



UCA

Universidad
de Cádiz

Escuela Superior
de Ingeniería

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS
INDUSTRIALES

**DISEÑO Y PROTOTIPADO DE UNA
PEDALERA DIGITAL MULTIEFECTOS PARA
GUITARRA**

AUTOR: CÉSAR CINOS SÁNCHEZ

Cádiz, junio 2024



Universidad
de Cádiz

Escuela Superior
de Ingeniería

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS
INDUSTRIALES

**DISEÑO Y PROTOTIPADO DE UNA
PEDALERA DIGITAL MULTIEFECTOS PARA
GUITARRA**

DIRECTOR: EDUARDO ALEJANDRO ROMERO BRUZÓN

CO-DIRECTOR: ARTURO MORGADO ESTÉVEZ

AUTOR: CÉSAR CINOS SÁNCHEZ

Cádiz, junio 2024

A Avicii y Linkin Park, descubrirme la música.
A Kardinal Punch, por enseñarme cómo funciona la música.
Y a mis amigos, por compartir y disfrutar de la música conmigo todos los días.

ÍNDICE

1.	INTRODUCCIÓN	11
1.1.	OBJETIVOS Y ALCANCE DEL PROYECTO	13
1.1.1.	ETAPAS DEL PROYECTO	14
2.	DESARROLLO.....	15
2.1.	ELECCIÓN DEL HARDWARE	15
2.1.1.	ETAPAS BÁSICAS	15
2.1.2.	REQUISITOS DEL ADC.....	16
2.1.3.	REQUISITOS DEL MICROCONTROLADOR	17
2.1.4.	REQUISITOS DEL DAC.....	18
2.1.5.	OPCIONES Y DECISIÓN	18
2.2.	DISEÑO DE LA PCB.....	21
2.2.1.	CIRCUITO PREVIO ADC.....	21
2.2.2.	CIRCUITO POSTERIOR DAC	26
2.2.3.	IMPRESIÓN DE LA PCB	29
2.3.	DISEÑO DEL SOFTWARE.....	31
2.3.1.	DMA (DIRECT MEMORY ACCESS).....	32
2.3.2.	CONFIGURACIÓN DE PERIFÉRICOS	34
2.3.3.	EFFECTOS.....	39
2.3.4.	DIBUJO DE LAS PANTALLAS DE LA INTERFAZ DE USUARIO.....	46
2.3.5.	DIAGRAMA DE FUNCIONAMIENTO	50
2.4.	DISEÑO DEL ENCAPSULADO	59
3.	PRUEBAS	64
4.	PRESUPUESTO	67
5.	MEJORAS PROPUESTAS	68
6.	CONCLUSIONES FINALES	69
7.	BIBLIOGRAFÍA	70

ÍNDICE DE FIGURAS

Figura 1: Funcionamiento de una pastilla de guitarra eléctrica (Doxrud, 2019)	11
Figura 2: Cadena de pedales (Thomann, 2018)	12
Figura 3: 'Guitar Rig' de Native Instruments.....	13
Figura 4: Diagrama de Gantt del proyecto.....	14
Figura 5: Etapas principales del proyecto	15
Figura 6: Conversión ADC.....	16
Figura 7: Ejemplo de aliasing (National Instruments, 2023)	17
Figura 8: Arduino DUE (Arduino, 2019)	19
Figura 9: STM32F407VET6. (Hudak, 2018).....	19
Figura 10: ILI93441 conectado a la placa mediante pines dedicados (ACELEX, n.d.)	20
Figura 11: Esquema de diseño de la pedalera	20
Figura 12: Características del ADC	21
Figura 13: Circuito previo ADC	22
Figura 14: Teorema de superposición.....	23
Figura 15: Simulación de la respuesta en el dominio del tiempo del circuito previo al ADC	25
Figura 16: Simulación de la respuesta en frecuencia del circuito previo al ADC	26
Figura 17: Voltajes de salida del DAC.....	26
Figura 18: Circuito posterior DAC	27
Figura 19: Simulación de la respuesta en el dominio del tiempo del circuito posterior al DAC.....	28
Figura 20: Simulación de la respuesta en frecuencia del circuito posterior al DAC	28
Figura 21: Diseño de la PCB en EAGLE	29
Figura 22: Diseño de la impresión de la PCB con capas GND	30
Figura 23: PCB con los componentes soldados.....	30
Figura 24: STM32Cube IDE.....	31
Figura 25: Esquema de navegación del usuario.....	32
Figura 26: Diagrama de funcionamiento Half Complete y Complete Callback.....	33
Figura 27: Ecuación para la configuración del timer.....	34
Figura 28: configuración de TIM2	34
Figura 29: Configuración del ADC1	35
Figura 30: Configuración del DMA para el ADC1	36
Figura 31: Configuración del DAC1	36
Figura 32: configuración del DMA para el ADC1.....	37
Figura 33: Configuración del GPIO del pin PB5	37
Figura 34: Activación de las interrupciones externas	38
Figura 35: Configuración de los periféricos necesarios para la pantalla LCD	38
Figura 36: Ejemplo de señal de Overdrive	39
Figura 37: Esquema y funcionamiento del efecto de delay (Ramirez, 2017)	40
Figura 38: Diferencia entre Vibrato y Tremolo (Espinosa, 2014).....	42
Figura 39: Función seno LFO a 1Hz	43
Figura 40: Función seno LFO a 1Hz transformada	43
Figura 41: Efecto tremolo a una señal seno de 5Hz.....	44
Figura 42: Pantalla de menu	46
Figura 43: Pantalla del modo PLAY	47
Figura 44: Pantalla de configuración de presets.....	47

Figura 45: Pantalla de configuración de los efectos del preset	48
Figura 46: Pantalla de configuración del efecto Overdrive.....	48
Figura 47: Pantalla de configuración del efecto Delay	49
Figura 48: Pantalla de configuración del efecto Chorus	49
Figura 49: Pantalla de configuración del efecto Tremolo	50
Figura 50: Diagrama de flujo del menú principal.....	51
Figura 51: Diagrama de flujo del modo PLAY.....	52
Figura 52: Diagrama de flujo de la función ADC_ConvHalfCpltCallback.....	53
Figura 53: Diagrama de flujo de la función ADC_ConvCpltCallback	55
Figura 54: Diagrama de flujo de las interrupciones externas	56
Figura 55: Diagrama de flujo de la selección del preset a configurar	57
Figura 56: Diagrama de flujo del preset elegido para configurar	58
Figura 57: Diagrama de flujo de la configuración del efecto a modificar	59
Figura 58: Diseño de la carcasa 3D	60
Figura 59: Diseño 3D de la pieza superior 1.....	60
Figura 60: Diseño 3D de la pieza superior 2.....	61
Figura 61: Diseño 3D de la pieza inferior	61
Figura 62: Proceso de pintado de las piezas superiores	62
Figura 63: Mejora de la pieza inferior	62
Figura 64: Pedalera ensamblada.....	63
Figura 65: Prueba del efecto Overdrive	64
Figura 66: Prueba del efecto Delay	64
Figura 67: Prueba del efecto Chorus.....	65
Figura 68: Prueba del efecto Tremolo	65
Figura 69: Prueba de la combinación de Overdrive y Tremolo.....	66

1. INTRODUCCIÓN

Desde que Fender sacara al mercado la primera guitarra eléctrica tal y como la concebimos hoy en día en 1950, guitarristas y productores han ido experimentando con el sonido de la misma según avanzaba la industria musical. (Wikipedia, 2023b)

En el caso de la guitarra eléctrica, al no tener ninguna caja de resonancia como tienen las guitarras acústicas y españolas, el sonido debe de amplificarse y para ello se genera una señal eléctrica que irá a un amplificador. Esta señal se genera gracias a las ‘pastillas’ (*‘pick-ups’* en inglés), cuyo funcionamiento se basa en unos imanes rodeados por bobinas que generan un campo magnético y este al ser alterado por la vibración de las cuerdas de acero, inducen una señal eléctrica que saldrá hacia al amplificador, tal como se indica en la Figura 1. Este sonido se ve afectado por varios factores como la madera del cuerpo de la guitarra, cómo el tipo de pastilla que se está usando, si el bobinado es doble o no, etc. Pero en este texto vamos a centrarnos en la modificación del sonido una vez que ya ha salido de la guitarra.

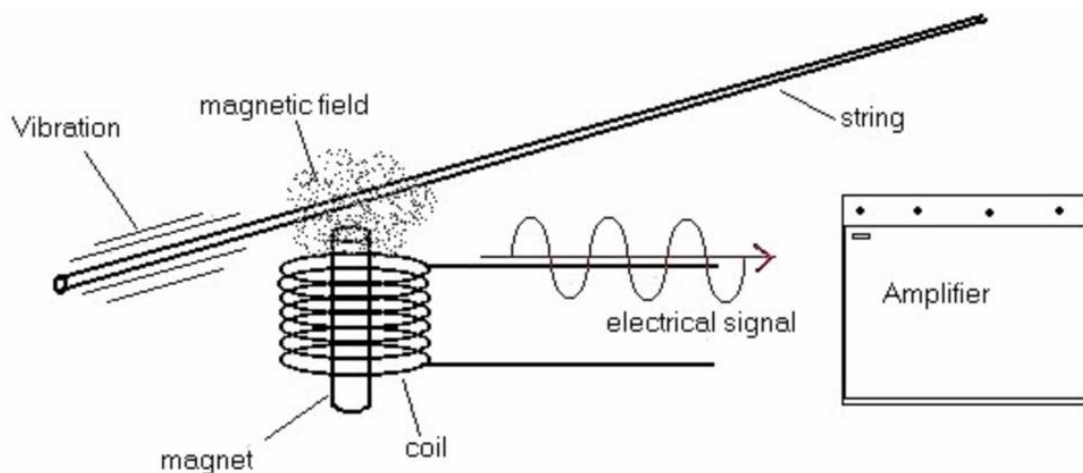


Figura 1: Funcionamiento de una pastilla de guitarra eléctrica (Doxrud, 2019)

Según Wikipedia, “...el concepto de manipulación de la señal comenzó, con la interacción de la saturación. Para ese entonces, los amplificadores de guitarra eran de baja fidelidad, y sólo producían el efecto de distorsión por error al superar el umbral de volumen que el sistema electrónico podía permitir (conocido ahora como *‘Overdrive’*). En esa época, la distorsión era vista como un error en el sonido que debía ser reparado inmediatamente.” (2023a). Pero poco a poco guitarristas como Jimi Hendrix fueron aceptando ese sonido y dándole uso en sus canciones y se fue extendiendo el uso de la distorsión. Algunos guitarristas usaban técnicas tan innovadoras como agujerear los altavoces con un lápiz o una cuchilla de afeitar para obtener este efecto, pero esto quedó atrás gracias a la aparición de los pedales de efectos que imitaban este sonido de manera electrónica.

Fue entonces cuando en los 60s y los 70s las empresas empezaron a sacar al mercado sus innovaciones en este campo y creando distintos pedales con distintos efectos como el anteriormente mencionado *‘Overdrive’*, *‘Delay’*, *‘Reverb’*, *‘Chorus’*, etc. Estos pedales se pueden conectar en cadena, formando un sonido único para cada guitarrista, algunas de estas cadenas pueden llegar a tener varios componentes como podemos ver en la Figura 2.



Figura 2: Cadena de pedales (Thomann, 2018)

Con el avance del ‘procesamiento de señales digitales’ (siglas DSP en inglés), las marcas han ido simulando estos pedales de manera digital facilitando al usuario su acceso mediante la incorporación de los mismos en programas ‘*standalone*’ en ordenadores o como ‘*plugins*’ en los denominados DAW (‘*Digital Audio Workstation*’). El DSP ha ido evolucionando hasta tal punto que no sólo ha conseguido simular los pedales, sino que también ha llegado al punto de simular amplificadores específicos de fabricantes, altavoces e incluso el método de grabación de guitarras en la que entran muchísimas variables a simular como por ejemplo el tipo de micrófono, la distancia al altavoz, la inclinación, la posición y la habitación en la que se encuentra. Ejemplo de ello es el famoso programa ‘Guitar Rig’ de Native Instruments, como vemos en la Figura 3.



Figura 3: 'Guitar Rig' de Native Instruments

En este proyecto nos centraremos en los algunos de los efectos más básicos y conocidos que simulen pedales de pie para guitarra.

