provee de una función que optimiza este cálculo:

*void arm_rms_f32(const float32_t *pSrc,uint32_t blockSize,float32_t *pResult);*

Parámetros:

- [in] pSrc: puntero al vector de datos de entrada.
- [in] blockSize: número de muestras del vector de entrada.
- [out] pResult: puntero a la variable donde se almacena el resultado.

Descripción de funcionamiento

En esta sección describiremos el funcionamiento del código principal del bloque DSP. El código utilizado y todos los detalles de la implementación se encuentran en el Anexo B.

Generalmente el programa que corre en un microcontrolador sigue un esquema de instrucciones secuenciales. Sin embargo, al usar DMA e interrupciones el código deja de tener un flujo claro de funcionamiento en base a un paradigma secuencial. Podemos esquematizar su operación a partir de máquinas de estado cuyos parámetros de entrada están dados por los mecanismos asíncronos comandados por DMA.

En esencia, la base de funcionamiento del programa está basada en la interfaz de audio I2S, la cual es una máquina asíncrona que toma datos de entrada cuando se produce una interrupción DMA, determina el valor correspondiente al buffer de salida en base a la variable de estado del sistema y coloca dichos datos en su interfaz. Paralelamente, existe otra máquina de estados asíncrona (que corresponde a la presentada en la sección 0) cuya interfaz de control es UART y su salida, el valor de la variable de estado del sistema. Esta variable dicta a la primera máquina qué rutina debe seguir para calcular la salida de los datos. La interrelación entre estas máquinas se representa en la Figura 66.

Como datos de salida de la máquina de procesamiento de audio se incluyen, además de las señales I2S de salida, las salidas del vúmetro y el valor de MSE que se trasmite por UART hacia el bloque de control de usuario.

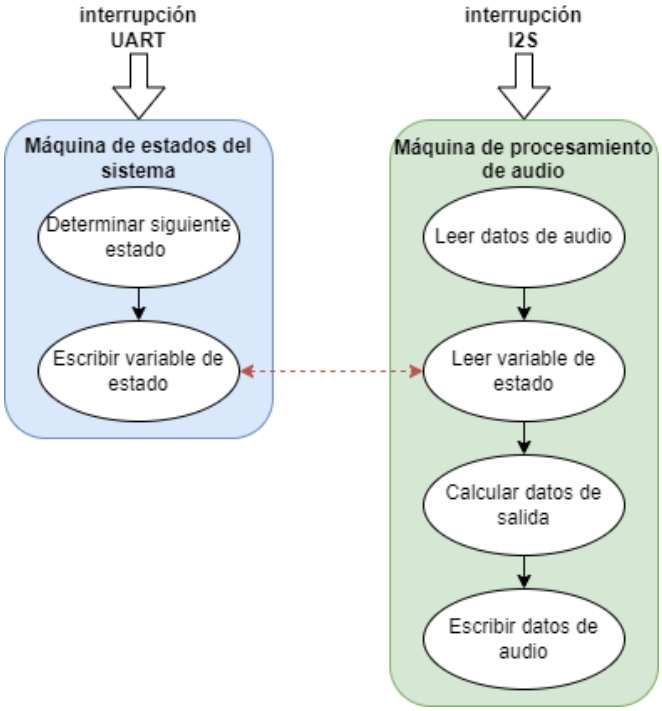


Figura 66 - Esquema de funcionamiento asíncrono

Máquina de estados del sistema

Esta parte del procesamiento se corresponde con la Figura 54 presentada como el diagrama de estados del sistema. Básicamente su funcionalidad es determinar en qué estado debe operar el bloque a partir de los comandos que recibe por UART.

Como ya fue presentada la lógica de funcionamiento, nos adentraremos en algunos detalles de implementación del protocolo de comunicación UART entre este bloque y el Control de usuario y cómo se produce la transición entre estados.

Para la comunicación entre el control de usuario y el DSP definimos una trama de 4 caracteres de largo fijo y que sigue la estructura presentada en la Tabla 19.

Trama UART Rx			
Caracter de inicio	**Comando**		**Caracter de fin**
%	X	X	#

Tabla 19 - Trama UART Rx

Con los caracteres de inicio y cierre la interfaz de recepción UART del bloque DSP puede determinar el inicio y fin de la trama recibida e interpretar correctamente los dos caracteres del comando. Cada vez que la interfaz UART recibe un byte se dispara un

callback donde primeramente se guarda el caracter recibido en un buffer de recepción y a continuación se busca el caracter de fin para limpiar dicho buffer. Luego se utiliza una función que extrae los caracteres de comando del buffer recibido y genera un String que representa el comando recibido. Por último, dicho String es enviado a otra función que interpreta el comando recibido y cambia el valor de la variable de estado del sistema según corresponda.

En la Tabla 20 se listan todos los comandos del protocolo definido junto con la acción de la máquina de estados al recibirlos. En la Figura 67 se representa un diagrama de flujo simplificado de las operaciones descriptas.

Trama recibida	Comando	Nemónico	Descripción
%ST#	ST	Stop	Indica pasar al estado STOP (0).
%AF#	AF	Adapt fast	Indica pasar al estado ADAPT (1) con un parámetro μ alto, asegurando una adaptación rápida. Llama a la función de inicialización del filtro NLMS.
%AS#	AS	Adapt slow	Indica pasar al estado ADAPT (1) con un parámetro μ bajo, asegurando una adaptación lenta. Llama a la función de inicialización del filtro NLMS.
%SV#	SV	Save	Indica guardar los coeficientes del filtro adaptativo en la memoria FLASH.
%FI#	FI	Filter	Indica pasar al estado FILTER (2). Llama a la función de inicialización del filtro FIR.
%LB#	LB	Loopback	Indica pasar al estado LOOPBACK (3).

Tabla 20 - Comandos del protocolo de comunicación UART entre bloques

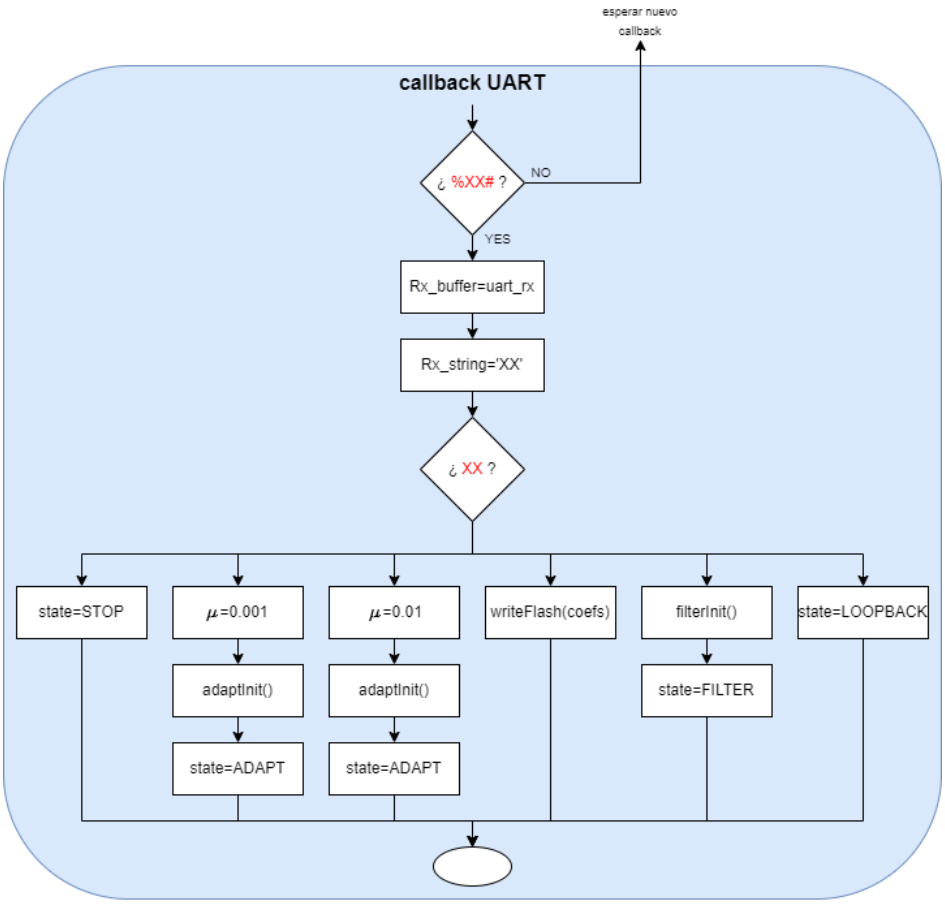


Figura 67 - Diagrama de flujo del callback UART

Máquina de procesamiento de audio

En paralelo a la máquina de estados del sistema, funciona la máquina de procesamiento de audio. La misma sigue una técnica de doble buffer, por lo que se tienen dos callbacks: uno para cuando se ha llenado la primera mitad del buffer de entrada, y a su vez se ha vaciado la primera mitad del buffer de salida; y otro cuando se ha llenado la segunda mitad del buffer de salida, y también se ha vaciado la segunda mitad del buffer de salida. Esta operación ya fue representada en la Figura 55.

Para conocer qué parte del buffer debe ser procesada, definimos una variable global de estado de callback, que puede tomar 3 valores. Cuando es 0, indica que no hay ningún bloque de datos disponibles para procesar; cuando vale 1, que la primera mitad del buffer se ha completado; y al valer 2, que la segunda mitad del buffer se ha completado. En el bucle main() del programa se chequea constantemente su valor y cuando es distinto de cero, se procede a realizar el procesamiento de audio y se la vuelve a 0. Por otro lado, cuando se disparan cada uno de los callbacks, se setea la variable a su valor correspondiente. Esto se representa en la Figura 68.

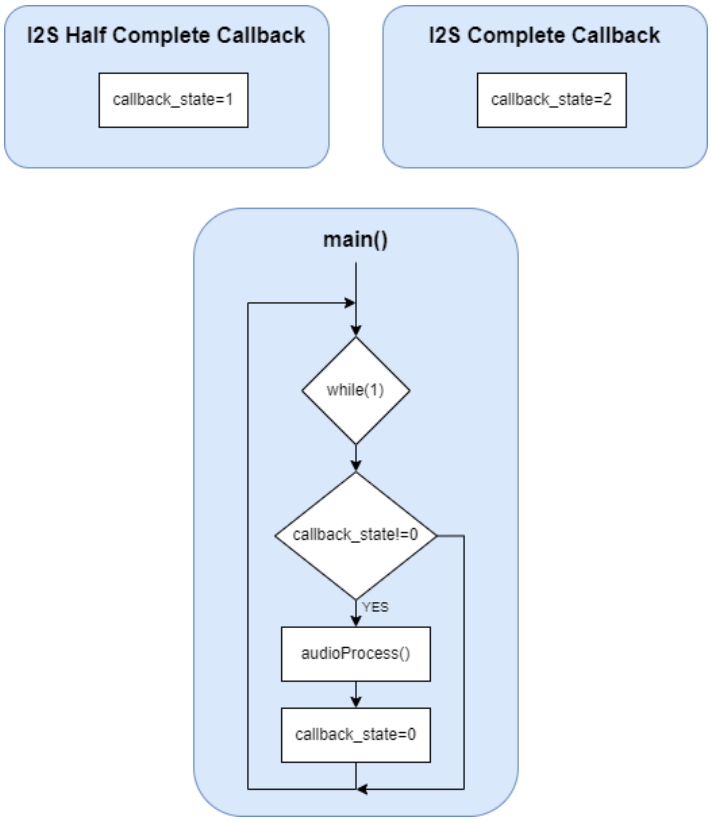


Figura 68 - Diagrama de flujo de procesamiento con técnica de doble buffer

En la función de procesamiento de audio, leyendo la variable de callback se puede conocer qué parte del buffer entrada es la que se debe copiar para su procesamiento y en que sección del buffer de salida se debe depositar la información resultante. Esto se realiza mediante un puntero que es idéntico para los buffers de entrada y salida. Considerando un tamaño de buffer total de 2N datos, cuando se completa la primera mitad del buffer el puntero apunta a la posición 0 del buffer, mientras que cuando se completa la segunda mitad, el puntero apunta a la posición N. En ambos casos se leen N datos y estos son los utilizados para el procesamiento. El funcionamiento del puntero y la variable de callback se ilustra en la Tabla 21.

Estado del buffer de entrada/salida	callback_state	buff_ptr	Datos del buffer de procesamiento
Completando una transferencia	0	-	-
Primera mitad llena	1	0	0-N
Segunda mitad llena	2	N	N-2N

Tabla 21 - Variables de procesamiento para el buffer circular

Luego, deben ser separados los datos correspondientes a cada canal. Para ello se debe tener en cuenta la organización de las muestras en los datos de I2S, lo cual fue explicado anteriormente y se puede visualizar en la Figura 58. Esto da lugar a dos vectores de entrada y dos vectores de salida, correspondientes a los dos canales de audio.

La información de entrada y salida de la interfaz I2S se representa como enteros de 32 bits. Sin embargo, las funciones de procesamiento de audio están implementadas en punto flotante, por lo que en la entrada se deben convertir los datos a punto flotante, procesarlos según convenga, y luego se los devuelve a su representación en punto fijo para escribirlos en el buffer de salida.

