

PROYECTO FIN DE GRADO

TÍTULO: Diseño e implementación de un dispositivo multicanal de grabación y reproducción basado en Raspberry Pi

AUTOR/A: Pablo O. Ebohon Serna

TITULACIÓN: Grado en Ingeniería de Imagen y Sonido

TUTOR/A: Lino Pedro García Morales

DEPARTAMENTO: IAC

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: Eloy Portillo Aldana

TUTOR: Lino Pedro García Morales

SECRETARIO: Jorge Grundman Isla

Fecha de lectura:

Calificación:

El Secretario,

Resumen

El objetivo principal de este proyecto es el diseño y desarrollo de un sistema de grabación y reproducción de audio multicanal utilizando la plataforma Raspberry Pi. Este sistema permite manejar hasta 8 canales de entrada y salida, grabando y reproduciendo audio de alta calidad en tiempo real. Además, busca ser escalable, versátil y de bajo costo, haciéndolo adecuado para su uso en entornos educativos y profesionales, como los laboratorios de la Universidad Politécnica de Madrid. En este contexto, se espera que el sistema sirva como una herramienta de apoyo en asignaturas relacionadas con el procesamiento de señales y diseño de sistemas embebidos.

El sistema emplea componentes clave como conversores analógico-digitales (ADC) y digital-analógicos (DAC), los cuales son fundamentales para la conversión de señales de audio. También se utilizan conectores de audio y el protocolo de comunicación digital I2S, que permite una transmisión eficiente de audio entre los diferentes módulos del sistema. La elección de I2S garantiza la calidad del audio y la compatibilidad con otros dispositivos de audio digital, además de permitir la expansión a más canales si fuese necesario en el futuro.

Una característica importante del proyecto es la inclusión de una interfaz gráfica, desarrollada en Python, que permite una gestión sencilla y eficaz del sistema. Python ha sido elegido por su flexibilidad y facilidad para incorporar futuras funcionalidades, como procesamiento de audio avanzado o integración con sistemas de almacenamiento en red. Este enfoque facilita la adaptabilidad del sistema a las necesidades de los usuarios.

Este proyecto no solo proporciona una solución técnica para la grabación y reproducción de audio multicanal, sino que también establece una plataforma flexible y de bajo costo que puede evolucionar para satisfacer distintas necesidades en entornos académicos y profesionales. La posibilidad de escalar el sistema y añadir nuevas funciones asegura su relevancia a largo plazo, tanto en el ámbito educativo como en el profesional.

Abstract

The main objective of this project is the design and development of a multichannel audio recording and playback system using the Raspberry Pi platform. This system allows managing up to 8 input and output channels, recording and playing back high-quality audio in real-time. Additionally, it aims to be scalable, versatile, and cost-effective, making it suitable for use in educational and professional environments, such as the laboratories of the Polytechnic University of Madrid. In this context, the system is expected to serve as a support tool in courses related to signal processing and embedded systems design.

The system employs key components such as analog-to-digital (ADC) and digital-to-analog (DAC) converters, which are essential for audio signal conversion. Audio connectors and the I2S digital communication protocol are also used, enabling efficient audio transmission between the different system modules. The choice of I2S ensures audio quality and compatibility with other digital audio devices, while also allowing for the system's expansion to more channels if necessary in the future.

An important feature of the project is the inclusion of a graphical interface, developed in Python, which allows for simple and efficient system management. Python has been chosen for its flexibility and ease in incorporating future functionalities, such as advanced audio processing or integration with network storage systems. This approach facilitates the system's adaptability to users' needs.

This project not only provides a technical solution for multichannel audio recording and playback, but it also establishes a flexible and cost-effective platform that can evolve to meet various needs in academic and professional environments. The possibility of scaling the system and adding new functions ensures its long-term relevance in both educational and professional contexts.

Índice de figuras

Figura 1: Señal analógica (izquierda) y señal digital (derecha) [10]	4
Figura 2: Proceso de cuantificación [7]	4
Figura 3: Líneas de comunicación I ² S [5]	7
Figura 4: Modos de configuración de I ² S [5]	7
Figura 5: Diagrama funcional de sistema.....	8
Figura 6: Diagrama de conexiones.....	9
Figura 7: Aspecto físico del sistema.....	9
Figura 8: Pinout Raspberry Pi 4 y 5 [22]	11
Figura 9: Vista frontal de Raspberry Pi 5 [24]	12
Figura 10: Pulsador Momentáneo LED.....	13
Figura 11: PCM1808: Vista frontal y trasera del chip (izquierda) [27] y Diagrama de Bloques Funcional (derecha) [17]	15
Figura 12: Conector Combo (izquierda) y esquema de NCJ6FA-H (derecha) [29].....	15
Figura 13: MAX98357A: Vista frontal (izquierda) [30] y Diagrama de Bloques Funcional (derecha) [18].....	16
Figura 14: Potenciómetro Lineal Deslizante (izquierda) [31] y Potenciómetro Lineal Rotativo (derecha) [32].....	17
Figura 15: Terminales de un Potenciómetro [33].....	18
Figura 16: MCP3008: Vista frontal (izquierda) [34] y Diagrama de Bloques Funcional (derecha) [19].....	19
Figura 17: Vista frontal y trasera del Display [35]	20
Figura 18: Conexiones necesarias entre el núcleo de procesamiento y los Módulos de Captura y Reproducción, teniendo en cuenta las interfaces USB, SPDIF e I ² S.....	22
Figura 19: Dirección IP obtenida con IP Scanner	23
Figura 20: Escritorio de Raspberry Pi Os (64-bit).....	23
Figura 21: Archivo config.txt.....	24
Figura 22: Archivo config.txt.....	25
Figura 23: Listado de dispositivos de captura	26
Figura 24: Listado de dispositivos de reproducción.....	26
Figura 25: Archivo Config.txt	27
Figura 26: Esquemático del Núcleo de Procesamiento	29
Figura 27: Esquema de la vista superior y distribución de los pines del chip PCM1808 [17]	30
Figura 28: Diagrama de conexiones típicas del chip PCM1808 [17]	31
Figura 29: Chip PCM1808 implementado en PCB [55]	31
Figura 30: Esquemático del Módulo de Captura de Audio	34
Figura 31: Esquema de la vista superior y distribución de los pines del chip MAX98357A [18]	35
Figura 32: Configuración Estereo utilizando dos MAX98357A [18].....	36
Figura 33: Vista frontal y trasera del chip MAX98357A integrado en una PCB [56]	37
Figura 34: Diagrama de conexiones típicas del chip MAX98357A	37
Figura 35: Esquemático del Módulo de Reproducción de Audio.....	39

Índice de figuras

436: Esquema de la vista superior y distribución de los pines del chip MCP3008 [19]	40
Figura 37: Esquemático del Módulo de Control de Volumen	42
Figura 38: Ventana principal	43
Figura 39: Ventana grabador I.....	44
Figura 40: Ejemplo de interacción con la Ventana Gabador I.....	45
Figura 41: Ventana Grabador II.....	45
Figura 42: Ventana Save File	47
Figura 43: Ventana Reproductor I	48
Figura 44: Ejemplo de interacción con la Ventana Reproductor I.....	49
Figura 45: Ventana Reproductor II	50
Figura 46: Cable de Audio Apantallado	52
Figura 47: Módulo Reproductor de Audio	53
Figura 48: Módulo de Captura de Audio	53
Figura 49: Módulo de Control de Volumen.....	54
Figura 50: Cable Dupont Macho-Hembra (izquierda), cable Dupont Macho-Macho (centro) y cable FFC/FPC (derecha)	54
Figura 51: Vista frontal grabador/reproductor de audio basado en Raspberri Pi	55
Figura 52: Vista trasera grabador/reproductor de audio basado en Raspberri Pi.....	55
Figura 53: Medida Vrms del canal 1.....	56
Figura 54: Modo TDM 32 bits en MAX98357A.....	61

Lista de acrónimos

- ADC (Analog to Digital Converter): Convertidor Analógico-Digital.
- DAC (Digital to Analog Converter): Convertidor Digital-Analógico.
- I2S (Inter-Integrated Circuit Sound): Protocolo de comunicación de sonido.
- WS (Word Select): Señal de selección de palabra en comunicación I2S.
- SCK (Serial Clock): Reloj de datos en serie.
- MCLK (Master Clock): Reloj maestro para sincronización.
- SD (Serial Data): Datos transmitidos en serie.
- SPI (Serial Peripheral Interface): Interfaz de comunicación en serie para periféricos.
- GPIO (General Purpose Input/Output): Entradas/Salidas de propósito general.
- LCD (Liquid Crystal Display): Pantalla de cristal líquido.
- PCB (Printed Circuit Board): Placa de circuito impreso.
- TDM (Time Division Multiplexing): Multiplexación por división de tiempo.
- USB (Universal Serial Bus): Bus de comunicación universal en serie.
- SPDIF (Sony/Philips Digital Interface Format): Formato de interfaz digital Sony/Philips.
- RPI (Raspberry Pi): Plataforma de procesamiento de la Fundación Raspberry Pi.
- FLAC (Free Lossless Audio Codec): Códice de audio sin pérdida.
- WAV (Waveform Audio File Format): Formato de archivo de audio en forma de onda.

Índice de contenidos

Resumen.....	i
Abstract.....	iii
Índice de figuras	v
Lista de acrónimos	vii
Índice de contenidos.....	ix
1. Introducción	1
1.1 Objetivos del sistema.....	2
1.2 Especificaciones del sistema.....	2
2. Marco Tecnológico.....	3
2.1 Audio digital.....	3
2.2 Protocolo de comunicación I ² S	6
3. Visión general	8
4. Selección de componentes.....	10
4.1 Núcleo de procesamiento.....	10
4.2 Módulo de captura de audio	14
4.2.1 ADC.....	14
4.2.2 Conectores.....	15
4.3 Módulo de reproducción de audio	16
4.3.1 DAC.....	16
4.3.2 Conectores.....	17
4.4 Módulo de control de volumen	17
4.5 Módulo Interfaz gráfica	20
5. Selección de protocolo de audio	21
6. Diseño del Núcleo de Procesamiento	23
6.1 Configuraciones previas.....	23
6.2 Configuración de las interfaces	24
6.3 Esquema eléctrico.....	28
7. Diseño del Módulo de Captura de Audio.....	30
7.1 Conexiones.....	30
7.2 Diseño eléctrico	34
8. Diseño del Módulo de Reproducción de Audio.....	35
8.1 Conexiones.....	35
8.2 Esquema eléctrico.....	39
9. Diseño del Módulo de Control de Volumen.....	40
9.1 Conexiones.....	40
9.2 Esquema eléctrico.....	42

Índice de contenidos

10.	Software.....	43
10.1	Software del grabador.....	44
10.2	Software del reproductor.....	48
10.3	Software para el control de volumen	51
11.	Prototipado y montaje.....	52
12.	Presupuesto	57
13.	Impacto del proyecto.....	59
14.	Conclusiones	60
14.1	Líneas futuras	60
15.	Bibliografía	62
Anexo		69
A.1	Código del grabador	69
A.1.1.	Ventana grabador I	69
A.1.2.	Ventana grabador II.....	71
A.1.3.	Ventana Save File	74
A.1.4.	Captura de audio.....	75
A.1.5.	Sub-sub capítulo anexo	76
A.2	Código del reproductor	77
A.2.1.	Ventana reproductor I.....	77
A.2.2.	Ventana reproductor II.....	82
A.2.3.	Reproducción de audio	85
A.3	Código Ventana principal	88

1. Introducción

Desde que las innovaciones tecnológicas permitieron la conversión de señales acústicas en señales eléctricas a mediados del siglo XX, el campo del audio ha experimentado una evolución continua [1]. Este ámbito ha sido explorado desde múltiples perspectivas, abarcando disciplinas como la ingeniería, la ciencia y el arte, lo que ha resultado en mejoras significativas tanto en la calidad como en la versatilidad del sonido. Estas mejoras han ampliado considerablemente las aplicaciones del audio en áreas como la música, el cine, la educación y la investigación científica [2].

El procesamiento y manipulación de audio digital ha ganado popularidad en aplicaciones multimedia y de entretenimiento, presentando innumerables ventajas frente al audio analógico debido a su precisión, capacidad para evitar el ruido y la distorsión, y su flexibilidad creativa. La digitalización ha transformado dispositivos como grabadores y reproductores de audio, ofreciendo ventajas notables sobre los sistemas analógicos tradicionales. Estos sistemas digitales no solo permiten una mayor calidad y fidelidad en la captura y reproducción del sonido, sino que también ofrecen una personalización extensiva gracias a la programación, lo que permite incorporar funciones avanzadas y minimizar problemas [3].

En este contexto, el desarrollo de un grabador/reproductor de audio multicanal utilizando una plataforma versátil como la Raspberry Pi presenta una oportunidad significativa. La Raspberry Pi, con su bajo costo, versatilidad y suficiente capacidad de procesamiento, es ideal para manejar aplicaciones de audio complejas [4]. Este proyecto busca diseñar y desarrollar un sistema capaz de grabar y reproducir audio en tiempo real a través de múltiples canales de forma simultánea. Tal sistema será una herramienta valiosa en entornos profesionales, educativos y de entretenimiento, permitiendo la captura de sonido y la reproducción multicanal con alta fidelidad.

Este proyecto no solo ofrece un dispositivo funcional, sino que también establece una base para futuros desarrollos en el campo del procesamiento de audio digital. Gracias a la modularidad y escalabilidad de la Raspberry Pi, el sistema podrá expandirse y adaptarse a nuevas aplicaciones y tecnologías. Además, la implementación en entornos educativos, como los laboratorios de la Universidad Politécnica de Madrid, permitirá a los estudiantes interactuar con tecnología avanzada y aplicar sus conocimientos en proyectos reales, contribuyendo así al avance en la disciplina de la ingeniería de sonido.

En resumen, este proyecto promete una solución eficiente y versátil para la captura y reproducción de audio multicanal, con potencial para innovaciones futuras en el ámbito del procesamiento de audio digital, beneficiando tanto a la comunidad académica como a la industria.

1.1 Objetivos del sistema

Los principales objetivos por cumplir mediante el diseño y desarrollo del grabador/reproductor son los siguientes:

- **Escalabilidad:** Se busca que el proyecto permita grabar y reproducir audio en tiempo real, procesando digitalmente las señales, para su implementación en el laboratorio de audio de la Universidad Politécnica de Madrid, por lo que se utilizará para la docencia y futuros proyectos. Al ser un sistema libremente programable, se podrán desarrollar nuevas funcionalidades de forma sencilla, como la incorporación de efectos de audio o la compresión de los archivos capturados.
- **Versatilidad:** Se pretende desarrollar un sistema versátil, con la capacidad realizar diversas funciones como la grabación y la reproducción de audio multicanal, permitiendo al usuario poder decidir distintas formas de uso, como la selección de los canales por los que grabar y reproducir el sonido o que archivos reproducir.
- **Precio asequible:** Se escogerán componentes electrónicos comunes, fáciles de obtener y fabricados en masa, lo cual abarata los costes a la hora de su obtención, con la idea de que el reducido precio de los componentes no afecte en gran medida a la calidad y a las especificaciones del dispositivo.

1.2 Especificaciones del sistema

Con el objetivo de cumplir los requerimientos mencionados anteriormente, es necesario determinar una serie de especificaciones que caractericen el sistema.

- El software relativo a funciones de control de interfaz y procesamiento de audio será realizado en Python, un lenguaje óptimo para el tratamiento de audio en tiempo real y muy utilizado a la hora de programar en Raspberry Pi.
- El procesamiento de audio se realizará en tiempo real mediante la utilización de buffers de entrada y salida, por lo que debe ajustarse el tamaño de los buffers para que no se produzca una latencia perceptible.
- Grabar y reproducir audio de alta calidad por 8 canales de entrada y 8 de salida, cumpliendo con los estándares profesionales, dotando al sistema de una tasa de muestreo de 48 kHz y una profundidad de 24 bits.
- Transmisión de información entre los dispositivos de audio por el protocolo I²S.
- Se diseñará una interfaz gráfica para el control del software del dispositivo, esta debe ser sencilla de utilizar y entendible para el usuario.
- La plataforma principal debe ser capaz de controlar los periféricos hardware del dispositivo (botones y potenciómetros).
- Los dispositivos electrónicos serán soldados a mano en su mayoría.
- Aunar todos los elementos hardware en una carcasa con las medidas y el tamaño pertinente para su colocación en un rack de audio estándar.

2. Marco Tecnológico

En esta sección, se abordan las tecnologías clave que sustentan el presente proyecto, con el objetivo de proporcionar una comprensión integral del sistema desarrollado. Dado que el proyecto se centra en la manipulación y transmisión de datos de audio, se ha optado por utilizar el formato digital; por su capacidad para preservar la calidad del sonido y facilitar su procesamiento [3]. Se describe en detalle el audio digital y, en particular, se analiza el protocolo de comunicación I²S, el cual permite la interconexión eficiente de los diversos componentes del sistema encargados de la transmisión y recepción del audio [5].

Este enfoque es fundamental para el diseño del sistema modular, asegurando que todos los elementos involucrados sean compatibles y puedan interactuar de manera óptima dentro del entorno digital. Así, la explicación de estas tecnologías no solo establecerá las bases técnicas del proyecto, sino que también guiará las decisiones de implementación para lograr un rendimiento superior y una integración armoniosa de todos los componentes.

2.1 Audio digital

El audio digital se fundamenta en la transformación de las señales eléctricas en datos binarios, un formato que las computadoras y dispositivos electrónicos pueden procesar y entender con facilidad. Esta conversión permite no solo la transmisión eficiente de las señales de audio, sino también su almacenamiento y edición de manera precisa y flexible [6]. A continuación, se describen los procesos y características esenciales que definen el funcionamiento del audio digital, sentando las bases para su aplicación en sistemas de transmisión y procesamiento de sonido.

Muestreo

La señal de acústica se captura en intervalos de igual duración para obtener de la señal continua una señal discreta. Con este fin aparece el principal parámetro involucrado en este proceso, la tasa de muestreo [7].

La tasa de muestreo determina cuantas veces por segundo se toma una muestra de la señal analógica. La tasa de muestreo o frecuencia de muestreo se define mediante el teorema de Nyquist [6] [8], el cual establece que la frecuencia de muestreo tiene que ser por lo menos dos veces mayor que la máxima frecuencia de la señal analógica que se quiere procesar para evitar el solapamiento espectral (*aliasing*), esto se refleja en la siguiente fórmula:

$$f_s \geq 2 \cdot f_{max}$$

El oído humano percibe frecuencias hasta aproximadamente de 20 Hz a 20 kHz [9], por lo que siguiendo el teorema de Nyquist se puede afirmar que la frecuencia de muestreo tiene que ser al menos de 20 kHz para poder garantizar una señal digital fiel a la analógica. En la Figura 1, extraída del libro [10], se puede observar la diferencia de forma que presenta una señal analógica y una señal digital.

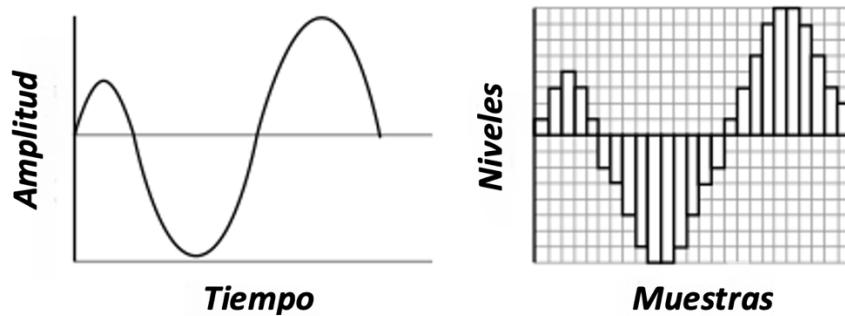


Figura 1: Señal analógica (izquierda) y señal digital (derecha) [10]

Cuantificación

En el proceso de cuantificación, se asignan valores numéricos a cada muestra recogida, representando la amplitud de la señal en cada instante temporal concreto. Para establecer los niveles discretos que puede tomar la señal en formato digital se define la profundidad de bit.

Los niveles que puede tomar cada muestra de la señal digital se conocen como escalones de cuantificación, el número de escalones viene dado por la siguiente expresión:

$$\text{Número de escalones} = 2^{\text{Profundidad de bit}}$$

La resolución de cada escalón es inversamente proporcional al número de escalones, por lo que cuantos más escalones haya, la representación digital será más precisa y menor será el error de cuantificación, que es la diferencia entre el valor de la señal analógica y el valor asignado en el proceso de cuantificación.

La profundidad de bit indica la resolución de cada muestra, lo que significa que una mayor profundidad de bit proporciona un rango dinámico más amplio y una mayor fidelidad a la hora de representar el sonido [6].

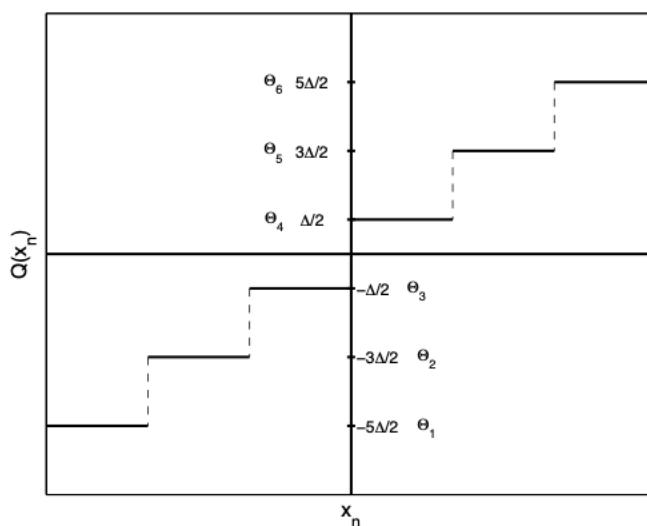


Figura 2: Proceso de cuantificación [7]

Latencia

En el ámbito del procesamiento digital de señales de audio, se conoce como latencia al tiempo que se tarda desde que se introduce una señal por la entrada del sistema de procesamiento hasta que sale de él por la salida correspondiente. La latencia comienza a ser perceptible para la mayoría de las personas a partir de los 10 ms pudiendo dificultar procesos como la grabación, las actuaciones en directo o la transmisión.

En audio digital las señales se procesan por bloques de un tamaño fijo de muestras, a lo que se le llama *buffering*. Un búfer es un espacio delimitado de memoria que sirve para intercambiar y procesar datos de audio digital. El tamaño de estos búferes es inversamente proporcional a la carga a la que someten al procesador. Si el tamaño del búfer es pequeño y por lo tanto la latencia es menor, el procesador tendrá mayor carga de trabajo que si el tamaño del búfer es mayor [11].

Estándares profesionales

Para garantizar, la calidad, la compatibilidad y la fiabilidad en la captura y la reproducción de audio, existen unos estándares profesionales a los que, en este caso, el grabador/reproductor de audio digital debe ceñirse.

- **Frecuencia de muestreo:** las más comunes en el sector son [10]:
 - **44,1 kHz:** Estándar del CD de audio, utilizado principalmente en el mercado de consumo.
 - **48 kHz:** Estándar en la producción de audio profesional.
 - **88,2 kHz y 96 kHz:** Frecuencias empleadas en producciones que requieren una mayor resolución en el procesamiento de audio.
 - **176,5 kHz y 192 kHz:** Utilizadas en aplicaciones que demandan una resolución extremadamente alta para grabación y reproducción.
- **Profundidad de bit:** Las más frecuentes son [10]:
 - **16 bits:** Es el estándar para CD proporcionando un rango dinámico de 96 dB.
 - **24 bits:** El estándar profesional, con un rango dinámico de 144 dB.
 - **32 bits de punto flotante:** Utilizado en procesamiento de audio durante las operaciones de mezcla y efectos.
- **Formatos de archivo:** Algunos de los más utilizados son [12]:
 - **WAV:** Un formato sin compresión compatible con una amplia gama de frecuencias de muestreo y profundidades de bit.
 - **AIFF:** Es el archivo de audio sin compresión estándar en MacOS y cuenta con características similares a las del WAV,
 - **FLAC:** Formato de compresión sin pérdida que reduce el tamaño del archivo sin sacrificar la calidad.
- **Latencia:** Un dispositivo puede ser utilizado para la grabación de señales acústicas de ámbito musical debe tener una latencia de menos de 10 ms, lo que la hace imperceptible [11].

- **Niveles:** El nivel de la salida de audio de consumo doméstico es de -10 dBV y el de audio profesional estándar es de +4 dBu [12].

2.2 Protocolo de comunicación I²s

En el ámbito de la transmisión de información entre elementos hardware, los protocolos de comunicación desempeñan un papel fundamental para garantizar la transmisión de la información de manera precisa y eficiente. A continuación, se detallará el funcionamiento del protocolo de comunicación utilizado para la conexión entre la interfaz principal y los dispositivos de audio.

El Inter-Integrated Circuit Sound o I²s es un estándar de interfaz de bus serie electrónico diseñado por Philips Semiconductors en el año 1986. Se utiliza para transferir datos PCM entre circuitos integrados en dispositivos electrónicos y para conectar dispositivos de audio digitales entre sí, como DAC, ADC u otros periféricos de audio [14].

La comunicación a través interfaz I²s se basa en el sincronismo de tres señales. El bus I²s separa las señales de reloj y de datos [15], lo que permite reducir las fluctuaciones de la señal en el eje temporal (*jitter*). Las señales involucradas en la comunicación mediante el protocolo I²s son las siguientes [5]:

- **WS:** El LRCLK (Left-Right Clock) o WS (Word Select) es la señal de reloj que indica qué canal de audio está transmitiendo información mediante el mecanismo de TDM (Time Division Multiplexing), dividiendo temporalmente el flujo de datos de audio para transmitir señales de audio estéreo o multicanal. La configuración básica del WS es estéreo, cuando WS = 0 se transmite el canal izquierdo y cuando WS = 1 el derecho.

La frecuencia del WS es igual a la frecuencia de muestreo de audio (fs), típicamente 44,1kHz o 48kHz.

$$\text{Frecuencia WS} = f_s$$

- **SCK:** El SCK (Continuous Serial Clock) es una señal de reloj que controla el ritmo al que los bits de datos de audio son transmitidos desde el dispositivo emisor al receptor. Su frecuencia se determina siguiendo la siguiente relación.

$$\text{Frecuencia SCK} = f_s \cdot \text{Número de canales} \cdot \text{Profundidad de bits}$$

- **MCLK:** El MCLK (Master Clock) es una señal de alta frecuencia que se utiliza para generar las señales WS y BCLK en sistemas que lo requieren. Las relaciones comunes de frecuencia son 256xfs, 384xfs y 512xfs
- **SD:** SD (Serial Data) transporta los datos de audio en forma de secuencia de bits. Estos bits transmitidos representan las muestras de audio digital emitidas por los equipos involucrados en la comunicación. La cantidad de bits que se envía en cada muestra es dependiente de la profundidad de bit, al ser este valor igual al número de bits por

muestra de cada canal de audio. SD es unidireccional, por lo que una señal de Serial Data solo podrá enviar o recibir datos.

En la Figura 3 se muestra un ejemplo general de funcionamiento de las diferentes líneas de comunicación que participan en el protocolo I²s.