1.1. OBJETIVOS Y ALCANCE DEL PROYECTO

El trabajo tendrá como fin diseñar y prototipar una pedalera para guitarra que simule los efectos principales utilizando DSP en un microcontrolador. Los efectos serán configurables mediante una pantalla táctil en la que se podrán modificar distintos parámetros de cada efecto para personalizar lo máximo posible el sonido. Además, contará con 3 botones de pie o '*footswitch*' con los que podremos activar o desactivar los efectos entre otras funciones.

Para poder modificar la señal, deberá tratarse primero analógicamente para adaptarla a la resolución del ADC del microcontrolador, así como posteriormente, después del DAC, se adaptará la señal para su entrada en el amplificador de guitarra. Todo el circuito se protegerá debidamente con una carcasa que sólo deje visibles los componentes necesarios para el uso por el usuario como la pantalla, los botones, las entradas y salidas de audio, etc.

El proyecto partirá de una placa de desarrollo programable en C y depurable mediante USB, una pantalla LCD y unos '*footswitch*' (botones para activarlos con el pie dándoles un pisotón). Se elaborará una selección del hardware necesario y se diseñará una PCB para adaptar la señal, se diseñará el software con ayuda de bibliotecas dadas por la propia placa o externas que podamos necesitar y se diseñará y construirá la carcasa exterior. Además, se realizarán pruebas de su funcionamiento.

1.1.1. ETAPAS DEL PROYECTO

El proyecto constará de las siguientes etapas:

- **Elección del Hardware y Software a utilizar:** En esta etapa del proyecto veremos las características de los distintos componentes elegidos, así como la justificación de los mismos.
- **Diseño de la adaptación analógica de la señal:** Diseñaremos los distintos circuitos para que el microcontrolador pueda leer la señal proveniente de la guitarra, así como adaptarla igualmente a la salida para la entrada en el amplificador. Se construirá sobre un circuito impreso.
- **Diseño y desarrollo del programa del microcontrolador:** aquí tras estudiar la naturaleza de los distintos efectos, diseñaremos los algoritmos que los simularán y los incluiremos dentro de un programa que permita su configuración por parte del usuario a través de la pantalla táctil y los 'footswitch'.
- **Diseño y construcción del encapsulado:** Para contener todo se construirá una carcasa, para ello se diseñará y se construirá mediante fabricación aditiva con impresoras 3D.
- **Realización de pruebas de funcionamiento:** Por último, se realizarán pruebas de funcionamiento para comprobar el mismo y discutir los resultados.

Finalmente, el proyecto siguió el siguiente diagrama de Gantt:

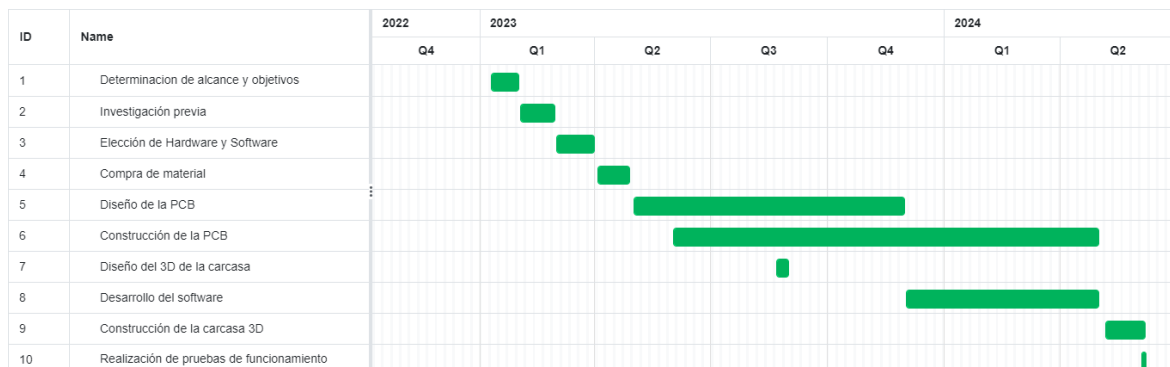


Figura 4: Diagrama de Gantt del proyecto

Como se puede apreciar, ha estado solapándose con la construcción de la PCB, esto se debe a que el circuito ha ido modificándose y solucionando errores hasta llegar al diseño final. De la misma manera que paralelamente se empezó a desarrollar el software y se fue modificando según se podían hacer pruebas del código con la PCB.

2. DESARROLLO

A continuación, se irán desarrollando las fases del proyecto resumidas en el apartado 1.1.1 para la construcción de la pedalera de guitarra hasta la consecución del producto final funcional.

2.1. ELECCIÓN DEL HARDWARE

2.1.1. ETAPAS BÁSICAS

En primer lugar, debemos de explicar cuáles son los requisitos más importantes a tener en cuenta para el funcionamiento de la pedalera, aunque algunos no son específicos para este proyecto si no que son indispensables en cualquier dispositivo enfocado en música y sonido en general.

Dado que este proyecto tiene como base la modificación de manera digital de una señal analógica de guitarra y se requiere de una señal analógica a la entrada del amplificador de guitarra, la pedalera necesita de 3 fases indispensables las cuáles serían:

1. **Conversión de la señal analógica en digital:** Lo primero que hay que hacer es convertir la señal analógica en digital para el DSP. Usaremos un ADC (*'Analog-to-Digital Converter'*).
2. **Modificación de la señal digital:** Con el microcontrolador y usando algoritmos modificaremos la señal para que simulen los distintos efectos de guitarra.
3. **Conversión de la señal digital modificada en analógica:** una vez tenemos la señal con los efectos deseados, para introducirla en el amplificador necesitamos que vuelva a ser analógica, por tanto, utilizaremos un DAC (*'Digital-to-Analog Converter'*).

Entonces los tres componentes principales de la pedalera serán el ADC, el procesador y el DAC. En la Figura 5 se muestra un esquema para facilitar la comprensión:

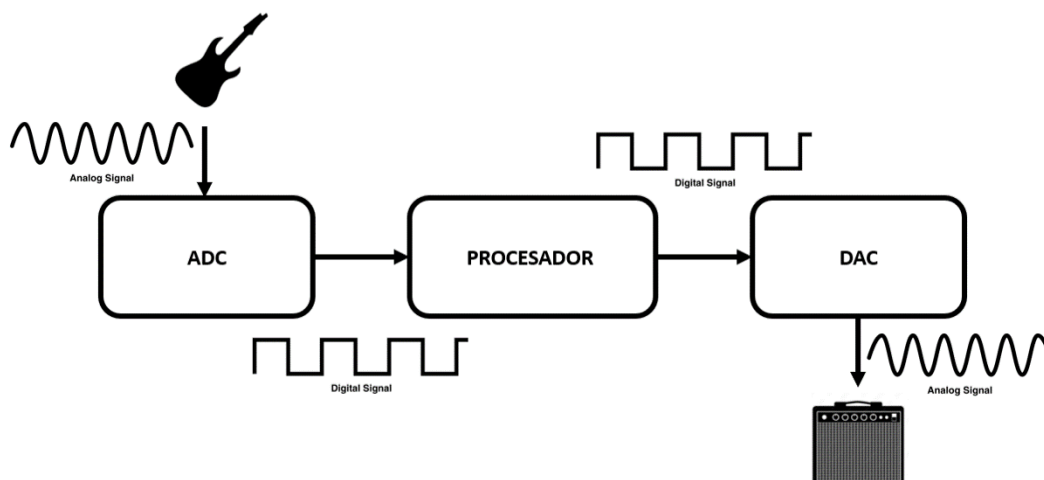


Figura 5: Etapas principales del proyecto

2.1.2. REQUISITOS DEL ADC

El funcionamiento del ADC, explicado de manera muy superficial (pero suficiente para entender su propósito para este proyecto), se basa principalmente en asignar un número a un valor de tensión a la entrada del mismo, es decir, si a la entrada del ADC entran 50mV, se devuelve un valor en binario y si entran 100mV se devolverá un valor binario mayor.

Si introdujéramos una señal sinusoidal, el ADC leería el valor de tensión a la entrada en cada instante que se le ordenara hacerlo y lo convertiría a un número, pero las señales sinusoidales al ser continuas pueden llegar a tener valores infinitos y es imposible que el ADC entregue valores infinitos en binario, la cantidad de valores que podrá entregar dependerá de la profundidad de bits de conversión que disponga. Entonces, dentro del rango de tensiones máximas y mínimas que acepte el ADC (esto se conoce como fondo de escala), según la profundidad de bits que disponga, dicho rango se podrá dividir en más o menos valores tal y como se indica en la Figura 6.

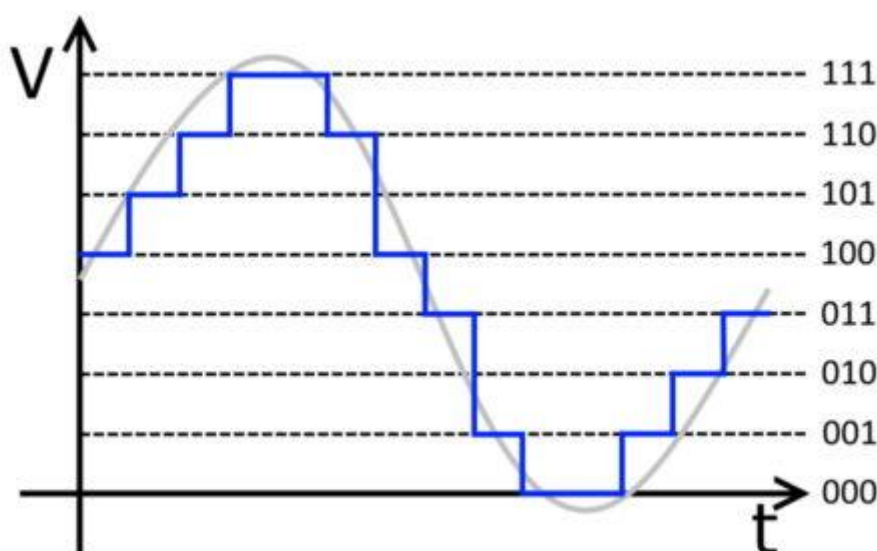


Figura 6: Conversión ADC

A mayor profundidad de bits, mayor será la resolución (unidad más pequeña medible) y por tanto, más se acercará la aproximación digital de la curva a la original analógica. Entonces para que la conversión sea lo más precisa posible tenemos que introducir la señal de manera que los máximos se acerquen lo posible al fondo de escala del ADC y tenemos que contar con un ADC con suficiente resolución para que no se distorsione la señal y la conversión sea precisa.

Para el primer objetivo, y contando con que las guitarras tienen una salida máxima de unos aproximadamente 500mV en pastillas pasivas y unos 4.5V en pastillas activas (varía sobre el tipo de pastilla y marca), tendremos que hacer un circuito que amplifique o atenúe la señal para que llegue al fondo de escala del ADC que elijamos. (Manualguitarraelectrica.com, 2020)

El segundo aspecto sí que será importante a la hora de decidir qué ADC tenemos que utilizar ya que como hemos dicho la señal puede distorsionarse y que suene robótico o simplemente ruido. Las tarjetas de audio estándar para grabación digital suelen rondar los 24 bits, si ya se tiene mayor presupuesto pueden llegar a 32 bits o más, pero hay algunas con suficiente calidad con 16 bits. Aun así, existen proyectos similares con Arduino Due, el cual tiene un ADC de 10 bits, donde se saca un sonido muy aceptable. Por tanto, cualquier ADC que tenga entre 10 y 24 bits de resolución será válido para este proyecto.

2.1.3. REQUISITOS DEL MICROCONTROLADOR

El microcontrolador supondrá el componente fundamental del proyecto, ya que en él se programará el funcionamiento del ADC, del DAC, los algoritmos para la modificación de la señal y controlar la pantalla LCD, así como el funcionamiento del menú interactivo para la modificación de los efectos por el usuario. Por tanto, debe ser potente y rápido para poder gestionar todas estas tareas a la vez.