Una vez obtenidos los bloques de información de los canales izquierdo y derecho en su representación flotante, se debe procesar esa información para luego convertirla y enviarla a la salida siguiendo el proceso inverso. En este punto se hace uso de la variable de estado del sistema, cuyo establecimiento depende de la máquina de estados del sistema ya explicada. Dependiendo del valor de dicha variable, se ejecutan funciones diferentes para cada estado, las cuales se resumen en la Tabla 22.

state	Función	Descripción	Función de procesamiento
STOP	Stop()	Sin importar las señales de entrada, se setean los bloques de salida derecho e izquierdo a cero.	l_buf_out={0}; r_buf_out={0};
ADAPT	Adapt()	En base a las señales de entrada (x, d) se calculan las señales de salida del filtro adaptativo (y, e) y se actualizan los coeficientes del filtro mediante el algoritmo NLMS. Además, utilizando el bloque de la señal de error se realiza la estimación del valor de MSE.	arm_lms_norm_f32(&adaptive_filter, &r_buf_in[offset_w_ptr], &l_buf_in[offset_w_ptr], &r_buf_out[offset_w_ptr], &l_buf_out[offset_w_ptr], BLOCK_SIZE_FLOAT);
FILTER	Filter()	Para ambos canales se utilizan los coeficientes almacenados del filtro para generar el bloque de datos de salida utilizando el filtrado FIR.	arm_fir_f32 (&firsettings_l, &l_buf_in[offset_w_ptr], &l_buf_out[offset_w_ptr],BLOCK_SIZE_FLOAT); arm_fir_f32 (&firsettings_r, &r_buf_in[offset_w_ptr], &r_buf_out[offset_w_ptr],BLOCK_SIZE_FLOAT);

LOOPBACK	Loopback()	La información de los canales de entrada se copia en sus respectivos canales de salida.	l_buf_out=l_buf_in;
			
			r_buf_out=r_buf_in;

Tabla 22 - Funciones de procesamiento para los estados de operación del DSP

Por otro lado, utilizando los bloques de datos de las señales de entrada, se procede a las funciones que controlan la funcionalidad de los vúmetros, uno para cada canal

(VumeterR() y VumeterL()). En dichas funciones, utilizando la librería CMSIS, se calcula el valor RMS de cada señal, el cual se puede representar porcentualmente como un valor de 0% a 100%. Este valor debe ser cuantizado para definir la activación de los LEDs del vúmetro de cada canal. Sin embargo, no se puede seguir una cuantización lineal por dos motivos: el primero es que la intención del vúmetro es que el usuario determine márgenes de operación confiables para las señales de entrada, por lo que setear un nivel de saturación al 66%, por ejemplo, haría que se desperdicie mucho rango de operación que no es perjudicial para el sistema. Por otro lado, la percepción auditiva del oído humano sigue una escala que no es lineal sino logarítmica, por lo que resulta conveniente que los márgenes de señal se relacionen también con la percepción del ser humano.

Por este motivo, definiremos los umbrales de activación de los LEDs en base a valores en dB. Esto se representa en la Figura 69. Todos los valores se encuentran normalizados con respecto al valor de voltaje máximo de señal. Esto hace que 0dB represente el máximo valor permisible de RMS (100%) y -inf dB, la ausencia de señal (0%). El valor de saturación fue definido de 0 a -0.5dB, el valor normal de señal, de -0.5 a -15dB y el valor bajo, de -15 a -40dB. Consideramos que la señal es nula si su valor está por debajo de -40dB.

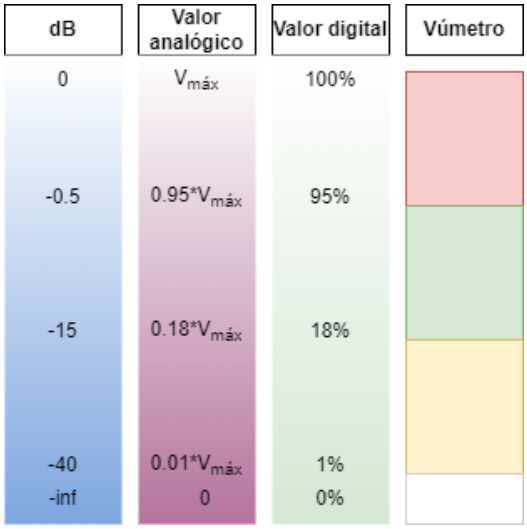


Figura 69 - Niveles de activación de los LEDs del vúmetro

La última funcionalidad del programa es el envío del valor de MSE por interfaz UART cuando se encuentra en el estado de adaptación. Si bien el MSE es calculado cada vez que un nuevo bloque de audio es procesado por el filtro adaptativo, el envío de su valor se realiza de forma asíncrona mediante interrupciones. Cuando la variable de estado pasa a ADAPT, se inicia la trasmisión de datos por UART, donde se indica a la interfaz que trasmita un buffer reservado de 16 bytes que es donde se escribe el valor de MSE. Cada vez que el periférico UART finaliza una trasmisión, dispara un callback que vuelve a iniciar la trasmisión. De esta manera se envía la información solamente cuando el periférico está disponible y no se pierde tiempo haciendo pooling de su estado de trasmisión, tiempo que se aprovecha para el resto de las tareas de procesamiento.

Al igual que para la recepción de datos UART, en la trama de trasmisión se utilizan caracteres de inicio y fin, con la diferencia que el tamaño de trama no es fijo.

Trama UART Tx		
Caracter de inicio	**Comando**	**Caracter de fin**
%	XXX……XXX	#

Tabla 23 - Trama UART Tx

4.1.2.2.2 Sistema de control de usuario

En este apartado describiremos el firmware del bloque Control de usuario, implementado en el módulo ESP32.

Diagrama de estados

La estructura de firmware del bloque está basada en el control de la GUI. A través de ella, el usuario navega por los diferentes modos de operación del sistema, y a partir de ellos, se van generando las salidas del sistema: los comandos UART con destino al bloque DSP, la generación de la señal de audio de ruido blanco, las nuevas pantallas a mostrar, etc. La forma de desarrollar el control de esta interfaz es también mediante una máquina de estados. Esta máquina de estados guarda relación con la presentada para el bloque DSP, pero agrega más sub-estados que tienen que ver con el menú de opciones de la GUI a través de los cuales el usuario debe navegar para controlar el sistema.

El diagrama de estados del bloque se muestra en la Figura 70. Vemos que, por tratarse de un menú de usuario, sigue una distribución jerárquica que parte del estado MAIN MENU, y se desdobla en 3 ramas correspondientes a los estados de adaptación, filtrado y loopback. En cada una de esas ramas existen varios estados que se relacionan con la interfaz de usuario.

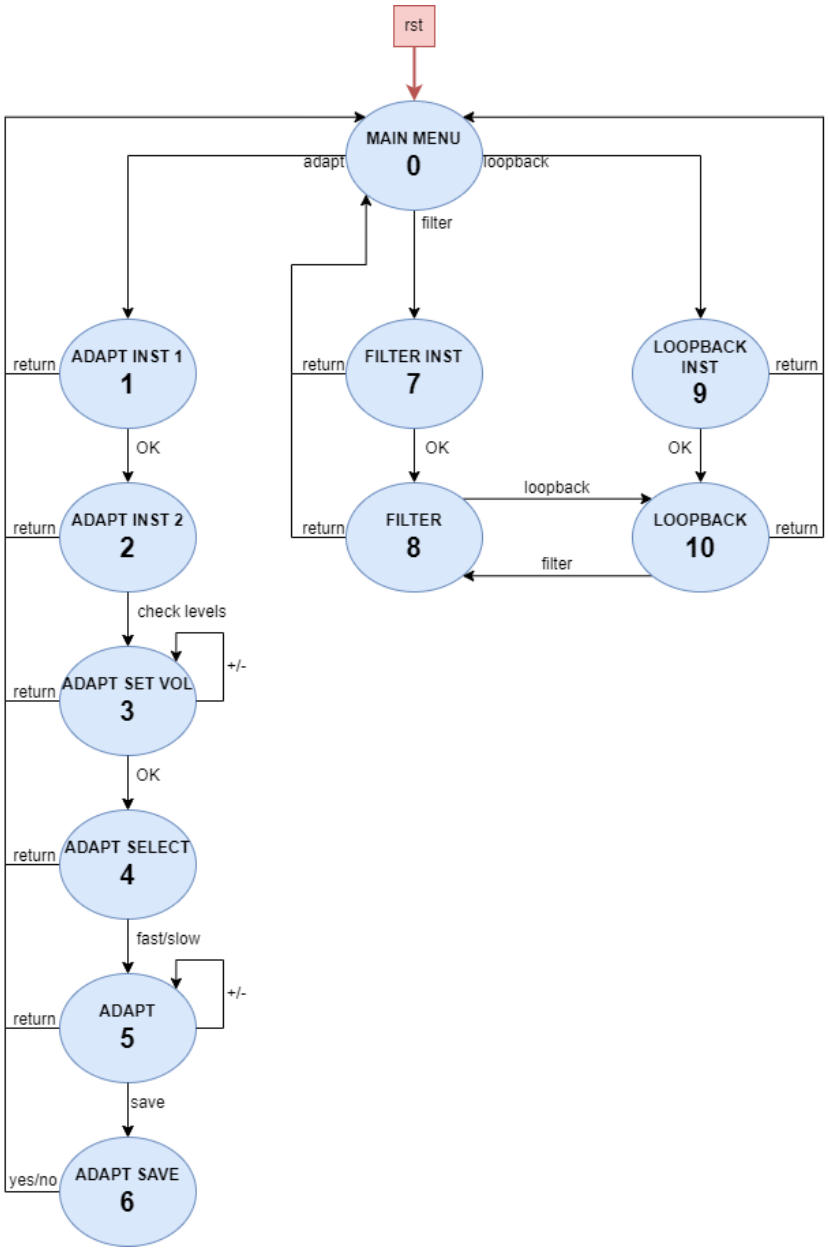


Figura 70 - Diagrama de estados Control de usuario

La variable de control de esta máquina de estados son las señales táctiles recibidas por la interfaz Touchscreen, que indican la selección del usuario dentro de cada estado de la GUI. Cada estado tiene una correspondencia biunívoca con una pantalla mostrada en el display. A su vez, en ciertas transiciones entre estados se envían comandos por la interfaz UART para comunicar el estado al bloque DSP. En algunas se habilita o deshabilita la generación de audio y se permite controlar su volumen. Además, la máquina de estados define la utilización de los datos recibidos por UART. En la Tabla 24 se describen detalladamente las funcionalidades de cada estado, junto con su interacción con los diferentes periféricos del bloque y como se relacionan con los estados del DSP.

Estado	Descripción	Display	Generador de audio I2S	UART Tx	UART Rx
MAIN MENU (0)	Estado base del sistema. Raíz del menú de la GUI. Se corresponde con el bloque DSP en STOP.	Cuando arranca el sistema muestra una pantalla temporal que permite dar tiempo al inicio del sistema. Luego, una pantalla con las opciones Adapt, Filter y Loopback.	Apagado.	-	-
ADAPT INST 1 (1)	Se sigue en STOP. Se indican instrucciones al usuario para el modo de adaptación.	Muestra instrucciones sobre el conexionado de señales. Las opciones son “Return”, para volver a MAIN MENU, y “OK”, para continuar.	Apagado.	-	-
ADAPT INST 2 (2)	Se sigue en STOP. Se indican más instrucciones al usuario para el modo de adaptación.	Muestra instrucciones sobre el control de volumen de la señal de salida. Las opciones son “Return”, para volver, y “OK”, para continuar.	Apagado.	Cuando se presiona “OK” se envía el comando “%LB#” para que el DSP pase a estado LOOPBACK.	-
ADAPT SET VOL (3)	En este momento se pone al DSP en modo LOOPBACK para que podamos escuchar en su salida la señal generada desde este bloque. Se configura el volumen de audio de salida.	Posee botones de “+” y “-” junto con un indicador porcentual de volumen que permite subir y bajar de volumen del ruido generado. También tiene las opciones “Return” y “OK”.	Encendido. En base a los controles de usuario se permite variar el volumen de la señal porcentualmente.	Cuando se presiona “OK” o “Return” se envía el comando “%ST#” para que el DSP pase a estado STOP.	-
ADAPT SELECT (4)	Se vuelve el DSP a STOP y en este estado el usuario selecciona la constante de adaptación (adaptación rápida o lenta).	Posee dos opciones: “Adapt fast” y “Adapt slow” junto con una breve descripción de las características de cada una. También tiene las opciones “Return” y “OK”.	Apagado.	Cuando se presiona “OK” se envía el comando “%AF#” o “%AS#” según lo que se haya seleccionado, para que el DSP pase a estado ADAPT.	-
ADAPT (5)	El sistema se encuentra en modo de adaptación. Se genera el ruido de salida para alimentar las señales de entrada del DSP y en la GUI se da feedback al usuario acerca de la evolución del proceso.	Se muestra una gráfica temporal de la evolución del MSE, donde se observa gráficamente su valor y también se lo indica numéricamente. Además, hay botones de “+” y “-” para modificar el volumen del audio de salida si fuera necesario. También tiene las opciones “Return” y “OK”.	Encendido. En base a los controles de usuario se permite variar el volumen de la señal porcentualmente.	Cuando se presiona “OK” o “Return” se envía el comando “%ST#” para que el DSP pase a estado STOP.	Se recibe el valor de MSE en crudo enviado por el DSP. Se debe procesar dicho valor para que pueda ser graficado por el display.
ADAPT SAVE (6)	Se vuelve el DSP a STOP y en este estado el usuario selecciona si desea almacenar los coeficientes obtenidos	Se consulta al usuario si desea guardar el filtro obtenido y se muestran las opciones “Yes” y	Apagado.	Si se presiona “Yes” se envía el comando “%SV#” para que el DSP	-