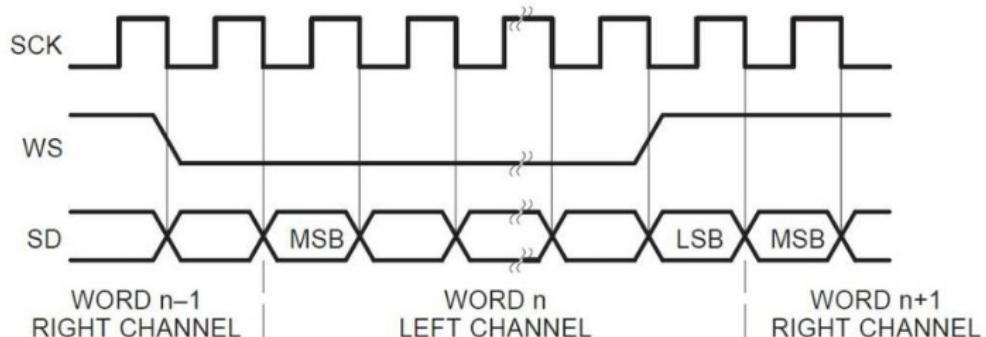


Figura 3: Líneas de comunicación I²s [5]

Para establecer el modo de gestión de las señales de reloj y de datos entre los dispositivos conectados por I²s, es necesario configurar a los dispositivos que forman parte del proceso de comunicación en modo maestro o esclavo. En la Figura 4 se muestran ejemplos de configuraciones representando dispositivos en modo maestro y esclavo.

- **Modo maestro:** El dispositivo en modo maestro se encarga de generar todas las señales de reloj necesarias, incluyendo MCLK, BCLK y WS. El maestro se encarga de la sincronización de toda la transferencia de datos, permitiendo que los dispositivos en modo esclavo utilicen las señales que el maestro ha generado.
- **Modo esclavo:** Los dispositivos en modo esclavo se encargan de recibir las señales de reloj y las utilizan para sincronizar sus operaciones. En este modo los dispositivos son completamente dependientes de las señales que recibe del dispositivo en modo maestro, lo que asegura la sincronización entre los dispositivos del sistema.

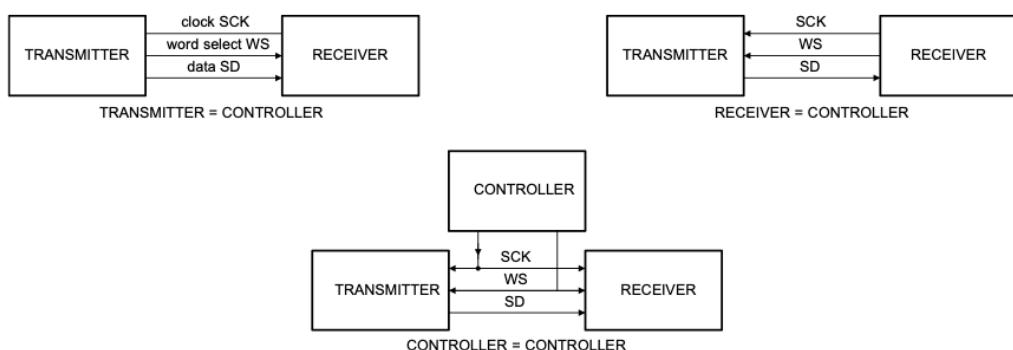


Figura 4: Modos de configuración de I²s [5]

3. Visión general

El grabador/ reproductor desarrollado en este proyecto se basa en el diagrama funcional de la Figura 5:

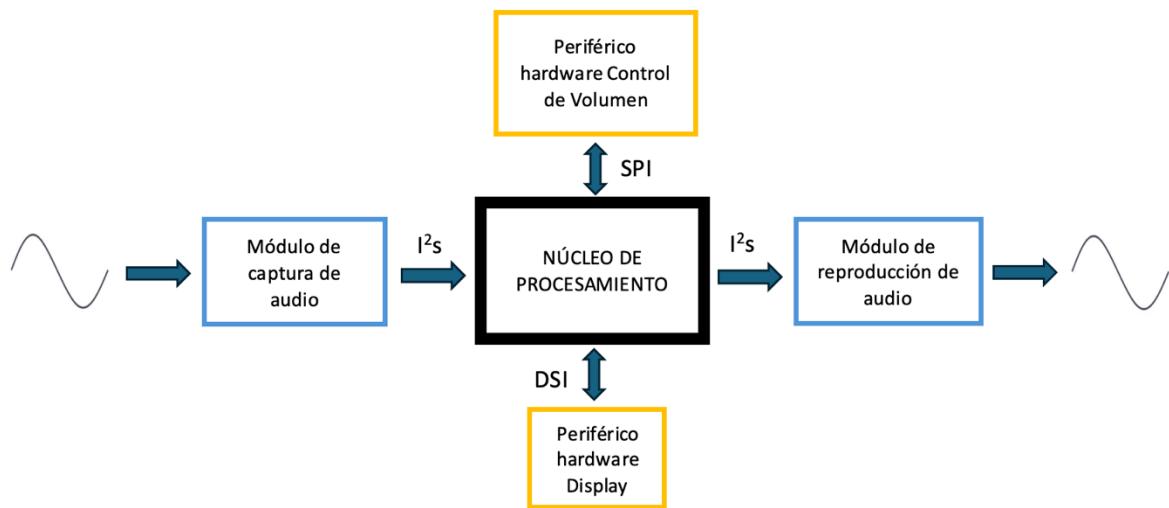


Figura 5: Diagrama funcional de sistema

A lo largo de la memoria se profundiza en los bloques que componen el sistema. A continuación, se resumen sus principales funciones:

- **Núcleo de procesamiento:** Es el cerebro del proyecto donde se realiza la fase de procesamiento de audio y se gestionan las comunicaciones con los periféricos hardware. Se ha escogido como plataforma principal una Raspberry Pi 5 [15], que se comunica con los periféricos mediante SPI, I²S, DSI y los pines GPIO.
- **Módulo de captura de audio:** Es el elemento que se encarga de capturar el audio analógico y convertirlo en digital para enviarlo al núcleo de procesamiento, esto se llevará a cabo mediante una interfaz de audio que realice las funciones de un ADC, compuesta principalmente por chips PCM1808 [16]. Este módulo se comunica por I²S con el núcleo de procesamiento. Este módulo posee 8 entradas combo, pudiendo conectar todo tipo de elementos que generen audio a través de un cable Jack o XLR.
- **Módulo de reproducción de audio:** Una vez procesado el audio por el núcleo de procesamiento, este llega en formato digital al módulo de reproducción de audio donde se convierte en analógico gracias a la interfaz de audio que realiza las funciones de un DAC. Esta interfaz de audio está compuesta principalmente por chips MAX98375A [17]. El módulo se comunica por I²S con el núcleo de procesamiento. Este módulo posee 8 salidas combo, pudiendo conectar todo tipo de elementos que reproduzcan audio a través de un cable Jack o XLR.
- **Periférico hardware control de volumen:** Controla el volumen mediante potenciómetros conectados a un conversor analógico digital, el MCP3008 [18], que a

su vez se comunica con el núcleo de procesamiento por SPI. Este módulo cuenta con 8 potenciómetros, para controlar el volumen de los 8 canales de entrada o los 8 de salida.

- **Periférico hardware Display:** Se trata de una pantalla táctil LCD que establece su comunicación con el núcleo central de procesamiento a través de DSI. Desde este periférico se muestra la interfaz gráfica que gestiona todo el software utilizable por el usuario.

El resultado de la implementación del sistema se detalla en el Diagrama de Conexiones de la Figura 6, mostrando los principales componentes físicos que definen el sistema

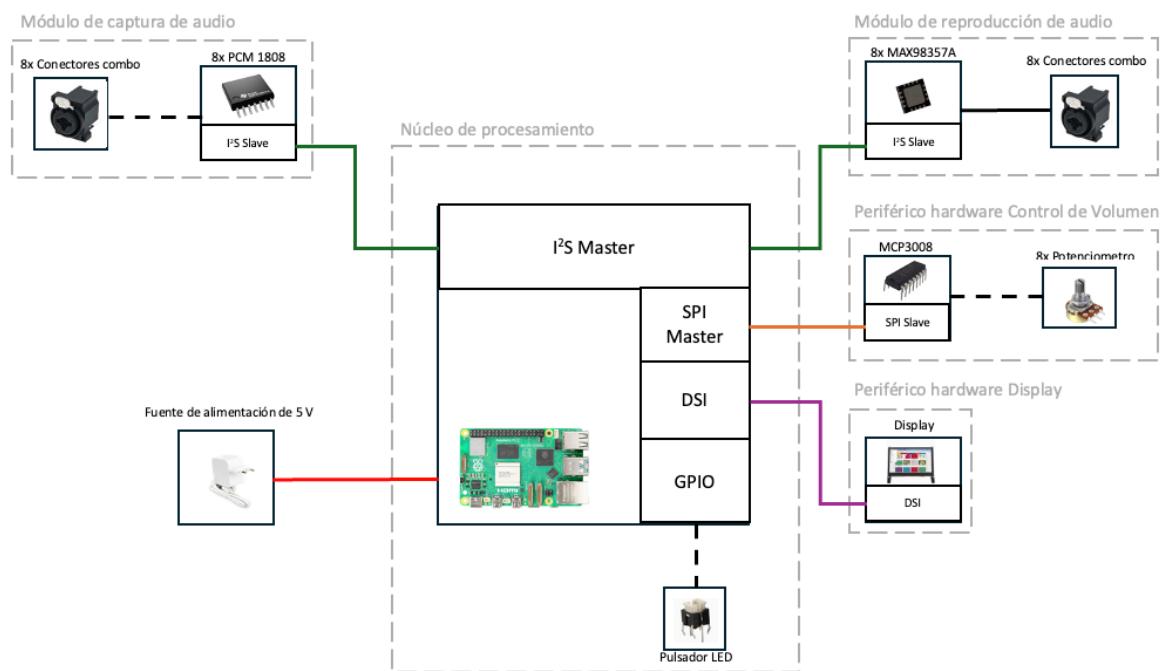


Figura 6: Diagrama de conexiones

La Figura 6 muestra la configuración básica del sistema, para completarlo se necesitan a la entrada elementos generadores de audio como instrumentos musicales o micrófonos y a la salida elementos que sean capaces de reproducir o recibir audio como altavoces o DAWs. En la figura 7 se muestra el aspecto físico del sistema.



Figura 7: Aspecto físico del sistema

4. Selección de componentes

El punto anterior ha dado una visión general sobre el funcionamiento del sistema y los elementos que lo conforman. En este apartado se examinan los principales componentes y dispositivos utilizados. El diseño y características de los elementos que componen el proyecto es fundamental para realizar un diseño fiable, garantizar un correcto rendimiento del sistema, y establecer un coste viable de acuerdo con el presupuesto. Por todo esto es necesaria una minuciosa selección de los componentes para que cumplan con su papel correctamente en el sistema.

4.1 Núcleo de procesamiento

Desde el inicio, se consideró la Raspberry Pi como la opción principal para ser el núcleo de este proyecto. La experiencia previa adquirida en la Universidad Politécnica de Madrid ha sido fundamental en esta elección. Tanto profesores como alumnos están más familiarizados con este dispositivo que con otros, y la información proporcionada por estas fuentes ha facilitado y continuará facilitando el desarrollo de este trabajo.

Las Raspberry Pi son microcomputadoras de bajo costo y tamaño reducido, pero con una gran flexibilidad en términos de programación, lo que permite llevar a cabo una amplia variedad de proyectos. Su extensa comunidad de usuarios proporciona una valiosa base de conocimientos y proyectos accesibles, lo que hace que trabajar con este dispositivo sea aún más sencillo.

Dado que existe una amplia gama de modelos de Raspberry Pi, es necesario analizar las características de cada uno para determinar cuál es el más adecuado para las funciones de núcleo de procesamiento en este proyecto [19].

Raspberry Pi 4 Modelo B

La Raspberry Pi 4 Modelo B [20] es una microcomputadora y placa de desarrollo que ofrece una amplia gama de funcionalidades, lo que le confiere una gran capacidad y versatilidad para el proyecto. Cuenta con la posibilidad de poder controlarla como si fuera una computadora (display, ratón y teclado), introduciendo una tarjeta SD con el Sistema Operativo que se deseé. Esto combinado con sus pines GPIO, la convierten en una placa de desarrollo muy versátil y programable.

Cuenta con un procesador Broadcom BCM2711 con cuatro núcleos y arquitectura ARM Cortex-A72 a 1,8 GHz. Esta placa cumple con la mayoría de los requerimientos planteados, soportando todas las tecnologías y periféricos necesarios para este proyecto, como puede observarse en el *pinout* de la Figura 8.



Figura 8: Pinout Raspberry Pi 4 y 5 [22]

En principio parece una opción adecuada y más barata que la Raspberry Pi 5, pero el motivo por el que no ha sido seleccionada es por el soporte que proporciona su interfaz I²S. Como se muestra en la Figura 8, la Raspberry Pi 4, cuenta con señales PCM CLK y PCM FS, que hacen referencia a las señales de reloj necesarias para poder transmitir datos a través de I²S, pero solo cuenta con una entrada y una salida de datos, PCM DIN y PCM DOUT, por cada uno de estos pines pueden recibir (PCM DIN) o emitir (PCM DOUT), únicamente dos canales (típicamente el derecho y el izquierdo en audio estéreo), por lo que no cumpliría el requerimiento para poder obtener 8 canales de entrada y 8 canales de salida. Esto podría solucionarse mediante una modificación en la TDM para que en vez de dos canales por pin se pudieran conseguir más, lo que complicaría en gran medida este proyecto al no contar esta Raspberry con una configuración adaptada a este sistema de funcionamiento [21].

Es la más utilizada en la UPM, por lo que el acceso a ella es fácil, y aunque no haya sido la seleccionada, ha sido útil para aprender los entresijos y la forma de funcionamiento de este tipo de dispositivo.

Raspberry Pi 5

La Raspberry Pi 5 [15], lanzada al mercado en 2023, es la última novedad de la organización educativa Raspberry Pi Fundation. Este ha sido el dispositivo seleccionado para ejercer el papel de núcleo de procesamiento del grabador/reproductor.

Cuenta con características similares a las de la Raspberry Pi 4 modelo B, en cuanto a sus características físicas y la forma de utilizarla, pero cuenta con un procesador más potente, el Broadcom BCM2712 con cuatro núcleos y arquitectura ARM Cortex-A76 a 2,4 GHz, se ha escogido el modelo de 8 GB de RAM. En la Figura 9 se muestra el aspecto físico de la placa [22].

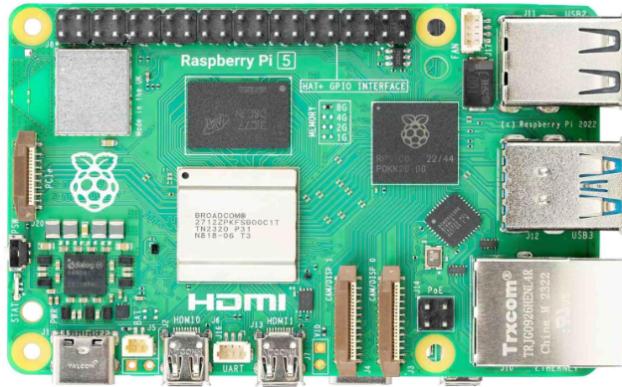


Figura 9: Vista frontal de Raspberry Pi 5 [24]

La principal ventaja frente a su predecesora es la incorporación del RP1, un chip de silicio que actúa controlador periférico. Según sus desarrolladores, es el programa de ingeniería más complicado y caro que han creado en Raspberry Pi. Este controlador gestiona la mayoría de las operaciones de entrada/salida, evitando que lo haga el procesador principal.

La flexibilidad de configuración de este chip RP1, dota a la Raspberry Pi 5 de la posibilidad de realizar multitud de funciones a los pines GPIO. La Tabla 1 extraída de la documentación oficial del RP1, muestra las opciones de configuración de los pines [22].

Tabla 1: Selección de funciones GPIO [25]

Function									
GPIO	a0	a1	a2	a3	a4	a5	a6	a7	a8
0	SPI0_SIO[3]	DPL_PCLK	UART1_TX	I2C0_SDA		SYS_RIO[0]	PROC_RIO[0]	PIO[0]	SPI2_CSn[0]
1	SPI0_SIO[2]	DPL_DE	UART1_RX	I2C0_SCL		SYS_RIO[1]	PROC_RIO[1]	PIO[1]	SPI2_SIO[1]
2	SPI0_CSn[3]	DPL_VSYNC	UART1_CTS	I2C1_SDA	UART0_JR_RX	SYS_RIO[2]	PROC_RIO[2]	PIO[2]	SPI2_CSn[0]
3	SPI0_CSn[2]	DPL_HSYNC	UART1_RTS	I2C1_SCL	UART0_JR_TX	SYS_RIO[3]	PROC_RIO[3]	PIO[3]	SPI2_SCLK
4	GPCLK[0]	DPL_D[0]	UART2_TX	I2C2_SDA	UART0_RI	SYS_RIO[4]	PROC_RIO[4]	PIO[4]	SPI3_CSn[0]
5	GPCLK[1]	DPL_D[1]	UART2_RX	I2C2_SCL	UART0_DTR	SYS_RIO[5]	PROC_RIO[5]	PIO[5]	SPI3_SIO[1]
6	GPCLK[2]	DPL_D[2]	UART2_CTS	I2C3_SDA	UART0_DCD	SYS_RIO[6]	PROC_RIO[6]	PIO[6]	SPI3_SIO[0]
7	SPI0_CSn[1]	DPL_D[3]	UART2_RTS	I2C3_SCL	UART0_DSR	SYS_RIO[7]	PROC_RIO[7]	PIO[7]	SPI3_SCLK
8	SPI0_CSn[0]	DPL_D[4]	UART3_TX	I2C0_SDA		SYS_RIO[8]	PROC_RIO[8]	PIO[8]	SPI4_CSn[0]
9	SPI0_SIO[1]	DPL_D[5]	UART3_RX	I2C0_SCL		SYS_RIO[9]	PROC_RIO[9]	PIO[9]	SPI4_SIO[0]
10	SPI0_SIO[0]	DPL_D[6]	UART3_CTS	I2C1_SDA		SYS_RIO[10]	PROC_RIO[10]	PIO[10]	SPI4_SIO[1]
11	SPI0_SCLK	DPL_D[7]	UART3_RTS	I2C1_SCL		SYS_RIO[11]	PROC_RIO[11]	PIO[11]	SPI4_SCLK
12	PWM0[0]	DPL_D[8]	UART4_TX	I2C2_SDA	AUDIO_OUT_L	SYS_RIO[12]	PROC_RIO[12]	PIO[12]	SPI5_CSn[0]
13	PWM0[1]	DPL_D[9]	UART4_RX	I2C2_SCL	AUDIO_OUT_R	SYS_RIO[13]	PROC_RIO[13]	PIO[13]	SPI5_SIO[1]
14	PWM0[2]	DPL_D[10]	UART4_CTS	I2C3_SDA	UART0_RX	SYS_RIO[14]	PROC_RIO[14]	PIO[14]	SPI5_SIO[0]
15	PWM0[3]	DPL_D[11]	UART4_RTS	I2C3_SCL	UART0_RX	SYS_RIO[15]	PROC_RIO[15]	PIO[15]	SPI5_SCLK
16	SPI1_CSn[2]	DPL_D[12]	MPI0_DSL_TE		UART0_CTS	SYS_RIO[16]	PROC_RIO[16]	PIO[16]	
17	SPI1_CSn[1]	DPL_D[13]	MPI1_DSL_TE		UART0_RTS	SYS_RIO[17]	PROC_RIO[17]	PIO[17]	
18	SPI1_CSn[0]	DPL_D[14]	I2S0_SCLK	PWM0[2]	I2S1_SCLK	SYS_RIO[18]	PROC_RIO[18]	PIO[18]	GPCLK[1]
19	SPI1_SIO[1]	DPL_D[15]	I2S0_WS	PWM0[3]	I2S1_WS	SYS_RIO[19]	PROC_RIO[19]	PIO[19]	
20	SPI1_SIO[0]	DPL_D[16]	I2S0_SD[0]	GPCLK[0]	I2S1_SDI[0]	SYS_RIO[20]	PROC_RIO[20]	PIO[20]	
21	SPI1_SCLK	DPL_D[17]	I2S0_SD[0]	GPCLK[1]	I2S1_SDI[0]	SYS_RIO[21]	PROC_RIO[21]	PIO[21]	
22	SDI00_CLK	DPL_D[18]	I2S0_SD[1]	I2C3_SDA	I2S1_SD[1]	SYS_RIO[22]	PROC_RIO[22]	PIO[22]	
23	SDI00_CMD	DPL_D[19]	I2S0_SD[1]	I2C3_SCL	I2S1_SD[1]	SYS_RIO[23]	PROC_RIO[23]	PIO[23]	
24	SDI00_DAT0	DPL_D[20]	I2S0_SD[2]	AUDIO_IN_CLK	I2S1_SD[2]	SYS_RIO[24]	PROC_RIO[24]	PIO[24]	SPI2_CSn[1]
25	SDI00_DAT1	DPL_D[21]	I2S0_SD[2]	AUDIO_IN_CLK	I2S1_SD[2]	SYS_RIO[25]	PROC_RIO[25]	PIO[25]	SPI3_CSn[1]
26	SDI00_DAT2	DPL_D[22]	I2S0_SD[3]	AUDIO_IN_DAT0	I2S1_SD[3]	SYS_RIO[26]	PROC_RIO[26]	PIO[26]	SPI3_CSn[1]
27	SDI00_DAT3	DPL_D[23]	I2S0_SD[3]	AUDIO_IN_DAT1	I2S1_SD[3]	SYS_RIO[27]	PROC_RIO[27]	PIO[27]	SPI4_CSn[1]

Como se puede observar en la Tabla 1, los pines 20, 22, 24 y 26 pueden ser configurados como pines de SDI (Serial Data In) y los pines 21, 23, 25 y 27 como SDO (Serial Data Out). En su configuración típica cada pin SDI y SDO puede emitir o recibir información por dos canales, lo que significa que se podrían conseguir hasta 8 canales de entrada y 8 canales de salida. Este

es el principal motivo por el que se ha escogido la Raspberry Pi 5 como núcleo de procesamiento del sistema.

Una posible desventaja es que, al incorporar un nuevo sistema de manejo de los periféricos hardware, algunas de las librerías más utilizadas de programación en Python no podrán usarse en este nuevo modelo.

Con el fin de aportar más información sobre el dispositivo seleccionado, se enumeran algunas de sus características principales extraídas de su página web oficial [15]:

- GPU VideoCore VII, compatible con OpenGL ES 3.1, Vulkan 1.2
- Salida de pantalla dual 4Kp60 HDMI® con soporte HDR
- Decodificador HEVC de 4Kp60
- Wi-Fi® 802.11ac de doble banda
- Ranura para tarjetas microSD, con soporte para el modo SDR104 de alta velocidad
- 2 puertos USB 3.0, que admiten una operación simultánea de 5 Gbps
- 2 puertos USB 2.0
- 2 transceptores de cámara/pantalla MIPI de 4 carriles
- Alimentación de 5V/5A CC a través de USB-C, con soporte de entrega de energía
- Cabezal estándar de 40 pines Raspberry Pi
- Botón de encendido

Pulsador momentáneo

Con el fin de controlar el correcto apagado y encendido del sistema se ha escogido un pulsador momentáneo LED (Un pulsador simple que incluye en su estructura un LED). El LED que incorpora el pulsador es de color azul. En la Figura 10 se muestra el aspecto del pulsador.



Figura 10: Pulsador Momentáneo LED

4.2 Módulo de captura de audio

Los componentes fundamentales para el diseño del Módulo de Captura de Audio son el ADC [24] y los conectores para la entrada de audio.

4.2.1 ADC

El elemento principal para realizar la captura de audio tiene que ser un ADC de uno o dos canales de entrada, que sea compatible con el protocolo I²S, que pueda gestionar datos de 24 bits y que acepte frecuencias de muestreo de 48 kHz.

PCM1808

El PCM1808 es un ADC estéreo que soporta 24 bits, tasas de muestreo de hasta 96 kHz y puede configurarse para utilizar el formato de transmisión de datos I²S en modo maestro y esclavo. A continuación, se enumeran algunas de las características extraídas del *datasheet* [16]:

- Convertidor A/D estéreo DeltaSigma de 24 bits
- Entrada de voltaje de un solo extremo: 3 Vpp • Alto rendimiento:
 - THD + N: – 93 dB (típico)
 - SNR: 99 dB (típico)
 - Rango dinámico: 99 dB (típico)
- Filtro de diezmado de sobremuestreo:
 - Frecuencia de sobremuestreo: × 64
 - Ondulación de banda de paso: ± 0,05 dB
 - Atenuación de banda de parada: – 65 dB
 - Filtro de paso alto en chip: 0,91 Hz (48 kHz)
- Interfaz de audio PCM flexible
 - Modo maestro/esclavo seleccionable
 - Formatos de datos: 24 bits I2S, 24 bits Left Justified
- Apagado y reinicio deteniendo el sistema Reloj
- Filtro Anti-alias analógico (LPF) incluido
- Frecuencia de muestreo: 8 kHz – 96 kHz
- Reloj del sistema: 256fs, 384fs, 512fs
- Fuentes de alimentación duales:
 - 5 V para analógico
 - 3,3 V para digital

En la Figura 11, se muestra el aspecto del chip y el diagrama de bloques funcional. En el apartado pertinente al diseño y conexionado de este dispositivo se profundizará en sus funciones y modos de configuración.

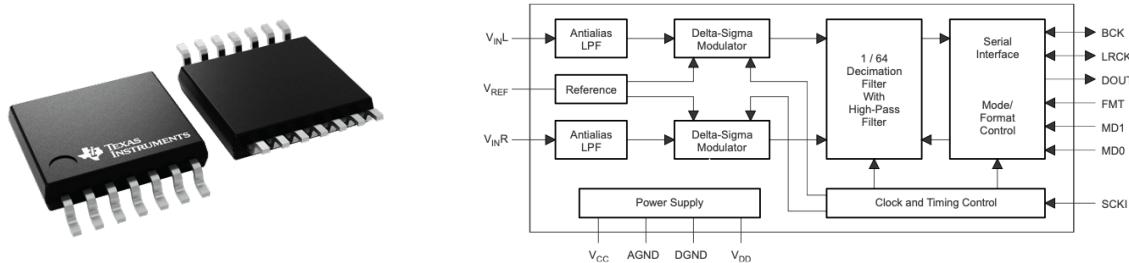


Figura 11: PCM1808: Vista frontal y trasera del chip (izquierda) [27] y Diagrama de Bloques Funcional (derecha) [17]

4.2.2 Conectores

Para completar el diseño, serán necesarios conectores dedicados al audio, a través de los cuales el usuario pueda introducir las señales acústicas en el sistema y lleguen a los ADC. En equipos de audio profesional, los conectores más utilizados son: el conector XLR y el Jack de $\frac{1}{4}$ de pulgada (6,35 mm) [25].

Conector de audio combinado

Un combo XLR/Jack es un conector que combina dos tipos de conexiones en un solo puerto, una conexión XLR hembra y una conexión Jack hembra de $\frac{1}{4}$ de pulgada (6,35 mm). Este diseño permite al usuario conectar tanto micrófonos (que generalmente usan conectores XLR), como instrumentos o dispositivos de línea (que suelen usar conectores Jack) al mismo puerto, proporcionando gran flexibilidad.