Pero la velocidad va a ser el factor más importante no solo por las tareas a gestionar sino también porque afectará a la calidad del sonido. Este aspecto viene dado por el “Teorema de muestreo de Nyquist-Shannon”. Este teorema nos dice que la frecuencia de muestreo debe ser del doble del espectro de señales que vamos a muestrear, o, dicho de otro modo, la señal a muestrear debe ser de la mitad de frecuencia que la de muestreo (llamada frecuencia de Nyquist). En el caso de que se muestreara una señal de frecuencia mayor a la frecuencia de Nyquist, aparecería el fenómeno llamado “*Aliasing*” el cual genera una onda de menor frecuencia que la “real”. En la Figura 7 vemos como si muestreamos una señal de 800kHz a una frecuencia de muestreo de 500kHz (la frecuencia de Nyquist sería 250kHz), la frecuencia de la onda percibida sería de 200kHz. (National Instruments, 2023)

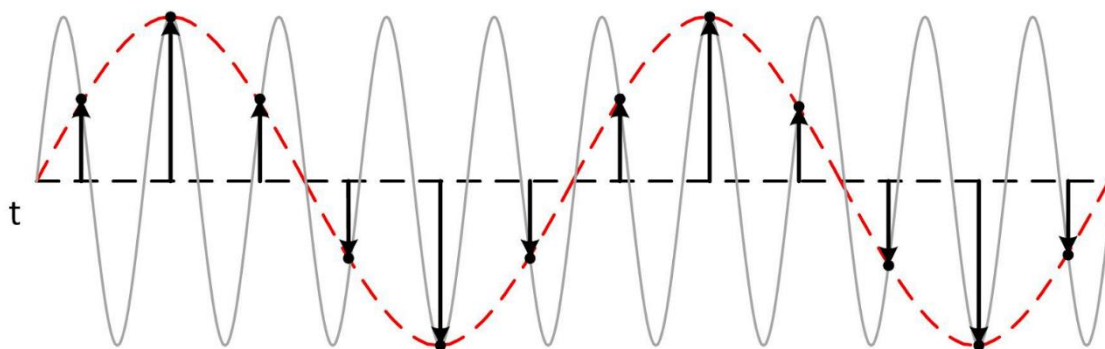


Figura 7: Ejemplo de aliasing (National Instruments, 2023)

En nuestro caso, el oído humano puede escuchar frecuencias desde los 20Hz a los 22kHz por lo que es por ello que en dispositivos de audio está estandarizada la frecuencia de muestreo a 44kHz, aunque son muy utilizados los 48kHz. En nuestro caso, la guitarra tiene sus frecuencias fundamentales hasta los 15kHz, contando con sus diferentes armónicos, por lo que, llegados a necesitarlo, podríamos bajar la frecuencia de muestreo para dejar tiempo a que se lleven a cabo otros procesos del programa.

En definitiva, necesitamos un microcontrolador que trabaje a altas frecuencias para que permita la frecuencia de muestreo de 48 kHz y el resto de procesos. Porque no solo es problema el '*aliasing*', sino también se necesita que la latencia del audio sea la mínima posible por no decir que debe ser nula. Cualquier pedalera puede tener los mejores sonidos y posibilidades en efectos, pero si hay latencia en el audio se hace imposible tocar y la pedalera no sirve de nada.

2.1.4. REQUISITOS DEL DAC

El DAC necesita únicamente que tenga los mismos bits de resolución que el ADC, ya que si no habría problemas entre las muestras leídas y luego habría que hacer un remuestreo de las mismas y se perdería tiempo y capacidad del programa.

Las tensiones de salida no serán un problema, puesto que luego habrá un procesamiento de la señal analógica generada, con filtros y con un control de volumen.

2.1.5. OPCIONES Y DECISIÓN

En resumen, con los apartados anteriores, los requisitos más importantes para nuestro hardware son:

- ADC y DAC con una profundidad de bits de entre 10 y 24 bits.
- Microcontrolador con alta velocidad de operación que permita una frecuencia de muestreo de 48kHz.

Por comodidad y por facilidad de adquisición se ha optado por buscar placas de desarrollo que cumplan con estos requisitos para la construcción del prototipo. En el caso de una producción en masa sería más interesante la adquisición por separado de los componentes y su implantación en un circuito específico, lo que ahorraría bastante espacio.

Cuando pensamos en placas de desarrollo lo primero en lo que pensamos es en Arduino, y el único que nos podía servir por el simple hecho de tener DAC disponibles era el Arduino DUE. Con ADC y DAC disponibles de 12 bits de resolución y una frecuencia de reloj de 84 MHz esta placa de desarrollo era más que aceptable para el cometido de este proyecto.

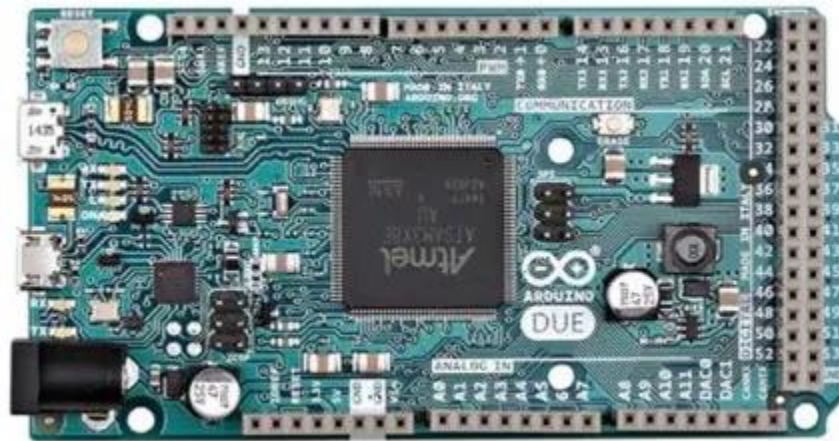


Figura 8: Arduino DUE (Arduino, 2019)

Pero buscando opciones en otros fabricantes nos encontramos con la placa de desarrollo basada en un ARM Cortex M4 de 32 bits de la compañía STMicroelectronics, la STM32F407VET6.

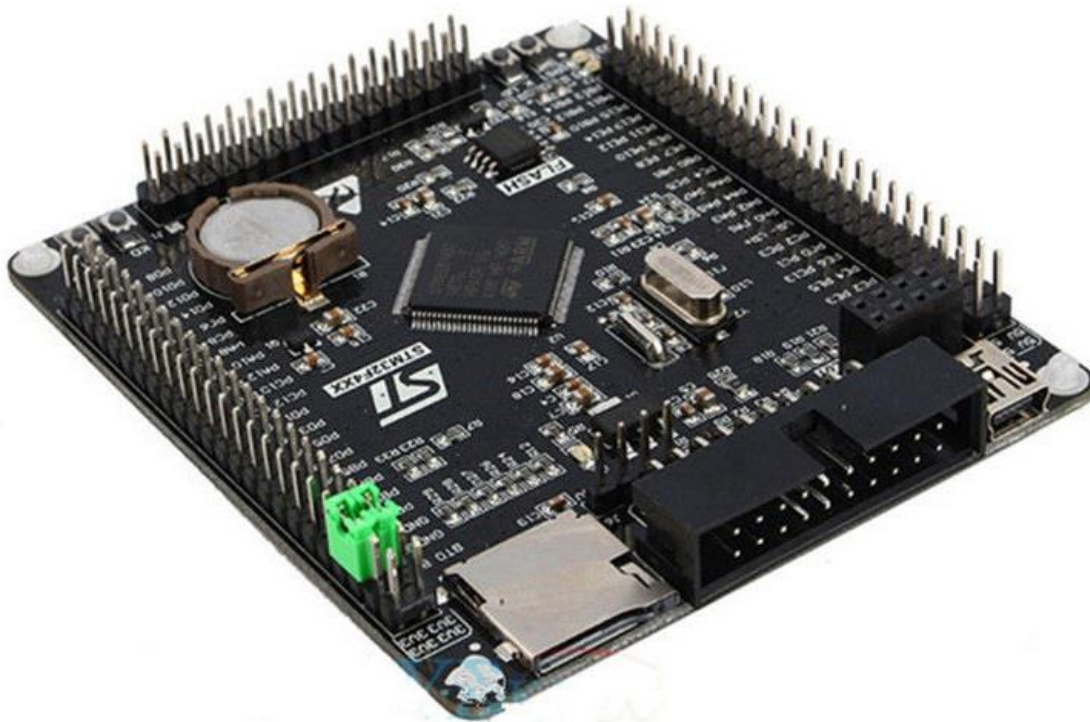


Figura 9: STM32F407VET6. (Hudak, 2018)

Esta placa de desarrollo cuenta con ADC y DAC de 12 bits de resolución y con una frecuencia de reloj mucho más alta que Arduino DUE con 168MHz. Y todo esto a un precio mucho menor que Arduino, ya que este cuesta unos 42€ mientras que la STM32 tiene un precio de 14€. Pero el factor económico no es el único que influye en la decisión, cuenta con varios modos de adquisición ADC, acceso directo

a memoria y sobre todo cuenta con pines dedicados exclusivamente para conectar una pantalla LCD TFT de 3.2 pulgadas con función táctil. Esto nos permitirá tener menos problemas en la conexión de los pines y también ahorrará tiempo en la configuración de los pines de la placa. La pantalla es una ILI9341, y su función táctil también nos permitirá ahorrarnos conexiones de botones que limitarán más aún el espacio disponible, pines de la placa y problemas en la conexión de los mismos.



Figura 10: ILI93441 conectado a la placa mediante pines dedicados (ACELEX, n.d.)

Usando los componentes elegidos, el esquema general del proyecto sería:

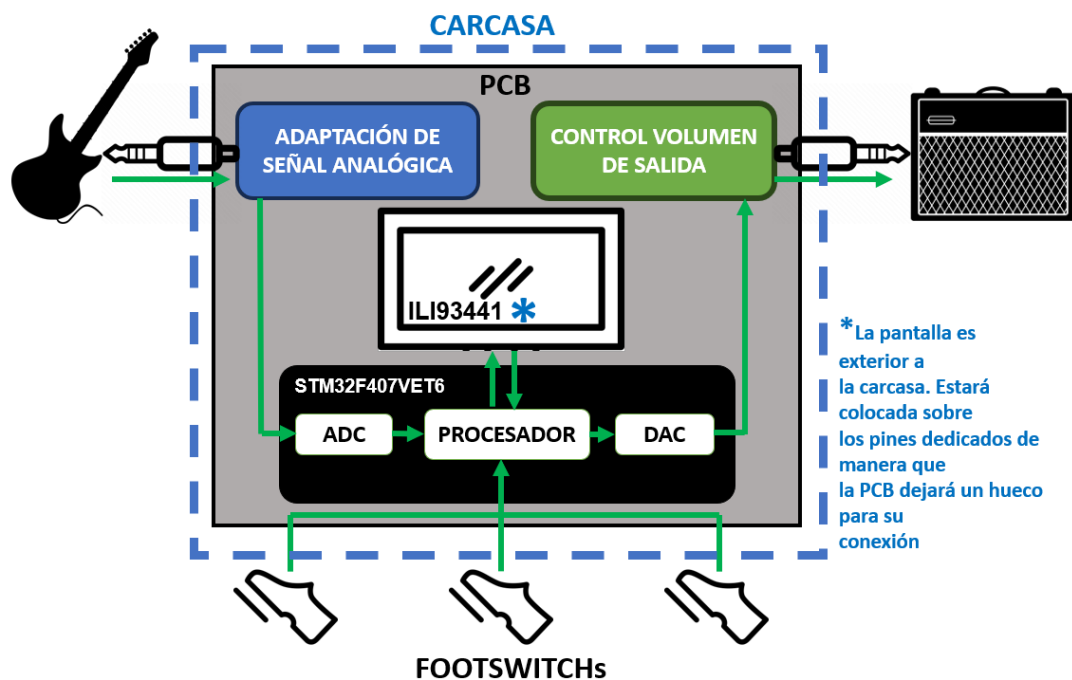


Figura 11: Esquema de diseño de la pedalera

La PCB contendrá los circuitos de entrada y salida de señal sobre la placa de desarrollo, la cual irá también soldada a esta PCB. La carcasa contendrá la PCB y a los *'footswitch'*, dejando a la vista del usuario sólo los elementos con los que interacciona directamente. La señal de guitarra se introducirá mediante el uso de un cable Jack de 6.35mm, y tras la adaptación de la señal, se introducirá a la placa de desarrollo por el ADC. El procesador modificará las muestras del ADC, así como también recibirá las señales de los *'footswitch'* y del táctil de la pantalla LCD además de dibujar las distintas pantallas sobre la misma. Las muestras modificadas pasarán al DAC e irán hacia el amplificador de guitarra pasando antes por un circuito de control del volumen. La salida de la pedalera se conecta al amplificador usando también un cable Jack de 6.35mm.