	en el proceso de adaptación.	"No". Luego se vuelve al MAIN MENU.		guarde en la FLASH los coeficientes.	
FILTER INST (7)	Se sigue en STOP. Se indican instrucciones al usuario para el modo de filtrado.	Muestra instrucciones sobre el conexionado de señales. Las opciones son "Return", para volver, y "OK", para continuar.	Apagado.	Cuando se presiona "OK" se envía el comando "%FI#" para que el DSP pase a estado FILTER.	-
FILTER (8)	El DSP se encuentra en estado FILTER. Se indica al usuario que está en ese modo y se le permite pasar directamente al modo LOOPBACK.	Muestra un esquema del bloque de filtrado con las señales de entrada y salida. Si se presiona el bloque se pasa al estado LOOPBACK. También tiene la opción "Return" para volver a MAIN MENU.	Apagado.	Cuando se presiona el bloque se envía el comando "%LB#" para que el DSP pase a estado LOOPBACK. Si se presiona "Return" se envía el comando "%ST#" para que el DSP pase a estado STOP.	-
LOOPBACK INST (9)	Se sigue en STOP. Se indican instrucciones al usuario para el modo de loopback.	Muestra instrucciones sobre el conexionado de señales. Las opciones son "Return", para volver, y "OK", para continuar.	Apagado.	Cuando se presiona "OK" se envía el comando "%LB#" para que el DSP pase a estado LOOPBACK.	-
LOOPBACK (10)	El DSP se encuentra en estado LOOPBACK. Se indica al usuario que está en ese modo y se le permite pasar directamente al modo FILTER.	Muestra un esquema del bloque de filtrado con las señales de entrada y salida. Si se presiona el bloque se pasa al estado FILTER. También tiene la opción "Return" para volver a MAIN MENU.	Apagado.	Cuando se presiona el bloque se envía el comando "%FI#" para que el DSP pase a estado FILTER. Si se presiona "Return" se envía el comando "%ST#" para que el DSP pase a estado STOP.	-

Tabla 24 - Estados de operación del bloque Control de usuario

Descripción de funcionamiento

En esta sección describiremos el funcionamiento del código principal del bloque. El código utilizado y todos los detalles de la implementación se encuentran en el Anexo C.

El programa obedece el esquema presentado en la máquina de estados de la Figura 70. Este diagrama de estados se encuentra organizado en base a la GUI que implementa el sistema, cuya variable de entrada son los valores leídos por el Touchscreen, y su salida principal, las ventanas mostradas en el display.

En primer lugar, comentaremos el código que permite implementar estas dos funcionalidades. Tanto el Touchscreen como el display son manejados por la librería TFT_eSPI. Para hacer referencia a la interfaz SPI con ambos dispositivos, la librería utiliza un objeto cuyos métodos permiten el control de ambos periféricos. En este caso el objeto "tft" que hace referencia al sistema touch-display se construye de la manera:

TFT_eSPI tft = TFT_eSPI();

4.1.2.2.3 Manejo del Touchscreen

La interfaz SPI del touchscreen envía constantemente 3 parámetros del dispositivo en forma de valores enteros. Los dos primeros parámetros representan las coordenadas horizontal y vertical en la pantalla, y el tercer parámetro es un valor que representa la presión aplicada en la pantalla.

La forma de tratar las coordenadas de la pantalla es mapear los valores recibidos a pixeles teniendo en cuenta la resolución de la misma y la orientación que se defina para el display. Esto se muestra en la Figura 71.

Por otro lado, en base a un umbral definido para el valor de la presión, se define si la pantalla ha sido presionada.

De esta forma, el programa debe leer constantemente el valor de las variables entregadas por la interfaz, mapear las coordenadas a pixeles y la presión a una variable binaria. En base a esto se definen regiones sensibles para cada una de las pantallas, determinadas por la variable de estado del programa, y se modifica dicha variable si se ha detectado presión en el área demarcada por un determinado ícono o botón de la pantalla. Esto se muestra en la Figura 72. Cuando se detecta que un botón ha sido presionado, antes de ejecutar la próxima función, se ejecuta un retardo de unos milisegundos a modo de antirrebote.

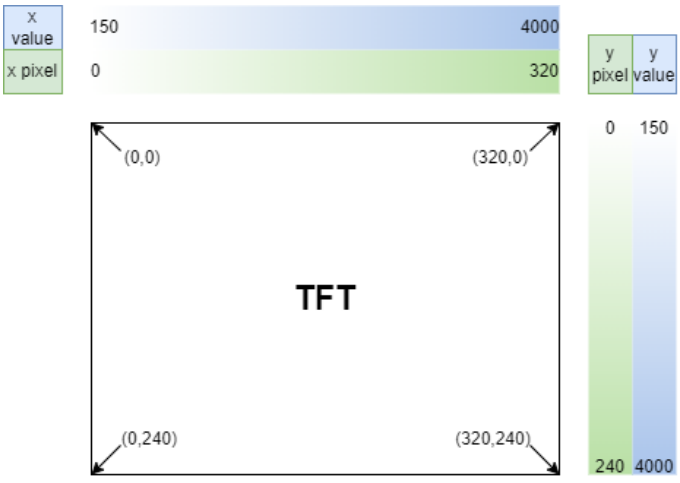


Figura 71 - Mapeo de coordenadas del Touchscreen

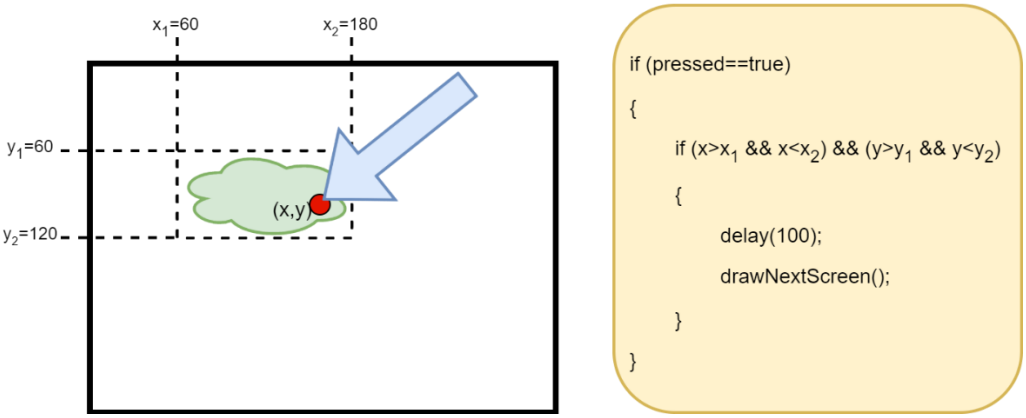


Figura 72 - Área sensible

Manejo del Display

Como vimos, el manejo del touchscreen implica que se esté consultando constantemente el estado del mismo y cuando se detecta que se presionó un botón en una determinada pantalla del sistema, se ejecuta la función que corresponde. Esta función es llamada por única vez luego de que el botón es presionado, y corresponde a la rutina que grafica la siguiente pantalla.

Las diferentes pantallas, que se corresponden con los estados de la máquina de estados, se implementan como funciones, que al ser ejecutadas dibujan nuevamente todos los objetos que deben aparecer en el display. En la Tabla 25 presentamos los métodos más importantes que nos da la librería para desarrollar cada una de estas pantallas.

Función	Descripción
tft.init();	Inicializa la comunicación con el display SPI.
tft.setRotation(rotation);	Configura la orientación de los ejes de la pantalla.
tft.fillScreen(color);	Llena toda la pantalla de un mismo color.
tft.drawPixel(x, y, color);	Dibuja un punto único.
tft.drawLine(x0, y0, x1, y1, color);	Dibuja una línea, indicando sus puntos de inicio y final.
tft.drawRect(uint16_x0, y0, w, h, color); tft.fillRect(x0, y0, w, h, color);	Funciones para dibujar rectángulos, tanto en su versión vacía como rellena. Se especifican sus coordenadas de inicio y su ancho y largo.
tft.drawRoundRect(x0, y0, w, h, radius, color); tft.fillRoundRect(x0, y0, w, h, radius, color);	Similares a las funciones anteriores, pero con los bordes redondeados. Se debe especificar el radio de las esquinas.
tft.drawCircle(x0, y0, r, color); tft.fillCircle(x0, y0, r, color);	Funciones para dibujar círculos, tanto en su versión vacía como rellena. Se especifican sus coordenadas centrales y su radio.
void drawTriangle(x0, y0, x1, y1, x2, y2, color); void fillTriangle(x0, y0, x1, y1, x2, y2, color);	Funciones para dibujar triángulos, tanto en su versión vacía como rellena. Se especifican las tres coordenadas de sus vértices.

tft.setTextDatum(text_align);
tft.setTextColor(main_color, edge_color);
tft.setFreeFont(font);
tft.drawString(str, x, y, GFXFF);
	Funciones para graficar texto en formato String. Se puede configurar el color, la alineación y la fuente del texto.

tft.drawFloat(value, decimals, x, y, GFXFF);	Grafica el valor de una variable float con un cierto número de posiciones decimales.

Tabla 25 - Métodos de la librería TFT_eSPI

Utilizando estas funciones se construyen cada una de las pantallas de la GUI. Cada pantalla está asociada a un estado de la variable de estado.

Señal de audio

Como vimos, la ventaja de utilizar la librería Audio Tools es que prácticamente toda la configuración, generación y ruteo de señales se debe realizar al comienzo del programa. En el bucle principal, solo resta activar y desactivar la señal de salida de ruido blanco y modificar su volumen si el usuario lo indica.

En primer lugar, daremos una breve explicación de la generación de la señal de ruido implementada por la librería. En la Figura 73 se muestra la implementación de la librería de la clase Noise Generator, que es la que hemos utilizado para esta función.

```
template <class T>
class NoiseGenerator : public SoundGenerator<T> {
    public:
        /// the scale defines the max value which is generated
        NoiseGenerator(T amplitude = 32767) {
            this->amplitude = amplitude;
        }
        /// Provides a single sample
        T readSample() {
            return (random(-amplitude, amplitude));
        }
    protected:
        T amplitude;
        // //range : [min, max]
        int random(int min, int max) {
            return min + rand() % (( max + 1 ) - min);
        }
};
```

Figura 73 - Generación de ruido blanco por la librería Audio Tools

La librería genera muestras aleatorias utilizando la función rand(), entre el valor negativo y positivo de la variable de amplitud de la señal definida por el usuario. La función rand() forma parte de la librería math de C/C++ y genera un valor aleatorio con distribución uniforme. La ventaja de esto es que el ruido generado nunca excederá los valores máximos especificados. El hecho de generar valores aleatorios muestra a muestra, a una frecuencia de 48000 muestras por segundo, nos asegura el ancho de banda requerido para la señal.

Para habilitar la salida de audio, debemos llamar al método copier.copy() de la clase StreamCopy como ya explicamos. Esto se realiza periódicamente en la parte del código que corresponde a los estados del sistema ADAPT SET VOL (3) y ADAPT (5) como está indicado en la Tabla 24. A su vez, en ambos estados el usuario dispone de controles para aumentar y bajar el volumen del sonido generado. Para conseguir esto, definimos una variable “volume”, la cual puede tomar valores flotantes entre 0 y 1. Cuando el usuario presiona los botones “+” o “-“ para modificar el volumen, se altera el valor de esta variable. Luego, se la utiliza como parámetro del método audio_volume.setVolume(volume) de la clase VolumeStream, consiguiendo así modificar el volumen en tiempo real de operación.

Comunicación UART

En lo que respecta al envío de datos, la comunicación UART implementa el protocolo de comandos ya explicado para el bloque DSP y también en la Tabla 24. Cuando el usuario presiona un botón que requiere el envío de una orden al bloque DSP, se hace uso del método Serial2.print("%XX#"). Esto trasmite el comando deseado a través de la interfaz.

Con respecto a la recepción de datos, el programa debe prepararse para recibir el valor de MSE de parte del DSP con el formato ya presentado en la Tabla 23. Cuando se pasa al estado ADAPT se llama periódicamente a la función readSerial, encargada de leer el puerto Rx de UART y limpiar la trama para obtener el valor de MSE.

Este valor de MSE arriba al microcontrolador aproximadamente cada 5 milisegundos, y es el valor "crudo" de la métrica calculado por el DSP en un bloque de 512 muestras de la señal de error. De acuerdo con lo expuesto en la sección 0, en la Figura 32, este tamaño de bloque genera una señal de alta varianza. Esta señal debe ser mostrada en la pantalla de la GUI, pero si se graficase de esta manera no sería una curva adecuada puesto que tendría mucha variación y no sería claro para el usuario.

Sumado a este problema, se tiene la situación de que, si bien sabemos que el MSE tiende a cero, no se conoce el valor máximo que puede tomar en el comienzo del proceso. Esto obligaría a que se deba ajustar continuamente la escala de la gráfica, dando menos claridad aún a la representación.

Para resolver estos problemas, se proponen una serie de transformaciones a la señal de MSE recibida para que su representación gráfica sea visualmente atractiva y aporte información de forma simple y clara.

La primera de ellas es la utilización de un filtro de suavizado de los datos de entrada. La intención es filtrar la señal mediante un filtro pasabajos, o de suavizado, tal que la curva de MSE siga un comportamiento suave y nos dé una gráfica más limpia. El filtro utilizado es un filtro recursivo de un polo, o también llamado filtro de media móvil exponencial.