Se ha seleccionado este conector por la capacidad de adaptación que proporciona al sistema. Hay equipos que cuentan solo con conectores Jack o solo con conectores XLR, lo que podría dificultar la interacción entre el Grabador/Reproductor y el equipo externo. En la Figura 12 se muestra el conector y un esquema. El esquema se ha obtenido de la página oficial de Neutrik [29] y aunque no sea de la misma marca el utilizado, servirá al ser igual la configuración de los pines. Se utilizará principalmente en la fase de Prototipado y Montaje para determinar que señal debe ir en cada pin.

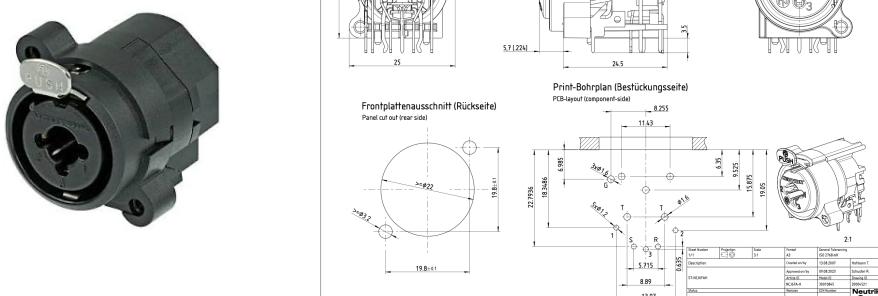


Figura 12: Conector Combo (izquierda) y esquema de NCJ6FA-H (derecha) [29]

4.3 Módulo de reproducción de audio

Los componentes fundamentales para el diseño del Módulo de Reproducción de Audio son el DAC y los conectores para la salida de audio.

4.3.1 DAC

Como elemento principal para la reproducción de audio se necesita un DAC [24] de uno o dos canales de salida, que sea compatible con el protocolo I²S, que pueda gestionar datos de 24 bits y que acepte frecuencias de muestreo de 48 kHz.

MAX98357A

Se ha escogido el chip MAX98357A incorpora un amplificador de clase D y un DAC mono, con decodificador de modulación de código de pulso digital (PCM), es de tamaño reducido, bajo coste y compatible con el protocolo I²S, solo necesitando para su funcionamiento las señales de reloj WS y SCLK, y la señal de datos SD, por lo que su configuración será siempre en modo esclavo. Se puede configurar con tasa de muestreo de 8 kHz a 96 kHz y en I²S se opera con datos de 16, 24 y 32 bits.

A continuación, se mostrarán algunas de las propiedades extraídas del *datasheet* [17] que caracterizan al dispositivo.

- Operación con suministro único (2,5 V a 5,5 V)
 - Potencia de salida de 3,2 W a 4 Ω y 5V
 - Corriente de reposo de 2,4 mA
 - 92% de eficiencia ($RL = 8\Omega$, $POUT = 1W$)
 - Ruido de salida de 22,8 μ VRMS ($AV = 15$ dB)
 - THD+N bajo de 0,013 % a 1 kHz
 - No se requiere MCLK
 - Frecuencias de muestreo de 8 kHz a 96 kHz
 - Admite salida izquierda, derecha o (izquierda/2 + derecha/2)
 - PSRR de 77 dB a 1 kHz
 - Circuito de reducción de clics y pop
 - Robusta protección térmica y contra cortocircuitos

En la Figura 13, se muestra el aspecto del chip y el diagrama de bloques funcional extraído de su *datasheet*. En el apartado pertinente al diseño y conexionado de este dispositivo se profundizará en sus funciones y modos de configuración.

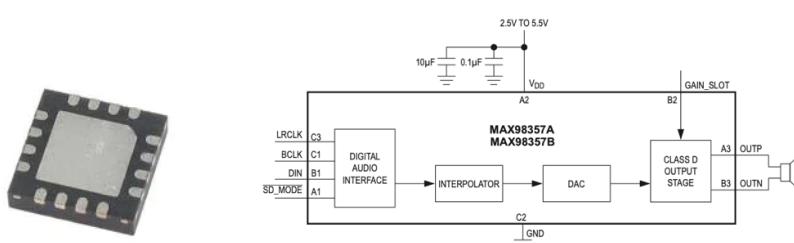


Figura 13: MAX98357A: Vista frontal (izquierda) [30] y Diagrama de Bloques Funcional (derecha) [18]

4.3.2 Conectores

Los conectores más utilizados para salidas de audio profesional son el XLR, el Jack de $\frac{1}{4}$ de pulgada (6,35 mm), el Speakon y el RCA. El Speakon es un conector orientado principalmente a transmitir señales ya amplificadas a altavoces y el RCA se suele utilizar para salidas de audio estéreo desbalanceadas [25].

Se ha prescindido de los Speakon y RCA porque no son tan utilizados en equipos de audio profesional en comparación con el XLR y el Jack. Por lo que como conectores de salida se escogerán los mismos que para la entrada. Estos son los conectores combinados XLR/Jack (explicados al detalle anteriormente).

4.4 Módulo de control de volumen

Se busca un método de control de volumen simple e intuitivo, por lo que se han escogido los potenciómetros analógicos como medio de interacción con el usuario. Para comunicar estos periféricos hardware con el núcleo de procesamiento, se ha optado por el chip MCP3008 [16], que facilita en gran medida el proceso de transmisión y recepción de información.

Potenciómetros analógicos

Estos dispositivos electrónicos permiten ajustar la resistencia eléctrica de un circuito, funcionando como resistencias variables. Son frecuentemente utilizados en gran variedad de equipos de audio para realizar diferentes funciones.

En este proyecto se utilizará para controlar el volumen de los canales de audio. Para esta función los más utilizados en equipos de audio profesional son los potenciómetros lineales deslizantes y los rotativos, se muestra un ejemplo de cada uno en la Figura 14.



Figura 14: Potenciómetro Lineal Deslizante (izquierda) [31] y Potenciómetro Lineal Rotativo (derecha) [32]

Ambos potenciómetros mostrados en la Figura 14, actúan de igual manera, aumentando o disminuyendo su resistencia al mover la perilla, pero el principal motivo por el que se ha seleccionado el rotativo es porque ocupa menos espacio.

Los potenciómetros lineales rotativos tienen tres terminales, tal y como se muestra en la Figura 14. Los terminales 1 y 3 (los de los extremos) están conectados a una pista resistiva interna y el terminal 2 (central) está conectado a un contacto deslizante o *wiper*. Al girar la

Selección de componentes

clavija del potenciómetro, el *wiper* se mueve a lo largo de la pista resistiva, modificando la proporción de resistencia. El potenciómetro se implementa en el circuito conectando uno de los terminales laterales a tierra y el del otro lateral otro a la alimentación, y el terminal central actúa como salida analógica. La Figura 15 muestra el esquema de los terminales de un potenciómetro.



Como se muestra en la Figura 15, la resistencia entre los terminales 1 y 3 siempre será la misma, pero en el terminal 2 varía. Hay potenciómetros de diferentes valores de resistencia, para este proyecto se han seleccionado de $10\text{ k}\Omega$. Se sabe que el máximo voltaje que se aplicará al circuito en el que se encuentren los potenciómetros es de 5 V, por lo que $10\text{ k}\Omega$ es más que suficiente para que actúen correctamente. El voltaje entre los terminales 2 y el que esté conectado a la alimentación (se supondrá el 1 para el ejemplo) es el que refleja el voltaje de la salida analógica. Este valor se calcula de la siguiente manera:

$$R_{13} = R_{Total}$$

$$R_{12} = \%Giro_{12} \cdot R_{13}$$

$$R_{13} = \%Giro_{23} \cdot R_{13}$$

$$V_{12} = \frac{V_f \cdot R_{12}}{R_{12} + R_{23}}$$

MCP3008

Es un ADC de 8 canales fabricado por Microchip Technology. Es ampliamente utilizado en proyectos de electrónica y sistemas embebidos para convertir señales analógicas en datos digitales.

Se ha escogido este dispositivo esencialmente por su fácil transmisión de datos a través de SPI, y sus 8 canales de entrada, lo cual se adecua al número de potenciómetros a implementar. También destaca su fácil manejo en Python, al haber librerías orientadas exclusivamente a su control. En la Figura 16 se muestra el aspecto del MCP3008 y su diagrama de bloque funcional [18].

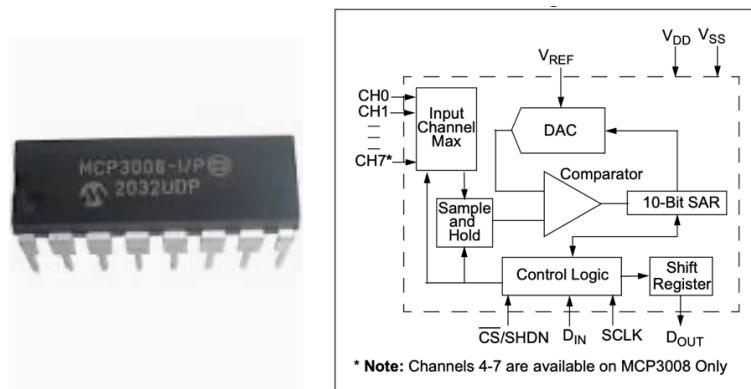


Figura 16: MCP3008: Vista frontal (izquierda) [34] y Diagrama de Bloques Funcional (derecha) [19]

A continuación, se enumeran algunas de sus características extraídas del *datasheet* [18]:

- Resolución de 10 bits
- ± 1 LSB de error máximo en la no linealidad diferencial (DNL)
- ± 1 LSB de error máximo en la no linealidad integral (INL)
- 4 (MCP3004) o 8 (MCP3008) canales de entrada
- Entradas analógicas programables como individuales o pares pseudo-diferenciales
- Muestra y retención integrados en el chip
- Interfaz serie SPI (modos 0,0 y 1,1)
- Operación con fuente de alimentación única: 2.7V - 5.5V
- Tasa máxima de muestreo de 200 ksps con VDD = 5V
- Tasa máxima de muestreo de 75 ksps con VDD = 2.7V
- Tecnología CMOS de bajo consumo
- Corriente en reposo típica de 5 nA, máximo 2 μ A
- Corriente activa máxima de 500 μ A a 5V
- Rango de temperatura industrial: -40°C a +85°C

4.5 Módulo Interfaz gráfica

Esta sección solo contará con un Display. En esta pantalla se mostrará la interfaz gráfica desde la que el usuario tendrá total acceso a las funciones del grabador/reproductor. El Display se ha escogido teniendo en cuenta las restricciones de espacio y según el grado de compatibilidad de conexión con la Raspberry Pi. En primera instancia, el control a través del Display fue planteado para que se llevara a cabo mediante botones, lo que supondría añadir otro elemento más al panel frontal, en el que no se dispone de mucho espacio. También dificultaría el proceso de programación y el aumento de conexiones al añadir periféricos. Por estos motivos se prescindió de los botones y se exploró la idea de utilizar una pantalla táctil.

Display

Tras realizar las medidas pertinentes se llegó a la conclusión de que el tamaño idóneo de la pantalla era de 4,3 pulgadas, por lo que se escogió el Display que se muestra en la Figura 17. El método de conexión entre la Raspberry Pi y el Display es el DSI [30]. Es una pantalla diseñada exclusivamente para Raspberry Pi por lo que su integración es óptima.



Figura 17: Vista frontal y trasera del Display [35]

A continuación, se resumen algunas de las características del Display [29]:

- Tamaño: 4,3 pulgadas
- Resolución 800x480
- Conector DSI, Plug and Play, no necesita controlador.
- Pantalla táctil capacitiva
- Compatible con Raspberry, Ubuntu MATE, Kali, RetroPie, OpenElec, OSMC, sistema Arch, etc.
- Utilice solo la interfaz DSI. La conexión a la placa Raspberry Pi incluye alimentación.
- La frecuencia de actualización es de unos 60Hz.
- Fuerte anti-interferencia.

5. Selección de protocolo de audio

Los protocolos de más utilizados para interconectar elementos de audio son el USB, el I²S y el SPDIF. Este apartado se centra en explicar porque el protocolo seleccionado para establecer la conexión con DACs y ADCs ha sido el I²S, imponiéndose a los otros dos.

Comparativa entre USB, SPDIF e I²S

Estas tres interfaces se utilizan para la transmisión de datos de audio digital. Cada una tiene sus propias características y aplicaciones. En la Tabla 2 se destacan las diferencias más relevantes.

Tabla 2: Comparación de protocolos de audio USB, SPDIF e I²S

Protocolo	Topología	Conexiones necesarias	Modo de transmisión
USB	Bus Jerárquico	4	Full Duplex
SPDIF	Punto a Punto	2	Simplex
I ² S	Punto a Punto o Bus Simple	Mínimo 3	Full Duplex

En cuanto a las velocidades de transmisión, la Raspberry Pi 5 [15] cuenta con puertos USB 2.0 y USB 3.0 cuyas velocidades pueden alcanzar los 480 Mbps y los 5 Gbps respectivamente. SPDIF ronda los 2 Mbps [30]. La velocidad del protocolo I²S depende de las frecuencias de las señales de reloj y la configuración de los datos, en este caso con 48 kHz de frecuencia de muestreo y 24 bits, la tasa de transferencia podría llegar a los 2.304 Mbps aproximadamente [5].

Para elaborar la Figura 18, no se han tenido en cuenta las conexiones de alimentación y de masa externas al protocolo. El bus USB contiene las señales de alimentación y masa, mientras que SPDIF [29] e I²S [5] necesitan una fuente externa para alimentar al dispositivo. Como el protocolo seleccionado ha sido el I²S, se ha considerado que lo más simple y cómodo es que la Raspberry Pi dote al sistema de la alimentación necesaria.

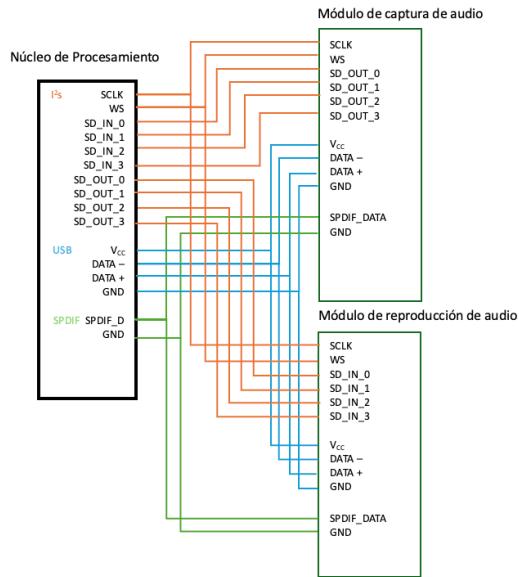


Figura 18: Conexiones necesarias entre el núcleo de procesamiento y los Módulos de Captura y Reproducción, teniendo en cuenta las interfaces USB, SPDIF e I²S

En la Figura 18 se observa que el protocolo I²S es el que más conexiones utiliza. El elevado número de señales que utiliza se traduce en fiabilidad al separar las funciones clave en líneas dedicadas.

El modo de transmisión de SPDIF es Simplex, por lo que solo se puede utilizar para enviar información y no para recibirla. Esta característica del protocolo lo hace inservible para este proyecto, ya que se pretende reproducir y capturar audio, cosa que el SPDIF no permite. Se podría utilizar SPDIF para la reproducción y otro protocolo para la captura, pero aumentaría la complejidad del sistema a la hora de la compatibilidad entre protocolos, la programación y el conexionado [29].

Para aplicaciones de audio donde la latencia baja y la calidad de sonido son prioritarios, I²S se presenta como la opción idónea. Su diseño especializado para la transmisión de audio asegura una experiencia de escucha de alta fidelidad con mínima demora, proporcionando una ventaja crucial sobre USB en este contexto. Mientras que USB ofrece versatilidad y alta velocidad para una gama más amplia de aplicaciones, I²S destaca en la entrega directa y precisa de datos de audio, haciendo de él la mejor opción para sistemas que valoran la calidad de audio y la eficiencia en la transmisión.

La utilización de USB hubiera sido completamente viable para la elaboración del proyecto, pero se ha querido profundizar en un protocolo menos explorado y estandarizado como es el I²S.

6. Diseño del Núcleo de Procesamiento

6.1 Configuraciones previas

A través de una tarjeta SD (en este caso de 64 GB), se introduce el Sistema Operativo en la Raspberry Pi. Se utiliza Raspberry Pi imager [30] para realizar la escritura en la tarjeta SD. En Raspberry Pi imager se ha seleccionado Raspberry Pi Os (64-bit) [31], una distribución del Sistema Operativo GNU/Linux basado en Debian y optimizado para su uso en Raspberry Pi. También mediante esta plataforma se han modificado campos como, la contraseña y se ha configurado la red Wifi.

Para controlar la Raspberry Pi se ha optado por VNC [32]. Esto permitirá utilizar la Raspberry a través de su interfaz gráfica. Para establecer la conexión a través de VNC, es necesario realizar algunos ajustes en el ordenador y en la Raspberry Pi:

1. Se descarga en el ordenador principal (se ha utilizado un MacBook Pro), XCode [33], MacPorts [34] y Putty [35].
2. Al haber conectado la Raspberry al wifi, se podrá obtener su dirección IP. Aplicaciones como IP Scanner [36] realizan esta función.

Discovered users & devices	IP addresses	MAC addresses
Raspberry Pi	192.168.0.24	D8:3A:DD:E3:12:68

Figura 19: Dirección IP obtenida con IP Scanner

3. Con Putty se accede por SSH [39] a la Raspberry Pi, introduciendo la dirección IP obtenida. Mediante el comando “sudo raspi-config” se accede al espacio de configuración de la Raspberry. En la ventana emergente se habilita el parámetro VNC, que estaba previamente desactivado.
4. Se instala en el ordenador principal RealVNC [37] e introduciendo la dirección IP de la Raspberry, el usuario y la contraseña (configuradas previamente), se habilita el acceso a la interfaz gráfica.

Teniendo acceso a la interfaz gráfica de Raspberry Pi Os (64-bit), se simplifica en gran medida la interacción con la Raspberry Pi 5. En la Figura 20, se muestra una imagen de su escritorio.



Figura 20: Escritorio de Raspberry Pi Os (64-bit)

6.2 Configuración de las interfaces

Es necesario modificar algunos ajustes para que las interfaces funcionen según las especificaciones del proyecto. Se utilizará SPI, GPIO y DSI para comunicarse con los periféricos hardware e I²S como interfaz para los datos de audio.

El archivo config.txt será fundamental para realizar muchos de los ajustes necesarios. La GPU lee el archivo config.txt antes de que la CPU Arm y Linux se inicialicen, por lo que cuando se realice un cambio en este archivo, se tendrá que reiniciar la Raspberry Pi para que se actualice la configuración realizada. A este archivo se accede introduciendo en el terminal el siguiente comando:

```
sudo nano /boot/firmware/config.txt
```

I²S

La Raspberry Pi debe ser configurada para que sea capaz de reconocer a los módulos de captura y reproducción de audio, también debe poder enviar información por ocho canales independientes (de entrada y de salida) y utilice I²S en modo maestro [5] [38].

El sistema que se encarga de controlar el audio en Linux es el Advanced Linux Sound Architecture (conocido como ALSA [38]). Gestiona la configuración y el manejo de todos los dispositivos de sonido conectados al sistema. ALSA será el medio básico para capturar y recibir audio y se configurará de acuerdo con los requerimientos de este proyecto

El primer paso es habilitar la interfaz I²S accediendo al archivo config.txt desde la terminal. Se quita el “#” de la línea en la que pone “dtparam=i2s=on” y se comenta “dtparam=audio=on”, como se muestra en la Figura 21.

The screenshot shows a terminal window with the title 'File Edit Tabs Help'. The command 'GNU nano 7.2 /boot/firmware/config.txt *' is at the top. The text in the editor is as follows:

```
GNU nano 7.2 /boot/firmware/config.txt *
# For more options and information see
# http://rptl.io/configtxt
# Some settings may impact device functionality. See link above for details

# Uncomment some or all of these to enable the optional hardware interface
#dtparam=i2c_arm=on
dtparam=i2s=on
#dtparam=spi=on

# Enable audio (loads snd_bcm2835)
#dtparam=audio=on
```

Figura 21: Archivo config.txt

Una vez habilitado el I²S, es necesario añadir un dispositivo de captura y uno de reproducción a la configuración de ALSA. Estos deben estar configurados para que cuenten con 8 canales de entrada (dispositivo de captura), 8 canales de salida (dispositivo de reproducción) y que su formato de transmisión de datos sea el I²S en modo maestro. Los dispositivos son interpretados por ALSA mediante los overlays. Los overlays son una forma de modificar o

extender el árbol de dispositivos (*Device Tree* [38]) sin tener que modificar el archivo principal de *Device Tree* que describe el hardware del sistema.

Raspbian Pi Os [31] incluye la carpeta que se encuentra siguiendo la ruta /boot/overlays, en cuyo interior hay gran variedad de overlays, para dispositivos de audio, video, pantallas, etc.... Ninguno de los overlays ubicados en esta carpeta cumple los requisitos deseados, lo que significa que será necesario encontrar o desarrollar un overlay que cumpla con las especificaciones.

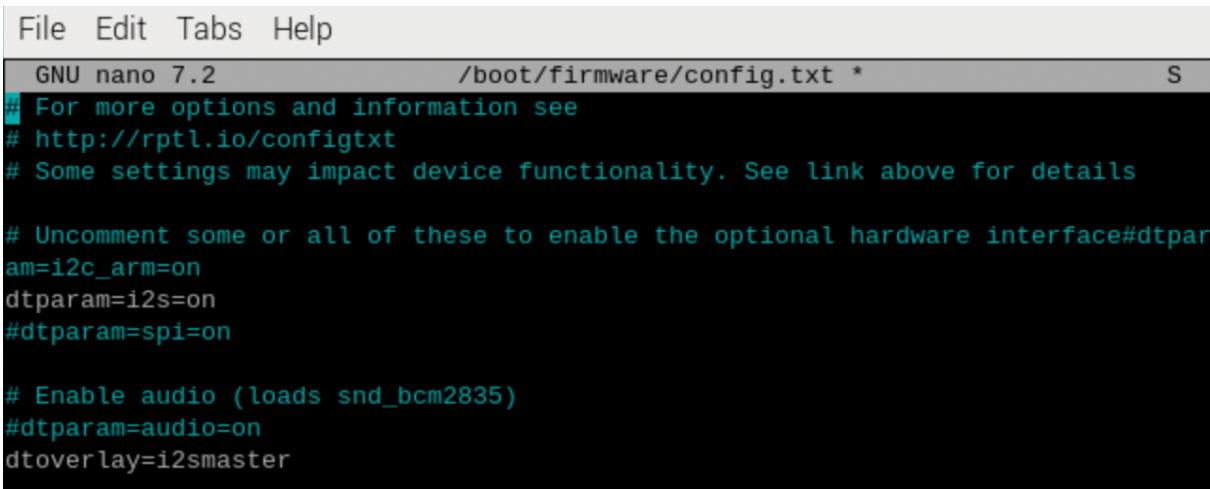
En GitHub se ha localizado un archivo fuente que cumple con lo previamente descrito [40]. El archivo tiene extensión .dts, por lo que habrá que compilarlo para obtener el archivo .dtbo definitivo (formato binario para aplicar overlays al *Device Tree*).

El archivo .dts se compila accediendo a la carpeta donde se encuentre el archivo e introduciendo en el terminal el siguiente comando:

```
dtc -@ -I dts -O dtb -o i2smaster.dtbo i2smaster.dts
```

Se obtiene el .dtbo y se copia el archivo en la carpeta overlays.

Se vuelve a modificar el archivo config.txt para cargar el overlay “i2smaster.dtbo”. Se añade la línea “dtoverlay = i2smaster” como se muestra en la Figura 22.



The screenshot shows a terminal window with the following content:

```
File Edit Tabs Help
GNU nano 7.2          /boot/firmware/config.txt *
# For more options and information see
# http://rpi.re�tina.org/configtxt
# Some settings may impact device functionality. See link above for details

# Uncomment some or all of these to enable the optional hardware interface
dtparam=i2c_arm=on
dtparam=i2s=on
#dtparam=spi=on

# Enable audio (loads snd_bcm2835)
#dtparam=audio=on
dtoverlay=i2smaster
```

Figura 22: Archivo config.txt

Se comprueba que la configuración es correcta escribiendo en el terminal el comando:

```
arecord -l
```

Y el comando:

```
aplay -l
```

Estos comandos listan respectivamente los dispositivos de captura y de reproducción configurados en el sistema. Se muestran los resultados obtenidos en las Figuras 23 y 24.

```
**** List of CAPTURE Hardware Devices ****
card 0: i2smaster [i2smaster], device 1: 1f000a0000.i2s-dir-hifi dir-hifi-1 [1f0
00a0000.i2s-dir-hifi dir-hifi-1]
    Subdevices: 1/1
    Subdevice #0: subdevice #0
```

Figura 23: Listado de dispositivos de captura

```
**** List of PLAYBACK Hardware Devices ****
card 0: i2smaster [i2smaster], device 0: 1f000a0000.i2s-dit-hifi dit-hifi-0 [1f0
00a0000.i2s-dit-hifi dit-hifi-0]
    Subdevices: 1/1
    Subdevice #0: subdevice #0
```

Figura 24: Listado de dispositivos de reproducción

Por último, desde el kernel de Linux, se configuran los pines GPIO para que al habilitar el I²S los pines GPIO 20, 21, 22, 23, 24, 25, 26 y 27 actúen como entradas y salidas de datos (los pares entradas y los impares salidas). Se enumeran los pasos a seguir para lograr este objetivo:

1. Se clona el repositorio de GitHub que contiene el kernel 6.6 [41] de Linux con el siguiente comando [41]:

```
git clone --depth=1 --branch rpi-6.7.y https://github.com/raspberrypi/linux
```

2. Se realiza la configuración de kernel introduciendo las siguientes líneas en el terminal:

```
cd linux
KERNEL=kernel_2712
make bcm2712_defconfig
```

3. Se modifica el archivo que gestiona los periféricos de la Raspberry Pi 5 [23]:

```
sudo nano ~/linux/arch/arm/boot/dts/broadcom/rp1.dtsi
```

En el apartado referente a la función “i2s0” se añaden todos los pines GPIO, enumerados al comienzo de este apartado, de la siguiente manera:

```
rp1_i2s0_18_21: rp1_i2s0_18_21 {
    function = "i2s0";
    pins = "gpio18", "gpio19", "gpio20", "gpio22", "gpio24", "gpio26", "gpio21", "gpio23",
    "gpio25", "gpio27";
    bias-disable;
};
```

4. Se compila el kernel para guardar los ajustes, se copian algunos archivos y se reinicia la Raspberry:

```
make -j4 Image.gz modules dtbs
sudo make modules_install
sudo cp arch/arm64/boot/dts/broadcom/*.dtb /boot/firmware/
sudo cp arch/arm64/boot/dts/overlays/*.dtb* /boot/firmware/overlays/
sudo cp arch/arm64/boot/dts/overlays/README /boot/firmware/overlays/
sudo cp arch/arm64/boot/Image.gz /boot/firmware/kernel_2712.img
sudo reboot
```

La configuración de los pines I²S queda de la siguiente manera:

Tabla 3: Configuración de los pines GPIO de la interfaz I²S en Raspberry Pi 5

GPIO	SEÑAL	ID	CANALES
18	SCLK	-	-
19	WS	-	-
20	DIN	0	1 y 2
21	DOUT	0	1 y 2
22	DIN	1	3 y 4
23	DOUT	1	3 y 4
24	DIN	2	5 y 6
25	DOUT	2	5 y 6
26	DIN	3	7 y 8
27	DOUT	3	7 y 8

Para conocer cómo se realiza la gestión de las señales generadas por I²S en la Raspberry Pi, se puede acudir al archivo que se encuentra en la ruta linux/sound/soc/dwc/dwc-i2s.c. Las señales de reloj se calculan en la función “dw_i2s_hw_params”. El controlador I²S está en modo maestro “(DW_I2S_MASTER)”, lo que significa que la señal de selección de palabra estará ligada a la frecuencia de muestreo, adquiriendo su valor, mientras que el reloj de bit se calcula de la siguiente manera:

```
u32 bitclk = config->sample_rate * config->data_width * 2;
```

SPI

Habilitar las funciones del Serial Peripheral Interface [45] es sencillo. Se accede al archivo de configuración config.txt y se quita el “#” de la línea en la que pone “dtparam=spi=on”, como se muestra en la Figura 25. Al reiniciar el sistema se podrán utilizar los pines por defecto que emiten las líneas de la interfaz SPI.

```
File Edit Tabs Help
GNU nano 7.2                      /boot/firmware/config.txt
# For more options and information see
# http://rpi.ti.io/configtxt
# Some settings may impact device functionality. See link above for details

# Uncomment some or all of these to enable the optional hardware interfaces
#dtparam=i2c_arm=on
dtparam=i2s=on
dtparam=spi=on
```

Figura 25: Archivo Config.txt

GPIO

La forma más sencilla de gestionar los pines de propósito general es utilizando Python. Este lenguaje de programación cuenta con varias librerías para la gestión de los pines GPIO, como RPi.GPIO, WiringPi, pigpio o gpiodzero. Como ya se explicó en el apartado correspondiente a la selección de la plataforma principal del núcleo de procesamiento, en la Raspberry Pi 5 se modificó el chip que gestiona a los periféricos, lo que hace inservibles a la mayoría de las librerías antiguas [23]. Después de probarlas todas se llegó a la conclusión de que con la librería gpiodzero [43], se pueden realizar operaciones sencillas con los pines GPIO, lo cual es suficiente para controlar un simple pulsador y proporcionar alguna otra señal.