El siguiente paso sería diseñar la PCB, que contendrá los circuitos de adaptación de la señal analógica y control del volumen.

2.2. DISEÑO DE LA PCB

El propósito principal de diseñar un circuito impreso es el de en primer lugar poder conectar la guitarra mediante un cable Jack de 6.35mm ya que la salida de todas las guitarras son con Jack hembra de 6.35mm entonces se usa siempre en todos los pedales un Jack macho-macho para conectar estas dos etapas y de la misma manera entre los pedales y otros pedales o el amplificador. Y en segundo lugar el de poder adaptar la señal de la guitarra a las especificaciones de lectura del ADC de nuestra placa de desarrollo, al igual que debemos de adaptar la salida del DAC a una señal aceptable por los amplificadores convencionales. Además, también conectaremos los *'footswitch'* y otros elementos para evitar el uso de cables que puedan inducir ruidos o desconexiones por el movimiento y generar problemas durante su uso.

2.2.1. CIRCUITO PREVIO ADC

Si consultamos el datasheet del STM32F407VET6 (Figura 12), podemos ver que el fondo de escala va desde los 0 V a los 3.3 V (es el voltaje de alimentación de la placa).

Table 67. ADC characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDA}	Power supply	-	1.8 ⁽¹⁾	-	3.6	V
V_{REF+}	Positive reference voltage	-	1.8 ⁽¹⁾⁽²⁾⁽³⁾	-	V_{DDA}	
V_{REF-}	Negative reference voltage	-	-	0	-	

Figura 12: Características del ADC

Los voltajes que entregan las pastillas de guitarra son muy variados según el modelo de pastilla que se esté usando. Las pastillas pasivas de bobina simple pueden tener entre 100 y 200mV de salida, mientras las de doble bobina (llamadas *'humbuckers'*) pueden entregar hasta 500mV. Y ya si contamos con las pastillas activas, que son pastillas alimentadas con un circuito preamplificador alimentado con una pila de 9V, se puede llegar a sobrepasar el voltio de salida. Esto hace que las

pastillas pasivas necesitan de amplificación para aprovechar al máximo el rango del ADC y que se lea con la mayor resolución posible y que las pastillas activas necesiten atenuar un poco su voltaje para no pasarse de esos 3.3V. También tenemos que tener en cuenta que la salida de la guitarra tiene valores positivos y negativos y el ADC solo acepta valores positivos hasta los 3.3V, por lo que necesitaremos agregar un offset de 1.65V para que los valores se mantengan en el valor establecido. Los 1.65V serán a partir de ahora nuestro nuevo '0'.

2.2.1.1. DISEÑO DEL CIRCUITO PREVIO ADC

Para conseguir las especificaciones hemos diseñado un circuito que tenga 2 etapas. La primera etapa será de amplificación, en la que haremos que la señal se acerque a una amplitud de 3.3 V pico-pico con la ayuda de un potenciómetro y la segunda etapa será de off-set, para introducir el '0 virtual' de 1.65 V. El circuito quedaría de la siguiente manera

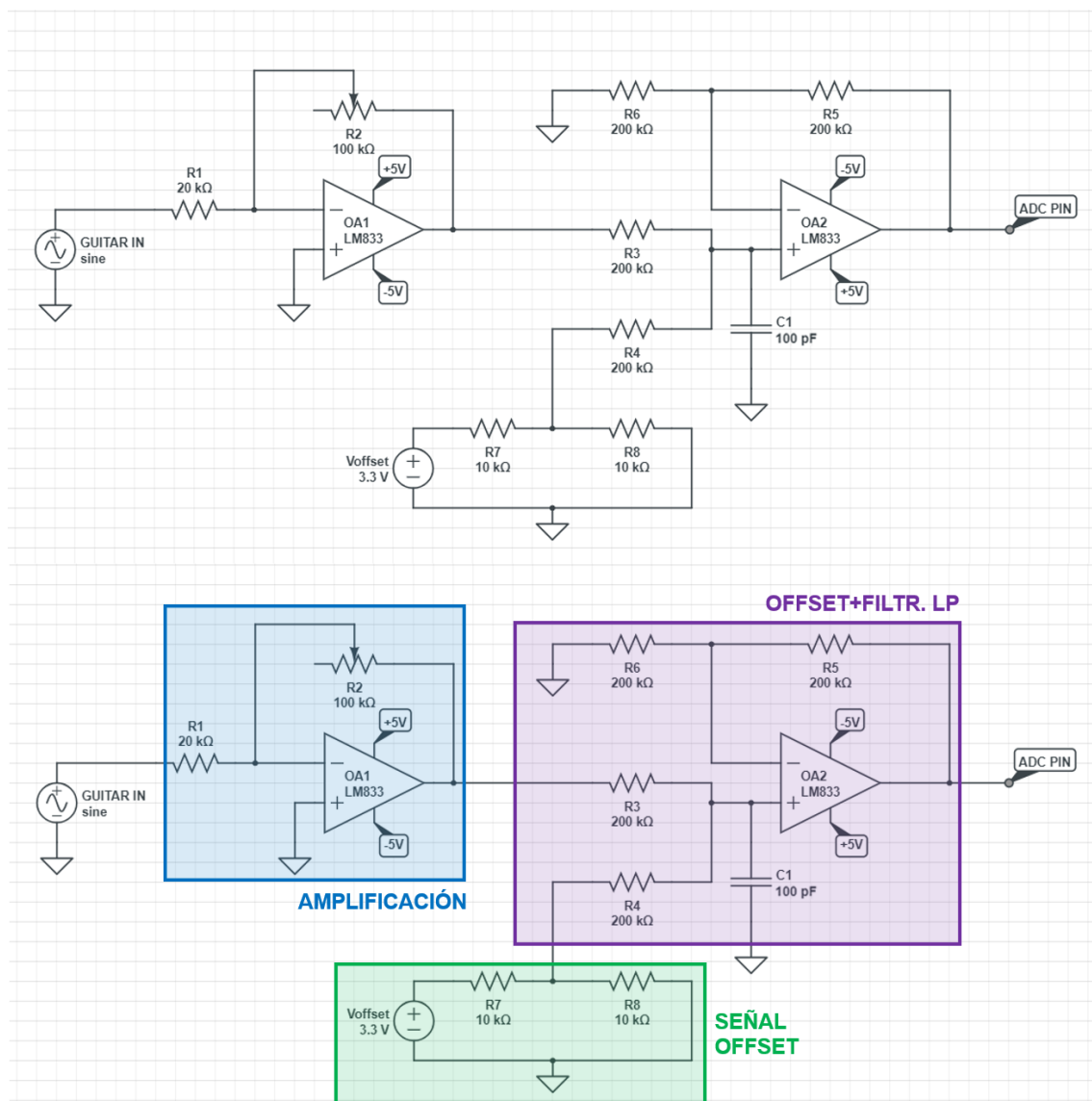


Figura 13: Circuito previo ADC

El circuito consta de varios elementos, primero que todo estará la entrada al circuito, que como ya dijimos será un Jack hembra de 6.35mm. La primera fase que no encontramos es la fase de amplificación, en la que como podemos ver en el esquema, hemos utilizado el amplificador operacional dual LM833D de STMicroelectronics, al igual que la placa de desarrollo. Este circuito integrado se caracteriza por ser diseñado para aplicaciones en audio gracias a su bajo nivel de ruido (4.5nV/VHz), muy baja distorsión (0.002%) y un ancho de banda de 15MHz. Para nuestra aplicación es perfecto puesto que necesitamos que la señal sea lo más limpia posible para que no haya errores en la lectura de la misma y el sonido sea lo más fiel posible al sonido que entregan las pastillas.

Entonces según las especificaciones del ADC tenemos que conseguir que nuestra señal se aproxime a los 3.3V pico-pico. Como nuestra señal no va a tener un voltaje fijo porque dependerá de la guitarra que usemos, la ganancia va a depender de un potenciómetro R3 de 100kΩ sobre la resistencia fija R2 de 20kΩ.

Siendo un montaje de amplificación con operacionales del tipo inversor, la señal se invertirá, pero no afecta en nada al sonido, ya que para el oído la onda se escucha igual. La ganancia de esta etapa vendrá dada por:

$$G_1 = -\frac{R_2}{R_1}$$

La ganancia entonces teniendo en cuenta el valor fijo de R2 y poniendo el potenciómetro en la posición mínima sería de G=0. Sin embargo, si ponemos el potenciómetro en su posición máxima la ganancia sería de G=-5. Esto podrá variar dependiendo de la impedancia de salida que tenga nuestra guitarra, ya que estará en serie con R1 y modificará la relación de amplificación. Esto no es un problema, ya que este ajuste no tiene que ser perfecto, es orientativo para aprovechar el máximo del ADC. Suponiendo impedancia de salida de la guitarra nula, entre las posiciones 20-0% del recorrido, el circuito atenuará la señal. Esto nos permitirá amplificar o atenuar la ganancia a la entrada del circuito según el tipo de pastilla que estemos utilizando en nuestra guitarra.

La siguiente etapa antes de la entrada de la señal en el ADC servirá para introducir el offset de 1.65 V (valor que vendrá dado por un divisor de tensión con factor ½ de una señal de 3.3V, tal y como se indica en la Figura 13 con R7 y R8 de 10kΩ), pero, además, también introduciremos un filtro paso bajo ('Low-Pass') pasivo para quitarnos de ruidos y frecuencias indeseadas que no aportan nada al sonido de la guitarra. Para el análisis de esta etapa, vamos a empezar por calcular el voltaje en la entrada positiva del operacional usando el análisis por superposición.

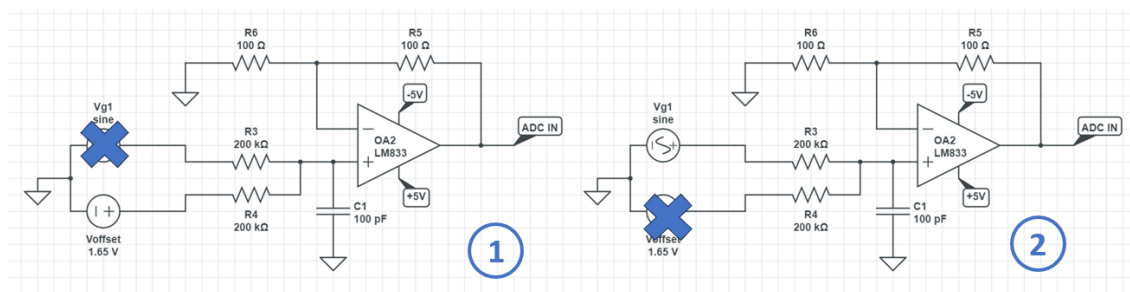


Figura 14: Teorema de superposición

En el caso 1, como aparece en la Figura 14, al eliminar la fuente de tensión que sería la entrada de la señal desde la anterior etapa y quedarnos con la fuente de offset la cual es corriente continua, el condensador no afectaría al cálculo y tendríamos un divisor de tensión entre R3 y R4.

$$V_{+1} = V_{offset} \cdot \frac{R_3}{R_3 + R_4}$$

Y si R3=R4=R nos queda que:

$$V_{+1} = \frac{V_{offset}}{2}$$

En el segundo caso en el que eliminamos la fuente del offset ya sí que afectarían las impedancias capacitivas, dejando otro divisor de tensión que vendrá dado por el paralelo:

$$V_{+2} = V_{in_2} \cdot \frac{R//C_1}{R + R//C_1} = V_{in_2} \cdot \frac{1}{RC_1s + 2}$$

Entonces si sumamos los dos valores de V+ tenemos que es:

$$V_{+} = V_{+1} + V_{+2} = \frac{V_{offset}}{2} + \frac{V_{in_2}}{RC_2s + 2}$$

Ahora por otro lado, vamos a relacionar la salida del amplificador, V_{ADC} , con la entrada positiva que acabamos de calcular. Como estamos en un montaje amplificador no inversor la ganancia será:

$$G_2 = \frac{V_{adc}}{V_{+}} = 1 + \frac{R_5}{R_6}$$

Que si tomamos R3=R4=R5=R6=R, nos queda que:

$$V_{adc} = 2 \cdot V_{+} = V_{offset} + \frac{2V_{in_2}}{RC_1s + 2}$$

Si analizamos la siguiente ecuación podemos observar que la salida al ADC será la tensión de offset (1.65V) sumada a la señal de la guitarra tras pasar la etapa anterior con una ganancia que depende de la frecuencia de la señal y de los valores de R y C1. Por tanto, podemos decir que la ganancia de la etapa amplificadora es:

$$G_2(s) = \frac{V_{adc} - V_{offset}}{V_{in_2}} = \frac{2}{RC_1s + 2}$$

Por tanto, veamos cómo se comporta en los extremos:

$$G_2(0) = \lim_{s \rightarrow 0} \frac{2}{RC_1 s + 2} = 1 = 0 \text{ dB}$$

$$G_2(\infty) = \lim_{s \rightarrow \infty} \frac{2}{RC_1 s + 2} = 0 = -\infty \text{ dB}$$

Como podemos comprobar cuando las frecuencias tienden a 0, la ganancia es de 0dB, mientras que cuando tienden a infinito la ganancia es de $-\infty$ dB. Esto nos indica que efectivamente existe un filtrado 'Low-Pas's en nuestro circuito. Para calcular los valores de R y de C1, se ha elegido 16kHz como frecuencia de corte ya que, a partir de dicha frecuencia, no hay mucho valor para el sonido. En nuestro circuito el filtro vendrá dado por las dos resistencias en paralelo a la entrada del amplificador con C1, por lo tanto, para la frecuencia de corte de 16kHz y siendo R=200kΩ:

$$F_{LP} = 16 \text{ kHz} = \frac{1}{2 \cdot \pi \cdot \frac{R}{2} \cdot C_1} \rightarrow C_1 = 99.4 \text{ pF}$$

Usando el valor más cercano posible, C1=100pF, la frecuencia de corte real sería 15.91kHz, valor igualmente válido para nuestra aplicación.