El filtro de media móvil exponencial es un filtro pasabajos discreto de respuesta infinita al impulso (IIR). Da mayor importancia a los datos recientes al descontar los datos antiguos de manera exponencial y se comporta de manera similar al filtro RC pasabajos de primer orden.

La ecuación en diferencias del filtro es:

$$y[n] = \alpha\, y[n-1] + (1-\alpha)x[n] \tag{50}$$

Donde y[n] es la señal de salida del filtro; x[n], la señal de entrada y α es una constante cuyo valor está entre 0 y 1 y fija la frecuencia de corte del filtro, o el suavizado que este introduce.

Se puede observar que el cálculo no requiere el almacenamiento de valores pasados de la señal de entrada, solo el valor anterior de la salida, lo que hace que este filtro sea

amigable en términos de memoria y procesamiento. Solo se necesitan una suma, una resta y dos multiplicaciones.

La constante α determina qué tan agresivo es el filtro. Puede variar entre 0 y 1 (inclusive). A medida que "α" se acerca a 1, el filtro se vuelve más y más agresivo, hasta que en α = 1, la entrada no tiene efecto en la salida. A medida que α se acerca a 0, el filtro deja pasar más de la entrada sin filtrar, hasta que en α = 0, el filtro no filtra en absoluto (la entrada pasa directamente a la salida).

El filtro se llama exponencial porque la contribución ponderada de las entradas anteriores disminuye exponencialmente a medida que la entrada evoluciona en el tiempo. En la Figura 74 se aprecia el efecto de α en el desempeño temporal del filtro. Se puede notar que a medida que se incrementa el valor de α, el filtro introduce mayor retraso a la señal de salida.

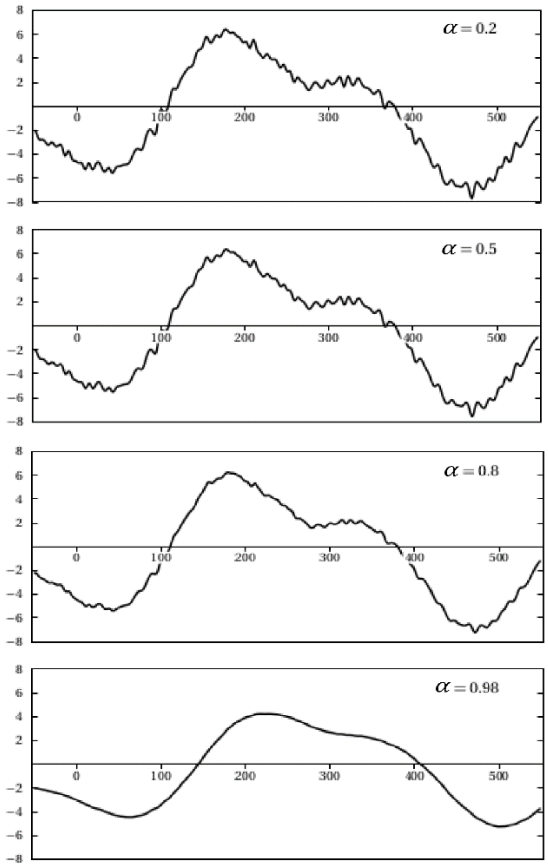


Figura 74 - Efecto de α en el suavizado del filtro

La función de transferencia del filtro está dada por la ecuación:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1-\alpha}{1-\alpha z^{-1}} \tag{51}$$

En la Figura 75 se muestra la respuesta en frecuencia para esta función de transferencia con un α=0.9. Se puede observar que claramente es un filtro pasabajos y que su respuesta en fase es no lineal.

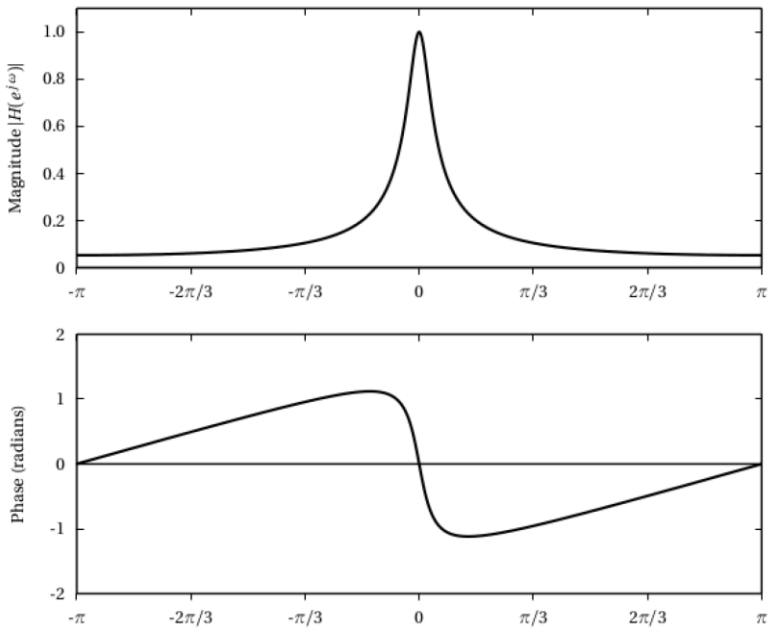


Figura 75 - Respuesta en frecuencia del filtro de media móvil exponencial

Mediante simulación, generamos la señal estimada de MSE que arriba al microcontrolador, la cual se muestra en la Figura 76.

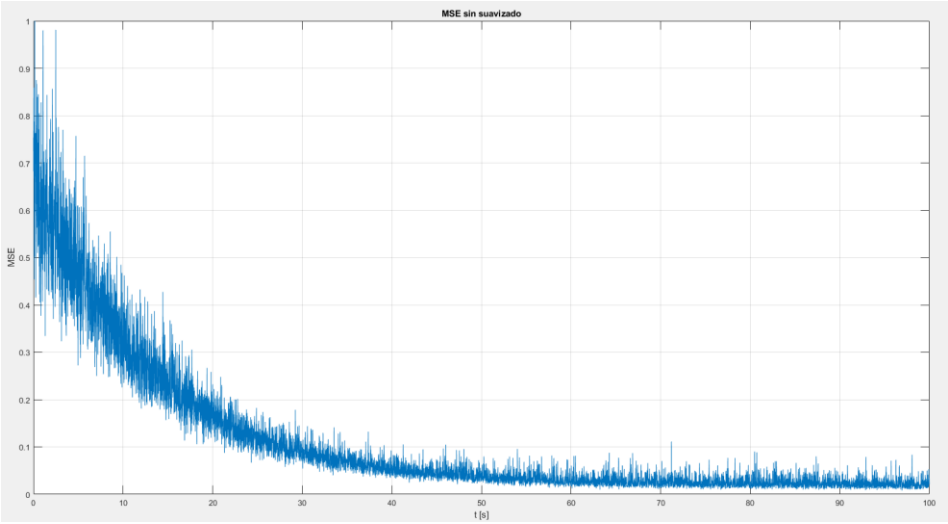


Figura 76 - Señal de MSE sin filtrar

En la Figura 77 se muestra la señal luego del filtro. Vemos que presenta un pequeño retardo pero que a los fines prácticos no es importante y que ahora sí se obtiene una curva suave que permite una representación gráfica más acorde a la buscada.

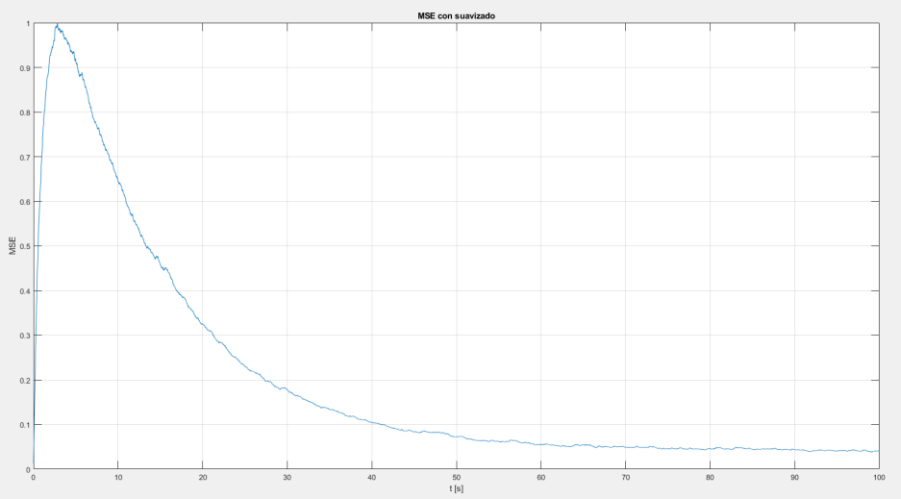


Figura 77 - Señal de MSE con filtrado

Como bien se dijo, el hecho de desconocer el valor máximo del MSE obligaría a que la gráfica de su valor sea escalada constantemente. Para evitar este inconveniente la solución que se tomó es graficar su valor porcentual respecto a un máximo, el cual supondremos que ocurre en los primeros segundos del proceso de adaptación. Esto se puede observar en la Figura 77. Vemos que el máximo de la señal se produce aproximadamente a los 5 segundos. El programa almacena el valor máximo recibido y grafica la señal:

$$MSE_{\%}[n] = \frac{MSE[n]}{MSE_{max}} \cdot 100\% \tag{52}$$

De esta manera, en los primeros segundos de adaptación arranca en el tope de la ventana del gráfico puesto que en este tiempo el máximo se encuentra en constante actualización. Luego, se establece el máximo en un valor constante y la gráfica porcentual se comporta de la manera prevista. Esto se observa en la Figura 78. Esta señal es la que efectivamente utilizará el microcontrolador para elaborar la gráfica de la métrica.

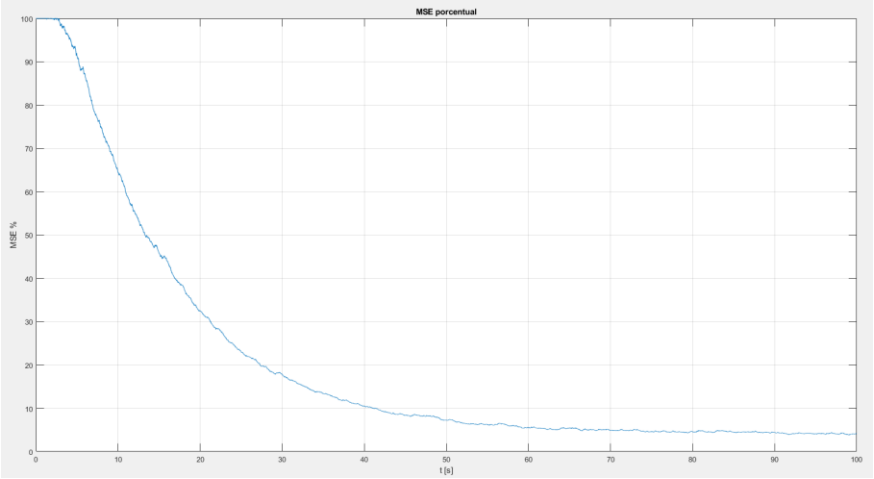


Figura 78 - Señal de MSE porcentual

4.1.3 Implementación

4.1.3.1 Diseño del circuito

Para el diseño del esquemático del circuito, y luego el PCB, se utilizó el software KiCad. El mismo permite realizar el circuito creando e importando los componentes utilizados y luego transfiriendo dicho esquemático a la interfaz que permite el diseño del PCB.

En primer lugar, se realizó el circuito esquemático de todo el sistema, donde se debió crear algunos componentes para los que no se disponían las librerías correspondientes. El esquemático completo del circuito se muestra en el Anexo D.

Luego se procedió a realizar el PCB de la placa. En este punto, debemos mencionar que las señales de audio de entrada y salida de los conversores Pmod I2S2 son entregadas en fichas estéreo hembra Jack de 3.5mm. Como para nuestra aplicación dichas señales se requieren por separado, se fabricaron conectores splitter para cada una de las señales estéreo. Cada uno de ellos posee un conector de audio Jack estéreo macho de 3.5mm en un extremo, mientras que en el otro se tienen dos conectores Jack mono hembra de 6.35 mm por separado, uno para la señal de cada canal. Estos últimos serán los conectores que se dispondrán en el chasis del dispositivo.

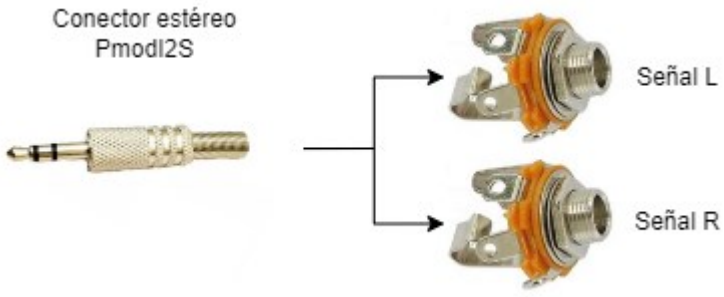


Figura 79 - Conector splitter

Por otro lado, el display y los LEDs también deben ubicarse en el frente del chasis. Por ello, en el PCB se dispondrán de conectores para cables flexibles que unan estos

dispositivos. La alimentación del circuito parte de la interfaz USB de la placa STM32F4DISCOVERY, la cual alimenta el resto de los componentes.