Solo es necesario un pin GPIO, configurado como salida para encargarse de encender y apagar el LED que el pulsador incorpora. También se utiliza una señal GPIO en estado alto en el Módulo de Reproducción de Audio.

DSI

DSI es un estándar recogido dentro de los estándares MIPI [28]. Específicamente es la interfaz definida por MIPI para la comunicación entre un procesador y una pantalla. DSI se utiliza para transferir datos de imagen de manera eficiente y rápida, mediante su conexión serie de alta velocidad, lo que permite transmitir grandes cantidades de datos utilizando menos pines y, por lo tanto, reduciendo el consumo de energía.

Para conectar el Display a la Raspberry Pi a través de DSI, no es necesaria ninguna configuración, se conecta directamente por un cable FFC/FPC al puerto de la Raspberry CAM/DISP 0 [23].

6.3 Esquema eléctrico

El núcleo de procesamiento se encarga de alimentar al resto del sistema. También envía y recibe señales de todos los elementos a los que está conectado.

En la Figura 26, se muestra el esquemático del Núcleo de Procesamiento. Para elaboración de los esquemas eléctricos se ha utilizado el Software KiCad [44].

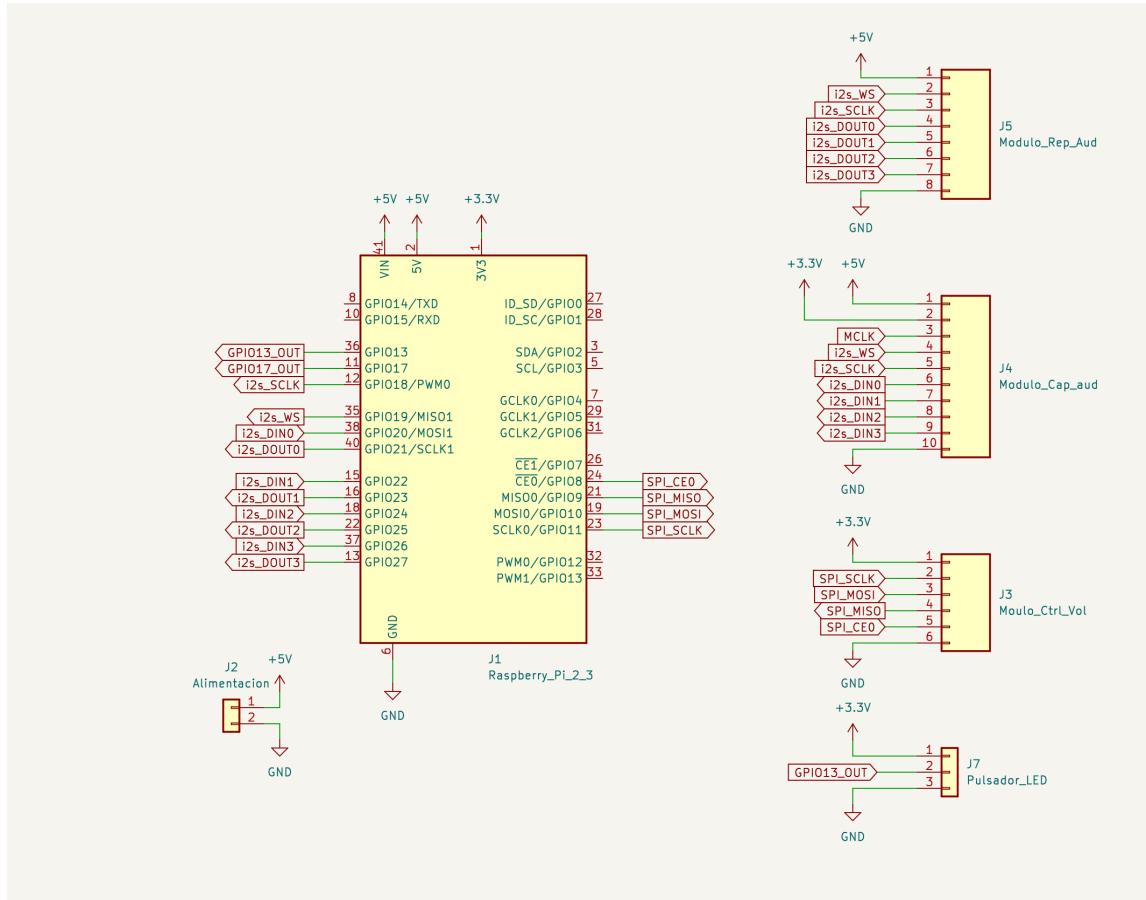


Figura 26: Esquemático del Núcleo de Procesamiento

El esquema eléctrico anterior da una visión general de cómo se realizan las conexiones entre la Raspberry Pi y el resto de los componentes que conforman el sistema. Se puede apreciar que los datos y la alimentación se transmiten a los periféricos de forma simple, a través de conexiones directas, suministrando a cada módulo las señales de la interfaz que precisa.

7. Diseño del Módulo de Captura de Audio

Se detallarán las configuraciones y conexiones realizadas, que conforman el Módulo de Reproducción de Audio.

7.1 Conexiones

El proceso de conversión analógico digital lo realizan chips PCM1808. Estos chips son estéreo, por lo que se utilizarán 4 para obtener los 8 canales de entrada deseados. Cada chip emite una señal de datos I²S y la envía a la Raspberry Pi, donde se realiza la multiplexación en tiempo para dividir la señal en dos canales de audio.

Se obtiene la información sobre los pines del chip acudiendo al *datasheet* [16]. En él se encuentra la función que desempeña cada uno de sus pines, como puede observarse en la Figura 27 y en la Tabla 4.

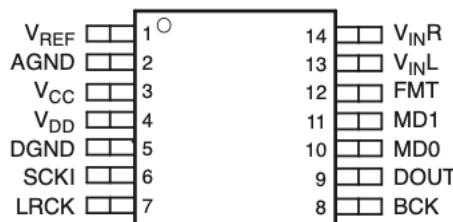


Figura 27: Esquema de la vista superior y distribución de los pines del chip PCM1808 [17]

Tabla 4: Descripción de los pines del chip PCM1808 [17]

TERMINAL NAME	PIN	I/O	DESCRIPTION
AGND	2	-	Analog GND
BCK	8	I/O	Audio data bit clock input/output ⁽¹⁾
DGND	5	-	Digital GND
DOUT	9	O	Audio data digital output
FMT	12	I	Audio interface format select ⁽²⁾
LRCK	7	I/O	Audio data latch enable input/output ⁽¹⁾
MD0	10	I	Audio interface mode select 0 ⁽²⁾
MD1	11	I	Audio interface mode select 1 ⁽²⁾
SCKI	6	I	System clock input; 256 f _S , 384 f _S or 512 f _S ⁽³⁾
V _{CC}	3	-	Analog power supply, 5-V
V _{DD}	4	-	Digital power supply, 3.3-V
V _{INL}	13	I	Analog input, L-channel
V _{INR}	14	I	Analog input, R-channel
V _{REF}	1	-	Reference voltage decoupling (= 0.5 V _{CC})

El *datasheet* también proporciona un diagrama típico de conexión (Figura 28). Se ha encontrado una PCB en el mercado que incorpora el chip PCM1808, que sigue el diagrama antes mencionado, por lo que para facilitar el proceso de montado se ha adquirido.

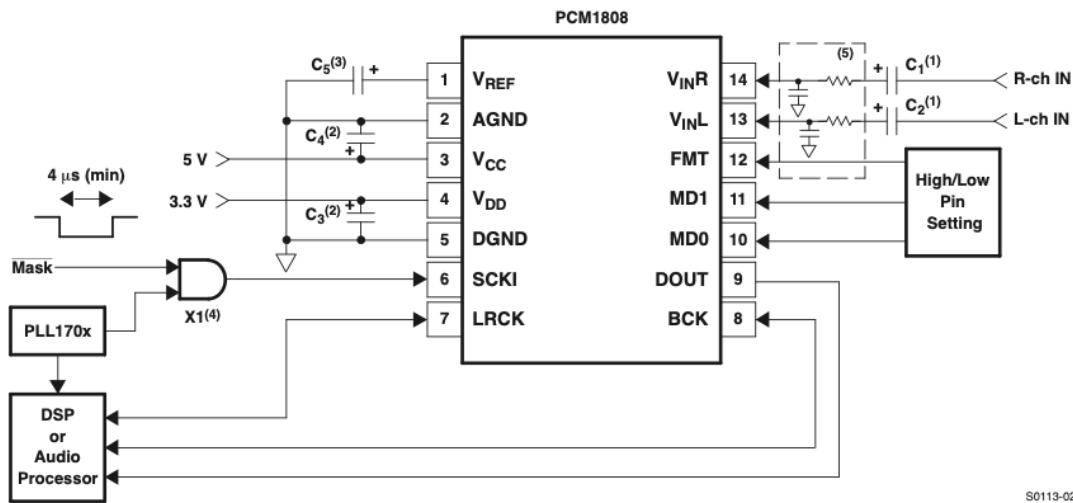


Figura 28: Diagrama de conexiones típicas del chip PCM1808 [17]

El aspecto de la PCB que implementa el diagrama de la Figura 29 es el siguiente.

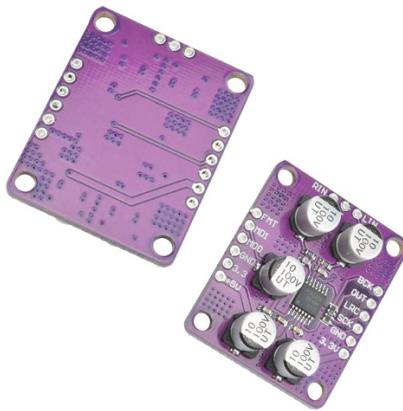


Figura 29: Chip PCM1808 implementado en PCB [55]

Con la información que las Figura 27 y la Tabla 4 aportan, se conoce que señales deben introducirse en cada pin de entrada y que señales salen de los pines de salida. Lo siguiente es saber los valores que debe tener cada señal para obtener el resultado deseado.

Modo de la interfaz

Lo primero es configurar el dispositivo en modo esclavo, para ello se conectan los pines MD0 y MD1 a tierra, como se indica en la Tabla 5, extraída del datasheet [16].

Tabla 5: Modos de Interfaz [17]

MD1 (Pin 11)	MD0 (Pin 10)	INTERFACE MODE
Low	Low	Slave mode (256 f _S , 384 f _S , 512 f _S autodetection)
Low	High	Master mode (512 f _S)
High	Low	Master mode (384 f _S)
High	High	Master mode (256 f _S)

Formato de datos

El pin FMT configura el formato de datos según la Tabla 6 extraída del *datasheet* [17]. Como se desea I²S, se conecta a tierra.

Tabla 6: Formato de datos [17]

FORMAT NO.	FMT (Pin 12)	FORMAT
0	Low	I ² S, 24-bit
1	High	Left-justified, 24-bit

Señales de reloj

Al estar configurado en modo esclavo, recibe las señales de reloj del dispositivo configurado en modo maestro, recibiendo las señales SCLK y WS generadas por la Raspberry Pi. Este dispositivo tiene un método de sincronización entre las señales de reloj algo diferente al usual en un equipo configurado como modo esclavo I²S. La diferencia radica en que, para una correcta transmisión de los datos, el PCM1808 debe recibir una señal de reloj maestra o SCKI. La frecuencia que debe tener la señal varía según la frecuencia de muestreo, pudiendo ser uno de los múltiplos de esta que se muestran en la Tabla 7, extraída del *datasheet* [16].

Tabla 7: Frecuencias de Muestreo y Frecuencias de Reloj Maestro [17]

SAMPLING FREQUENCY (kHz)	SYSTEM CLOCK FREQUENCY (f _{SCLK}) (MHz)		
	256 f _S	384 f _S	512 f _S
8	2.048	3.072	4.096
16	4.096	6.144	8.192
32	8.192	12.288	16.384
44.1	11.2896	16.9344	22.5792
48	12.288	18.432	24.576
64	16.384	24.576	32.768
88.2	22.5792	33.8688	45.1584
96	24.576	36.864	49.152

También se especifica que el valor de la frecuencia del reloj de bit debe ser la frecuencia de muestreo multiplicada por 64 o 48 (profundidad de bit multiplicada por dos como se muestra en la formula), lo que quiere decir que la profundidad de bit debe ser de 32 bits o 24 bits [16].

El requisito de la profundidad de bit se cumple, por lo que el PCM1808 no tendrá ningún problema a la hora de recibir la señal SCLK. Como la Raspberry Pi está configurada para emitir un reloj de selección de palabra de 48 kHz, se ha escogido una señal de reloj maestra de 12.288 MHz (256 x fs). La señal de reloj maestra la genera un cristal de cuarzo, ya que la Raspberry Pi no tiene ninguna función para generar señales de reloj precisas de frecuencia personalizada. Se intentó generar la señal de reloj con PWM, pero los resultados no eran del todo exactos.

Alimentación

Se conectan V_{CC} y V_{DD} a 5V y 3.3 V respectivamente. V_{CC} es la alimentación para el procesado analógico y V_{DD} para el procesado digital. AGND y DGND se conectan a tierra.

Salida de Datos

La señal que sale del pin DOUT del PCM es la señal digital que transporta los datos de audio, previamente convertidos a partir de la señal analógica. La señal de datos emitida por cada PCM1808 debe ser dirigida a los pines DIN de la Raspberry Pi.

Entrada de datos

En los pines V_{INL} V_{INR} se recibe la señal analógica por el canal izquierdo y derecho, y V_{REF} se conecta a tierra. Estos pines se conectan al conector de entrada.

Se ha elaborado la Tabla 8, la cual recoge que valores deben tener las señales para que cada chip PCM1808 una señal digital de 48 kHz de frecuencia de muestreo y 24 bits.

Tabla 8: Configuración de los pines del PCM1808 en el sistema

Pin PCM1808	Input/Output	Señal	Emisor
AGND	Input	Tierra	Raspberry Pi 5
DGND	Input	Tierra	Raspberry Pi 5
V_{CC}	Input	5 V	Raspberry Pi 5
V_{DD}	Input	3.3 V	Raspberry Pi 5
V_{INL}	Input	Canal L analógico	Fuente de audio
V_{INR}	Input	Canal R analógico	Fuente de audio
V_{REF}	Input	Tierra	Raspberry Pi 5
BCK	Input	64-BCK/frame	Raspberry Pi 5
SCKI	Input	256fs	Cristal de Cuarzo
LRCLK	Input	fs	Raspberry Pi 5
MDO	Input	Tierra	Raspberry Pi 5
MD1	Input	Tierra	Raspberry Pi 5
FMT	Input	Tierra	Raspberry Pi 5
I^2S DOUT	Output	I^2S DIN	PCM1808

La Tabla 9, relaciona cada dispositivo con el canal que transmite.

Tabla 9: Distribución de canales mediante la interfaz I^2S

PCM1808 ID	Pin Raspberry Pi I^2S DIN	CANAL ENTRADA
PCM1808_0	GPIO20	0 y 1
PCM1808_1	GPIO22	2 y 3
PCM1808_2	GPIO24	4 y 5
PCM1808_3	GPIO26	6 y 7

7.2 Diseño eléctrico

En la Figura 30 se muestra el esquemático del Módulo de Captura de Audio.

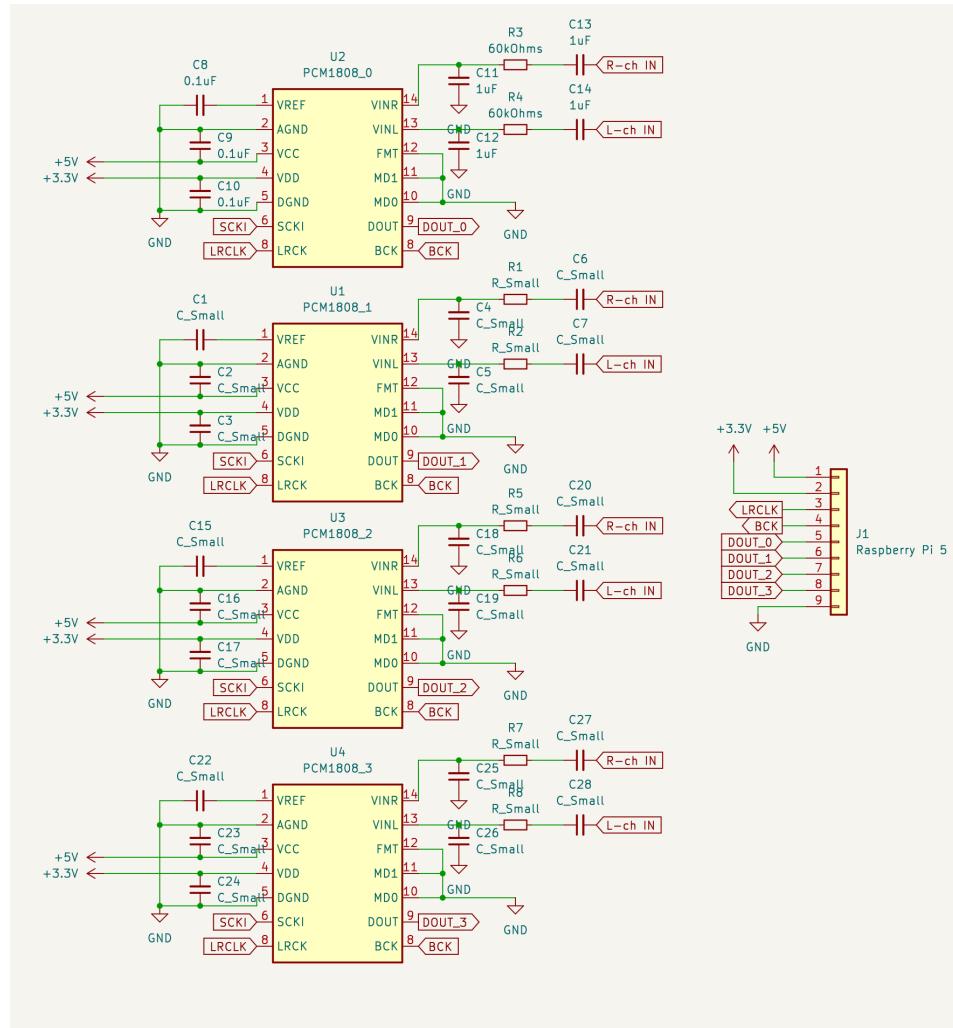


Figura 30: Esquemático del Módulo de Captura de Audio

8. Diseño del Módulo de Reproducción de Audio

Se detallarán las configuraciones y conexiones realizadas, que conforman el Módulo de Reproducción de Audio.

8.1 Conexiones

Se utilizarán 8 chips MAX98357A para realizar el proceso de conversión digital analógica. Al ser mono se necesita uno para cada canal de audio. Cada pin de salida de datos en formato I²S suministra dos canales (derecho e izquierdo o 0 y 1, 2 y 3, 4 y 5, 6 y 7), por lo que se conectarán dos chips MAX98357A a cada salida de datos. Para que cada chip reciba el canal que le corresponde se tendrán que realizar algunos ajustes.

Para poder configurar correctamente los chips MAX98357A, se acude a su *datasheet* [17]. El esquema y la tabla que más información aportan sobre el funcionamiento de sus pines se muestra en la Figura 31 y la Tabla 10.

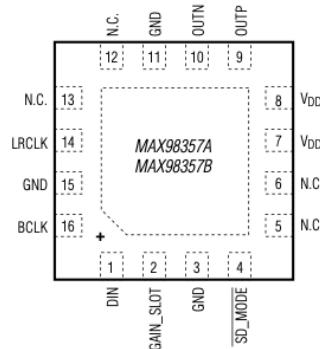


Figura 31: Esquema de la vista superior y distribución de los pines del chip MAX98357A [18]

Tabla 10: Descripción de los pines del chip MAX98357A [18]

PIN		NAME	FUNCTION
WLP	TQFN		
A1	4	SD_MODE	Shutdown and Channel Select. Pull SD_MODE low to place the device in shutdown. In I ² S or LJ mode, SD_MODE selects the data channel (Table 5). In TDM mode, SD_MODE and GAIN_SLOT are both used for channel selection (Table 7).
A2	7, 8	V _{DD}	Power-Supply Input
A3	9	OUTP	Positive Speaker Amplifier Output
B1	1	DIN	Digital Input Signal
B2	2	GAIN_SLOT	Gain and Channel Selection. In I ² S and LJ mode determines amplifier output gain (Table 8) In TDM mode, used for channel selection with SD_MODE (Table 7). In TDM mode, gain is fixed at 12dB.
B3	10	OUTN	Negative Speaker Amplifier Output
C1	16	BCLK	Bit Clock Input
C2	3, 11, 15	GND	Ground
C3	14	LRCLK	Frame Clock. Left/right clock for I ² S and LJ mode. Sync clock for TDM mode.
—	5, 6, 12, 13	N.C.	No Connection
—	—	EP	Exposed Pad. The exposed pad is not internally connected. Connect the exposed page to a solid ground plane for thermal dissipation.

Selección de canal

La selección de canales se realiza suministrando diferentes voltajes a los pines SD_MODE. Se utilizará un pin GPIO de la Raspberry Pi para que actúe como salida, proporcionando 3.3 V al pin SD_MODE. El canal en el que se configura el chip depende del voltaje que reciba SD_MODE, si recibe los 3.3 V el MAX98357A responde como canal izquierdo y si recibe 1.8 V, actúa como canal derecho (en DOUT0 los canales son 0 y 1 respectivamente). Para que disminuya el voltaje recibido el SD_MODE se añade una resistencia de 69.8 kΩ, entre el pin SD_MODE y la fuente. El *datasheet* [17] muestra de forma clara esta configuración, gracias a la Figura 32.

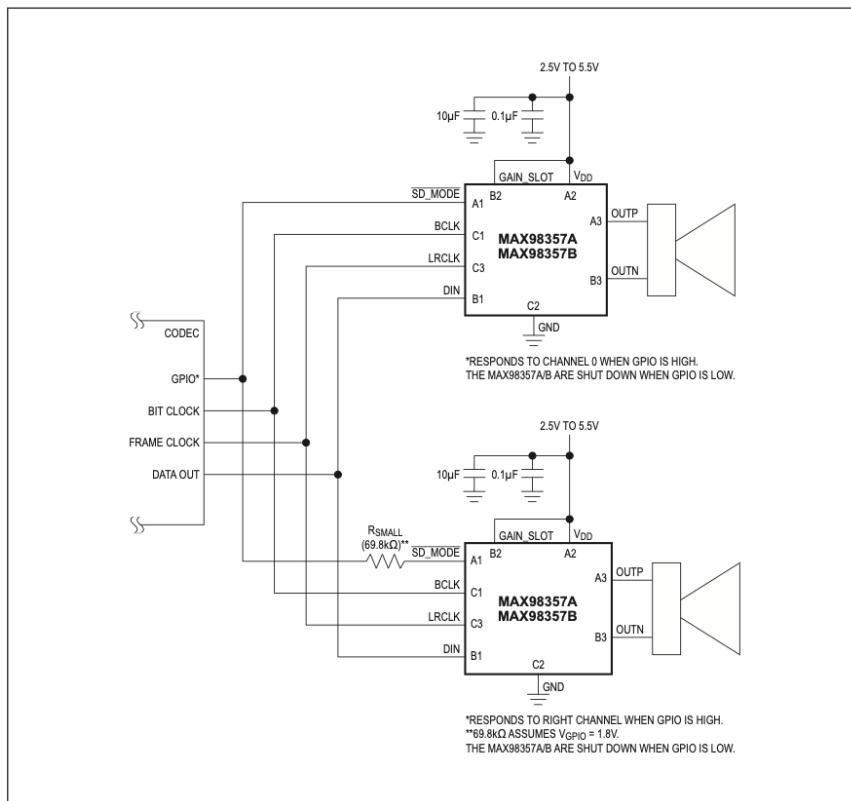


Figura 32: Configuración Estereo utilizando dos MAX98357A [18]

Ganancia

Se pueden configurar diferentes valores de ganancia según el voltaje que se suministre al pin GAIN_SLOT, como se muestra en la Tabla 11, extraída del *datasheet* [17].

Tabla 11: Selección de Ganancia [18]

GAIN_SLOT	I ² S/LJ GAIN (dB)
Connect to GND through 100kΩ ±5% resistor	15
Connect to GND	12
Unconnected	9
Connect to V _{DD}	6
Connect to V _{DD} through 100kΩ ±5% resistor	3

En un principio se elige la ganancia de 9 dB, sin conectar el pin GAIN_SLOT. En apartados posteriores de la memoria se detalla el proceso de selección de los niveles de salida.

A lo largo del *datasheet* [17] se aconsejan unas conexiones predeterminadas para cada pin. Con el objetivo de reducir la carga de trabajo y cumplir con todas estas recomendaciones se ha adquirido el chip integrado en una PCB que contiene los componentes recomendados (resistencias y condensadores). Además, su reducido tamaño, 17.7 mm x 19.1 mm, lo hace perfecto para que al ubicar los 8, no haya problemas de espacio. El aspecto y el esquemático de la PCB se muestran en las Figuras 33 y 34.