2.2.1.2. SIMULACIÓN DEL CIRCUITO PREVIO ADC

Una vez diseñado el circuito de entrada procedemos a simular con PSpice para comprobar que los cálculos son correctos y se obtiene la respuesta esperada.

En la Figura 15, podemos observar la evolución de la señal a lo largo de la etapa previa al ADC. En azul, tenemos la señal a la entrada de la etapa de amplificación, sería la señal desde la guitarra directamente. En verde tendríamos la señal tras pasar por la etapa de amplificación. Y en rojo la señal tras pasar la etapa en la que añadimos el 1.65V de offset.

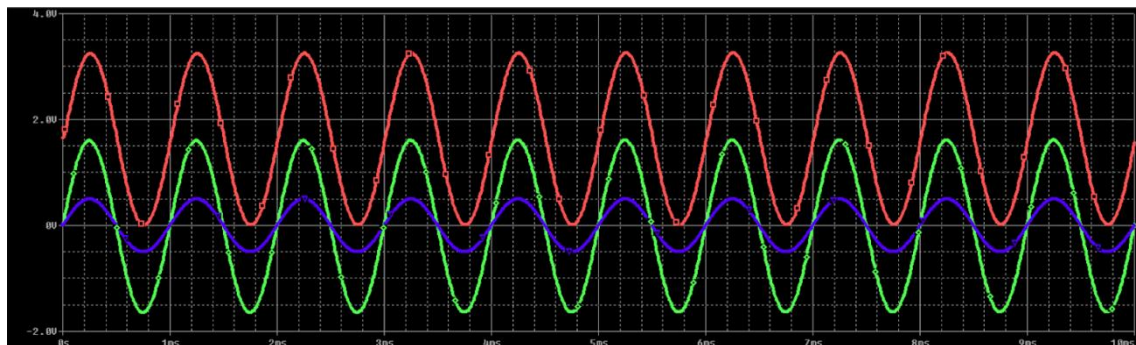


Figura 15: Simulación de la respuesta en el dominio del tiempo del circuito previo al ADC

Podemos ver claramente cómo se ajusta la señal muy cercana a ese rango de 0-3.3 V gracias a ese ‘0 virtual’ de 1.65V y a la ganancia ajustada por el potenciómetro de entrada, cumpliendo de nuevo con las especificaciones del diseño.

En cuanto a la respuesta en frecuencia del circuito, introdujimos un filtro ‘Low-Pass’ en la etapa de offset de 15.9kHz. En la Figura 16 podemos comprobar que estos valores se cumplen.

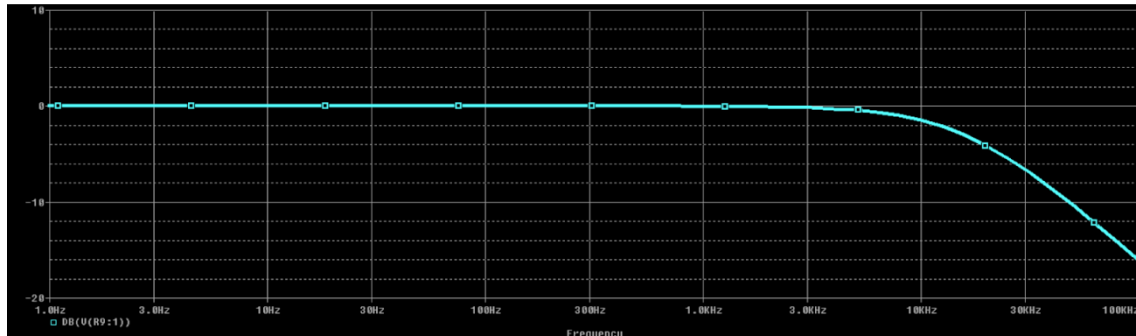


Figura 16: Simulación de la respuesta en frecuencia del circuito previo al ADC

Vemos como la frecuencia de corte se sitúa sobre los 15-16kHz, atenuando ruidos y frecuencias inservibles y cumpliendo con los cálculos anteriores.

2.2.2. CIRCUITO POSTERIOR DAC

El siguiente circuito que tenemos que diseñar nos servirá para ajustar la señal de vuelta a lo que sería una señal normal de guitarra (pero modificada con los distintos efectos). El datasheet del DAC del microcontrolador que está en la placa de desarrollo nos dice que el voltaje de salida es:

DAC_OUT _{min} ⁽²⁾	Lower DAC_OUT voltage with buffer ON	0.2	-	-	V	It gives the maximum output excursion of the DAC. It corresponds to 12-bit input code (0x0E0) to (0xF1C) at $V_{REF+} = 3.6\text{ V}$ and (0x1C7) to (0xE38) at $V_{REF+} = 1.8\text{ V}$
DAC_OUT _{max} ⁽²⁾	Higher DAC_OUT voltage with buffer ON	-	-	$V_{DDA} - 0.2$	V	
DAC_OUT _{min} ⁽²⁾	Lower DAC_OUT voltage with buffer OFF	-	0.5	-	mV	It gives the maximum output excursion of the DAC.
DAC_OUT _{max} ⁽²⁾	Higher DAC_OUT voltage with buffer OFF	-	-	$V_{REF+} - 1\text{LSB}$	V	

Figura 17: Voltajes de salida del DAC

Activemos o no el buffer del DAC, los parámetros son parecidos a los de entrada del ADC, donde tenía un rango de 0-3.3V. La señal de guitarra deberá tener valores positivos y negativos, por lo que tendremos que quitarle ese componente de continua. En cuanto al voltaje, se dispondrá de un potenciómetro con el que poder controlar el volumen de salida al amplificador.

2.2.2.1. DISEÑO DEL CIRCUITO POSTERIOR AL DAC

El circuito propuesto para ello es el siguiente.

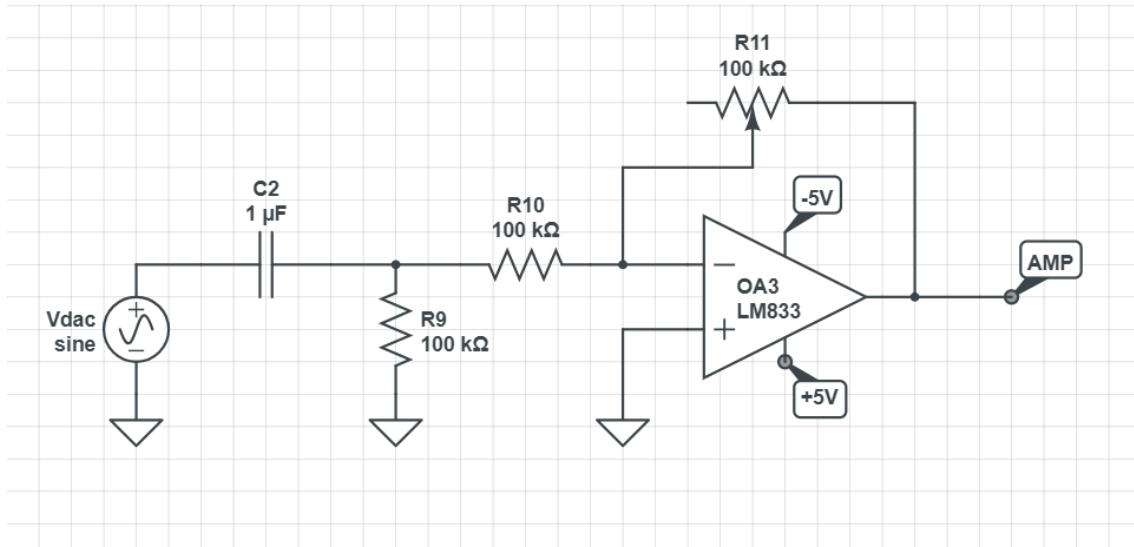


Figura 18: Circuito posterior DAC

Este circuito consiste simplemente en un filtro con el que quitaremos el componente de continua de la señal, es decir, el offset que introdujimos en el previo del ADC y una etapa de amplificación con montaje inversor, aunque deberíamos llamarla etapa de atenuación, puesto que R11 será un potenciómetro de 100kΩ con que podremos controlar el voltaje de salida al amplificador (volumen).

El filtro consiste en un 'High-Pass' que bloquee la señal continua procedente del DAC para que la señal vuelva a tener valores positivos y negativos, por lo que la frecuencia de corte será de unos 2Hz. Entonces los valores del condensador y la resistencia, suponiéndola de 100kΩ, deben de ser:

$$F_{HP_{DAC}} = \frac{1}{2 \cdot \pi \cdot C_2 \cdot R_9} \rightarrow C_2 = \frac{1}{2 \cdot \pi \cdot 2Hz \cdot 100k\Omega} = 0,8 \mu F$$

Usando un condensador de 1μF, la frecuencia de corte sería de 1.59Hz, que para bloquear la componente de continua es suficiente.

En la etapa de amplificación es simplemente un montaje inversor con un potenciómetro para controlar la ganancia. Que la señal se invierta en audio no influye nada, el sonido que percibirán nuestros oídos será el mismo, al igual que ocurría en el circuito de entrada al ADC:

$$G_3 = -\frac{R_{11}}{R_{10}}$$

Con $R_{10} = 100\text{k}\Omega$, si el potenciómetro R_{11} es de $100\text{k}\Omega$ en su posición máxima, en este estado funcionará como un buffer, mientras que si lo giramos irá disminuyendo el volumen hasta apagarlo.

2.2.2.2. SIMULACIÓN DEL CIRCUITO POSTERIOR DAC

En la Figura 19 vemos la simulación de este circuito en el dominio del tiempo, donde la línea roja representa la señal a la salida del DAC, la línea verde representa la señal tras pasar por el filtro y la línea azul es la señal que va hacia la salida de la pedalera.