El PCB fue diseñado en una capa, con la intención de ser fabricado mediante el método de planchado sobre una placa de cobre. En el Anexo E se muestra el diseño final del PCB. El software permite además importar los modelos 3D de los componentes y generar la vista tridimensional del diseño, la cual será utilizada para el diseño del chasis.

4.1.3.2 Diseño del chasis

Para el diseño del chasis se utilizó el software Autodesk Fusion 360. Se diseñó un chasis que contemple la sujeción de la placa en su base. Además, dispone de una apertura para el conector de alimentación. En su parte frontal posee los 6 conectores de las señales de entrada y salida del circuito, las cuales poseen etiquetas de cada una de ellas. Por último, en la tapa posee un hueco y sujeciones para los LEDs de los vúmetros y el display táctil.

La realización física del chasis se hizo mediante impresión 3D. En la Figura 80 se muestra el acabado final del dispositivo con el hardware, conectores y electrónica integrada.

Figura 80 - Dispositivo final

4.1.3.3 Validación y pruebas de funcionamiento

A continuación, desarrollaremos una plataforma y una serie de ensayos para verificar el correcto funcionamiento del dispositivo desarrollado. Para ello tomaremos como sistema desconocido un amplificador de guitarra, e intentaremos caracterizar su funcionamiento utilizando los métodos tradicionales y comparándolos con lo obtenido por nuestro emulador.

4.1.3.3.1 Sistema desconocido

El amplificador que utilizaremos como sistema desconocido para validar el dispositivo es el Marshall MG30FX.

Realizaremos el ensayo del mismo con un micrófono que toma la señal de salida del parlante del amplificador, en cuyo caso se considera como sistema desconocido el conjunto "preamplificador-gabinete-parlante-sala-micrófono". Esto se muestra en el diagrama de la Figura 81.

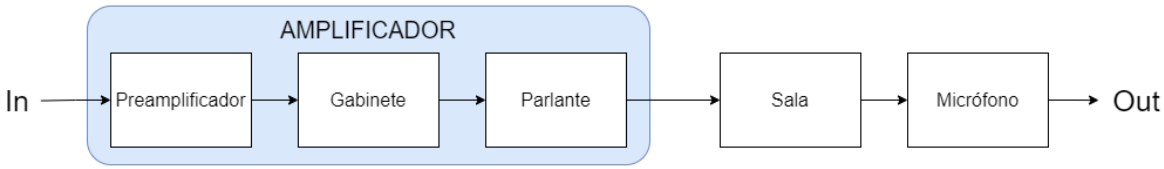


Figura 81 - Configuración para el sistema desconocido en la verificación

Puesto que para el ensayo el sistema desconocido se compone de bloques externos al amplificador (sala y micrófono), definiremos estos elementos a fin de aportar rigurosidad a la medición.

La sala donde realizaremos las mediciones es una habitación pequeña, de piso de madera, y paredes y techo de yeso. Cuenta con un armario y puertas de madera y una ventana de vidrio y marco de aluminio. A pesar de ser una habitación sin tratamiento acústico, debido a su pequeño tamaño no presenta una reverberación desmedida, por lo que podemos considerarla un recinto estándar o normal para un ensayo de estas características, con una influencia moderada o baja en el resultado del modelo que obtendremos.

Por otro lado, como micrófono de medición utilizaremos el Shure SM57. Este micrófono es del tipo cardioide, unidireccional y de baja impedancia, comúnmente usado en el refuerzo de audio en vivo y en grabaciones de estudio. El SM57 es popular entre los músicos debido a su construcción robusta y capacidad de trabajar bien con instrumentos de altas frecuencias, como por ejemplo percusiones y guitarras acústicas. El diseño del cardioide reduce los sonidos de fondo indeseables y la generación de feedback acústico.

4.1.3.3.2 Ensayos realizados

Para los ensayos de medición de la respuesta de los sistemas involucrados se utilizó el software REW (Room EQ Wizard).

Para nuestro caso, la herramienta nos permite realizar la medición de la respuesta en frecuencia de un sistema acústico. Para ello es necesario contar con una interfaz de audio a través de la cual podamos reproducir y grabar señales de audio de alta calidad y baja latencia. En nuestro caso utilizaremos la placa Focusrite Scarlett 2i2.

El procedimiento para realizar este tipo de mediciones consiste en generar un barrido senoidal de frecuencias, el cual es utilizado como entrada al sistema a evaluar. Con un micrófono se toma su salida y se realimenta al software nuevamente. Con esta información, REW puede calcular la respuesta en frecuencia y la respuesta al impulso del sistema. Otra funcionalidad que utilizaremos es la de exportar los datos de las respuestas al impulso obtenidas para su utilización en otras plataformas.

Respuesta del sistema desconocido

En primer lugar, obtendremos la respuesta "original" del amplificador para luego compararla con el modelo que obtendremos con nuestro dispositivo. Para ello realizamos las conexiones según lo descripto anteriormente, las cuales se muestran en la Figura 82. Los controles de ecualización del amplificador son los que se observan en la imagen y corresponden a la posición media de los mismos.

Figura 82 - Medición del amplificador

Una vez realizada la medición, el software nos arroja la respuesta en frecuencia obtenida, la cual se muestra en la Figura 83. También podemos visualizar la respuesta al impulso, representada en la Figura 84.

Figura 83 - Respuesta en frecuencia del amplificador

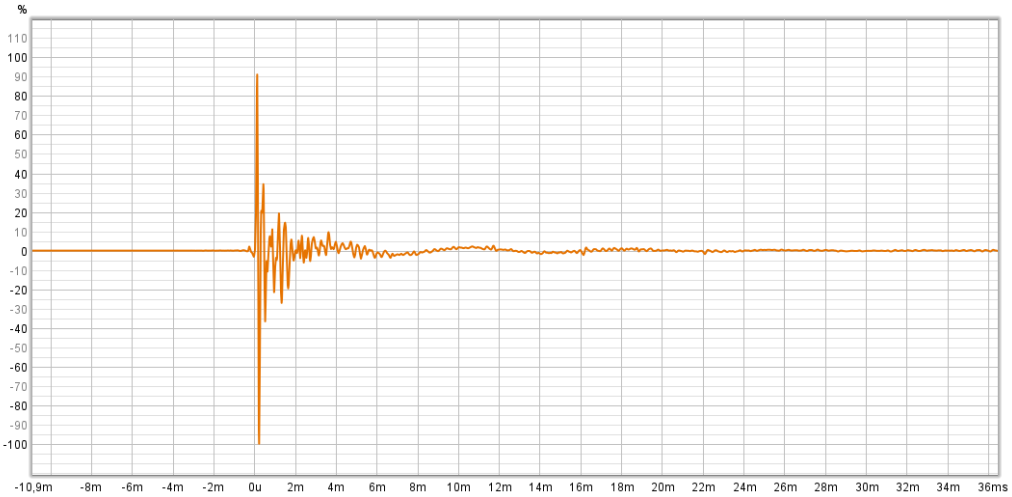


Figura 84 - Respuesta al impulso del amplificador

Modelo obtenido

Seguidamente, procederemos a utilizar el dispositivo construido para obtener el modelo correspondiente al amplificador. Detallaremos primero el procedimiento de adaptación, y luego mediremos la respuesta del dispositivo con la emulación obtenida.

Proceso de adaptación

Para efectuar la adaptación, debemos realizar el conexionado ya descripto en la sección Diagrama en bloques, el cual se esquematiza en la Figura 85. Se pueden conectar opcionalmente unos auriculares en la salida de error para monitorear auditivamente el avance de la adaptación.

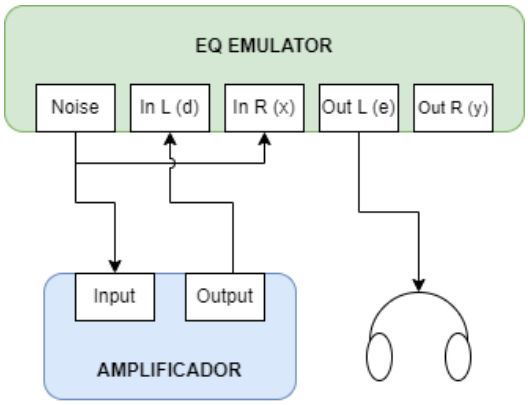


Figura 85 - Conexionado para adaptación (esquema)

Luego, se deben ajustar los volúmenes de la señal de ruido y de ganancia del amplificador a fin de que los niveles de ambas señales de entrada sean similares y acordes al volumen óptimo. Esto se muestra en la Figura 86.

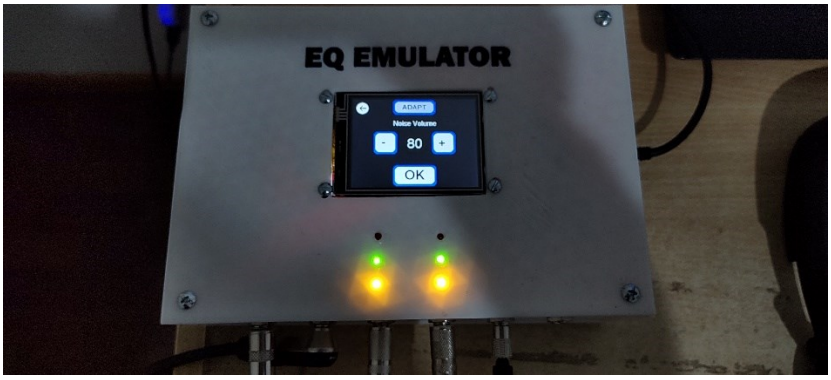


Figura 86 - Nivelación de volúmenes

Por último, se da comienzo al proceso de adaptación. Se testeó tanto el modo de adaptación rápido (μ alto) como el modo lento (μ bajo). En el primer caso se logró converger a un valor de MSE cercano al 20% en menos de 20 segundos, mientras que en el segundo, el tiempo de convergencia fue cercano a los 50 segundos, pero con un valor de MSE rondando en el 10%.

Figura 87 - Proceso de adaptación

Respuesta del dispositivo emulador

Una vez que se obtuvo el modelo y el mismo fue guardado, mediremos la respuesta del dispositivo utilizando el mismo método que para el amplificador. De esta manera tendremos una misma base de comparación para evaluar la adaptación obtenida y determinar si la misma cumple con las estimaciones realizadas en la etapa de simulación.

El conexionado se muestra en la Figura 88

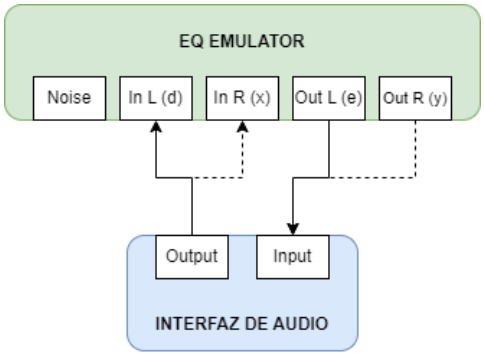


Figura 88 - Conexionado para medición (esquema)

Una vez realizada la medición, el software nos arroja la respuesta en frecuencia obtenida, la cual se muestra en la Figura 89. También podemos visualizar la respuesta al impulso, representada en la Figura 90.

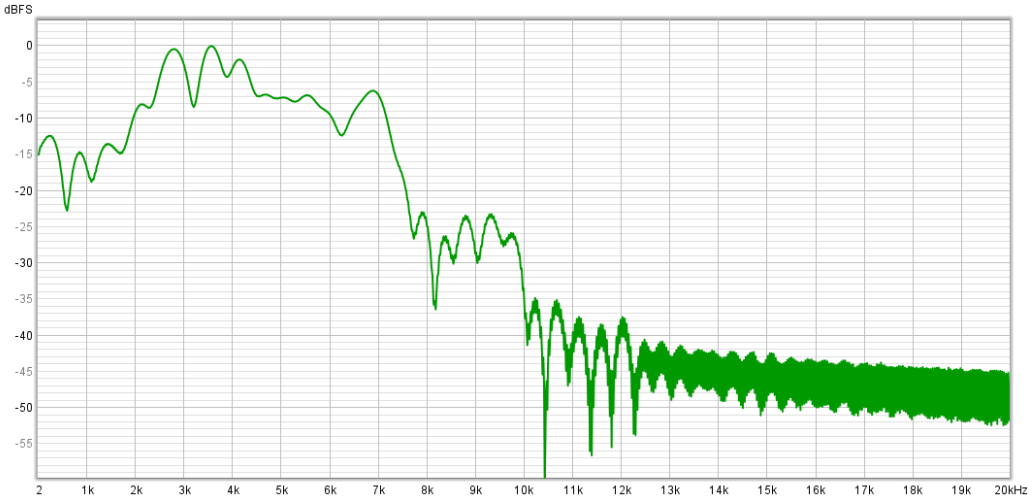


Figura 89 - Respuesta en frecuencia del modelo

Figura 90 - Respuesta al impulso del modelo

4.1.3.3.3 Comparativa de resultados

Habiendo obtenido las respuestas tanto del amplificador como de su emulación por el mismo método de medición, utilizaremos estos resultados para realizar una comparación que permita contrastar el funcionamiento del dispositivo desarrollado contra las simulaciones realizadas previamente y verificar que el mismo funcione de acuerdo a lo esperado.

Cabe aclarar que la prueba realizada cuenta con una complejidad avanzada para la operación del dispositivo, ya que en ella se incluyó no solamente un amplificador como elemento electroacústico, sino una sala sin tratamiento acústico y un micrófono cuya coloración no es despreciable en la cadena de audio. Esto hace que el ensayo contemple una serie de casos “límite” para la utilización del desarrollo realizado.