Figura 33: Vista frontal y trasera del chip MAX98357A integrado en una PCB [56]

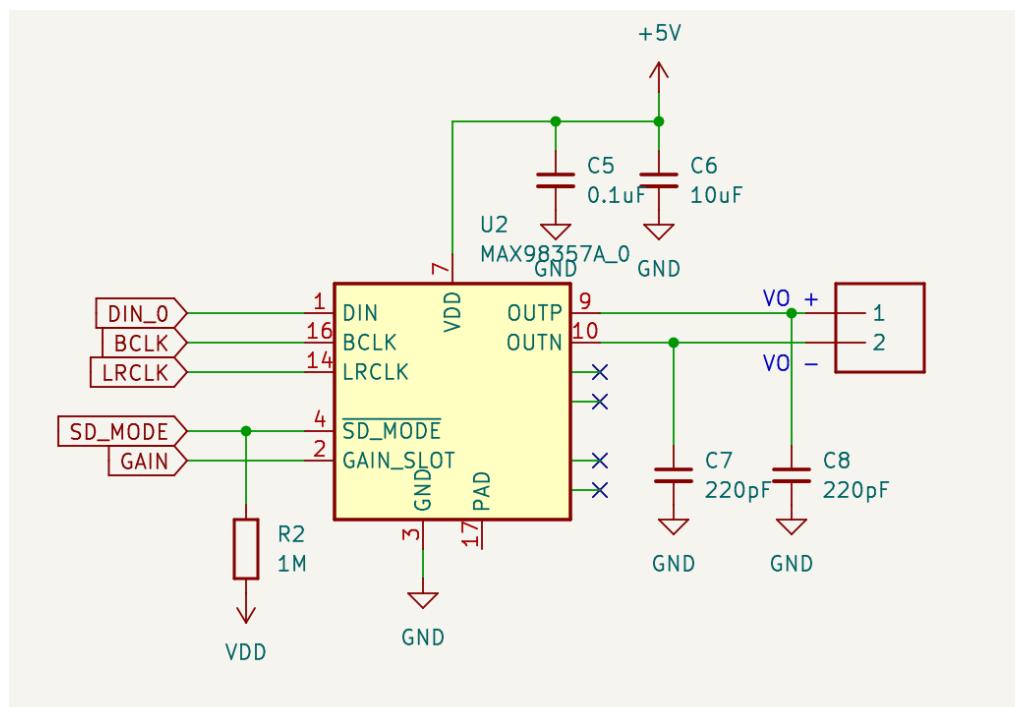


Figura 34: Diagrama de conexiones típicas del chip MAX98357A

Se ha elaborado la Tabla 12, la cual recoge que valores deben tener las señales para que cada chip MAX98357A procese la señal digital y la convierta en analógica según las especificaciones del sistema.

Tabla 12: Configuración de los pines del MAX98357A en el sistema

Pin MAX98357A	Input/Output	Señal	Emisor
GND	Input	Tierra	Raspberry Pi
V _{DD}	Input	5 V	Raspberry Pi
BCLK	Input	64-BCLK/frame	Raspberry Pi
LRCLK	Input	48 kHz	Raspberry Pi
GAIN_SLOT	Input	NC	-
SD_MODE	Input	GPIO17	Raspberry Pi
OUTP	Output	Receptor de audio	Receptor de audio
OUTN	Output	Receptor de audio	Receptor de audio

La Tabla 13, relaciona cada dispositivo con el canal que transmite.

Tabla 13: Distribución de canales mediante la interfaz I²S

MAX98357A ID	Pin Raspberry I ² S DOUT	CANAL SALIDA
MAX98357A_0	GPIO21	0
MAX98357A_1	GPIO21	1
MAX98357A_2	GPIO22	2
MAX98357A_3	GPIO22	3
MAX98357A_4	GPIO23	4
MAX98357A_4	GPIO23	5
MAX98357A_6	GPIO24	6
MAX98357A_7	GPIO24	7

8.2 Esquema eléctrico

En la Figura 35 se muestra el esquemático del Módulo de Reproducción de Audio.

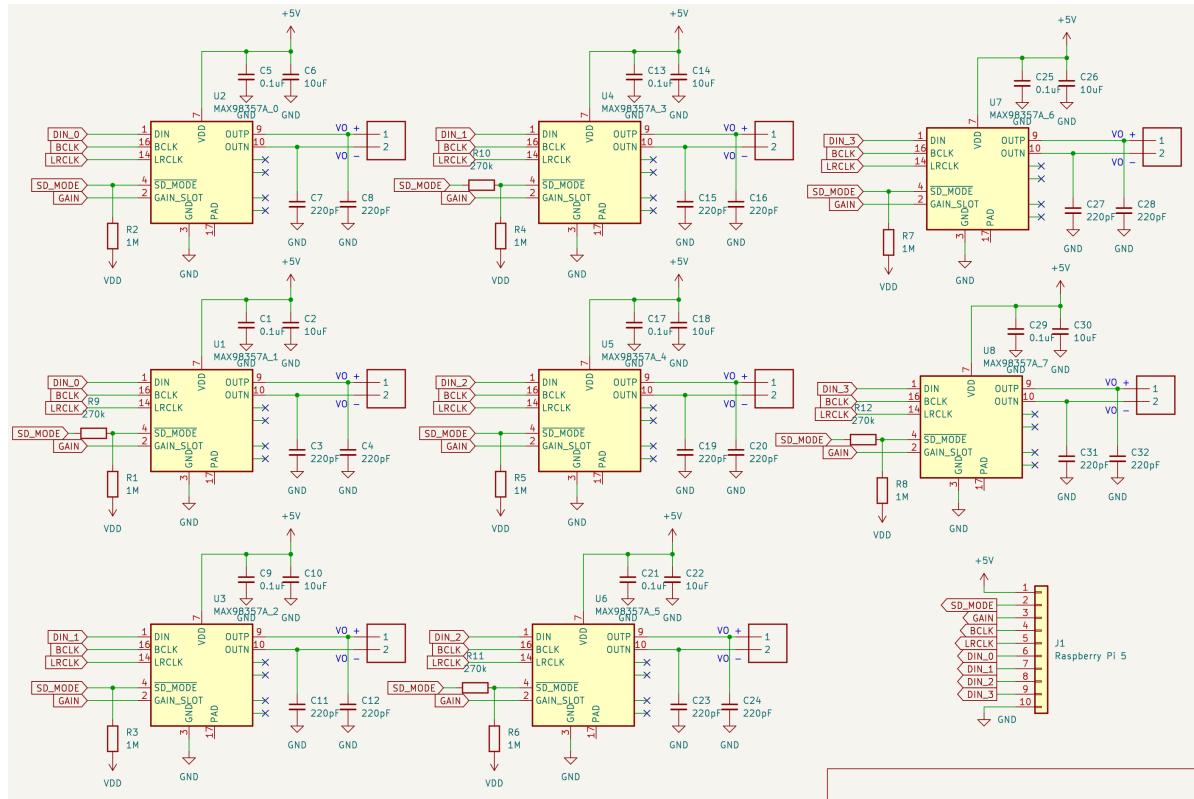


Figura 35: Esquemático del Módulo de Reproducción de Audio

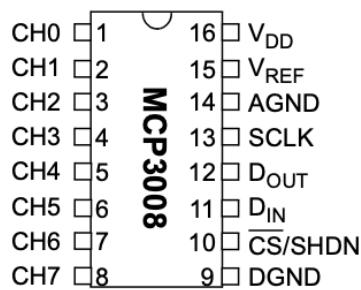
9. Diseño del Módulo de Control de Volumen

Se detallarán las configuraciones y conexiones realizadas, que conforman el Módulo de Control de Volumen.

9.1 Conexiones

Los elementos principales que conforman este módulo son, un chip MCP3008 y 8 potenciómetros lineales rotativos. El MCP3008 actúa como conversor analógico digital, convirtiendo las señales que recibe de cada potenciómetro a un idioma que la Raspberry Pi pueda comprender.

El chip se comunica con la Raspberry Pi 5 a través de la interfaz SPI, configurando a la Raspberry Pi en modo maestro y al chip en modo esclavo. Para saber que función desempeña cada pin y que señal emite o debe recibir, se acude a su *datasheet* [18] (Figura 36 y Tabla 14).



436: Esquema de la vista superior y distribución de los pines del chip MCP3008 [19]

Tabla 14: Descripción de los pines del chip MCP3008 [19]

MCP3008	Symbol	Description
PDIP, SOIC		
1	CH0	Analog Input
2	CH1	Analog Input
3	CH2	Analog Input
4	CH3	Analog Input
5	CH4	Analog Input
6	CH5	Analog Input
7	CH6	Analog Input
8	CH7	Analog Input
9	DGND	Digital Ground
10	CS/SHDN	Chip Select/Shutdown Input
11	D _{IN}	Serial Data In
12	D _{OUT}	Serial Data Out
13	SCLK	Serial Clock
14	AGND	Analog Ground
15	V _{REF}	Reference Voltage Input
16	V _{DD}	+2.7V to 5.5V Power Supply
-	NC	No Connection

La comunicación con el chip es sencilla, solo necesitando cuatro señales de la interfaz SPI, señal de alimentación, tierra y las señales analógicas de entrada.

Señales SPI

Simplemente habilitando SPI [42] en Raspberry Pi se obtienen las señales necesarias para comunicarse con el chip, no es necesaria ninguna configuración adicional.

- El pin **CS/SHDN** se conecta la señal CS0 (Chip Select 0), que se utiliza para seleccionar al esclavo con el que el maestro quiere comunicarse.
- El pin **DIN** recibe la señal de datos que envía el maestro, esta señal SPI se conoce como MOSI (Master Out Slave In)
- El pin **DOUT** emite los datos que son enviados del esclavo al maestro y la Raspberry Pi lo recibe mediante la señal MISO (Master In Slave Out). En este caso la información emitida es la convertida de analógico a digital proveniente de los potenciómetros.
- La señal emitida por la Raspberry Pi SCLK (Serial Clock), es la señal de reloj que sincroniza la transferencia de datos (MOSI y MISO), entre el maestro y el esclavo. El pin del MCP3008 que recibe la señal es el pin 13 o SCLK.

Señales de alimentación

La alimentación del Módulo de Control de Volumen se llevará a cabo desde los pines de alimentación de la Raspberry Pi. V_{DD} y V_{REF} se conectan a 3.3 V y AGND y DGND se conectan a tierra.

En el caso de los potenciómetros el terminal izquierdo se conecta a 3.3 V y el derecho a tierra (visto el potenciómetro desde atrás). Con esta configuración al girar la perilla a la derecha del todo la resistencia entre el terminal de voltaje y el central es mínima, por lo que el voltaje emitido por el potenciómetro es el mayor posible, mientras que al girar la perilla a la izquierda ocurre lo contrario.

Señales analógicas

El chip MCP3008 cuenta con 8 canales dedicados a la entrada de datos analógicos (pines del 1 al 8). Por cada uno de estos canales se introduce la señal analógica que emite de cada potenciómetro, conectando el pin central de los potenciómetros a uno de los pines analógicos del MCP3008.

La Tabla 15, recoge los valores que deben tener las señales para que el chip MCP3008 se comunique correctamente con la Raspberry Pi 5.

Tabla 15: Configuración de los pines del MCP3008 en el sistema

Pin MCP3008	Input/Output	Señal	Emisor
V_{DD}	Input	3.3 V	Raspberry Pi 5
V_{REF}	Input	3.3 V	Raspberry Pi 5
AGND	Input	Tierra	Raspberry Pi 5
DGND	Input	Tierra	Raspberry Pi 5
SCLK	Input	GPIO11 (SCLK)	Raspberry Pi 5
DOUT	Output	GPIO9 (MISO)	MCP3008
DIN	Input	GPIO10 (MOSI)	Raspberry Pi 5
CS/SHDN	Input	GPIO8 (CS0)	Raspberry Pi 5

Diseño del Módulo de Control de Volumen

La conexión entre los potenciómetros y el MCP se muestra en la Tabla 16(orden de los terminales visto desde atrás):

Tabla 16: Conexiones entre el MCP3008 y los Potenciómetros

Potenciómetro ID	Terminal izquierdo	Terminal central	Terminal derecho
Potenciómetro_0	3.3 V	Pin 1 MCP3008	Tierra
Potenciómetro_1	3.3 V	Pin 2 MCP3008	Tierra
Potenciómetro_2	3.3 V	Pin 3 MCP3008	Tierra
Potenciómetro_3	3.3 V	Pin 4 MCP3008	Tierra
Potenciómetro_4	3.3 V	Pin 5 MCP3008	Tierra
Potenciómetro_5	3.3 V	Pin 6 MCP3008	Tierra
Potenciómetro_6	3.3 V	Pin 7 MCP3008	Tierra
Potenciómetro_7	3.3 V	Pin 8 MCP3008	Tierra

9.2 Esquema eléctrico

En la Figura 37 se muestra el esquemático del Módulo de Control de Volumen.

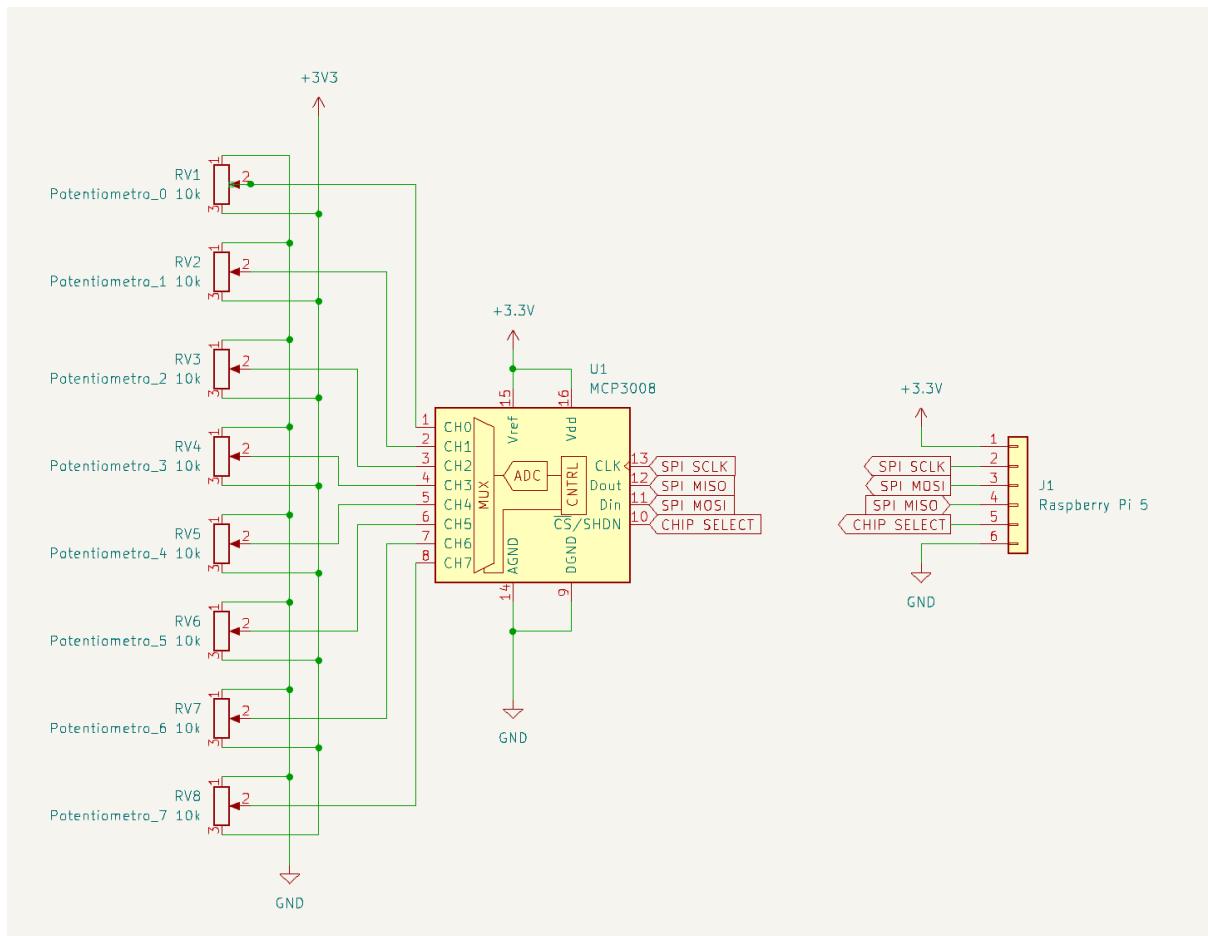


Figura 37: Esquemático del Módulo de Control de Volumen

10. Software

En este apartado se explica cómo se ha desarrollado el software que controla los procesos de grabación y reproducción de audio. Se ha diseñado el sistema para que las funciones del grabador/reproductor, sean controladas por el usuario mediante una Interfaz gráfica. También se detallará la programación relacionada con los periféricos hardware. Se ha utilizado Thonny [47] como entorno de desarrollo integrado y Python 3.11.2 [48] como lenguaje de programación, por las facilidades que proporciona a la hora de controlar los pines GPIO (librerías como gpiodzero) y para el procesado de audio (librerías como pyaudio) en la Raspberry Pi 5. Todas las librerías utilizadas son de código abierto

Para la programación de la interfaz gráfica se ha utilizado la librería Tkinter [49]. Python incluye esta librería por defecto, siendo su biblioteca estándar para la creación de interfaces gráficas de usuario. Para aprender los conceptos básicos de la programación con esta librería se han seguido las directrices que proporciona el tutorial [51]. Se ha seguido un método de programación de ventanas superpuestas, lo que significa que se ha creado una ventana principal que contiene el mainloop (ciclo que mantiene la ventana abierta) y el resto de las ventanas son secundarias, pero se ejecutan dentro del mainloop principal.

La ventana principal de la interfaz gráfica se muestra al encender el grabador/reproductor. Para definir a esta ventana como la principal es necesario utilizar la clase TK() al principio de la función en la que llama a la ventana .

```
ventana = TK()
```

y al final de la función se llama a mainloop().

```
ventana.mainloop()
```

En la ventana principal se ofrece la posibilidad de iniciar el grabador de audio, pulsando el botón “Record” o iniciar el reproductor audio, pulsando el botón “Play”. En la Figura 38 puede verse la ventana.

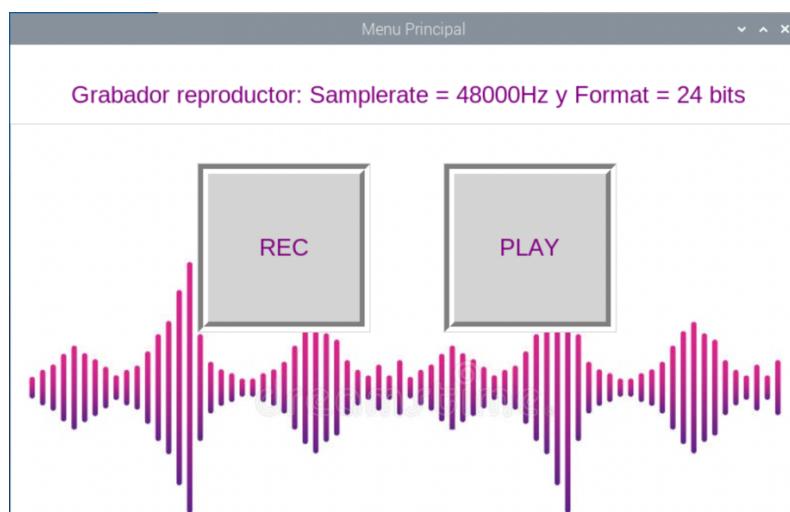


Figura 38: Ventana principal

10.1 Software del grabador

Al pulsar el botón “REC” (Figura 38) se accede al software que realiza las funciones de grabador de audio. Se ha escogido la biblioteca pyaudio [50] para gestionar las operaciones de grabación. Para generar una ventana secundaria como la que se muestra en la Figura 39, se utiliza la clase de Tkinter, Toplevel() (El resto de ventanas secundarias se han creado siguiendo el mismo proceso).

```
ventana2 = Toplevel()
```

La apariencia de la ventana que gestiona las funciones del grabador se muestra en la Figura 39.

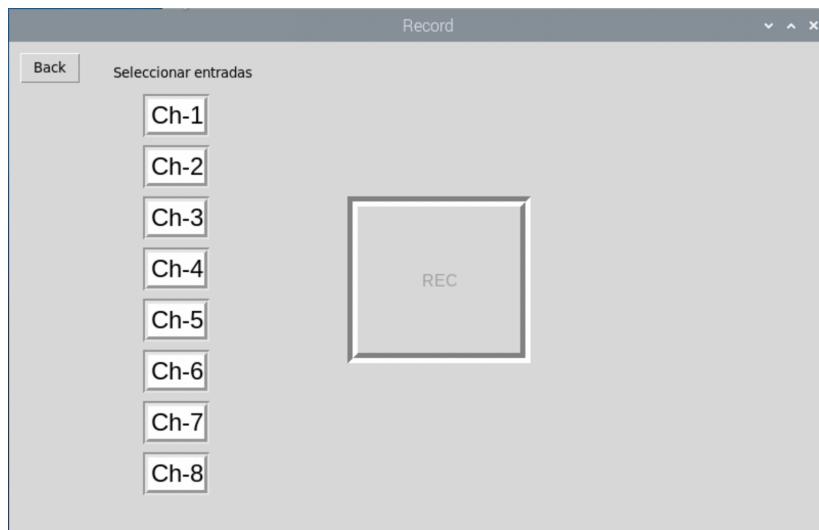


Figura 39: Ventana grabador I

En esta ventana se arman las pistas por las que se pretende capturar audio. A continuación, se explica la funcionalidad de cada botón:

- **Back:** Cierra la ventana del grabador y vuelve a la ventana principal.
- **Botones de selección de entradas:** Cuando los botones están en color rojo significa que la pista a la que hace referencia se ha armado para capturar audio, si está en blanco, cuando se inicie la grabación dicho canal no recibe audio.
- **REC:** Al pulsar el botón “REC” se abre la siguiente ventana y se inicia la grabación. Si no hay ningún canal armado (botón de selección de entrada en rojo), el botón permanece deshabilitado.

En el ejemplo de la Figura 40, se han armado las tres primeras pistas y el botón “REC” está habilitado.

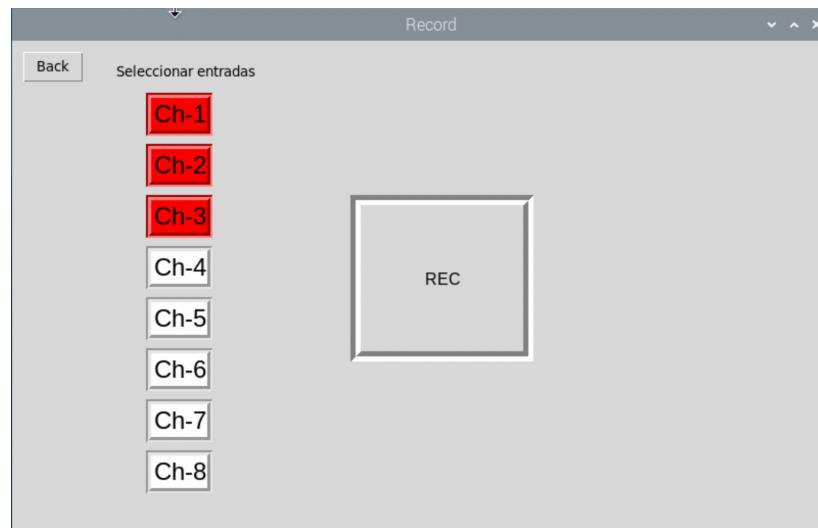


Figura 40: Ejemplo de interacción con la Ventana Gabador I

Grabación

Una vez pulsado el botón “REC”, se accede a la siguiente ventana secundaria. En esta ventana se inicia el proceso de grabación y un cronómetro que controla el tiempo transcurrido desde el inicio de la grabación, tal y como se muestra en la Figura 41. Solo hay un botón en esta ventana, el “STOP”, que se encarga de detener el proceso de grabación, cierra la ventana y abre la anterior. Las barras verticales de nivel calculan el valor RMS (muy comúnmente utilizado en audio [2]) de cada canal del búfer de salida, siguiendo la siguiente formula:

$$x_{rms} = \sqrt{\frac{1}{N} \sum_{i=1}^N x_i^2}$$

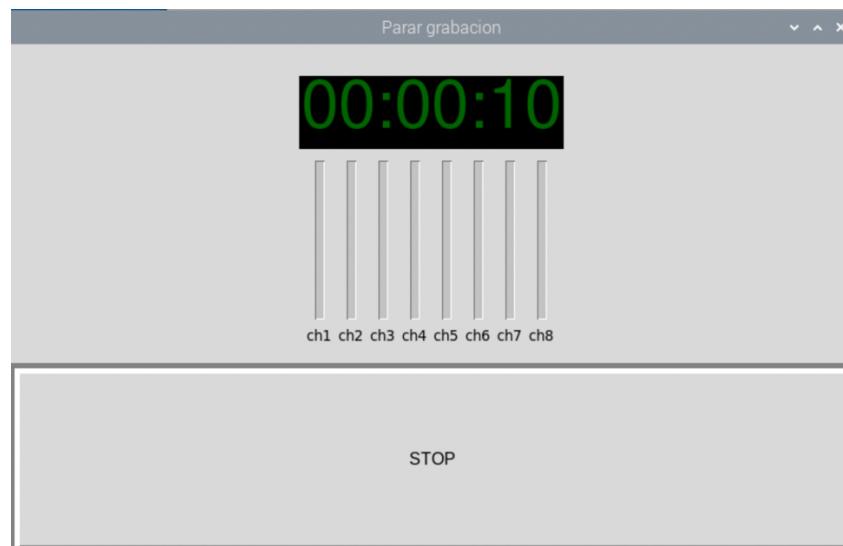


Figura 41: Ventana Grabador II

Software

La grabación se realiza a través de un hilo [52], para que se ejecute de manera concurrente y se pueda mostrar la ventana de la Figura 41, mientras se graba el audio. La función que inicializa el hilo es la siguiente:

```
def grabar():
    audio_thread=threading.Thread(target=Grabador.iniciar_grabacion)
    audio_thread.daemon = True
    audio_thread.start()
    schedule_check(audio_thread)
```

Al pulsar el botón “STOP” es necesario que el hilo haya finalizado su ejecución, de no ser así podrían producirse errores. Gracias a las siguientes funciones se lee el estado del hilo cada segundo y hasta que no haya finalizado no se cierra la ventana.

```
def schedule_check(audio_thread):
    ventana3.after(1000, check_if_done, audio_thread)
def check_if_done(audio_thread):
    if not audio_thread.is_alive():
        ventana3.destroy()
        ventana2.mostrar_ventana()
    else:
        schedule_check(audio_thread)
```

Inicializar parámetros de la grabación

El código de la grabación se ha desarrollado principalmente gracias a pyaudio [27]. Se ha creado un Stream de audio de entrada y otro de salida. El Stream de entrada se inicializa con el fin de capturar la señal de cada pista previamente armada y el de salida se utiliza para reproducir la información de la pista respectiva. Cada Stream se configura según el formato de datos, número de canales, la frecuencia de muestreo, si es de entrada o salida, el dispositivo que procesa la señal y las muestras por búfer. Es importante saber cuál es el identificador del dispositivo de captura y cuál es el de reproducción. Para hallar este número identificatorio, se listan los dispositivos de audio configurados, escribiendo en Python las siguientes líneas:

```
Import sounddevice as sd
print(sd.query_devices())
```

Se muestra el siguiente resultado:

```
0 i2smaster: 1f000a0000.i2s-dit-hifi dit-hifi-0 (hw:0,0), ALSA (0 in, 8 out)
1 i2smaster: 1f000a0000.i2s-dir-hifi dir-hifi-1 (hw:0,1), ALSA (8 in, 0 out)
2 sysdefault, ALSA (0 in, 128 out)
3 lavrate, ALSA (0 in, 128 out)
4 samplerate, ALSA (0 in, 128 out)
5 speexrate, ALSA (0 in, 128 out)
6 pulse, ALSA (32 in, 32 out)
7 speex, ALSA (0 in, 1 out)
8 upmix, ALSA (0 in, 8 out)
9 vdownmix, ALSA (0 in, 6 out)
10 dmix, ALSA (0 in, 2 out)
* 11 default, ALSA (32 in, 32 out)
```

Gracias a esta información se obtiene el número identificatorio de los dispositivos que se utilizan, el 0 hace referencia al dispositivo de reproducción y el 1 al de captura. Una vez obtenidos estos valores se pueden iniciar los Streams de audio, tal y como recoge el siguiente código:

```
FORMAT = pyaudio.paInt24
CHANNELS = 8
RATE = 48000
CHUNK = 256

audio = pyaudio.PyAudio()

stream_in = audio.open(format=FORMAT,
                       channels=CHANNELS,
                       rate=RATE,
                       input=True,
                       output_device_index=1,
                       frames_per_buffer=CHUNK)

stream_out = audio.open(format=FORMAT,
                       channels=CHANNELS,
                       rate=RATE,
                       output=True,
                       output_device_index=0,
                       frames_per_buffer=CHUNK)
```

Guardar archivos en formato WAVE

El proceso de grabación finaliza con el guardado (o no) de la información capturada. La información obtenida mediante la grabación puede guardarse o no, mediante la ventana de la Figura 42, en ella aparece el nombre con el que se guardaría el archivo siguiendo el siguiente formato, f"Record of {date}.wav" (siendo date la fecha y hora).