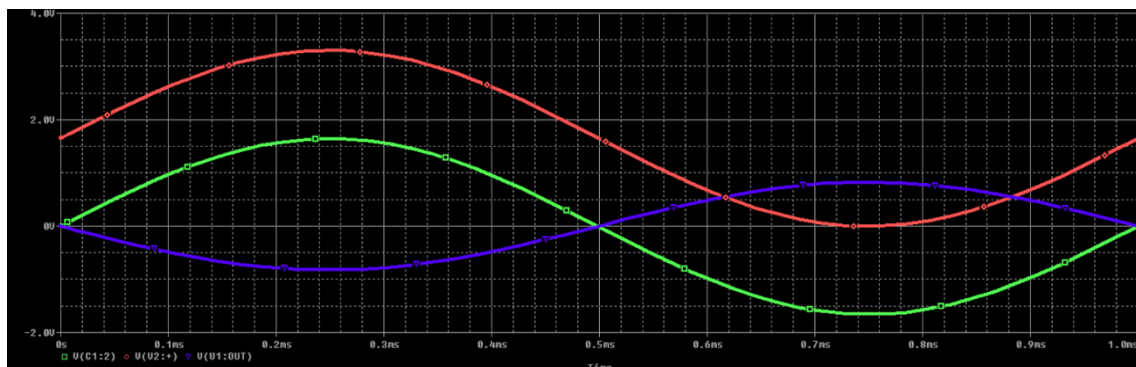


Figura 19: Simulación de la respuesta en el dominio del tiempo del circuito posterior al DAC

Como podemos observar, la señal que sale del DAC tiene un rango aproximado de 0-3.3 V, y tras el filtro, “bajamos” la señal al 0 y pasa a tener un rango de $\pm 1.65\text{V}$, ya si es una señal como la original de la guitarra. Luego en la etapa de amplificación hemos simulado como si el potenciómetro estuviera en la posición media, por lo que estaría atenuando la señal por la mitad y pasaría a tener un rango de $\pm 825\text{mV}$. En la Figura 20 tenemos el diagrama de Bode de la etapa de salida, donde podemos ver claramente que el filtro ‘High-Pass’ cumple con su función.

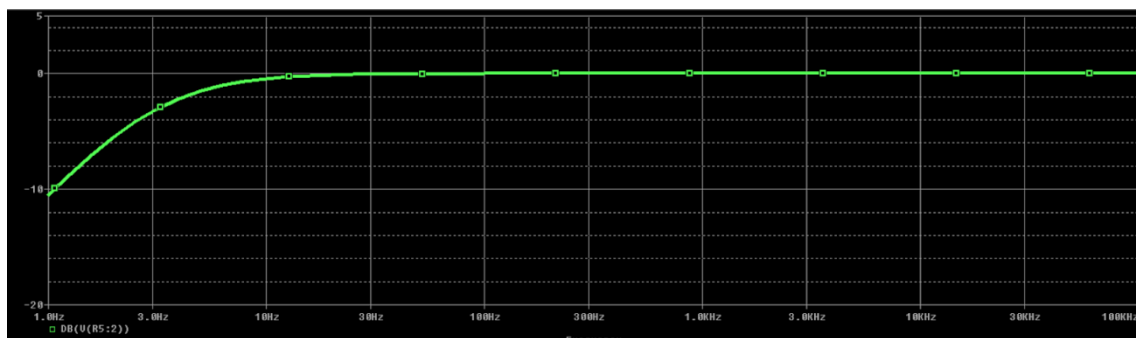


Figura 20: Simulación de la respuesta en frecuencia del circuito posterior al DAC

2.2.3. IMPRESIÓN DE LA PCB

Todos los componentes irán soldados a una PCB para evitar todo lo posible el uso de cables que puedan inducir a ruidos e interferencias en el sonido. Para el diseño se ha usado el programa de diseño de diagramas y PCBs “EAGLE” de AutoDesk. El diseño consiste en una PCB de 135x145mm, la entrada Jack de 6.35mm estará situada en la parte superior izquierda y la salida en la parte superior derecha, al lado de cada una estará su respectivo potenciómetro de control de ganancia a la entrada y de volumen a la salida. Entre los potenciómetros estará situado el módulo inversor de voltaje “CJMCU-2662 LM2662” que será el encargado de convertir 5V en -5V para poder alimentar los amplificadores operacionales. Todos los componentes eléctricos estarán situados en la capa superior de la PCB, sin embargo, los agujeros para los pines de la placa de desarrollo están pensados para que esta se conecte por debajo de la PCB, para que la pantalla quede en la parte superior, se ha creado un espacio por el que a través de unos extensores la pantalla LCD pueda acceder a los pines de la placa de desarrollo. Además, se han introducido agujeros para poder atornillar los soportes para la pantalla LCD y para poder atornillar la PCB a la carcasa. En la parte inferior se colocarán 3 conectores de terminales para conectar los cables de conexión de los ‘footswitch,’ estos son los únicos componentes que llevan cables obligatoriamente, pero al no influir en el sonido no generan ningún problema. Por último, un LED rojo al lado de la entrada Jack será el encargado de avisar al usuario cuando la señal está saturando los límites del ADC o del DAC para que modifique la ganancia a la entrada del circuito. En la Figura 21 podemos ver el diseño de EAGLE.

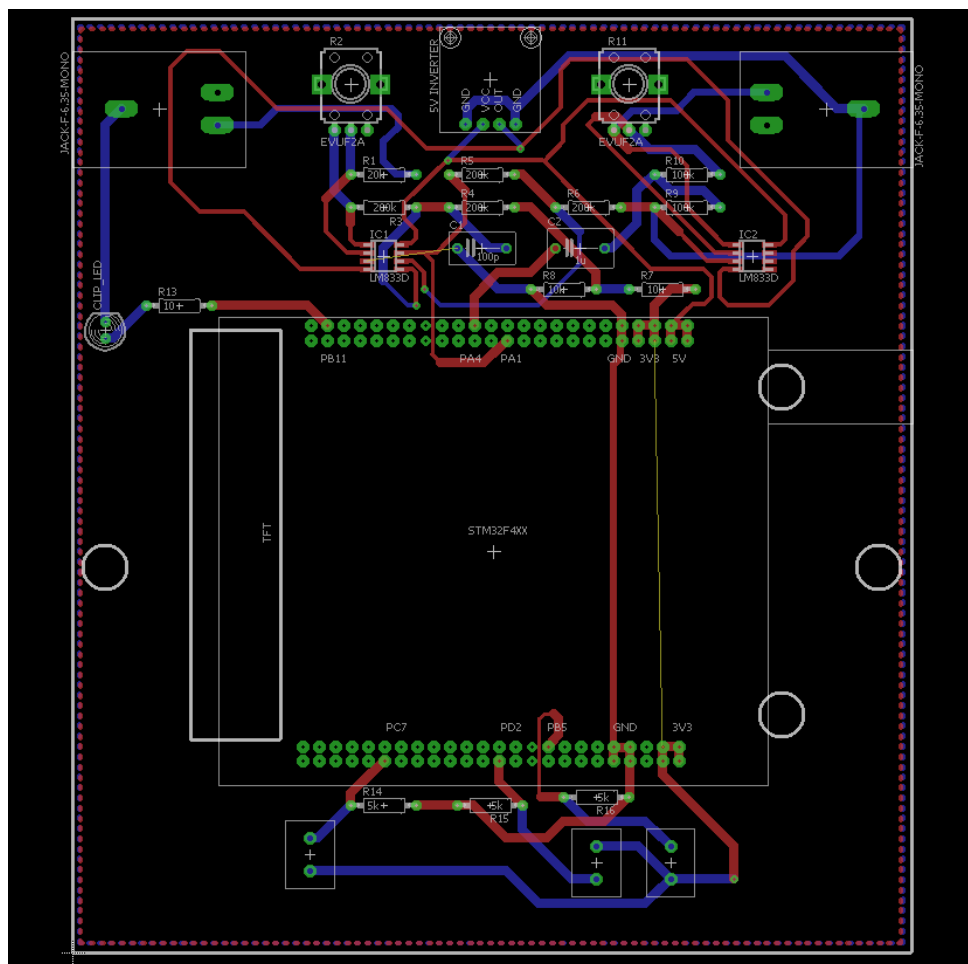


Figura 21: Diseño de la PCB en EAGLE

La línea de puntos representa las capas GND para unificar las tierras de todas las conexiones y facilitar también la conexión de estas. El diseño de las capas superior e inferior entonces quedarían de la siguiente manera

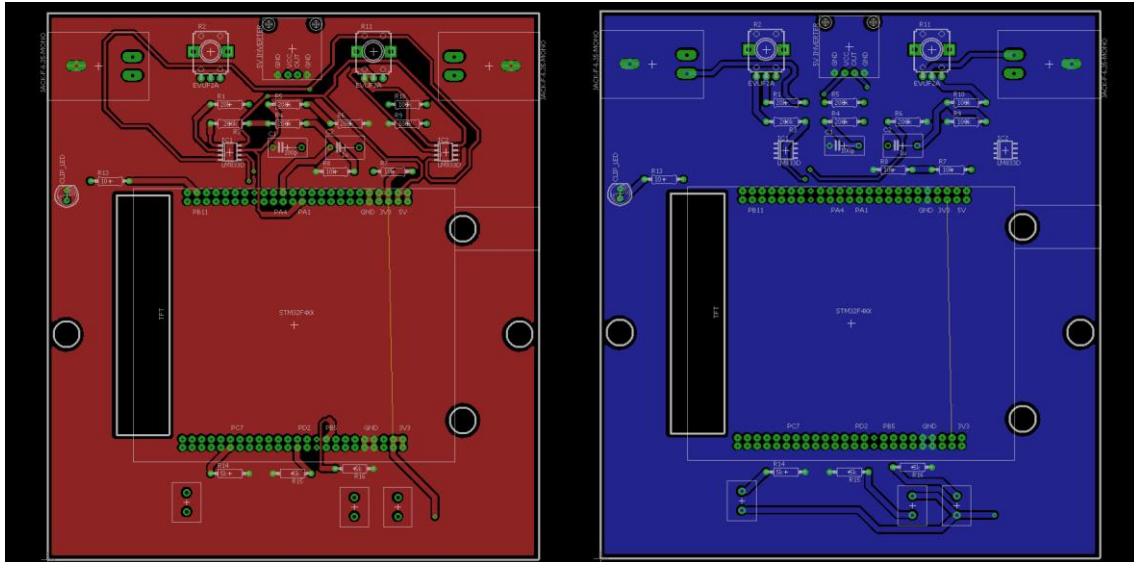


Figura 22: Diseño de la impresión de la PCB con capas GND

Para la impresión de la PCB se ha utilizado la máquina de trazados de PCB “LPKF Protomat S103” ubicada en el laboratorio D13 de la Escuela Superior de Ingeniería de la Universidad de Cádiz. La máquina usa el fresado para ir delimitando las pistas y pads de los componentes sobre una placa revestida con cobre laminado. Tras la finalización del proceso, se le ha aplicado laca protectora aislante “AISLARCO/1” para proteger la placa de humedades, ácidos y álcalis, así como de la corrosión. A continuación, se procede a la soldadura de todos los componentes (Aerosol Laca Protectora Aislante A/1, n.d.).

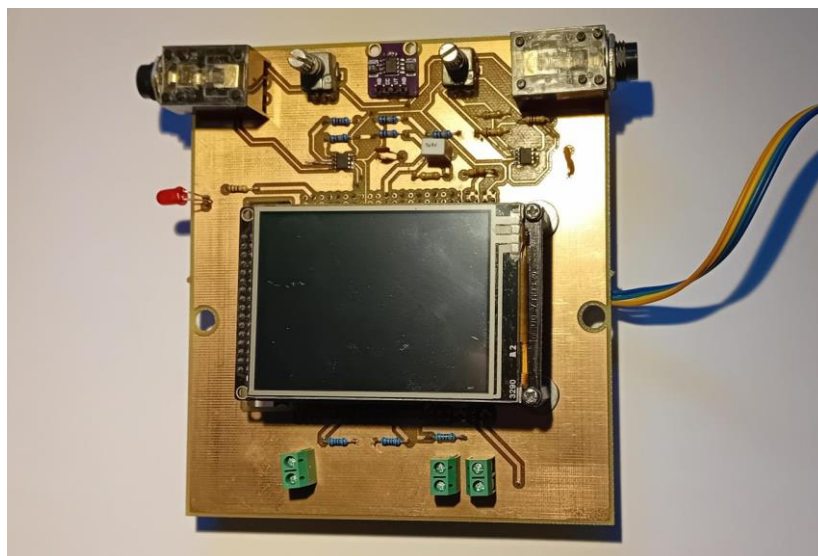


Figura 23: PCB con los componentes soldados

2.3. DISEÑO DEL SOFTWARE

Para la programación de la pedalera, entre los IDE de programación disponibles como por ejemplo “Keil uVision” o el IDE de Arduino, se optó por utilizar el “STM32Cube IDE”, un software de programación desarrollado por la empresa STM, la misma que desarrolla el microcontrolador utilizado. El lenguaje de programación será C, y la ventaja que tiene este IDE frente a los demás y fue el motivo principal por el que se optó por él, es porque tiene la capacidad de autogenerar las líneas de código necesarias para la configuración de todos los periféricos y otros aspectos del microcontrolador, lo cual nos ha ahorrado muchísimo tiempo en escribir líneas de código, así como el aprendizaje de su configuración, ya que su uso es muy intuitivo. En la Figura 24 se puede observar cómo es el entorno de configuración mencionado.