Para realizar la comparación, exportamos las respuestas al impulso del amplificador y del emulador del software REW para su utilización y análisis en MATLAB. En la Figura 91 se grafican en superposición ambas respuestas obtenidas.

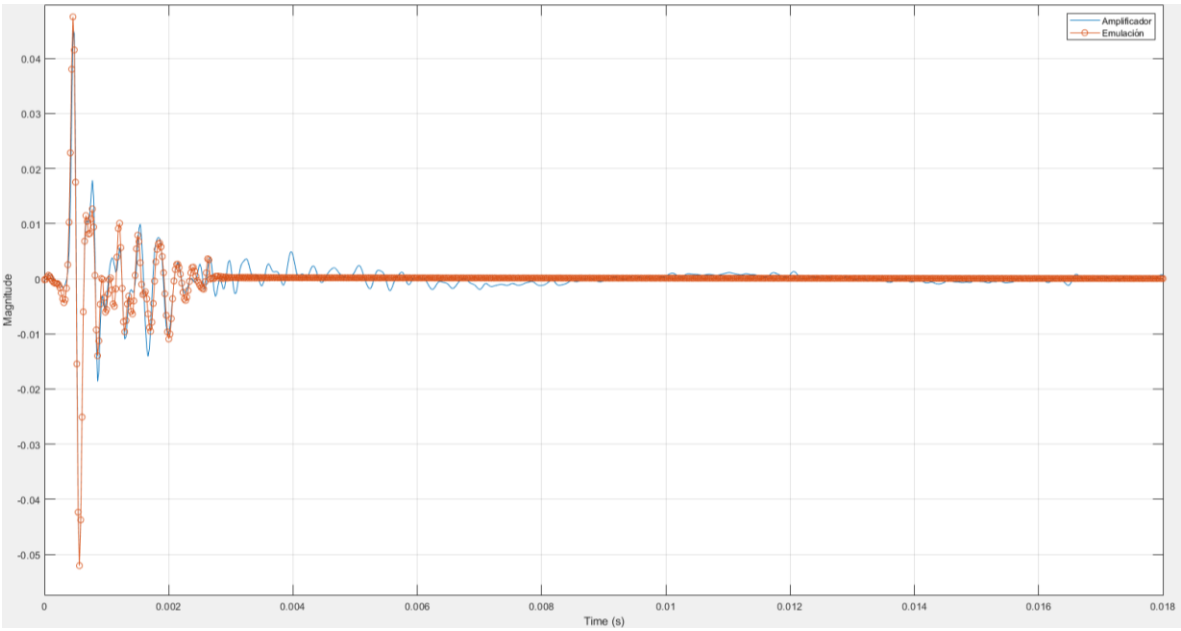


Figura 91 - Respuesta al impulso amplificador vs emulación

Se puede observar, al igual que en las simulaciones, la correspondencia entre ambas respuestas en los primeros instantes. También se aprecia el efecto de truncado de la respuesta de la emulación. Recordando que el filtro cuenta con 128 coeficientes y que la frecuencia de muestreo es de 48000 muestras por segundo, la duración de la respuesta al impulso del emulador es de:

$$T_{RI} = \frac{N}{F_s} = 2.67\ ms \tag{53}$$

Que es precisamente lo que se observa en la gráfica, con el corte abrupto de la respuesta en ese tiempo. Esto corrobora los resultados obtenidos en la simulación para la respuesta temporal.

En la Figura 92 se muestra la respuesta en frecuencia del amplificador y de la emulación obtenida. Vemos que se consigue una buena representación de la respuesta hasta 13kHz aproximadamente y aparece el mismo piso de ruido observado en las simulaciones de algunos sistemas. Esto puede deberse a que en la medición se incluyeron los efectos de la sala y el micrófono que introducen variaciones en alta frecuencia de baja magnitud que el dispositivo no puede seguir. Sin embargo, a nivel auditivo estas características al encontrarse debajo de los -25dB son prácticamente imperceptibles. El resultado obtenido es muy positivo dada la complejidad de la medición y el gran número de factores de ruido y errores que no se han tenido en cuenta a la hora de efectuarla. Sumado a esto, a los efectos prácticos la emulación en la escucha reproduce muy fielmente las características del amplificador, por lo que se puede decir que las inferencias enunciadas en el apartado Conclusiones del algoritmo son válidas para el dispositivo implementado.

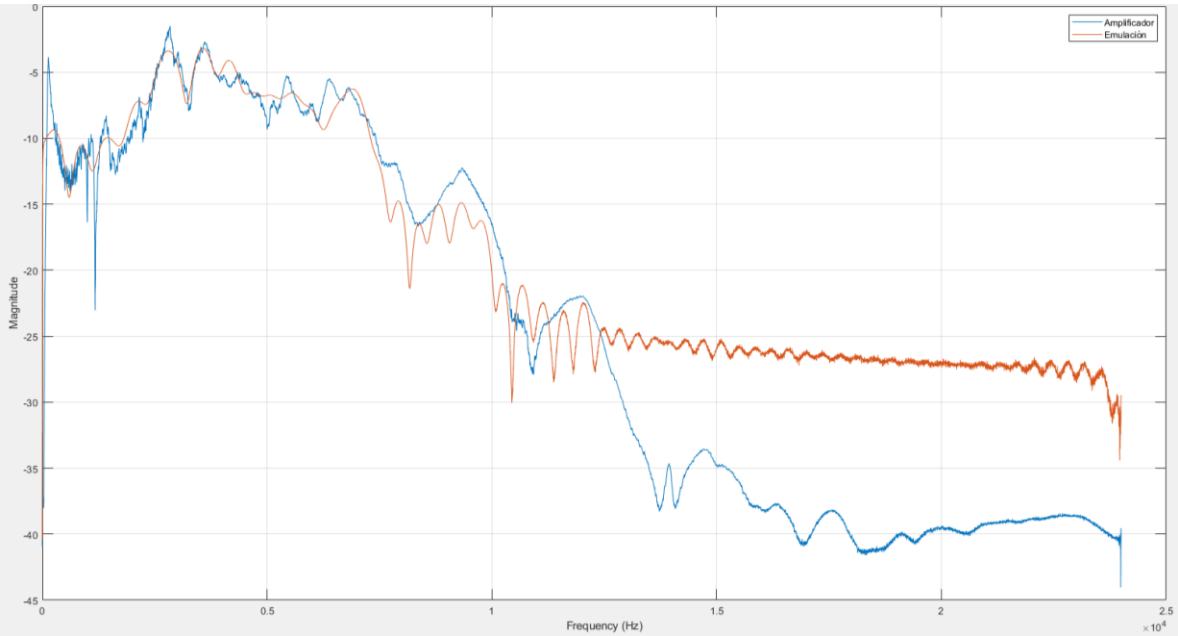


Figura 92 - Respuesta en frecuencia amplificador vs emulación

4.2 FACTIBILIDAD ECONÓMICA

Para la realización del análisis de factibilidad económica se ha supuesto que el producto en cuestión será vendido con el fin de determinar su impacto en el mercado y cuáles serían las ganancias esperadas y los tiempos de recuperación de ser el caso. Se han calculado, de forma estimativa, los gastos y costos de realización de este proyecto. Todos los valores aquí expuestos corresponden a dólares estadounidenses (U$D).

El análisis de factibilidad económica propiamente dicho se ha dividido en dos partes.

La primera es para un VAN igual a cero en un periodo de 5 años de producción. El objetivo de este primer análisis es determinar la cantidad y precio de venta de los dispositivos necesarios para recuperar la inversión inicial al finalizar el quinto año.

La segunda parte considera el mismo periodo de análisis, pero la cantidad de dispositivos fabricados y vendidos en cada año se fija en base a las posibilidades de producción, es decir, en base a la cantidad que efectivamente puede producirse año a año con la consideraciones y suposiciones tomadas.

4.2.1 Costos y precio de venta

En primer término, se analizan los costos de fabricación de una unidad de EQ Emulator, los cuales se detallan en la Tabla 26.

PRODUCTO	CANTIDAD	COSTO	TOTAL
EQ EMULATOR			
Módulo STM32	1	$ 20,74	$ 20,74
Módulo ESP32	1	$ 9,80	$ 9,80
Pmod I2S2	2	$ 24,99	$ 49,98
Pantalla táctil	1	$ 20,90	$ 20,90
Led	6	$ 0,10	$ 0,60
Resistencia	6	$ 0,10	$ 0,60
Conector plug	6	$ 0,80	$ 4,80
Conectores PCB	1	$ 7,00	$ 7,00
Placa de cobre	1	$ 4,00	$ 4,00
Conexionado adicional	1	$ 2,00	$ 2,00
Carcasa	1	$ 10,00	$ 10,00
Sujeción	1	$ 2,00	$ 2,00
Cable/fuente alimentación	1	$ 5,00	$ 5,00
TOTAL			**$ 137,42**

Tabla 26 - Costos de fabricación

Realizando una comparativa con precios de otras alternativas comerciales, se define un precio de venta de U$D 200, el cual es inferior a todos los productos de la competencia y permite un margen de ganancia de U$D 62.58, que corresponde a un 32%.

Como parte de la inversión inicial del proyecto, se considera la compra de herramientas y equipos electrónicos necesarios para la fabricación y puesta a punto de los dispositivos. Las herramientas tienen una vida útil de 5 años, dando una inversión en las mismas de U$D 230, mientras que los equipos electrónicos, una vida útil de 3 años, lo que obliga a adquirir nuevas unidades en el tercer año del proyecto. El detalle de estos ítems se presenta en la Tabla 27. La depreciación de estos ítems se realizó siguiendo el método de la línea recta y se muestra en la Tabla 28.

RECURSOS			
Ítem	**CANTIDAD**	**COSTO**	**TOTAL**
Herramientas			
Herramientas varias	1	$ 30,00	$ 30,00
Micrófono	1	$ 50,00	$ 50,00
Placa de grabación	1	$ 150,00	$ 150,00
SUB TOTAL			**$ 230,00**
Equipos electrónicos			
Computadora	1	$ 350,00	$ 350,00
Componentes de prueba	1	$ 20,00	$ 20,00
SUB TOTAL			**$ 370,00**
TOTAL			**$ 600,00**

Tabla 27 - Inversión inicial

TABLA DE DEPRECIACIONES						
DEPRECIACIÓN HERRAMIENTAS						
Vida útil	**5**					
ITEM	**0**	**1**	**2**	**3**	**4**	**5**
Valor de adquisición	$ 230					
Gasto depreciación		$ 46	$ 46	$ 46	$ 46	$ 46
Depreciación acumulada		$ 46	$ 92	$ 138	$ 184	$ 230
Valor en libros		$ 184	$ 138	$ 92	$ 46	$ -
EQUIPOS ELECTRÓNICOS						
Vida útil	**3**					
ITEM	**0**	**1**	**2**	**3**	**4**	**5**
Valor de adquisición	$ 370					
Gasto depreciación		$ 123	$ 123	$ 123		
Depreciación acumulada		$ 123	$ 247	$ 370		
Valor en libros		$ 247	$ 123	$ -		
EQUIPOS ELECTRÓNICOS						
Vida útil	**3**					
ITEM	**0**	**1**	**2**	**3**	**4**	**5**
Valor de adquisición				$ 370		
Gasto depreciación					$ 123	$ 123
Depreciación acumulada					$ 123	$ 247
Valor en libros					$ 247	$ 123
TOTAL DE DEPRECIACIONES		**$ 169**	**$ 169**	**$ 169**	**$ 169**	**$ 169**

Tabla 28 - Depreciaciones

Con respecto a los costos fijos, se tendrá en cuenta el gasto en servicios y alquiler de un local, lo que se detalla en la Tabla 29.

COSTOS FIJOS	
Ítem	**COSTO**
Servicios	
Luz	$ 2,00
Agua	$ 1,50
Internet	$ 10,00
TOTAL	**$ 13,50**
Alquiler local	
Mensual	$ 50,00
Anual	$ 600,00
TOTAL	**$ 600,00**
TOTAL	

Tabla 29 - Costos fijos

4.2.2 Análisis para VAN = 0

Para este caso, se considerará el mínimo de producción posible que asegure un VAN lo más cercano a cero. En primer lugar, consideraremos una producción inicial en el primer año del proyecto de 3 unidades para obtener el capital de trabajo por el método del máximo acumulado. Luego, se procede a realizar el flujo completo de caja para toda la duración del proyecto, ajustando iterativamente la producción anual en cada año hasta llegar a un VAN cercano a cero.

Para el capital de trabajo se considera que el cobro de la producción se divide en un 50% al realizar la compra del producto y el otro 50% 30 días después. Además, el equipo se produce en el mes donde se ejecuta la compra. El cálculo del mismo se muestra en la Tabla 30.