En caso de que se pulse el botón “si” el archivo se guarda en la carpeta “Grabaciones”, cuya ruta es “/home/pi3/Desktop/TFG/Grabaciones”.

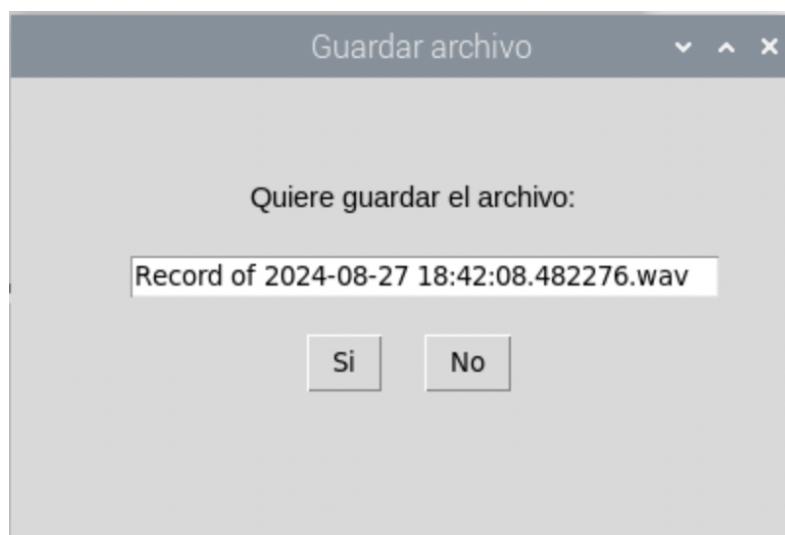


Figura 42: Ventana Save File

El formato que se ha escogido de archivo de audio es el WAVE, con la frecuencia de muestreo y el formato de datos que se han usado al inicializar los Streams de audio. La datos de audio de cada iteración se guardan en el array dinámico frames[], inicializado antes de iniciar el bucle de grabación. Al finalizar el bucle y cerrar los Streams, frames[] contiene los datos correspondientes a las pistas en las que se ha introducido señal. Los datos de las pistas se suman, obteniendo el archivo .wav. El código es el siguiente:

```
wf = wave.open(f"Grabaciones/Record of {date}.wav", 'wb')
wf.setnchannels(1)
wf.setsampwidth(audio.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()
```

10.2 Software del reproductor

Cuando se pulsa el botón “PLAY” en la ventana principal (Figura 38) se accede al software que realiza la reproducción de audio. Se muestra en el Display la ventana secundaria que aparece en la Figura 43. En esta ventana se puede configurar el número de archivos que se quieren reproducir, las pistas en las que se reproducen los archivos e iniciar la reproducción.

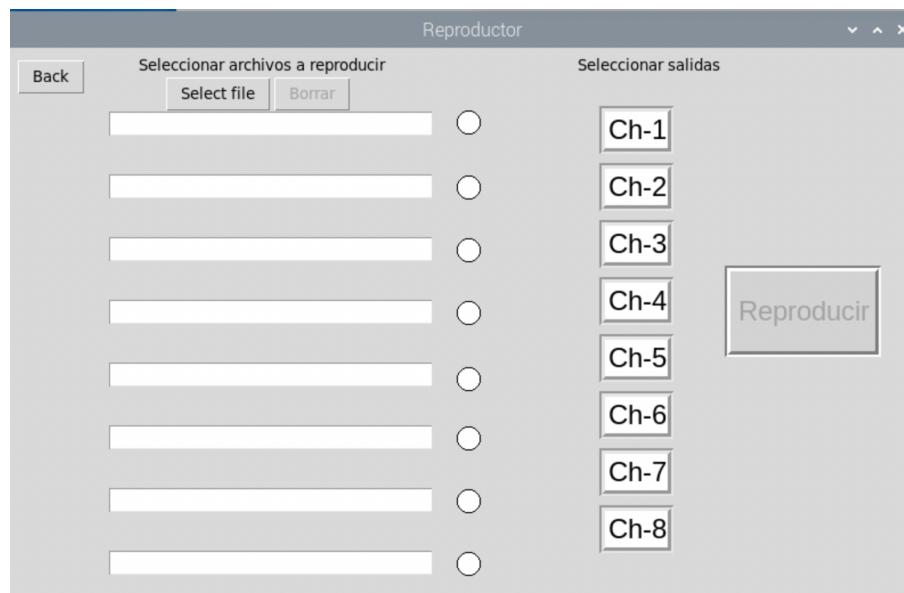


Figura 43: Ventana Reproductor I

A continuación, se explica la funcionalidad de cada botón:

- **Back:** El botón “Back” cierra la ventana del grabador y vuelve a la ventana principal.
- **Select file:** Accede a la carpeta “Grabaciones”, siguiendo la ruta “/home/pi3/Desktop/TFG/Grabaciones”. En ella se encuentran los archivos .wav, resultado de las grabaciones realizadas. Al seleccionar un archivo, aparece su nombre en el respectivo campo de texto y se colorea el círculo a su derecha. Los colores que tomarán los círculos en orden son: verde, rojo, azul, ...

- **Borrar:** Borra el último archivo seleccionado.
- **Botones de selección de salida:** Los botones pueden tomar los colores de los círculos coloreados (referentes a cada archivo seleccionado) según se vayan pulsando. Si el botón ha adquirido el color del último círculo coloreado y se pulsa, el botón volverá a su estado inicial (color blanco). Este intuitivo sistema de colores, sirve para indicar porque canal se quiere reproducir cada archivo. Si el botón de selección de salida comparte color con uno de los círculos, por ese canal (al pulsar el botón “PLAY”)se reproduce el archivo cuyo nombre aparece a la izquierda de ese círculo.
- **PLAY:** Inicia la reproducción, cierra la ventana y abre la siguiente. El botón permanece inhabilitado si no se ha seleccionado ningún archivo

En la Figura 44 se muestra un ejemplo de posible configuración del reproductor, en la que se han seleccionado dos archivos “output_mono_1.wav” y “output_mono_2.wav”. Los círculos a la derecha de los campos de texto en los que aparecen los nombres de los archivos han tomado los colores verde y rojo respectivamente. El primer botón de selección de canal “CH-1” se ha coloreado de verde (pulsando una vez el botón) y el botón “CH-2” se ha coloreado de rojo (pulsando dos veces el botón). Si se pulsa el botón “PLAY”, por el canal 1 se reproduce el archivo “output_mono_1.wav” y por el canal 2 el archivo “output_mono_2.wav”.

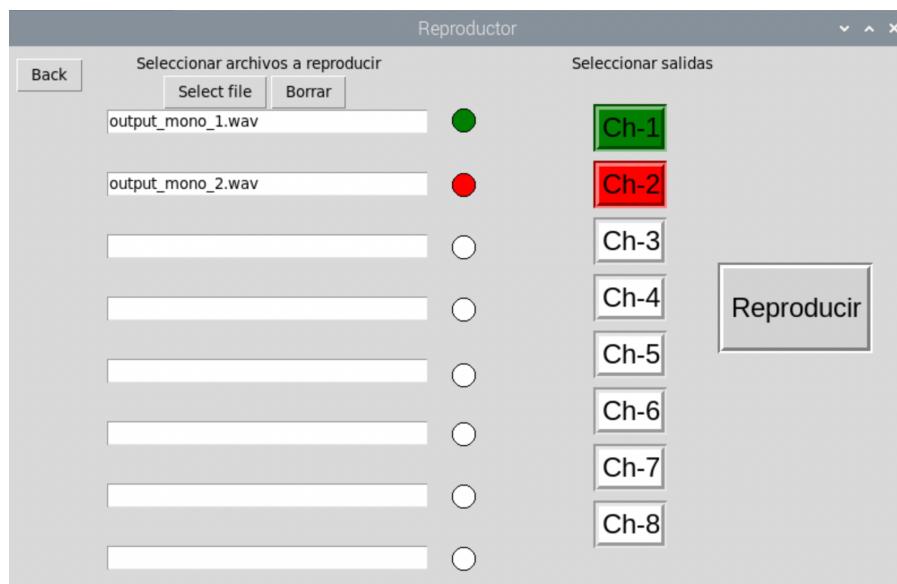


Figura 44: Ejemplo de interacción con la Ventana Reproductor I

Reproducción

Cuando se pulsa el botón “Reproducir” se accede a la siguiente ventana secundaria (Figura 45). La ventana muestra ocho campos de texto, cada uno correspondiente a un canal de audio (nombre del canal a la izquierda), estos campos de texto se llenan con los nombres de los archivos seleccionados en cada canal. A la derecha de los nombres aparece una barra de nivel que calculan el valor RMS (muy comúnmente utilizado en audio [2]) de cada canal del búfer, de la misma forma que las de la figura 41.

En la parte inferior de la ventana hay dos botones:

- >/||: Este botón pausa o continua la reproducción. Si el botón se pulsa cuando se muestra el símbolo “||”, la reproducción se detiene y aparece el símbolo “>”, si el botón vuelve a pulsarse la reproducción continua.
- **Finish:** Al pulsar este botón se finaliza la reproducción de los archivos de audio, se cierra la ventana y se vuelve a la anterior.

Si todos los archivos finalizan su reproducción ocurre lo mismo que al pulsar el botón finish, pero automáticamente.



Figura 45: Ventana Reproductor II

La grabación se realiza a través de un hilo [52], para que se ejecute de manera concurrente y se pueda mostrar la ventana de la Figura 45. Se quiere reproducir los archivos previamente grabados, para ello será necesaria una gestión adecuada de los .wav. Estos archivos se obtienen del array “file” que proporciona la ventana anterior. Lo primero es abrir los archivos:

```
for x in range(len(file)):  
    if os.path.isfile(file[x]):  
        wav.append(file[x])  
        wf.append(wave.open(wav[size], 'rb'))  
        size = size+1  
    else:  
        size = size
```

El código para inicializar el Stream Out es similar al del apartado del grabador. Una vez iniciado el bucle principal, se lee el número de simples previamente configurado de cada archivo .wav y después se transforma al formato de los simples obtenidos a “int24”:

```

for i in range(size):
    data.append(wf[i].readframes(CHUNK))

    if all(elemento == 0 for elemento in data[i]):
        r = True
        pass

    else:
        samples.append(np.frombuffer(data[i], dtype=np.int24))

```

Se crea un búfer de salida (“output_buffer”), un array de ceros de tantas columnas como canales tiene el sistema y con el formato de datos establecido. Este búfer se rellena en cada iteración teniendo en cuenta el array de colores, ya que los colores de los círculos de la Figura 44, definen los canales por los que se envía la información.

```

output_buffer = np.zeros((num_samples, 8), dtype=np.int24)

for t in range(len(colors)):

    if colors[t] == 0:
        pass
    else:
        output_buffer[:,t] = samples[colors[t]-1]

```

El bucle finaliza cada iteración limitando los valores del búfer de salida a un rango específico, para que no se exceda del formato de datos. Por último, se convierten los valores del búfer de salida a una secuencia de bytes, la cual se envía al sistema de audio para reproducir el sonido.

```

output_buffer = np.clip(output_buffer, -32768, 32767)

act_barra_volumen(output_buffer)

output_data = output_buffer.tobytes()

stream.write(output_data)

```

10.3 Software para el control de volumen

El código para el control de volumen gestionado por el chip MCP3008 y los potenciómetros, se ha incluido en las funciones principales de grabación y reproducción. No se ha desarrollado como una función, ya que se probó y se observó que se producían errores. El código es el siguiente:

```

try:
    for i in range(8):
        pot = MCP3008(channel=i)
        output_buffer[:,i] = output_buffer[:,i] *round(pot.value,1)
except Exception as e:
    break

```

11. Prototipado y montaje

Una vez elaborado el diseño del sistema, se implementa de forma física, módulo por módulo. El objetivo es integrar todos los componentes en una carcasa apta para su colocación en un rack de audio.

Prototipado en Protoboard

Al tratarse de un prototipo la mayoría de los circuitos se han integrado en Protoboards. Se comenzó implementando el Módulo Reproductor de Audio, de esta forma se puede obtener sonido de la Raspberry Pi 5, lo cual sirve de guía para el desarrollo del resto de funciones. Una vez se consiguió emitir sonido por uno de los chips MAX98357A, se probó con dos y posteriormente se implementó el diseño propuesto.

Como conexión entre los MAX98357A y los conectores combos se ha realizado mediante cables de audio apantallados. Se han escogido cables apantallados con el fin de mantener la integridad de la señal de audio y reducir las interferencias externas. En el caso de este módulo el apantallamiento no es imprescindible al ser amplificada la señal que viaja por él, pero en el caso de las entradas es necesario. El cable utilizado se muestra en la figura 46.



Figura 46: Cable de Audio Apantallado

Se han cortado 8 fragmentos de cable, uno de los extremos de cada fragmento se ha unido a la clema de salida soldada a cada PCB del MAX98357A y el otro extremo se ha soldado a un conector combo. La señal que emiten los chips siempre es mono, por lo que se han soldado los pines de cada conector, para que no haya ningún problema de conexionado. Se han unido los pines 1, S, 3, R y los pines T y 2 por separado, por el primer grupo de pines se conectan al negativo del chip MAX98357A y el resto al positivo. El aspecto final del Módulo Reproductor de Audio en la Protoboard se muestra en la Figura 47.

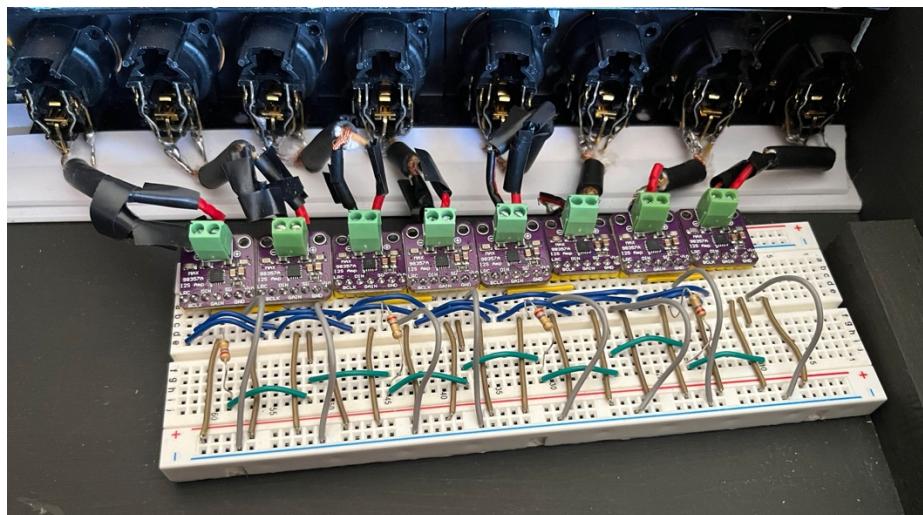


Figura 47: Módulo Reproductor de Audio

Para implementar el Módulo Reproductor de Audio se ha seguido el esquemático de la Figura, las conexiones con los conectores combo se han realizado de la misma manera que en el caso del Módulo de Reproductor de Audio, con la diferencia de que los pines del conector que se sueldan para transmitir el positivo de la señal son el 2, R, 3 y T y para el negativo el 1 y el S. El aspecto final del Módulo de Captura de Audio en la Protoboard se muestra en la Figura 48.

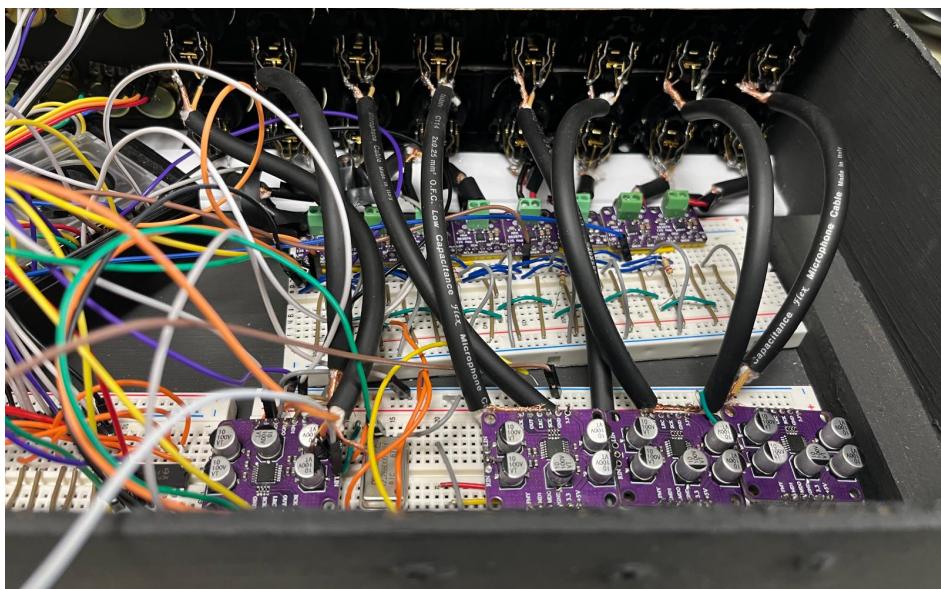


Figura 48: Módulo de Captura de Audio

El Módulo de Control de Volumen se ha desarrollado físicamente tal y como se muestra en el esquemático de la Figura 37. Su aspecto se puede observar en la Figura 49. Para establecer las conexiones entre los terminales de los potenciómetros y el controlador MCP3008 se ha utilizado cable Dupont macho-macho.

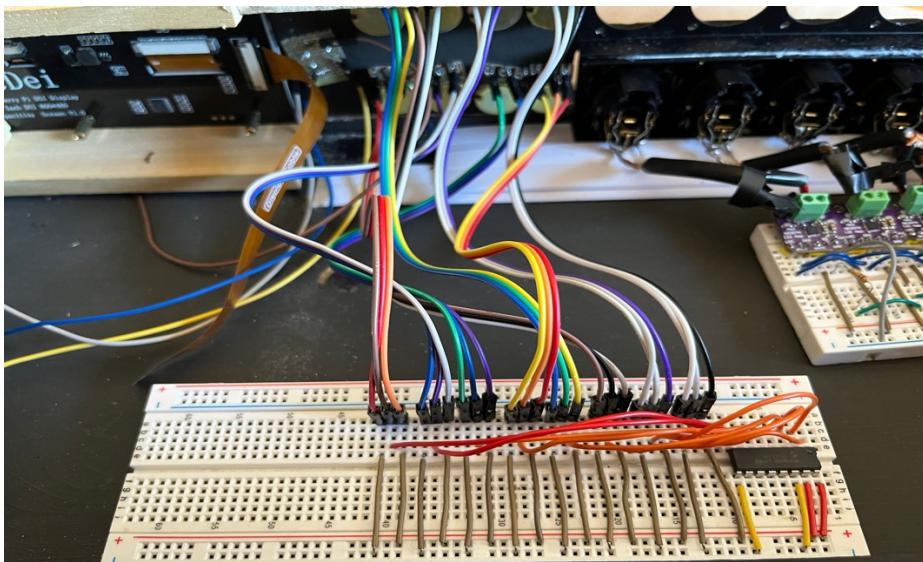


Figura 49: Módulo de Control de Volumen

Todos los módulos del sistema, menos el Display, reciben las señales de la Raspberry Pi a través de cables Dupont macho-hembra. Estos cables de un solo hilo conductor suelen utilizarse en proyectos de Arduino y Raspberry Pi, permitiendo hacer conexiones rápidas y temporales. Para la comunicación con el Display, se ha escogido el cable FFC/FPC, también conocido como cable de pantalla DSI/MIPI, diseñado para transmitir datos de video y gráficos de manera eficiente y rápida entre una fuente y el Display, proporcionando video de alta calidad usando pocos pines. Los conectores recientemente mencionados se muestran en la Figura 50.



Figura 50: Cable Dupont Macho-Hembra (izquierda), cable Dupont Macho-Macho (centro) y cable FFC/FPC (derecha)

La carcasa en la que se han incluido todos los elementos del proyecto debe poder ubicarse en un rack estándar. Por la gran cantidad de elementos hardware con los que cuenta el panel frontal, se ha decidido diseñar de 19 pulgadas y 2U, medidas que proporcionan el espacio suficiente y cumplen con los estándares. La carcasa se ha construido a mano con madera Tablex. Se ha utilizado madera por su fácil manipulación. Se ha dado forma a este material mediante herramientas como un taladro, sierras, serruchos, clavos tornillos etc.... Una vez conformada la carcasa, se ha pintado con pintura negra y se ha barnizado para proteger el material. Los conectores combo se han colocado en láminas de aluminio con orificios redondos. En la Figura 51 se muestra el aspecto físico del resultado final del proyecto desarrollado.



Figura 51: Vista frontal grabador/reproductor de audio basado en Raspberri Pi

La parte trasera alberga el conector USB-C a través del cual se alimenta el sistema. Su aspecto se muestra en la Figura 52.



Figura 52: Vista trasera grabador/reproductor de audio basado en Raspberri Pi

Ajustes de Nivel

Una vez montado el sistema se procede a ajustar los niveles de salida a los estándares. Al principio de la memoria se menciona cuáles son los niveles típicos de audio de consumo y audio profesional, en este caso se ha escogido a la salida el nivel de audio profesional +4 dBu o 1,23 V_{rms} [13]. Para realizar el ajuste de nivel se ha generado en Python una señal sinusoidal de 1 kHz y amplitud de 1,739 V, este valor se ha calculado mediante las siguientes operaciones:

$$V_{\text{rms}} = \frac{V_{\text{pico}}}{\sqrt{2}} = \frac{1}{\sqrt{2}} = 0,707 \text{ V}_{\text{rms}}$$

$$\text{Factor de escala} = \frac{1.23}{0,707} = 1,739$$

Se ha medido con un multímetro cada salida. El objetivo es obtener 1,23 V_{rms} en cada salida, para ello se ha ajustado mediante software el volumen de cada canal, multiplicando el buffer

Prototipado y montaje

de salida por el factor de escala que más se aproxime al objetivo. Se han medido todos los canales y se ha comprobado que al multiplicar el buffer por un factor de 0,55 se obtiene el valor deseado. En la Figura 52, se muestra el ejemplo del valor obtenido en el canal 1 tras el ajuste.

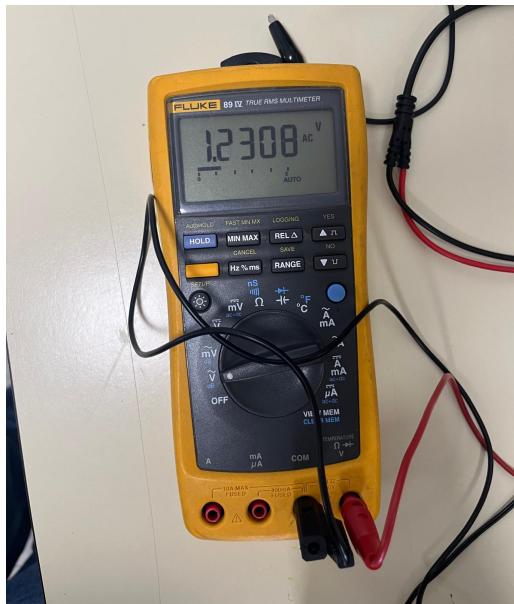


Figura 53: Medida Vrms del canal 1

El código utilizado es el siguiente:

```
factor = 0.55

try:
    for i in range(8):
        pot = MCP3008(channel=i)
        output_buffer[:,i] = output_buffer[:,i]*round(pot.value,1)*factor
except Exception as e:
    break
```

12. Presupuesto

El proyecto ha sido financiado en su totalidad por el estudiante que ha redactado la memoria, sin ayuda de ninguna financiación externa. Para sacar adelante ha sido necesario la adquisición de los equipos y materiales necesarios para el prototipado y el montaje del diseño final. La Tabla 17, recoge las compras realizadas durante el desarrollo del proyecto. En este listado no figuran las herramientas necesarias para realizar el montaje ni algunos de los materiales de los que se disponía previamente.