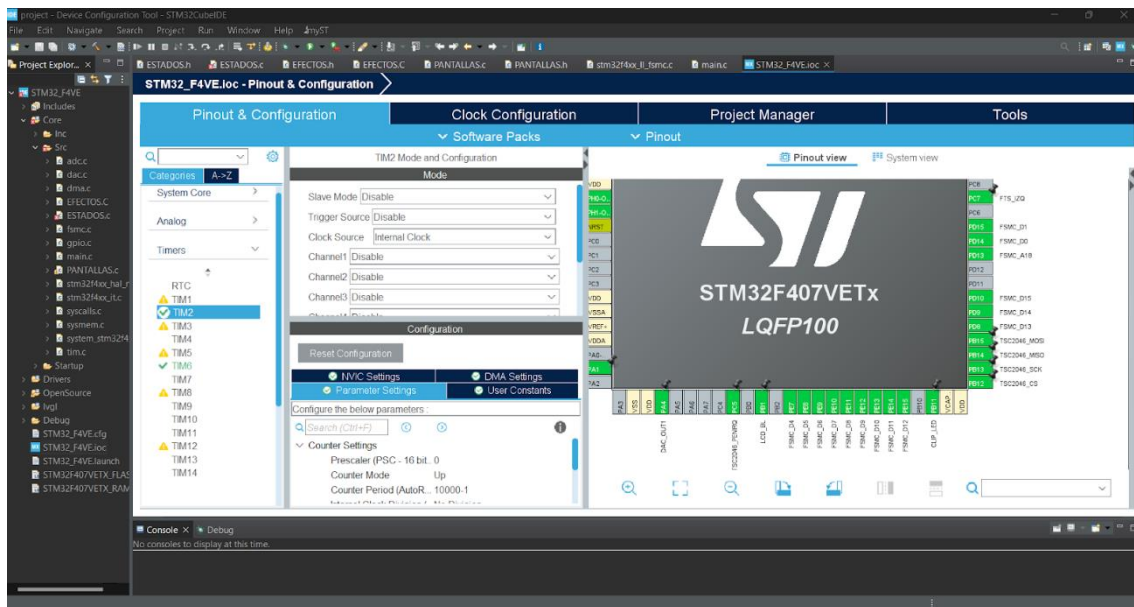


Figura 24: STM32Cube IDE

El objetivo de esta etapa de programación es programar 3 presets que serán configurables por el usuario usando la pantalla táctil. Habrá un modo para tocar, en el que los presets serán intercambiables usando los ‘footswitch’, y otro modo de configuración de los efectos, donde el usuario podrá modificar variables que afectan al comportamiento de los efectos, así como también qué efectos se aplican a la señal de 4 posibles, los cuáles serán: ‘Overdrive’, ‘Delay’, ‘Tremolo’ y ‘Chorus’. El esquema de utilización de la pedalera por el usuario sería:

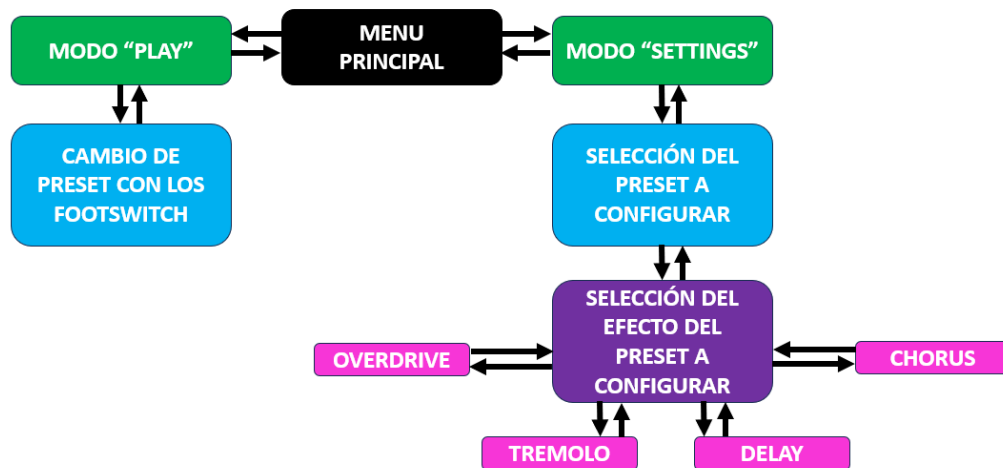


Figura 25: Esquema de navegación del usuario

Para cambiar entre estas etapas se hará mediante el uso del táctil de la pantalla LCD.

2.3.1. DMA (DIRECT MEMORY ACCESS)

El STM32 tiene un controlador llamado DMA (*'Direct Memory Access'*), el cual significa *'Acceso Directo a Memoria'*, y tal como nos explican en el manual del microcontrolador, es un bus para la transferencia de información a alta velocidad entre memoria y periféricos. Los datos pueden ser transferidos por el DMA sin que la CPU tenga que intervenir en ningún momento, dejándolo libre para otras operaciones. Este recurso será fundamental para el proyecto, ya que necesitaremos de la CPU para que lea el táctil en caso de que queramos salir al menú principal además de cambiar de preset cuando pulsemos los *'footswitch'*.

El funcionamiento para este caso consiste en 2 arrays, *"adc_buffer[]"* y *"dac_buffer[]"*, en los que el ADC irá almacenando las muestras tomadas en el primero y el DAC irá tomando las muestras del segundo para convertirlas en tensión. Para pasar de un buffer a otro se usarán las funciones *void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc)* y *void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)*.

En estas funciones se modificarán las muestras recogidas por el ADC según las funciones creadas para los efectos. Algunos efectos se aplicarán a la muestra y otros no, eso dependerá de la configuración del preset seleccionado, y una vez modificada, la almacenaremos en el buffer del DAC. Estas funciones *'Callback'* se ejecutarán cuando el ADC haya rellenado la primera mitad del array y la segunda función cuando se rellene la segunda mitad del array.

Cuando se escriba la primera mitad del array del ADC, se ejecutará la función *void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc)*, estos datos se modificarán según los efectos configurados y pasarán a la primera mitad del array del DAC y cuando se rellene la segunda mitad del array del ADC, se ejecutará la función *void*

HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef hadc)*, los datos se modificarán de la misma manera y pasarán a la segunda mitad del array.

Mientras el ADC vuelve a rellenar la primera mitad del array, el DAC estará leyendo los datos de la primera mitad de su array, es decir, los datos que se pasaron de la anterior vez que se ejecutó el *void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc)* . Ídem cuando empiece a rellenar la segunda mitad. En la Figura 26 se muestra gráficamente.

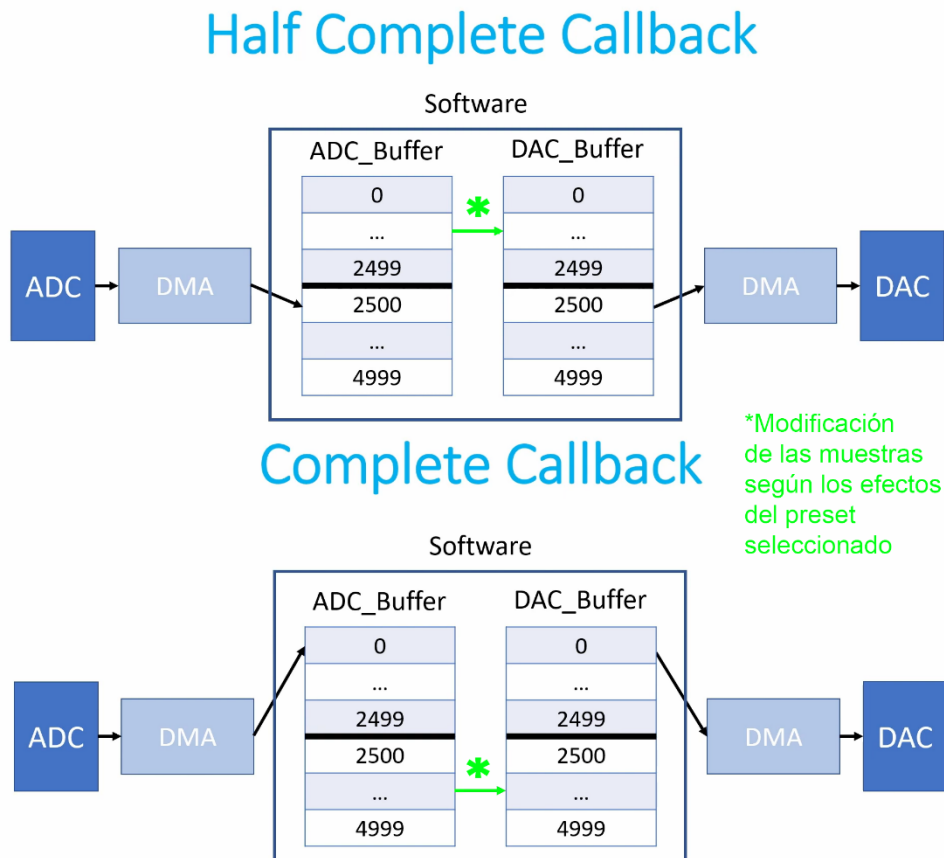


Figura 26:Diagrama de funcionamiento Half Complete y Complete Callback

Este método nos asegurará que no se sobrescriban los datos y que la frecuencia de muestreo se mantiene. Pero hay que tener cuidado con el tamaño del array, ya que esto genera un retardo en la señal y queremos que cuando toquemos una nota la sensación es que suene a la vez que tocamos, no con retardo. Por ello hemos elegido un tamaño del array de 512 muestras (estándar en muchas interfaces de audio), lo que nos genera un retardo de 10ms a los 48kHz de muestreo que funciona nuestro código. 10ms es una latencia aceptable ya que los efectos del retardo se suelen empezar a percibir sobre los 15-20ms.

2.3.2. CONFIGURACIÓN DE PERIFÉRICOS

2.3.2.1. ADC, DAC y TIM2

El ADC y el DAC son los componentes fundamentales del proyecto, ya que son los encargados de recoger el sonido de la guitarra y luego generar el sonido con las muestras modificadas. Para poder tomar muestras cada 48kHz, se ha configurado un timer que marcará cuando se deben hacer las conversiones de ADC y DAC, en este caso se ha elegido el TIMER2. Para la configuración del timer, tenemos que tener en cuenta la siguiente expresión:

$$SampleRate = \frac{F_{CLK}}{(PSC + 1) \cdot (ARR + 1)}$$

Figura 27: Ecuación para la configuración del timer

En ella tenemos 3 valores. “F_{clk}” hace referencia a la velocidad del reloj principal, en nuestro caso serían los 168 MHz, PSC sería el contador de preescala y ARR sería el ‘Auto-Reload Register’, que es el período de conteo.

Como nuestro reloj interno funciona a 168MHz, para que el timer vaya a 48kHz se ha configurado con un periodo de conteo de 3500 poniendo el contador de preescala a 0. Entonces 168MHz/3500=48kHz.

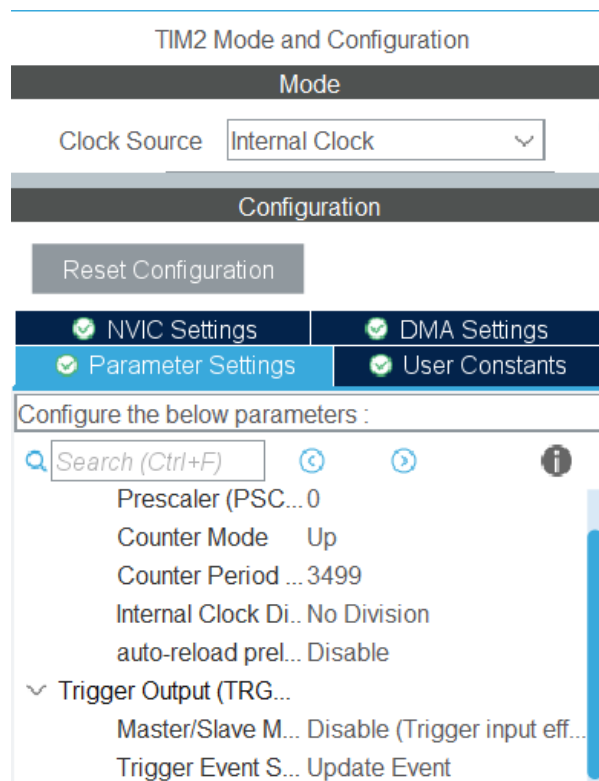


Figura 28: configuración de TIM2

También debemos de activar el '*Trigger Event Selection*' como '*Update Event*' para que el DMA pueda tomarlo de referencia para las conversiones del ADC y del DAC.