INGRESO MENSUAL AÑO UNO		Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre
Precio de venta		$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00
Unidades vendidas		0	0	1	1	1	0	0	0	0	0	0	0
Total de ventas		$0,00	$0,00	$200,00	$200,00	$200,00	$0,00	$0,00	$0,00	$0,00	$0,00	$0,00	$0,00
Realización del cobro													
Cobro al momento de la compra	50 %	$0,00	$0,00	$100,00	$100,00	$100,00	$0,00	$0,00	$0,00	$0,00	$0,00	$0,00	$0,00
Cobro 30 días después de la compra	50 %	$0,00	$0,00	$0,00	$100,00	$100,00	$100,00	$0,00	$0,00	$0,00	$0,00	$0,00	$0,00
Ingreso total		**0**	**0**	**100**	**200**	**200**	**100**	**0**	**0**	**0**	**0**	**0**	**0**
PRODUCCIÓN													
Stock		0	0	0	0	0	0	0	0	0	0	0	0
Ventas		0	0	1	1	1	0	0	0	0	0	0	0
Producción Mensual		0	0	1	1	1	0	0	0	0	0	0	0
EGRESOS													
Costos Fijos		$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50
Costos de Fabricación		$0,00	$0,00	$137,42	$137,42	$137,42	$0,00	$0,00	$0,00	$0,00	$0,00	$0,00	$0,00
Egreso total		**$63,50**	**$63,50**	**$200,92**	**$200,92**	**$200,92**	**$63,50**	**$63,50**	**$63,50**	**$63,50**	**$63,50**	**$63,50**	**$63,50**
CAPITAL DE TRABAJO													
Ingresos		0	0	100	200	200	100	0	0	0	0	0	0
Egresos		$63,50	$63,50	$200,92	$200,92	$200,92	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50
Acumulado		-$63,50	-$127,00	-$227,92	-$228,84	-$229,76	-$193,26	-$256,76	-$320,26	-$383,76	-$447,26	-$510,76	-$574,26
Máximo Acumulado		**-$574,26**											

Tabla 30 - Capital de trabajo (VAN=0)

Se observa que el valor máximo acumulado necesario es de U$D 574,26. Se considerará un valor de capital de trabajo para el flujo de caja con un margen extra del 20%.

CAPITAL DE TRABAJO SIN MARGEN		**$574,26**
CAPITAL DE TRABAJO CON MARGEN	**1,2**	**$689,11**

Tabla 31 - Margen para capital de trabajo (VAN=0)

Una vez determinado el capital de trabajo necesario se realizó el flujo de caja para los 5 años de duración del proyecto y considerando una tasa de interés del 12%. Aquí se consideró un aumento de la producción creciente año a año y se ajustaron estas cantidades hasta arribar a un valor de VAN cercano a cero. Iterando se llegó a una cantidad mínima de unidades por año que deben ser vendidas para asegurar la rentabilidad del proyecto. Estas corresponden a 3, 5, 7, 9 y 10 unidades producidas y vendidas para cada uno de los 5 años de duración del proyecto. El flujo de caja se muestra en la Tabla 32.

Flujo de caja	0	1	2	3	4	5
Unidades vendidas		3	5	7	9	10
Ingreso		$ 600	$ 1.000	$ 1.400	$ 1.800	$ 2.000
Precio unitario		$ 200	$ 200	$ 200	$ 200	$ 200
Costos fijos						
Total costos fijos		$ 762	$ 762	$ 762	$ 762	$ 762
Costos variables						
Costo de venta del producto		$ 137	$ 137	$ 137	$ 137	$ 137
Herramientas e instrumentos	$ 230	$ -	$ -	$ -	$ -	$ -
Equipos electrónicos	$ 370	$ -	$ -	$ 370	$ -	$ -
Costos variables totales		$ 137	$ 137	$ 507	$ 137	$ 137
Depreciación herramientas		$ 169	$ 169	$ 169	$ 169	$ 169
Utilidad antes de impuestos		$ -469	$ -69	$ -39	$ 731	$ 931
Impuesto a las ganancias (35%)		$ -164	$ -24	$ -14	$ 256	$ 326
Utilidad después de impuestos		$ -305	$ -45	$ -25	$ 475	$ 605
Depreciación herramientas		$ 169	$ 169	$ 169	$ 169	$ 169
Inversión inicial						
Herramientas y equipos electrónicos	$ -600					
Capital de trabajo	$ -689					
Recuperación capital de trabajo						$ 689
Valor de desecho						$ -
Flujo del proyecto	**$ -1.289**	**$ -135**	**$ 125**	**$ 144**	**$ 645**	**$ 1.464**

Tabla 32 - Flujo de caja (VAN=0)

Tasa de interés	12%
VAN	**$ 32**
TIR	**13%**

Tabla 33 - VAN y TIR (caso 1)

4.2.3 Análisis para VAN > 0

En este otro análisis se considerarán las posibilidades reales de producción para analizar la rentabilidad real del proyecto llevado a cabo en estos términos. Para ello definiremos la capacidad de producción en unidades para cada año del proyecto.

Inicialmente supondremos que la producción y ventas lo realizará exclusivamente el líder del proyecto por los primeros dos años. Esto a fin de no aumentar los costos de contratación en el inicio del proyecto. A partir del tercer año se considera la contratación de un empleado extra que permita aumentar la capacidad de producción.

En primer lugar, se definen las horas necesarias para la fabricación de una unidad, lo cual se representa en la Tabla 34.

HORAS EQ EMULATOR	
FASE	HORAS
Armado de placas	20
Ensamblaje	20
Programación	5
Verificaciones y puesta en marcha	5
TOTAL	**50**

Tabla 34 - Horas de fabricación

Teniendo en cuenta media jornada de trabajo por empleado (4 horas diarias) y un promedio de 20 días hábiles por mes, se obtienen la cantidad de horas de trabajo mensuales y anuales. Esto permite calcular la capacidad de producción teórica de la empresa por año, a la cual se la ajusta y redondea con un margen del 60% para obtener un número más conservador que contemple imprevistos en la fabricación y producción.

Año	1	2	3	4	5
Cantidad de empleados	1	1	2	2	2
Horas mensuales	80	80	160	160	160
Horas anuales	960	960	1920	1920	1920
Capacidad de producción teórica	19,2	19,2	38,4	38,4	38,4
Capacidad de producción con margen	12	12	23	23	23

Tabla 35 - Capacidad de producción

Se considerará para el flujo de caja el sueldo de este empleado extra a partir del tercer año, cuyo valor se muestra en la Tabla 36.

Sueldo anual	
Horas mensuales	80
Costo por hora	$ 1,50
Sueldo mensual	$ 120,00
Total	**$ 1.440,00**

Tabla 36 - Sueldo anual

Para el capital de trabajo se tomará en cuenta el flujo de caja del primer año con las 12 unidades proyectadas a vender. Nuevamente, se considera que el cobro de la producción se divide en un 50% al realizar la compra del producto y el otro 50% 30 días después. Además, el equipo se produce en el mes donde se ejecuta la compra. El cálculo del mismo se muestra en la Tabla 37.

INGRESO MENSUAL AÑO UNO		Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre
Precio de venta		$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00
Unidades vendidas		1	1	1	1	1	1	1	1	1	1	1	1
Total de ventas		$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00	$200,00
Realización del cobro													
Cobro al momento de la compra	50 %	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00
Cobro 30 días despues de la compra	50 %	$0,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00	$100,00
Ingreso total		**100**	**200**	**200**	**200**	**200**	**200**	**200**	**200**	**200**	**200**	**200**	**200**
PRODUCCIÓN													
Stock		0	0	0	0	0	0	0	0	0	0	0	0
Ventas		1	1	1	1	1	1	1	1	1	1	1	1
Producción Mensual		1	1	1	1	1	1	1	1	1	1	1	1
EGRESOS													
Costos Fijos		$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50	$63,50
Costos de Fabricación		$137,42	$137,42	$137,42	$137,42	$137,42	$137,42	$137,42	$137,42	$137,42	$137,42	$137,42	$137,42
Egreso total		**$200,92**	**$200,92**	**$200,92**	**$200,92**	**$200,92**	**$200,92**	**$200,92**	**$200,92**	**$200,92**	**$200,92**	**$200,92**	**$200,92**
CAPITAL DE TRABAJO													
Ingresos		100	200	200	200	200	200	200	200	200	200	200	200
Egresos		$200,92	$200,92	$200,92	$200,92	$200,92	$200,92	$200,92	$200,92	$200,92	$200,92	$200,92	$200,92
Acumulado		-$100,92	-$101,84	-$102,76	-$103,68	-$104,60	-$105,52	-$106,44	-$107,36	-$108,28	-$109,20	-$110,12	-$111,04
Máximo Acumulado		**-$111,04**											

Tabla 37 - Capital de trabajo (VAN>0)

Se observa que el valor máximo acumulado necesario es de U$D 111,04. Se considerará un valor de capital de trabajo para el flujo de caja con un margen extra del 20%.

CAPITAL DE TRABAJO SIN MARGEN		**$111,04**
CAPITAL DE TRABAJO CON MARGEN	**1,2**	**$133,25**

Tabla 38 - Margen para capital de trabajo (VAN>0)

Una vez determinado el capital de trabajo necesario se realizó el flujo de caja para los 5 años de duración del proyecto y considerando una tasa de interés del 12%. El flujo de caja se muestra en la Tabla 39.

Flujo de caja	0	1	2	3	4	5
Unidades vendidas		12	12	23	23	23
Ingreso		$ 2.400	$ 2.400	$ 4.600	$ 4.600	$ 4.600
Precio unitario		$ 200	$ 200	$ 200	$ 200	$ 200
Costos fijos						
Total costos fijos		$ 762	$ 762	$ 762	$ 762	$ 762
Costos variables						
Sueldo empleados		$ -	$ -	$ 1.440	$ 1.440	$ 1.440
Costo de venta del producto		$ 137	$ 137	$ 137	$ 137	$ 137
Herramientas e instrumentos	$ 230	$ -	$ -	$ -	$ -	$ -
Equipos electrónicos	$ 370	$ -	$ -	$ 370	$ -	$ -
Costos variables totales		$ 137	$ 137	$ 507	$ 137	$ 137
Depreciación herramientas		$ 169	$ 169	$ 169	$ 169	$ 169
Utilidad antes de impuestos		$ 1.331	$ 1.331	$ 3.161	$ 3.531	$ 3.531
Impuesto a las ganancias (35%)		$ 466	$ 466	$ 1.106	$ 1.236	$ 1.236
Utilidad después de impuestos		$ 865	$ 865	$ 2.055	$ 2.295	$ 2.295
Depreciación herramientas		$ 169	$ 169	$ 169	$ 169	$ 169
Inversión inicial						
Herramientas y equipos electrónicos	$ -600					
Capital de trabajo	$ -133					
Recuperación capital de trabajo						$ 133
Valor de desecho						$ -
Flujo del proyecto	**$ -733**	**$ 1.035**	**$ 1.035**	**$ 2.224**	**$ 2.465**	**$ 2.598**

Tabla 39 - Flujo de caja (VAN>0)

Tasa de interés	12%
VAN	**$ 5.639**
TIR	**164%**

Tabla 40 - VAN y TIR (caso 2)

Por último, considerando estos números, se estima el plazo de la recuperación de la inversión PRI, el cual es de 0,71 años para la situación considerada.

PRI			
PERIODO	FLUJO	ACUMULADO	
0	$ -733		
1	$ 1.035	$ 1.035	$ 301
2	$ 1.035	$ 2.069	$ 1.336
3	$ 2.224	$ 4.293	$ 3.560
4	$ 2.465	$ 6.758	$ 6.025
5	$ 2.598	$ 9.356	$ 8.623

Tabla 41 - PRI (VAN>0)

4.2.4 Aproximación al valor actual neto

El Valor Actual Neto (VAN) consiste en actualizar los cobros y pagos de un proyecto o inversión y calcular su diferencia. Para ello el VAN actualiza todos los flujos de caja al momento presente descontándolos a un tipo de interés determinado. El VAN va a expresar una medida de rentabilidad del proyecto en términos absolutos netos.

De acuerdo con el análisis realizado en el segundo caso, donde se consideró una cantidad de unidades vendidas de acuerdo con las capacidades de producción, se obtuvo un VAN de U$D 5639. Este valor implica una alta rentabilidad del proyecto respecto a la tasa de interés considerada, por lo que permite definir positivamente la rentabilidad del proyecto.

4.2.5 Tasa interna de retorno

La Tasa Interna de Retorno (TIR) es la tasa de interés o rentabilidad que ofrece una inversión. Es decir, es el porcentaje de beneficio o pérdida que tendrá una inversión para las cantidades que no se han retirado del proyecto.

En el caso considerado, la TIR arrojó un resultado del 164%, altamente superior a la tasa de interés considerada. Por ello se puede concluir, esta vez mediante esta herramienta, la buena rentabilidad del proyecto.

4.2.6 Payback o plazo de recuperación

Es una manera muy sencilla de mostrar al inversor en que tiempo aproximado recuperará su inversión inicial con lo cual da una idea muy global de la conveniencia o no de invertir en el proyecto. Este es un criterio sencillo pero estático porque no tiene en cuenta la desvalorización del dinero. Sin embargo, es muy utilizado por los inversores como una primera aproximación de análisis de rentabilidad.

El cálculo del mismo se realiza mediante la siguiente ecuación:

$$PRI = a + \frac{I_0 - b}{F_t} \tag{54}$$

• a: es el número del periodo inmediatamente anterior hasta recuperar el desembolso inicial.

• Io: es la inversión inicial del proyecto.

• b: es la suma de los flujos hasta el final del periodo “a”.

• Ft: es el valor del flujo de caja del año en que se recupera la inversión.

Para el caso considerado, se observa en la Tabla 41 que el proyecto recupera su inversión inicial en el primer año de operación. Específicamente en 0.71 años de acuerdo con la fórmula, lo que equivale aproximadamente a 9 meses.