Tabla 17: Compras realizadas para el proyecto

Elemento	Precio por Unidad	Unidades	Precio Total
Plataforma de desarrollo y chips			
Raspberry Pi 5 (8GB RAM)	95,41 €	1	95,41 €
MAX98357A en PCB	1,20 €	11	13,20 €
PCM1808 en PCB	3,00 €	5	15,00 €
MCP3008	2,84 €	2	5,68 €
Si5351A en PCB	2,16 €	2	4,32 €
Cables			
Cable USB C a USB C	9,53 €	1	9,53 €
Cable Jumper macho a macho	0,03 €	40	0,99 €
Cable Jumper macho a hembra	0,03 €	40	0,99 €
Cable Jumper hembra a hembra	0,03 €	40	0,99 €
Kit cable puente	9,99 €	1	9,99 €
Cable de audio apantallado	2,99 €	2 m	4,98 €
Cable de altavoces	0,39 €	1 m	0,68 €
Componentes electrónicos			
Resistencia	0,04 €	30	1,26 €
Condensador electrolítico	0,17 €	2	0,34 €
Condensador cerámico	0,63 €	2	1,26 €
Potenciómetro	0,38 €	10	3,76 €
Pulsador momentáneo	0,20 €	8	1,60 €
Pulsador momentáneo LED	0,30 €	8	2,79 €
Oscilador de Cristal de Cuarzo	3,30 €	2	6,60 €
Conectores			
Conector minijack	0,30 €	1	0,30 €
Conector Jack 6.35 mm	0,82 €	1	0,82 €
Conector XLR	1,61 €	1	1,61 €
Conector Combo	0,74 €	16	11,84 €
Pantalla			
Display LCD Táctil	17,36 €	1	17,36 €
Herramientas			
Madera	10,00 €	1	10,00 €
Tornillo	0,16 €	20	3,20 €
Visagra	1,39 €	2	1,39 €
Soporte Rack 2U 19	12,50 €	1	12,50 €
Panel XLR	3,11 €	4	12,44 €
Pintura negra 250 ml	7,99 €	1	7,99 €
Barniz	8,49 €	1	8,49 €
Cola blanca	7,49 €	1	7,49 €
Multímetro Digital AoKoZo 21D	16,09 €	1	16,09 €
Cinta selladora Ceys	9,49 €	1	9,49 €
Equipo de soldadura	20,00 €	1	20 €
Otros			
Lector de tarjetas SD	6,79 €	1	6,79 €
Software	0,00 €	1	0,00 €
Carcasa Raspberry Pi 5 con ventilador	9,95 €	1	9,95 €
Fuente de alimentación USB C 27W	14,46 €	1	14,46 €
Tarjeta microSD Kingston 64 GB	6,96 €	1	6,96 €
TOTAL			358,54 €

Prototipado y montaje

Por otro lado, se muestra en la Tabla 18 el coste total del proyecto, contando las horas trabajadas por un ingeniero de telecomunicaciones con un salario de 20 €/hora.

Tabla 18: Coste total del proyecto

Elemento	Coste
Ingeniero de Telecomunicaciones (350 horas)	7.000,00 €
Compras realizadas para el proyecto	358,54 €
TOTAL	7.358,54 €

Por último, se muestra en la Tabla 19 el gasto de los materiales utilizados para montar o replicar el Grabador/Reproductor, contando con el núcleo de procesamiento, el Módulo de Captura de Audio, el Módulo de Reproducción de Audio, el Módulo de Control de Volumen y el Display. El precio final resultante es de 240,52€, un precio asequible en comparación a otras opciones del mercado.

Tabla 19: Gasto montaje

Elemento	Precio por Unidad	Unidades	Precio Total
Raspberry Pi 5 (8GB RAM)	95,41 €	1	95,41 €
MAX98357A en PCB	1,20 €	8	9,60 €
PCM1808 en PCB	3,00 €	4	12,00 €
MCP3008	2,84 €	1	2,84 €
Oscilador de Cristal de Cuarzo	3,30 €	1	3,30 €
Cable USB C a USB C	9,53 €	1	9,53 €
Cable Jumper macho a macho	0,03 €	40	1,00 €
Cable Jumper macho a hembra	0,03 €	40	1,00 €
Kit cable puente	9,99 €	1	9,99 €
Cable de audio apantallado	2,99 €	1	2,99 €
Resistencia	0,04 €	4	0,16 €
Potenciómetro	0,38 €	8	3,01 €
Pulsador momentáneo LED	0,30 €	1	0,30 €
Conector Combo	0,74 €	16	11,84 €
Display LCD Táctil	17,36 €	1	17,36 €
Madera	10,00 €	1	10,00 €
Tornillo	0,16 €	8	1,28 €
Soporte Rack 2U 19	12,50 €	1	12,50 €
Panel XLR	3,11 €	4	12,44 €
Pintura negra 250 ml	7,99 €	1	7,99 €
Barniz	8,49 €	1	8,49 €
Cola blanca	7,49 €	1	7,49 €
TOTAL			240,52 €

13. Impacto del proyecto

El diseño del Grabador/Reproductor de audio de alta calidad basado en una Raspberry Pi, tiene el potencial de generar un significativo impacto en varios de los ODS (Objetivos de Desarrollo sostenible) [57]. A continuación, se describe como este proyecto puede contribuir a estos objetivos:

- **ODS 4. Educación de calidad:** Uno de los principales objetivos de este proyecto es poder implementar el Grabador/Reproductor en los laboratorios de audio de la Universidad Politécnica de Madrid, ligándolo directamente con el sector educativo. La intención es que los alumnos de la Universidad realicen prácticas de laboratorio con el aparato diseñado, facilitando el aprendizaje del audio digital, la captura y la reproducción de audio. Por la proximidad entre el sonido y el arte, el dispositivo es idóneo para fomentar la creatividad de los estudiantes, utilizándolo para enseñar materias como la producción musical o la edición del sonido. Por la flexibilidad de programación que presenta el sistema, puede sentar las bases para futuros proyectos académicos y profesionales.
- **ODS 9. Industria, innovación e infraestructura:** La Raspberry Pi es una plataforma de bajo costo, y desarrollar un Grabador/Reproductor sobre esta base promueve la creación de dispositivos tecnológicos accesibles. Esto puede inspirar la innovación en comunidades con recursos limitados. La integración de tecnologías como la Raspberry Pi en proyectos grabación y reproducción de sonido, puede impulsar la investigación en campos como el procesamiento de audio digital, la electrónica y la fabricación digital, especialmente en contextos de bajo costo.
- **ODS 10. Reducción de las desigualdades:** Al ser una herramienta de bajo costo, un grabador y reproductor de sonido basado en Raspberry Pi puede ser distribuido en comunidades desfavorecidas, permitiendo que personas de diferentes orígenes socioeconómicos accedan a la tecnología de grabación y reproducción de sonido de alta calidad. Esto puede ayudar a reducir la brecha digital y permitir que más personas participen en la creación y consumo de contenidos multimedia.
- **ODS 12. Producción y consumo responsable:** La Raspberry Pi es conocida por su bajo consumo de energía y su eficiencia. El desarrollo del proyecto en esta plataforma promueve el uso de tecnologías que minimizan el impacto ambiental. Además, se fomenta la reutilización de componentes y la reducción de desperdicios electrónicos. Al basarse en una plataforma abierta y flexible, este dispositivo puede ser reparado, actualizado y modificado, alargando su vida útil y promoviendo prácticas de consumo responsables.
- **ODS 17. Alianzas para lograr los objetivos:** Este proyecto surge de la colaboración de un alumno con la Universidad Politécnica de Madrid, específicamente con su departamento de Ingeniería Audiovisual y Comunicaciones. La UPM ha facilitado recursos como la utilización de los laboratorios, lo cual ha ayudado a lograr el resultado final.

14. Conclusiones

Tras la finalización del diseño y montaje, se considera que el proyecto de grabador/reproductor de audio multicanal basado en Raspberry Pi cumple con los objetivos y especificaciones descritos. Se destacan los siguientes aspectos clave:

- El sistema está construido en base a una Raspberry Pi 5, una microcomputadora accesible y escalable, que constituye el núcleo del dispositivo. Este componente, junto con otros como los conversores ADC y DAC, asegura el procesamiento de audio en tiempo real.
- El diseño modular del sistema permite una gran versatilidad en su funcionamiento, ya que puede ser programado para incorporar nuevas funciones fácilmente. Además, su estructura abierta y basada en software libre facilita futuras actualizaciones y modificaciones.
- Los componentes utilizados, como conectores y chips de audio, son asequibles y ampliamente disponibles en el mercado, lo que reduce significativamente el coste de fabricación. Esto permite que el dispositivo sea una solución viable tanto en entornos profesionales como educativos.
- El dispositivo cumple con los requerimientos del audio profesional, realizando un procesamiento de 48 kHz, 24 bits y se han obtenido niveles de salida de +4dBu.
- Es un sistema sencillo de replicar. El coste y la accesibilidad a los componentes que conforman el sistema hace que sean fáciles de adquirir. Una vez montada la parte física, el software puede copiarse desde la tarjeta SD en la que se ha desarrollado el proyecto. Una forma simple de realizar la copia es mediante la herramienta SD Card Copier, introduciendo la tarjeta en la que se quiere copiar la información, en el puerto USB de la Raspberry Pi 5 utilizando un adaptador USB.

En conclusión, se ha obtenido un sistema funcional y sencillo de replicar, que cumple con las expectativas de calidad y versatilidad. Se presenta como una útil herramienta para el desarrollo de proyectos de audio digital y para su utilización en contextos educativos.

14.1 Líneas futuras

Una posible línea de desarrollo futuro para el grabador/reproductor de audio multicanal es la ampliación del número de canales mediante la técnica de Multiplexación por División de Tiempo (TDM). Esta tecnología permite transmitir múltiples señales de audio a través de una única línea de datos, dividiendo el flujo de bits en intervalos de tiempo asignados a cada canal. Implementar TDM en este proyecto permitiría manejar más canales de entrada y salida.

Para llevar a cabo esta modificación, se debería reconfigurar el protocolo de comunicación utilizado, en este caso el I2S, para que en lugar de gestionar solo dos canales (izquierdo y derecho), pueda multiplexar hasta 16 o más canales en una única línea. Esto implicaría un ajuste en la frecuencia de los relojes de sincronización y en la gestión de los buffers de entrada

y salida, asegurando que no haya latencias perceptibles en la captura y reproducción del audio.

Por ejemplo, el DAC utilizado en este proyecto (MAX98357A) informa en el *datasheet* [18], sobre su capacidad para soportar TDM. El emisor puede enviar hasta 8 señales de datos por un solo bus, por lo que como la Raspberry Pi 5 tiene la posibilidad de emitir 4 señales de datos I²S, se podrían emitir con este DAC hasta 32 canales. La selección del canal que recibe cada chip se realiza mediante diferentes niveles de entrada en los pines “SD_MODE” y “GAIN_SLOT”. Lo que diferencia a este proceso de transmisión al I²S estándar utilizado en este proyecto es la línea de reloj de selección de palabra. En I²S la señal WS varía periódicamente entre estado alto y bajo con un ciclo de trabajo del 50%, mientras que en modo TDM esta señal permanece en estado alto mucho menos tiempo que en bajo por cada periodo, lo que se conoce como pulso de sincronización. Estos pulsos de sincronización delimitan el inicio de la trama de datos de cada canal, tal y como se muestra en la figura 49.

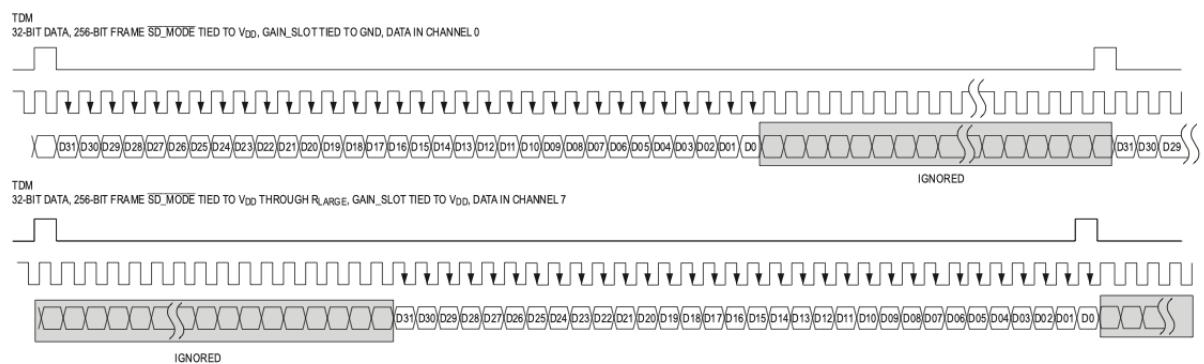


Figura 54: Modo TDM 32 bits en MAX98357A

El aumento del número de canales mediante TDM haría al sistema más escalable, permitiendo su uso en aplicaciones más avanzadas, para realizar funciones que requieren la gestión simultánea de múltiples fuentes de audio. Además, esto ofrecería una ventaja significativa en términos de eficiencia, al reducir el número de conexiones físicas y componentes, manteniendo la calidad de audio.

Sin embargo, este desarrollo implicaría también desafíos técnicos, como la optimización del procesamiento de datos para asegurar que la Raspberry Pi pueda manejar una mayor carga de trabajo sin generar retrasos y el ajuste de las líneas de reloj y datos para garantizar precisión en la transmisión.

15. Bibliografía

- [1] L. Bazara, «Pequeña historia del audio digital: un recorrido por las máquinas sonoras del siglo XX,» *Revista Bricolaje*, nº 7, pp. 39-51, 2020.
- [2] T. Piñeiro Otero y L. Pedrero Esteban, «La comunicación Sonora ante el renacimiento del audio digital,» *Profesional de la información*, vol. 31, nº 5, p. 2022.
- [3] L. P. García Morales, Introducción al audio digital, BOD GmbH, 2020.
- [4] E. L. Aldea, Raspberry Pi fundamentos y aplicaciones, Ra-Ma Editorial, 2017.
- [5] NXP Semiconductors, «I2S bus specification,» UM11732, 2022.
- [6] K. C. Pohlmann, «Principles of digital audio,» *McGraw-Hill Professional*, 2000.
- [7] R. Cádiz, «Introducción a la música computacional,» *Publicación electrónica auspiciada por la Pontificia Universidad Católica de Chile, provista por el docente*, 2008.
- [8] M. Semeria, «Los tres teoremas: Fourier - Nyquist - Shannon,» 2015. [En línea]. Available: <https://www.econstor.eu/bitstream/10419/130833/1/844215546.pdf>. [Último acceso: Julio 2024].
- [9] J. M. M. d. I. Fuente, «La percepción acústica: física de la audición,» *Revista de Ciencias*, nº 2, pp. 19-26.
- [10] S. V. Angeles, «Analisis de audio,» 2014.
- [11] A. P. McPherson, «Action-sound latency: Are our tools fast enough?,» *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2016.
- [12] FileFormat, «docs.fileformat,» [En línea]. Available: <https://docs.fileformat.com/es/audio/wav/>. [Último acceso: 17 Julio 2024].
- [13] Hispasonic, «hispanic niveles linea,» [En línea]. Available: <https://www.hispasonic.com/tutoriales/son-niveles-linea-10dbv-4dbu/43052#:~:text=Así%20los%20niveles%20de%20las,definidas%20como%20elaciones%20entre%20voltajes..> [Último acceso: 19 Julio 2024].
- [14] Ricable, «ricable,» [En línea]. Available: <https://www.ricable.com/es/il-protocollo-i2s-e-il-cavo-supreme-hdmi/#:~:text=El%20protocolo%20I2Sse%20creó,5%20de%20junio%20de%201996..> [Último acceso: 16 Junio 2024].

- [15] ANALOG DEVICES, «Common Inter-IC Digital Interfaces for Audio Data Transfer,» Technical Article MS-2275, 2012.
- [16] Raspberry Pi Fundation, «raspberry pi 5,» [En línea]. Available: <https://www.raspberrypi.com/products/raspberry-pi-5/>. [Último acceso: 7 Agosto 2024].
- [17] TEXAS INSTRUMENTS, «PCM1808 Single-Ended, Analog-Input 24-Bit, 96-kHz Stereo ADC,» *SLES177B*, 2015.
- [18] ANALOG DEVICES/Maxim Integrated, «MAX98357A/ MAX98357B,» *Tiny, Low-Cost, PCM Class D Amplifier with Class AB Performance*, 2019.
- [19] Microchip Technology Inc, «MCP3004/3008,» *2.7V 4-Channel/8-Channel 10-Bit A/D Converters with SPI Serial Interface*, 2008.
- [20] Raspberry Pi Fundation, «raspberry pi hardware,» [En línea]. Available: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>. [Último acceso: 8 Agosto 2024].
- [21] Raspberry Pi Fundation, «raspberry pi 4 model b,» [En línea]. Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>. [Último acceso: 3 Agosto 2024].
- [22] «Raspberry Pi Pinout,» [En línea]. Available: <https://pinout.xyz>.
- [23] Raspberry Pi Fundation, «bcm2711 peripherals,» [En línea]. Available: https://datasheets.raspberrypi.com/bcm2711/bcm2711-peripherals.pdf?_gl=1*17t9btc*_ga*NTIzOTExODUxLjE3MjU0NDQ0Njg.*_ga_22FD70LWDS*MTcyNTQ0NDQ3OS45LjEuMTcyNTQ0NTIwNy4wLjAuMA... [Último acceso: 3 Agosto 2024].
- [24] Raspberry Pi Fundation, «raspberry pi 5 product brief,» [En línea]. Available: https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf?_gl=1*1725wf9*_ga*NTIzOTExODUxLjE3MjU0NDQ0Njg.*_ga_22FD70LWDS*MTcyNTQ0NDQ3OS45LjEuMTcyNTQ0NDU3MC4wLjAuMA... [Último acceso: 7 Agosto 2024].
- [25] Raspberry Pi Fundation, «rp1 peripherals,» [En línea]. Available: https://datasheets.raspberrypi.com/rp1/rp1-peripherals.pdf?_gl=1*15l0gnr*_ga*NTIzOTExODUxLjE3MjU0NDQ0Njg.*_ga_22FD70LWDS*MTcyNTQ0NDQ3OS45LjEuMTcyNTQ0NTAxOC4wLjAuMA... [Último acceso: 1 Agosto 2024].

Bibliografía

- [26] H. Mora, «Sistemas de adquisición y Procesamiento de datos,» *Fundamentos de los Computadores*, 2011.
- [27] TEXAS INSTRUMENTS, «ti.PCM1808,» [En línea]. Available: https://www.ti.com/product/PCM1808?utm_source=google&utm_medium=cpc&utm_campaign=asc-null-null-GPN_EN-cpc-pf-google-wwe&utm_content=PCM1808&ds_k=PCM1808&DCM=yes&gad_source=1&gbraid=0A AAAAC068F10Pz5xuEn4oj4wYxi9ykGkq&gclid=CjwKCAjwoJa2BhBPEiwA0l0ImJG_OjDnH. [Último acceso: 22 Agosto 2024].
- [28] D. Laureano, «Apunte de Cátedra: Cadena de audio,» *Universidad Nacional de la Plata*, 2020.
- [29] NEUTRIK, «neutrik,» [En línea]. Available: <https://www.neutrik.com/en/product/ncj6fa-h>. [Último acceso: 14 Agosto 2024].
- [30] DigiKey, «MAX98357AETE-T,» [En línea]. Available: <https://www.digikey.com/en/products/detail/analog-devices-inc-maxim-integrated/MAX98357AETE-T/4936122>. [Último acceso: 20 Agosto 2024].
- [31] RS, «Rs Potenciómetros,» [En línea]. Available: <https://es.rs-online.com/web/p/potencios/2499418>. [Último acceso: 12 Agosto 2024].
- [32] ElectrónicaOnline, «Potencímetro Rotativo,» [En línea]. Available: <https://electronicaonline.net/componentes-electronicos/resistor/potenciometro-rotatorio/>. [Último acceso: 12 Agosto 2024].
- [33] smartkits, «Como Funciona o Potenciômetro com Chave,» [En línea]. Available: <https://blog.smartkits.com.br/como-funciona-o-potenciometro-com-chave/>. [Último acceso: 12 Agosto 2024].
- [34] AliExpress, « MCP3008-I/P Mcp3008 ADC de 8 canales y 10 bits con interfaz SPI para Raspberry Pi,» [En línea]. Available: <https://es.aliexpress.com/item/1005005944889779.html>. [Último acceso: 13 Agosto 2024].
- [35] P. F. Rodrigues, «MIPI Display Serial Interface Interoperability Tests,» *Master's thesis, Instituto Politecnico do Porto*, 2016.
- [36] HLFEC Global Store, «aliexpress,» [En línea]. Available: https://es.aliexpress.com/item/1005006737177076.html?spm=a2g0o.order_list.order_list_main.45.57a6194dMJMg9y&gatewayAdapt=glo2esp. [Último acceso: 27 Junio 2024].

- [37] D. Anderson y J. Trodden, *USB 3.0 Technology*, MindShare Press, 2013.
- [38] C. Dunn y M. J. Hawksford, «Is the AES/EBU/SPDIF Digital Audio Interface Flawed?», *In Audio Engineering Society Convention 93. Audio Engineering Society*, 1992.
- [39] Raspberry Pi Fundation, «raspberrypi software,» [En línea]. Available: <https://www.raspberrypi.com/software/>. [Último acceso: 12 Febrero 2024].
- [40] Raspberry Pi Fundation, «raspberry pi operating systems,» [En línea]. Available: <https://www.raspberrypi.com/software/operating-systems/#raspberry-pi-os-64-bit>. [Último acceso: 15 Mayo 2024].
- [41] Universitat Jaume I, «vnc multimedia,» 16 Agosto 2020. [En línea]. Available: <https://www3.uji.es/~galdu/vnc-multimedia/vnc-multimedia.pdf>. [Último acceso: 10 Febrero 2024].
- [42] Apple, «developer.apple,» [En línea]. Available: <https://developer.apple.com/xcode/>. [Último acceso: 25 Junio 2024].
- [43] The MacPorts Project, «mscports,» [En línea]. Available: <https://www.macports.org>. [Último acceso: 25 Junio 2024].
- [44] Simon Tatham, «puttygen,» [En línea]. Available: <https://www.puttygen.com/download-putty>. [Último acceso: Febrero 10 2024].
- [45] Matias Labs, «IP Scanner,» [En línea].
- [46] Arsys, «arsys,» 22 Abril 2024. [En línea]. Available: <https://www.arsys.es/blog/ssh>. [Último acceso: 10 Mayo 2024].
- [47] RealVNC, «realvnc,» [En línea]. Available: <https://www.realvnc.com/es/connect/download/viewer/>. [Último acceso: 9 Febrero 2024].
- [48] Linux, «Linux and the Devicetree,» [En línea]. Available: <https://docs.kernel.org/devicetree/usage-model.html>. [Último acceso: 12 Junio 2024].
- [49] Linux, «alsa.project,» [En línea]. Available: https://www.alsa-project.org/wiki/Main_Page. [Último acceso: 4 Junio 2024].
- [50] Akiyuki Okayasu, «github-RaspberryPi_i2s_Master,» [En línea]. Available: https://github.com/AkiyukiOkayasu/RaspberryPi_I2S_Master/blob/master/i2smaster.dts. [Último acceso: 14 Junio 2024].

Bibliografía

- [51] Linux, «github-Linux,» [En línea]. Available: <https://github.com/raspberrypi/linux>. [Último acceso: 17 Junio 2024].
- [52] ANALOG DEVICES, «SPI Interface,» AN-1248 APPLICATION NOTE, 2015.
- [53] B. Nuttal y D. Jones, «gpiozero,» [En línea]. Available: <https://gpiozero.readthedocs.io/en/latest/>. [Último acceso: 4 Junio 2024].
- [54] KiCad, «kicad,» [En línea]. Available: <https://www.kicad.org>. [Último acceso: 20 Agosto 2024].
- [55] Amazon, «Amazon-Módulo amplificador PCM1808 de 24 bits PCM1808,» [En línea]. Available: <https://www.amazon.es/Youmile-amplificador-decodificador-reproductor-terminaci%F3n/dp/B09BTLPCM4>. [Último acceso: 14 Junio 2024].
- [56] Amazon, «Amazon-Max98357 I2S 3W Clase D Amplificador Breakout Interface Dac,» [En línea]. Available: <https://www.amazon.es/Pmkvgdy-Max98357-Amplificador-Interface-Filterless/dp/B0BHSMMT68>.
- [57] F. Leens, «An introduction to I2C and SPI protocols,» ieee, 2009.
- [58] Thonny, «Thonny Python IDE for beginners,» [En línea]. Available: <https://thonny.org>. [Último acceso: 3 Mayo 2024].
- [59] Python Software Foundation, «Python 3.11.2,» [En línea]. Available: <https://www.python.org/downloads/release/python-3112/>. [Último acceso: 20 Junio 2024].
- [60] Python Software Foundation, «docs.python Tkinter,» [En línea]. Available: <https://docs.python.org/es/3/library/tkinter.html>. [Último acceso: 1 Julio 2024].
- [61] A. Alvarez, «github,» [En línea]. Available: <https://github.com/eliluminado>. [Último acceso: 28 Junio 2024].
- [62] Hubert Pham, «people.csail.mit.edu,» [En línea]. Available: <https://people.csail.mit.edu/hubert/pyaudio/docs/>. [Último acceso: 4 Julio 2024].
- [63] Hispasonic, «hispanic,» [En línea]. Available: <https://www.hispasonic.com/tutoriales/presion-sonora-sonoridad-ii-son-valor-rms-vumetro/43343>. [Último acceso: 23 Agosto 2024].
- [64] Python Software Foundation , «docs.python threading,» [En línea]. Available: <https://docs.python.org/es/3.8/library/threading.html>. [Último acceso: 5 Julio 2024].
- [65] PyAudio, «PyAudio,» [En línea]. Available: <https://pypi.org/project/PyAudio/>.

-
- [66] Naciones Unidas, «un,» [En línea]. Available: <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>. [Último acceso: 24 Agosto 2024].
 - [67] J. Lewis, «Common inter-IC digital interfaces for audio data transfer,» *EDN-Electronic Design News*, vol. 57, nº 16, p. 46.
 - [68] Raspberry Pi Fundation, «raspberry pi 5 mechanical drawing,» [En línea]. Available: https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-mechanical-drawing.pdf?_gl=1*1odv5c7*_ga*NTIzOTExODUxLjE3MjU0NDQ0Njg.*_ga_22FD70LWDS*MTcyNTQ0NDQ3OS45LjEuMTcyNTQ0NDc0MC4wLjAuMA... [Último acceso: 8 Agosto 2024].
 - [69] S. S. Oommen, R. Dhanabal, N. Anand y G. Joseph, «Design and implementation of a high speed Serial Peripheral Interface,» *International Conference on Advances in Electrical Engineering (ICAEE)*, nº IEEE, pp. 1-3, 2014.