4.2.7 Productos y servicios de otros fabricantes

Como se expuso en la sección Estado actual, en el mercado de emuladores de amplificadores existen diferentes tipos de ofertas, clasificadas según el formato físico en el que las mismas se presentan. Sin embargo, en este caso se compara el dispositivo desarrollado contra alternativas comerciales que lleven a cabo emulaciones en hardware dedicado, puesto que es el tipo de equipo que más se asemeja al sistema implementado.

En este sentido cabe destacar que el equipo desarrollado no encuentra en el mercado competencia directa, puesto que se trata de un dispositivo que realiza la función de emulación utilizando una metodología alternativa y orientado a un mercado de consumo muy diferente. En general, los últimos dispositivos de emulación comerciales se encuentran integrados en sistemas de hardware con potentes procesadores de efectos de muy alto coste que superan los U$D 1000 y que incluyen muchas más funcionalidades que la que aquí se busca.

Es por ello que se seleccionaron dos equipos que resultan comparables en cierta medida por algunas de sus especificaciones y por presentar un precio y mercado de consumo más acorde al de nuestra implementación. La comparativa se muestra en la Tabla 42.

Producto	EQ Emulator	Joyo Revolution Cab Box R-08	Simplifier MK2
			
Descripción	Emulador personalizable estéreo basado en filtros adaptativos.	Pedal de modelado de gabinete, que cuenta con la posibilidad de carga de IR (Respuestas de Impulso). Cuenta con 4 simulaciones de amplificadores valvulares, 20 modelos de gabinete y 11 modelos clásicos de micrófono.	Amplificador de guitarra estéreo, analógico, de cero vatios con reverberación estéreo, repleto de opciones de enrutamiento y esculpido de tono.
Precio	U$D 200	U$D 238	U$D 556
Método de emulación	Filtro adaptativo FIR	IR	IR
Grado de personalización	Permite emular cualquier amplificador que el usuario dese con un grado bueno de fidelidad. No cuenta con carga de IRs.	Cuenta con emulaciones preestablecidas. Permite cargar IRs externamente pero no su generación.	No cuenta con posibilidad de carga de IRs pero si con numerosas opciones de ecualización (personalización manual).
Portabilidad	Si	Si	Si

Tabla 42 - Comparativa de emuladores

5. CONCLUSIONES

El proyecto ha logrado cumplir con los objetivos planteados al inicio del informe al desarrollar un dispositivo modular, portátil y económico capaz de emular sistemas electroacústicos complejos de manera sencilla y personalizada utilizando algoritmos y técnicas avanzadas de procesamiento digital de señales en tiempo real.

Durante el proceso de realización, se llevó a cabo un exhaustivo estudio y simulación que, aunque consumió tiempo considerable, proporcionó un alto nivel de rigurosidad técnica y sentó las bases para la evaluación y verificación del dispositivo implementado. La implementación de un sistema de tiempo real presentó un desafío tecnológico debido a la escasa disponibilidad de componentes que se ajustaran a las exigencias técnicas y económicas del proyecto, así como a la falta de bibliografía y documentación específica. En este sentido, este trabajo pretende servir como referencia para futuros desarrollos en este campo.

Este desarrollo ha introducido innovaciones tanto en el ámbito técnico como algorítmico. La utilización de filtros adaptativos ha demostrado ser un área de interés debido a su simplicidad para abordar problemas complejos. El estudio de estos filtros, su adaptación para su uso en sistemas electroacústicos y la búsqueda y el desarrollo de nuevas técnicas algorítmicas que mejoraron su rendimiento, han llevado a resultados sólidos en el marco de este proyecto, que pueden extrapolarse a otras áreas de investigación y aplicaciones diversas.

En cuanto a su viabilidad comercial, el dispositivo desarrollado ofrece una solución de bajo costo en comparación con las alternativas existentes en el mercado, al tiempo que permite un alto grado de personalización. Esto no solo brinda una ventaja económica, sino también funcional, ya que permite a los usuarios crear sus propias emulaciones de manera sencilla, intuitiva y novedosa en comparación con las opciones convencionales. Todo esto, además, se verifica mediante el análisis de factibilidad económica realizado, el cual permite un plazo de recuperación muy corto y márgenes altos de ganancia y rentabilidad.

Los resultados obtenidos en la verificación del sistema son altamente positivos en relación con las expectativas planteadas durante su desarrollo. Aunque se ha observado una pérdida aceptable de precisión en comparación con el método de respuesta al impulso, esta reducción de complejidad, tanto en términos de practicidad como de costo, es perfectamente aceptable para las aplicaciones previstas. Además, la evaluación auditiva ha superado las expectativas de los usuarios en términos de emulación de respuestas de amplificadores y equipos de audio. Sin embargo, se concluye que este dispositivo no es adecuado para la emulación de recintos reverberantes debido a su complejidad, lo cual plantea desafíos más allá del alcance de un dispositivo de estas características.

Como líneas de trabajo futuro, se proponen mejoras adicionales para el módulo desarrollado, como la posibilidad de almacenar y seleccionar múltiples filtros, lo que permitiría la selección de un filtro para cada canal estéreo o la conmutación entre diferentes emulaciones almacenadas. También se sugiere la exploración de técnicas de procesamiento más rápidas para soportar filtros de mayor orden, lo que permitiría respuestas de mayor longitud temporal y, por lo tanto, una mayor precisión. Otras

mejoras incluyen la visualización de la respuesta obtenida para identificar picos o valles en el sistema evaluado y corregir su ecualización, lo que ampliaría su utilidad técnica y comercial como dispositivo de medición. Además, surgen nuevas aplicaciones potenciales, como sistemas de ecualización automática o control activo de ruido, que podrían aprovechar una arquitectura adaptativa similar.

En resumen, el proyecto ha logrado sus objetivos, presentando avances significativos en el desarrollo de un dispositivo modular y económico para emular sistemas electroacústicos complejos. Además, se han identificado oportunidades de mejora y líneas de investigación futuras que podrían ampliar aún más su utilidad y aplicabilidad en diversos campos de estudio y aplicación.

6. ANEXOS

6.1 ANEXO A – CÓDIGO SIMULADOR

Se adjuntan los scripts de MATLAB utilizados para la etapa de simulación. Los archivos se pueden descargar del repositorio adjunto en bibliografía.

6.2 ANEXO B – CÓDIGO DSP

Se adjunta el programa principal en C del firmware del bloque DSP, implementado sobre un microprocesador STM32F407VG. El proyecto completo se puede descargar del repositorio adjunto en bibliografía.

6.3 ANEXO C – CÓDIGO CONTROL DE USUARIO

Se adjunta el programa en C++ del firmware del bloque control de usuario, implementado sobre un microprocesador Dual core Tensilica Xtensa LX6 (ESP32). El proyecto completo se puede descargar del repositorio adjunto en bibliografía.

6.4 ANEXO D – CIRCUITO ESQUEMÁTICO

6.5 ANEXO E - PCB

6.6 MANUAL DE USUARIO

7. BIBLIOGRAFÍAS Y REFERENCIAS BIBLIOGRÁFICAS

[1] Cuadrado, F.J. and Domínguez, J.J. (2019) Teoría y técnica del sonido. Madrid: Editorial Síntesis.

[2] Miyara, F. (2000) Acústica y sistemas de sonido. México: Musical Iberoamericana.

[3] Domingo, A.M. (2014) Apuntes de Acústica, Archivo digital UPM. Disponible en: https://oa.upm.es/70136/ (Accedido: 2022).

[4] Ferreyra, S.P. et al. (2012) Identificación y Análisis de Modos propios de recintos a partir de sus respuestas impulsivas, Mecánica Computacional Vol XXXI, págs. 3969-3989. Disponible en: https://www.investigacion.frc.utn.edu.ar/cintra/pub/file/mecom2012-ferreyra.pdf

[5] Hernandez, E. (2003) Transductores - University of Las Palmas de Gran Canaria, Electroacústica troncal de ITT Sonido e Imagen. Disponible en: https://www2.ulpgc.es/hege/almacen/download/26/26541/tema1elatransductores.pdf

[6] Ünsalan, C., Yücel, M.E. y Gürhan, H.D. (2018) Digital signal processing using ARM cortex-M based microcontrollers: Theory and practice. Cambridge: arm Education Media.

[7] Tokatli, A. (2004) Design and implementation of a DSP based active noise controler for headsets. Thesis. METU.

[8] Manuele, P. (2015) Filtro Digital Adaptativo para Control Activo de Ruido, Universidad Tecnológica Nacional.

[9] Franco, F. (1970) Implementación del Algoritmo LMS en la Plataforma DSPIC para la eliminación de Ruido Acústico sinusoidal, Gerencia tecnológica informática. Disponible en: https://biblat.unam.mx/es/revista/gerencia-tecnologica-informatica/articulo/implementacion-del-algoritmo-lms-en-la-plataforma-dspic-para-la-eliminacion-de-ruido-acustico-sinusoidal

[10] Prasad, S.R. and Godbole, B.B. (2017) Optimization of LMS algorithm for system identification, arXiv.org. Disponible en: https://arxiv.org/abs/1706.00897

[11] Parlanti, G. (2000) Filtros adaptativos, Universidad Nacional de Córdoba. Disponible en: http://www.facultad.efn.uncor.edu/webs/investigacion/DSP/documentos/Filtros_adaptivos.pdf

[12] Morán, J.A. y Carrié, J.S. (2014) Filtros adaptativos, Universidad Oberta de Catalunya. Disponible en: https://openaccess.uoc.edu/bitstream/10609/77806/1/Proceso%20avanzado_M%C3%B3dulo%203_Filtros%20adaptativos.pdf

[13] STM32F429/439 Reference manual (2021) STMicroelectronics. Disponible en: https://www.st.com/en/microcontrollers-microprocessors/stm32f429-439/documentation.html

[14] Discovery Kit with STM32F407VG MCU User manual (2020) STMicroelectronics. Disponible en: https://www.st.com/content/ccc/resource/technical/document/user_manual/70/fe/4a/3f/e7/e1/4f/7d/DM00039084.pdf/files/DM00039084.pdf/jcr:content/translations/en.DM00039084.pdf

[15] ESP32 Technical reference manual (2023) ESPRESSIF systems. Disponible en: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[16] ESP32-WROOM-32 Datasheet (2021) Espressif Systems. Disponible en: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
[17] PMOD I2S2 Reference Manual (2018) Digilent. Disponible en: https://digilent.com/reference/pmod/pmodi2s2/reference-manual
[18] (No date) 2.8inch SPI module MSP2807 User Manual. Disponible en: https://www.dragonwake.com/download/LCD/2.8inch_spi/2.8inch_SPI_Module_MSP2807_User_Manual_EN.pdf
[19] Burgess, P. (2023) Adafruit GFX Graphics Library, Adafruit Industries. Disponible en: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-gfx-graphics-library.pdf
[20] QD-TFT2803 specification (2018) QDtech. Disponible en: https://www.dragonwake.com/download/LCD/2.8inch_spi/QD-TFT2803%20specification_v1.1.pdf
[21] Marshall Amplification MG30FX Manual de Usuario (2008) Marshall Amplification. Disponible en: https://manualsbrain.com/es/manuals/1210522/
[22] SM57 User Guide (2010) Shure. Disponible en: https://pubs.shure.com/guide/SM57/en-US

7.1 ENLACES

[1] http://www.grgr.de/IR/
[2] https://drive.google.com/file/d/1sw11vJPne33Oh9aVFctduwD3aB31esgn/view
[3] https://jimamsden.wordpress.com/2015/12/24/creating-an-acoustic-guitar-impulse-response-for-line6-helix/
[4] https://audiodsplab.wordpress.com/ping-pong-buffer-audio-stream/
[5] https://www.st.com/content/st_com/en.html
[6] https://www.rohde-schwarz.com/lat/productos/prueba-y-medicion/essentials-test-equipment/digital-oscilloscopes/entendiendo-el-uart_254524.html#:~:text=UART%20significa%20Transmisor%2DReceptor%20As%C3%ADncrono,ambos%20lados%20de%20la%20conexi%C3%B3n.
[7] https://www.espressif.com/en/products/socs
[8] https://digilent.com/reference/pmod/pmodi2s2/reference-manual?redirect=1
[9] https://www.elprocus.com/i2s-protocol/
[10] http://www.lcdwiki.com/2.8inch_SPI_Module_ILI9341_SKU:MSP2807
[11] https://vidaembebida.wordpress.com/2017/02/08/protocolo-de-comunicacion-spi/
[12] https://deepbluembedded.com/stm32-dma-tutorial-using-direct-memory-access-dma-in-stm32/
[13] https://civilpedia.org/p/?t=STM32-DMA-Cheat-Sheet&pid=315
[14] https://predictabledesigns.com/auto-generated-firmware-code-deconstructing-an-stm32cubemx-project/
[15] https://tttapa.github.io/Pages/Mathematics/Systems-and-Control-Theory/Digital-filters/Exponential%20Moving%20Average/Exponential-Moving-Average.html
[16] https://blog.mbedded.ninja/programming/signal-processing/digital-filters/exponential-moving-average-ema-filter/
[17] https://www.roomeqwizard.com/

7.2 LIBRERÍAS

Librería FLASH: https://github.com/MYaqoobEmbedded/STM32-Tutorials/tree/master/Tutorial%2030%20-%20FLASH%20Memory

CMSIS-DSP: https://www.keil.com/pack/doc/CMSIS/DSP/html/index.html

Audio tools: https://github.com/pschatzmann/arduino-audio-tools

TFT_eSPI: https://github.com/Bodmer/TFT_eSPI

7.3 REPOSITORIO

Archivos del proyecto: https://github.com/franjf97/EQ_emulator