Anexo

A.1 Código del grabador

A.1.1. Ventana grabador I

```

#Se importan las librerías
from tkinter import *
import tkinter
from tkinter import ttk
import ventana1
import ventana3
import Grabador
import threading

# Vuelve a la ventana principal
def regresar():
    ventana2.destroy()
    ventana1.get_raiz().iconify()
    ventana1.get_raiz().deiconify()

# Se introduce un nuevo valor booleano en el array yesno[] y se cambia el color de los
# botones CH
def select_channel(btnCH,posicion):
    if yesno[posicion] == True:
        yesno[posicion] = False
    else:
        yesno[posicion] = True

    if yesno[posicion] == True:
        btnCH.config(bg = 'red')
    else:
        btnCH.config(bg = 'white')

    actualizar_boton()

# Cierra la ventana actual y se inicia la grabacion
def grabar():
    ventana2.destroy()
    ventana3.mostrar_ventana()

# Devuelve el array yesno
def get_yesno():
    return yesno

# Habilita y deshabilita el boton REC
def actualizar_boton():
    x = 0
    for v in range(8):
        if yesno[v] == True:
            x = x+1
        else:
            x = x

    if x == 0:
        rec['state']='disabled'

```

```
else:
    rec.config(state=tkinter.NORMAL)

# Se inicializa la ventana
def mostrar_ventana():
    global ventana2
    global btnCH
    global yesno
    global rec

    ventana2 = Toplevel()
    ventana2.title("Record")
    ventana2.geometry("800x480")
    yesno = [False, False, False, False, False, False, False]
    btnCH =[Button(ventana2, font=("arial",18),bg="white",
                   fg="black",padx=1,pady=1,relief="groove",bd=5)   for _ in range(8)]

    label = Label(ventana2,text="Seleccionar entradas")
    label.place(x=100,y=20)

    btn = Button(ventana2,text="Back",command=regresar)
    btn.place(x=10,y=10)

    btnCH[0].config(text ='Ch-1',command=lambda:select_channel(btnCH[0],0))
    btnCH[0].place(x=130,y=50)

    btnCH[1].config(text ='Ch-2',command=lambda:select_channel(btnCH[1],1))
    btnCH[1].place(x=130,y=100)

    btnCH[2].config(text ='Ch-3',command=lambda:select_channel(btnCH[2],2))
    btnCH[2].place(x=130,y=150)

    btnCH[3].config(text ='Ch-4',command=lambda:select_channel(btnCH[3],3))
    btnCH[3].place(x=130,y=200)

    btnCH[4].config(text ='Ch-5',command=lambda:select_channel(btnCH[4],4))
    btnCH[4].place(x=130,y=250)

    btnCH[5].config(text ='Ch-6',command=lambda:select_channel(btnCH[5],5))
    btnCH[5].place(x=130,y=300)

    btnCH[6].config(text ='Ch-7',command=lambda:select_channel(btnCH[6],6))
    btnCH[6].place(x=130,y=350)

    btnCH[7].config(text ='Ch-8',command=lambda:select_channel(btnCH[7],7))
    btnCH[7].place(x=130,y=400)

    rec = Button(ventana2,text ='REC',width=15, height=7,font=("arial",12),
                 relief="groove",bd=10,command=grabar)
    rec.place(x=330,y=150)
    rec.config(state=tkinter.DISABLED)
```

A.1.2. Ventana grabador II

```

#Se importan las librerias
from tkinter import *
import tkinter
from tkinter import ttk
import threading
import Grabador
import ventana2
import ventana1
import time

# Actualizacion valor del cronometro
def act_cron():
    global inicio
    global duracion

    if duracion == 0:
        inicio = time.time()
    else:
        inicio = time.time() - duracion

    ventana3.after(500,actualizar_cron)

# Actualizar la imagen del cronometro
def actualizar_cron():
    global inicio
    global duracion

    duracion = time.time() - inicio
    horas = int(duracion//60//60)
    minutos = int(duracion//60%60)
    segundos = int(duracion%60)

    tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")
    tarea = ventana3.after(500,actualizar_cron)

# Decuelve la varianle progress
def get_barra_volumen():
    return progress

# Vuelve a la ventana anterior al acabarse la grabacion
def regresar():
    ventana3.destroy()
    ventana2.mostrar_ventana()

#Inicia la grabacion
def grabar():
    #global audio_thread
    audio_thread = threading.Thread(target=Grabador.iniciar_grabacion, args =
(ventana2.get_yesno()))
    audio_thread.daemon = True
    audio_thread.start()
    schedule_check(audio_thread)

```

```
#Detiene la grabacion
def parar():
    print('Grabacion finalizada.')
    stop.set(True)
    #ventana.destroy()
    #ventana2.mostrar_ventana()

# Devuelve el valor stop
def get_stop():
    return stop.get()

# comprueba el estado del hilo cada segundo
def schedule_check(audio_thread):
    ventana3.after(1000, check_if_done, audio_thread)

# Comprueba si el hilo ha finalizado
def check_if_done(audio_thread):
    if not audio_thread.is_alive():
        ventana3.destroy()
        ventana2.mostrar_ventana()
    else:
        schedule_check(audio_thread)

# Inicializa la ventana
def mostrar_ventana():

    global ventana3
    global stop
    global detener_hilo
    global segundos,minutos,horas,inicio,duracion,tarea,tiempo
    global cronometro
    global progress

    segundos = 0
    minutos = 0
    horas = 0
    inicio = 0
    duracion = 0

    tarea = None
    tiempo = tkinter.StringVar()
    tiempo.set(f"00:00:00")

    ventana3 = Toplevel()
    ventana3.title("Parar grabacion")
    ventana3.geometry("800x480")

    stop = tkinter.BooleanVar()
    stop.set(False)

    grabar()
    act_cron()

    STOP = Button(ventana3,text ='STOP',width=84,
height=8,font=("arial",12),relief="groove",bd=10,command=parar)
```

```
STOP.place(x=0,y=300)

cronometro = tkinter.Label(ventana3, textvariable=tiempo, font=("Helvetica",
48),bg="black",fg="darkgreen")
cronometro.pack(pady=30)

nombre_canal = [Label(ventana3) for _ in range(8)]

progress = [tkinter.DoubleVar()for _ in range(8)]
barra_volumen = [ttk.Progressbar(ventana3,variable=progress[1],maximum=100,
orient="vertical", length=150, mode="determinate")
                 for l in range(8)]
style = ttk.Style()
style.configure("TProgressbar", thickness=5)

nombre_canal[0].config(text = "ch1")
nombre_canal[1].config(text = "ch2")
nombre_canal[2].config(text = "ch3")
nombre_canal[3].config(text = "ch4")
nombre_canal[4].config(text = "ch5")
nombre_canal[5].config(text = "ch6")
nombre_canal[6].config(text = "ch7")
nombre_canal[7].config(text = "ch8")

nombre_canal[0].place(x=280,y=265)
nombre_canal[1].place(x=310,y=265)
nombre_canal[2].place(x=340,y=265)
nombre_canal[3].place(x=370,y=265)
nombre_canal[4].place(x=400,y=265)
nombre_canal[5].place(x=430,y=265)
nombre_canal[6].place(x=460,y=265)
nombre_canal[7].place(x=490,y=265)

barra_volumen[0].place(x=290,y=110)
barra_volumen[1].place(x=320,y=110)
barra_volumen[2].place(x=350,y=110)
barra_volumen[3].place(x=380,y=110)
barra_volumen[4].place(x=410,y=110)
barra_volumen[5].place(x=440,y=110)
barra_volumen[6].place(x=470,y=110)
barra_volumen[7].place(x=500,y=110)
```

A.1.3. Ventana Save File

```
# Se importan las librerias
From tkinter import *
import tkinter
from tkinter import ttk
import ventana2

# Pone la variable guardar a True y cierra la ventana
def si():
    guardar.set(True)
    savefile.destroy()

# Pone la variable guardar a False y cierra la ventana
def no():
    guardar.set(False)
    savefile.destroy()

# Devuelve el valor de la variable guardar
def guardarwav():
    return guardar.get()

# Se inicializa la ventana
def mostrar_ventana(nombre):
    global savefile
    global guardar

    savefile = Toplevel()
    savefile.title("Guardar archivo")
    savefile.geometry("400x240")

    guardar = tkinter.BooleanVar()
    guardar.set(False)
    texto = tkinter.StringVar()

    campo_texto = tkinter.Entry(savefile)
    campo_texto.insert(0, nombre)
    campo_texto.config(width=37)
    campo_texto.place(x=60,y=90)

    label = tkinter.Label(savefile, text="Quiere guardar el archivo:",font=("arial", 11),)
    label.place(x=120,y=50)

    btn1 = Button(savefile,text="Si",command=si)
    btn1.place(x=150,y=130)
    btn2 = Button(savefile,text="No",command=no)
    btn2.place(x=210,y=130)
    savefile.wait_window()
```

A.1.4. Captura de audio

```
#Se importan las librerias
import threading
import pyaudio
import wave
import numpy as np
from datetime import datetime
import SaveFile
import ventana3
import ventana2
from tkinter import *
import tkinter
from tkinter import ttk
import Potenciómetro

# Se actualiza la barra de volumen en cada iteracion
def act_barra_volumen(output_buffer):

    for g in range(8):
        vol = 0
        progreso = ventana3.get_barra_volumen()
        if np.isnan(abs(np.sqrt(np.mean(np.square(output_buffer[:,g]))))):
            pass
        else:
            vol = abs(np.sqrt(np.mean(np.square(output_buffer[:,g]))))
            print(vol)
            progreso[g].set(vol)

# Se inicia el proceso de grabacion
def iniciar_grabacion(*args):
    rec_channels = args
    to_rec = []
    for i in range(8):
        if rec_channels[i] == True:
            to_rec.append(i)

    print(to_rec)

    FORMAT = pyaudio.paInt24
    CHANNELS = 8
    RATE = 48000
    CHUNK = 256
    CANAL_GRABAR = to_rec
    audio = pyaudio.PyAudio()
    factor = 0.55

    frames = []
    stop = False

    # Se inicializan los Streams
    stream_in = audio.open(format=FORMAT,
                           channels=CHANNELS,
                           rate=RATE,
                           input=True,
                           output_device_index=1,
                           frames_per_buffer=CHUNK)
```

```
stream_out = audio.open(format=FORMAT,
                        channels=CHANNELS,
                        rate=RATE,
                        output=True,
                        output_device_index=0,
                        frames_per_buffer=CHUNK)

print('Grabacion iniciada...')
# Bucle principal
while True:

    stop = ventana3.get_stop()
    if stop == True:
        break
    data = stream_in.read(CHUNK)
    audio_data = np.frombuffer(data, dtype=np.int24)
    audio_data = audio_data.reshape(-1, CHANNELS)

    output_buffer = np.zeros((len(audio_data), 8), dtype=np.int24)

    for i in range (len(to_rec)):
        output_buffer[:,to_rec[i]] = audio_data[:,to_rec[i]]
    try:
        for i in range(8):
            pot = MCP3008(channel=i)
            output_buffer[:,i] = output_buffer[:,i] *round(pot.value,1)*factor
    except Exception as e:
        break

    output_buffer = np.clip(output_buffer, -32768, 32767)
    act_barra_volumen(output_buffer)
    mono_data = np.mean(output_buffer, axis=1).astype(np.int24)
    frames.append(mono_data.tobytes())
    stream_out.write(mono_data.tobytes())

    stream_in.stop_stream()
    stream_in.close()
    stream_out.stop_stream()
    stream_out.close()
    audio.terminate()

    date = datetime.now()
    ventana3.ventana3.withdraw()
    SaveFile.mostrar_ventana(f"Record of {date}.wav")

    guardar = SaveFile.guardarwav()

    if guardar == True:
        wf = wave.open(f"/home/pi3/Desktop/TFG/Grabaciones/Record of {date}.wav", 'wb')
        wf.setnchannels(1)
        wf.setsampwidth(audio.get_sample_size(FORMAT))
        wf.setframerate(RATE)
        wf.writeframes(b''.join(frames))
        print("Guardado correctamente")
    else:
        print("No se ha guardado")
```

A.2 Código del reproductor

A.2.1. Ventana reproductor I

```

# Se importan las librerias
from tkinter import *
import tkinter
from tkinter import ttk
import ventana1
from tkinter import filedialog
import Prueba
import os
import numpy as np
import ventana5
import matplotlib.pyplot as plt
import ventana1

# Vuelve a la ventana principal
def regresar():
    ventana4.destroy()
    ventana1.get_raiz().iconify()
    ventana1.get_raiz().deiconify()

# Se selecciona el archivo a reproducir
def select_file():
    valor = 0
    for j in range(8):
        if file[j] == '':
            valor = j
            break
        elif not file[7] == '':
            valor = 7
            break

    file_path =
tkinter.filedialog.askopenfilename(initialdir='/home/pi3/Desktop/TFG/Grabaciones',title=
'Files',filetypes=[('archivos wav','*.wav')])
    string_var[valor].set(file_path)
    file_name = os.path.basename(file_path)
    campo_texto[valor].delete(0, tkinter.END)
    campo_texto[valor].insert(0, file_name)
    file[valor] = file_path

    if file[0] == '':
        btnB['state']='disable'
    else:
        btnB.config(state=tkinter.NORMAL)
        cambiar_color_circulo(circulo[valor])
        btnB['state']='normal'

    if not file[7] == '':
        btn2['state']='disable'

# Se habilita y deshabilita el boton PLAY
def actualizar_boton():
    x = 0
    blanco = 0
    for i in range (8):

```

```
if btnCH[i].cget("background") == colores[0]:
    blanco = blanco + 1
else:
    pass
if blanco == 8:
    btn18['state']='disabled'
else:
    btn18.config(state=tkinter.NORMAL)

# Cambia el color de los circulos
def cambiar_color_circulo(circulo):

    valor = 0
    for i in range(9):
        if i == 0:
            pass
        elif valores[i] == False:
            valores[i] = True
            valor = i
            break

    canvas.itemconfig(circulo, fill= colores[valor])

# Cambia a color blanco
def cambiar_a_blanco(circulo):
    color = canvas.itemcget(circulo,"fill")
    if color == colores[0]:
        valores[0] == True
    else:
        valor = 0
        for x in range(9):
            if color == colores[x]:
                valor = x
                break
        else:
            pass

    valores[valor] = False

    canvas.itemconfig(circulo, fill=colores[0])

# Borra el ultimo archivo introducido
def borrar():
    valor = 0
    for i in range(8):
        if file[i] == '':
            valor = i
            break
        elif not file[7] == '':
            valor = 8

    file[valor-1] = ''
    campo_texto[valor-1].delete(0, tkinter.END)

    for i in range(8):
        if canvas.itemcget(circulo[i],"fill") == btnCH[i].cget("background"):
            btnCH[i].config(bg = colores[0])

    cambiar_a_blanco(circulo[valor-1])
    string_var[valor-1].set('')
```

```

if file[0] == '':
    btnB['state']='disable'
    actualizar_boton()
    btn2['state']='normal'

# Inicia la reproduccion
def reproducir():
    #file_list()
    ventana4.withdraw()
    ventana5.mostrar_ventana()
    #Prueba.reproducir(file,btnCH,colores)

# Devuelve el array file
def file():
    return file

# Devuelve el boton btnCH
def btnCH():
    return btnCH

# devuelve el array colores
def colores():
    return colores

# Se seleccionan los canales por los que se quiere reproducir
def select_channel(btnCH):
    valor = 0
    color = btnCH.cget("background")
    nuevo = 0
    no_true = 0
    if color == colores[8]:
        nuevo = 0
    else:
        for x in range(9):
            if color == colores[x]:
                valor = x
            else:
                pass

    for i in range(valor+1, 9):
        if valores[i] == True:
            nuevo = i
            no_true = 1
            break
        else:
            pass

    if no_true == 1:
        btnCH.config(bg = colores[nuevo])
    else:
        btnCH.config(bg = colores[0])

    actualizar_boton()

# Se inicializa la ventana
def mostrar_ventana():
    global ventana4
    global string_var
    global btn18

```

```
global colores
    global circulo
    global canvas
    global valores
    global btnCH
    global file
    global btn2

ventana4 = Toplevel()

ventana4.title("Reproductor")
ventana4.geometry("800x480")

string_var = [tkinter.StringVar() for _ in range(8)]
campo_texto = [tkinter.Entry(ventana4, width=35) for _ in range(8)]
btnB = tkinter.Button(ventana4)
btnCH =[Button(ventana4, font=("arial",18),bg="white",
               fg="black",padx=1,pady=1,relief="groove",bd=5) for _ in range(8)]
canvas = tkinter.Canvas(ventana4, width=50, height=500)
canvas.pack()
x0, y0 = 20, 20
x1, y1 = 40, 40
colores =
["white","green","red","blue","purple","orange","yellow","lightblue","lightgreen"]
circulo = [canvas.create_oval(x0, y0, x1, y1, fill=colores[0], outline="black")for _
in range(8)]
valores = [True,False,False,False,False,False,False]
file = ['','','','','','','','']

label1 = Label(ventana4,text="Seleccionar salidas")
label1.place(x=500,y=5)

label2 = Label(ventana4,text="Seleccionar archivos a reproducir")
label2.place(x=115,y=5)

btn = Button(ventana4,text="Back",command=regresar)
btn.place(x=10,y=10)

btn2 = Button(ventana4,text="Select file",command=lambda:select_file())
btn2.place(x=140, y=25)

btnB.config(text="Borrar",command=lambda:borrar())
btnB.config(state=tkinter.DISABLED)
btnB.place(x=235, y=25)

campo_texto[0].place(x=90,y=55)

canvas.move(circulo[0],2,35)

campo_texto[1].place(x=90,y=110)

canvas.move(circulo[1],2,92)

campo_texto[2].place(x=90,y=165)

canvas.move(circulo[2],2,147)

campo_texto[3].place(x=90,y=220)
```

```
canvas.move(circulo[3],2,202)

campo_texto[4].place(x=90,y=275)

canvas.move(circulo[4],2,260)

campo_texto[5].place(x=90,y=330)

canvas.move(circulo[5],2,312)

campo_texto[6].place(x=90,y=385)

canvas.move(circulo[6],2,367)

campo_texto[7].place(x=90,y=440)

canvas.move(circulo[7],2,422)

btnCH[0].config(text = 'Ch-1',command=lambda:select_channel(btnCH[0]))
btnCH[0].place(x=520,y=50)

btnCH[1].config(text = 'Ch-2',command=lambda:select_channel(btnCH[1]))
btnCH[1].place(x=520,y=100)

btnCH[2].config(text = 'Ch-3',command=lambda:select_channel(btnCH[2]))
btnCH[2].place(x=520,y=150)

btnCH[3].config(text = 'Ch-4',command=lambda:select_channel(btnCH[3]))
btnCH[3].place(x=520,y=200)

btnCH[4].config(text = 'Ch-5',command=lambda:select_channel(btnCH[4]))
btnCH[4].place(x=520,y=250)

btnCH[5].config(text = 'Ch-6',command=lambda:select_channel(btnCH[5]))
btnCH[5].place(x=520,y=300)

btnCH[6].config(text = 'Ch-7',command=lambda:select_channel(btnCH[6]))
btnCH[6].place(x=520,y=350)

btnCH[7].config(text = 'Ch-8',command=lambda:select_channel(btnCH[7]))
btnCH[7].place(x=520,y=400)

btn18 = Button(ventana4,text="Reproducir",font=("arial",18),bg="lightgray",
               fg="black",padx=5,pady=20,relief="groove",bd=5,command=reproducir)
btn18.place(x=630, y=190)
btn18.config(state=tkinter.DISABLED)
```

A.2.2. Ventana reproductor II

```
# Se importan las librerias
from tkinter import *
import tkinter
from tkinter import ttk
import ventana4
import Prueba
import threading
import matplotlib.pyplot as plt
from scipy.io import wavfile
import numpy as np
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import os

# Detener la reproduccion
def parar():
    nuevo_valor = not stop.get()
    if nuevo_valor == True:
        pause.config(text='>')
    else:
        pause.config(text='||')
    stop.set(nuevo_valor)

# Devuelve el valor de stop
def get_stop():
    return stop.get()

# Devuelve el valor de finish
def get_finish():
    return finish.get()

# Comprueba el estado del hilo cada segundo
def schedule_check(audio_thread):
    ventana5.after(1000, check_if_done, audio_thread)

# Comprueba si el hilo ha terminado su ejecucion
def check_if_done(audio_thread):
    if not audio_thread.is_alive():
        ventana5.destroy()
        ventana4.mostrar_ventana()
    else:
        schedule_check(audio_thread)

# Se inicia la reproduccion en un hilo
def reproducir():
    global audio_thread
    audio_thread = threading.Thread(target=Prueba.reproducir,
args=(ventana4.file,ventana4.btnCH,ventana4.colores))
    audio_thread.daemon = True
    audio_thread.start()
    schedule_check(audio_thread)

# Devuelve el valor de progress
def get_barra_volumen():
    return progress
```

```

# Se inicializa la ventana
def mostrar_ventana():
    global ventana5
    global stop
    global finish
    global pause
    global progress

    ventana5 = Toplevel()
    ventana5.title("Reproduciendo")
    ventana5.geometry("800x480")

    reproducir()

    stop = tkinter.BooleanVar()
    stop.set(False)

    finish = tkinter.BooleanVar()
    finish.set(False)

    nombre_canal = [Label(ventana5) for _ in range(8)]

    texto = [tkinter.Entry(ventana5, width=45) for _ in range(8)]

    progress = [tkinter.DoubleVar() for _ in range(8)]

    barra_volumen = [ttk.Progressbar(ventana5, variable=progress[1], maximum=100,
orient="horizontal", length=200, mode="determinate")
                     for l in range(8)]

    def finished():
        finish.set(True)

    pause = tkinter.Button(ventana5, text='|||', width=84,
height=3, font=("arial", 12), relief="groove", bd=10, command=parar)
    pause.place(x=0, y=300)

    finito = tkinter.Button(ventana5, text='finalizar', width=84,
height=3, font=("arial", 12), relief="groove", bd=10, command=finished)
    finito.place(x=0, y=390)

    colors = []

    for y in range(8):
        for n in range(9):
            if ventana4.btnCH[y].cget('background') == ventana4.colores[n]:
                colors.append(n)
            else:
                pass

    for i in range(8):
        for x in range(9):
            if x == 0:
                pass
            else:
                if colors[i] == x:
                    nombre_file = os.path.basename(ventana4.file[x-1])
                    texto[i].insert(0, nombre_file)

```

```
else:  
    pass  
  
texto[0].place(x=90,y=55)  
texto[1].place(x=90,y=80)  
texto[2].place(x=90,y=105)  
texto[3].place(x=90,y=130)  
texto[4].place(x=90,y=155)  
texto[5].place(x=90,y=180)  
texto[6].place(x=90,y=205)  
texto[7].place(x=90,y=230)  
  
nombre_canal[0].config(text = "ch1:")  
nombre_canal[1].config(text = "ch2:")  
nombre_canal[2].config(text = "ch3:")  
nombre_canal[3].config(text = "ch4:")  
nombre_canal[4].config(text = "ch5:")  
nombre_canal[5].config(text = "ch6:")  
nombre_canal[6].config(text = "ch7:")  
nombre_canal[7].config(text = "ch8:")  
  
nombre_canal[0].place(x=40,y=55)  
nombre_canal[1].place(x=40,y=80)  
nombre_canal[2].place(x=40,y=105)  
nombre_canal[3].place(x=40,y=130)  
nombre_canal[4].place(x=40,y=155)  
nombre_canal[5].place(x=40,y=180)  
nombre_canal[6].place(x=40,y=205)  
nombre_canal[7].place(x=40,y=230)  
  
style = ttk.Style()  
style.configure("TProgressbar", thickness=5)  
  
barra_volumen[0].place(x=470,y=63)  
barra_volumen[1].place(x=470,y=88)  
barra_volumen[2].place(x=470,y=113)  
barra_volumen[3].place(x=470,y=138)  
barra_volumen[4].place(x=470,y=164)  
barra_volumen[5].place(x=470,y=188)  
barra_volumen[6].place(x=470,y=213)  
barra_volumen[7].place(x=470,y=238)
```

A.2.3. Reproducción de audio

```

# Se importan las librerias
import pyaudio
import wave
import numpy as np
import os
import tkinter
import matplotlib.pyplot as plt
from scipy.io import wavfile
from matplotlib.animation import FuncAnimation
import ventana5
import time
import Potenciómetro
from gpiozero import MCP3008
import reloj

# Se actualizan las barras de volumen en cada iteracion
def act_barra_volumen(output_buffer):

    for g in range(8):
        vol = 0
        progreso = ventana5.get_barra_volumen()
        if np.isnan(abs(np.sqrt(np.mean(np.square(output_buffer[:,g]))))):
            pass
        else:
            vol = abs(np.sqrt(np.mean(np.square(output_buffer[:,g]))))
            print(vol)
            progreso[g].set(vol)

# Se inicializa e inicia la reproducción
def reproducir(file,btnCH,colores):
    reloj.clock()
    CHUNK = 256
    volume=1
    print(file)
    wav = []
    wf = []
    factor = 0.55

    size = 0
    for x in range(len(file)):
        if os.path.isfile(file[x]):
            wav.append(file[x])
            wf.append(wave.open(wav[size], 'rb'))
            size = size+1
        else:
            size = size
    colors = []

    for y in range(8):
        for n in range(9):
            if btnCH[y].cget('background') == colores[n]:
                colors.append(n)
            else:
                pass
    print (colors)

p = pyaudio.PyAudio()

```

```
p = pyaudio.PyAudio()

stream = p.open(format=pyaudio.paInt24,
                 channels=8,
                 rate=48000,
                 output=True,
                 output_device_index=0,
                 frames_per_buffer=CHUNK)

print("Reproduciendo audio...")
stop = False
finish = True
# Bucle principal
try:
    while True:
        stop = ventana5.get_stop()
        finish = ventana5.get_finish()

        if finish == True:
            break
        while stop:
            finish = ventana5.get_finish()
            if finish == True:
                break
            time.sleep(1)
            stop = ventana5.get_stop()

        data = []
        samples = []
        r = False
        for i in range(size):
            data.append(wf[i].readframes(CHUNK))

        if all(elemento == 0 for elemento in data[i]):
            r = True
            pass

        else:
            samples.append(np.frombuffer(data[i], dtype=np.int124))

        if r == True:
            break

    num_samples = len(samples[0])

    output_buffer = np.zeros((num_samples, 8), dtype=np.int24)

    for t in range(len(colors)):

        if colors[t] == 0:
            pass
        else:
            output_buffer[:,t] = samples[colors[t]-1]
    try:
        for i in range(8):
            pot = MCP3008(channel=i)
            output_buffer[:,i] = output_buffer[:,i] *round(pot.value,1)*factor
    except Exception as e:
        break
```

```
output_buffer = np.clip(output_buffer, -32768, 32767)

act_barra_volumen(output_buffer)

output_data = output_buffer.tobytes()

stream.write(output_data)

except KeyboardInterrupt:
    pass

print("Finalizando...")
stream.stop_stream()
stream.close()
for u in range(len(wf)):
    wf[u].close()
p.terminate()
```

A.3 Código Ventana principal

```
# Se importan las librerias
from tkinter import *
import tkinter
from tkinter import ttk
import ventana2
import ventana3
import ventana4
from PIL import Image, ImageTk

# Ir al grabador
def go2():
    ventana.withdraw()
    ventana2.mostrar_ventana()

# Ir al reproductor
def go4():
    ventana.withdraw()
    ventana4.mostrar_ventana()

# Devuelve la ventana principal
def get_raiz():
    return ventana

# Inicializa la ventana principal
def mostrar_ventana():
    global ventana
    #global raiz

    ventana = Tk()
    ventana.title("Menu Principal")
    ventana.geometry("800x480")
    ventana.config(bg="white")

    image_path = "/home/pi3/Desktop/TFG/Interfaz/wave.jpg"
    imagen = Image.open(image_path)
    photo = ImageTk.PhotoImage(imagen)
    tkinter.Label(ventana, image= photo).place(x=0,y=80)

    texto = tkinter.StringVar()
    texto.set("Grabador reproductor: Samplerate = 48000Hz y Format = 24 bits ")
    label = tkinter.Label(ventana,
    textvariable=texto,bg="white",fg="purple",font=("arial", 18),)
    label.place(x=60,y=35)

    btn1 = Button(ventana,text="REC", width=10,
    height=5,font=("arial",18),bg="lightgray",
    fg="purple",relief="groove",bd=10,command=go2)
    btn1.place(x=190,y=120)

    btn3 = Button(ventana,text="PLAY", width=10, height=5,
    font=("arial",18),bg="lightgray",
    fg="purple",relief="groove",bd=10,command=go4)
    btn3.place(x=440,y=120)

    ventana.mainloop()
```