El ADC tendremos que configurarlo para que haga conversiones de 12bits, '*Right-Alignment*' y activaremos los requerimientos continuos del DMA. Además, pondremos como trigger externo el Timer 2, que previamente configuramos a la frecuencia de muestreo de 48 kHz. Elegimos el ADC1 en la entrada IN1, que corresponde al pin A1 de la placa.

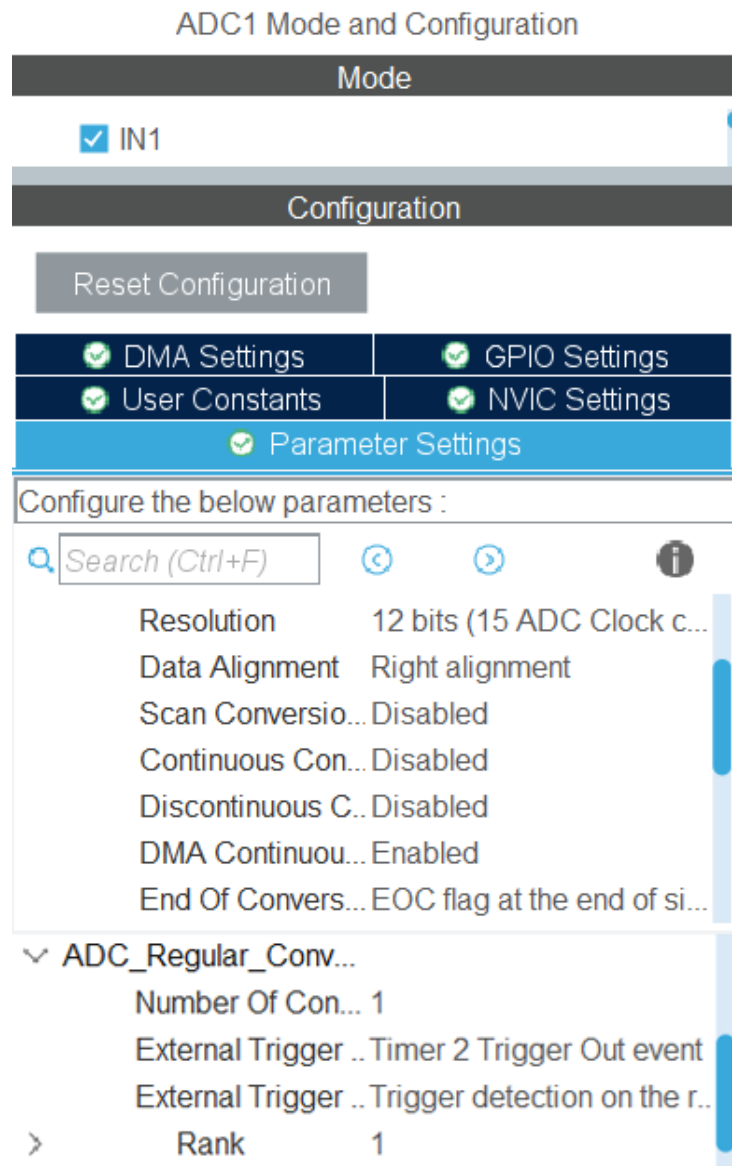


Figura 29: Configuración del ADC1

Habrá también que configurar el DMA para el ADC en la pestaña de "DMA Settings", en la que añadiremos un requerimiento de DMA, lo configuraremos en modo Circular para que cuando termine una conversión empiece con otra automáticamente. Además, configuraremos el tamaño de palabra en Word.

Parameter Settings		User Constants		NVIC Settings		DMA Settings		GPIO Settings	
DMA Request		Stream		Direction		Priority			
ADC1		DMA2 Stream 0		Peripheral To Memory		Low			
Add		Delete							

DMA Request Settings

Peripheral				Memory	
Mode	Circular	Increment Address	☐		
Use Fifo	☐	Threshold		Data Width	Word
				Burst Size	

Figura 30: Configuración del DMA para el ADC1

La configuración del DAC es bastante parecida, tendremos que activarlo y activar el buffer de salida para que no existan efectos de impedancia a la salida. Al igual que con el ADC, configuraremos como trigger el Timer 2. La resolución del DAC es de 12 bits también por defecto, no da lugar a cambio. Se ha elegido el DAC1, cuyo pin corresponde al PA4.

DAC Mode and Configuration

Mode

☒ OUT1 Configuration

☐ OUT2 Configuration

Configuration

Reset Configuration

DMA Settings

User Constants

GPIO Settings

NVIC Settings

Parameter Settings

Configure the below parameters :

Search (Ctrl+F)

↺

↻

ⓘ

▼ DAC Out1 Settings

Output Buffer

Enable

Trigger

Timer 2 Trigger Out event

Wave generation..

Disabled

Figura 31: Configuración del DAC1

36

De la misma forma que con el ADC, activaremos el requerimiento del DMA.

The screenshot shows the 'DMA Settings' tab in the STM32CubeMX configuration tool. At the top, there are tabs for 'Parameter Settings', 'User Constants', 'NVIC Settings', 'DMA Settings' (which is selected), and 'GPIO Settings'. Below these, a header bar displays 'DAC1', 'DMA1 Stream 5', 'Memory To Peripheral', and 'Low'. There are 'Add' and 'Delete' buttons. The main section is titled 'DMA Request Settings' and contains a table with two columns: 'Peripheral' and 'Memory'. The 'Peripheral' column has a 'Mode' dropdown set to 'Circular', an 'Increment Address' checkbox (unchecked), and a 'Use Fifo' checkbox (unchecked). The 'Memory' column has a checkbox (checked) and a 'Threshold' dropdown. Below the table, there are 'Data Width' and 'Burst Size' dropdowns, both set to 'Word'.

Figura 32: configuración del DMA para el ADC1

2.3.2.2. GPIO INPUTS

Para los *'footswitch'*, necesitaremos configurar 3 entradas GPIO que generen una interrupción para cambiar de preset. Los *'footswitch'* estarán conectados a los pines PC7, PD2 y PB5. Se usarán con la resistencia *'Pull-Down'* interna que viene con la placa de desarrollo y se activarán cuando se detecte un flanco de subida. La Figura 33 muestra el ejemplo de configuración.

The screenshot shows the 'GPIO Configuration' for pin PB5. It has four rows: 'GPIO mode' set to 'External Interrupt Mode with Rising edge t', 'GPIO Pull-up/Pull-down' set to 'Pull-down', and 'User Label' set to 'FTS_DER'. The first row is highlighted with a blue bar.

Figura 33: Configuración del GPIO del pin PB5

También debemos de activar las interrupciones externas para estos pines, las cuales son EXTI2, EXTI5 y EXTI7.

Configuration			
Group By Peripherals			
☒ FSMC	☒ RCC	☒ SYS	☒ NVIC
☒ GPIO	☒ ADC	☒ DAC	
NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
EXTI line2 interrupt	☒	0	0
EXTI line[9:5] interrupts	☒	0	0

Figura 34: Activación de las interrupciones externas

De la misma manera que con el timer 2, tendremos que copiar la función

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);
```

la cual está situada en el archivo "stm32f4xx_hal_gpio.c" ya que también es de tipo "__weak". En esta función daremos las instrucciones necesarias para el cambio del preset que se está ejecutando.

2.3.2.3. PANTALLA LCD

Para la pantalla LCD, usaremos la librería para la ILI9341 de "pic18fx.blogspot.com" (R, n.d.), la cual nos permite mediante funciones sencillas dibujar, líneas, rectángulos, circunferencias y texto en pantalla, así como la lectura de la pantalla táctil. Para su correcto funcionamiento, debemos de configurar el RCC, el SYS, el FMSC y algunos pines GPIO.

FSMC Mode and Configuration		RCC Mode and Configuration																																																									
Mode NOR Flash/PSRAM/SRAM/ROM/LCD 1 ☒ NE1 Chip Select Memory type LCD Interface LCD Register Select A18 Data 16 bits		Mode High Speed Clock (HSE) Crystal/Ceramic Resonator Low Speed Clock (LSE) Disable ☐ Master Clock Output 1 ☐ Master Clock Output 2 ☐ Audio Clock Input (I2S_CKIN)																																																									
Configuration Reset Configuration ☒ NOR/PSRAM 1 ☒ User Constants ☒ GPIO Settings		SYS Mode and Configuration Mode Debug Serial Wire ☒ System Wake-Up Timebase Source SysTick																																																									
Configure the below parameters : Search (Ctrl+F) Memory type LCD Interface Bank Bank 1 NOR/PSRAM 1 Write operation Enabled Extended mode Disabled NOR/PSRAM timing Address setup time in H. 6 Data setup time in HCL. 6 Bus turn around time in ... 0		<table border="1"> <thead> <tr> <th>Pin N...</th> <th>User Label</th> <th>Signal</th> <th>GPIO ...</th> <th>GPIO ...</th> <th>GPI...</th> <th>Ma...</th> <th>M...</th> </tr> </thead> <tbody> <tr> <td>PB13</td> <td>TSC2046_SCK</td> <td>n/a</td> <td>Low</td> <td>Output...</td> <td>No ...</td> <td>Low</td> <td>☒</td> </tr> <tr> <td>PC5</td> <td>TSC2046_PENI...</td> <td>n/a</td> <td>n/a</td> <td>Input ...</td> <td>No ...</td> <td>n/a</td> <td>☒</td> </tr> <tr> <td>PB15</td> <td>TSC2046_MOSI</td> <td>n/a</td> <td>Low</td> <td>Output...</td> <td>No ...</td> <td>Low</td> <td>☒</td> </tr> <tr> <td>PB14</td> <td>TSC2046_MISO</td> <td>n/a</td> <td>n/a</td> <td>Input ...</td> <td>No ...</td> <td>n/a</td> <td>☒</td> </tr> <tr> <td>PB12</td> <td>TSC2046_CS</td> <td>n/a</td> <td>High</td> <td>Output...</td> <td>No ...</td> <td>Low</td> <td>☒</td> </tr> <tr> <td>PB1</td> <td>LCD BL</td> <td>n/a</td> <td>Low</td> <td>Output...</td> <td>No ...</td> <td>Low</td> <td>☒</td> </tr> </tbody> </table>		Pin N...	User Label	Signal	GPIO ...	GPIO ...	GPI...	Ma...	M...	PB13	TSC2046_SCK	n/a	Low	Output...	No ...	Low	☒	PC5	TSC2046_PENI...	n/a	n/a	Input ...	No ...	n/a	☒	PB15	TSC2046_MOSI	n/a	Low	Output...	No ...	Low	☒	PB14	TSC2046_MISO	n/a	n/a	Input ...	No ...	n/a	☒	PB12	TSC2046_CS	n/a	High	Output...	No ...	Low	☒	PB1	LCD BL	n/a	Low	Output...	No ...	Low	☒
Pin N...	User Label	Signal	GPIO ...	GPIO ...	GPI...	Ma...	M...																																																				
PB13	TSC2046_SCK	n/a	Low	Output...	No ...	Low	☒																																																				
PC5	TSC2046_PENI...	n/a	n/a	Input ...	No ...	n/a	☒																																																				
PB15	TSC2046_MOSI	n/a	Low	Output...	No ...	Low	☒																																																				
PB14	TSC2046_MISO	n/a	n/a	Input ...	No ...	n/a	☒																																																				
PB12	TSC2046_CS	n/a	High	Output...	No ...	Low	☒																																																				
PB1	LCD BL	n/a	Low	Output...	No ...	Low	☒																																																				

Figura 35: Configuración de los periféricos necesarios para la pantalla LCD

2.3.3. EFECTOS

2.3.3.1. OVERDRIVE

El efecto de 'overdrive' es un efecto que surgió con el fin de imitar el ruido generado al saturar las válvulas de los amplificadores de guitarra antiguos, lo que resulta en un tono saturado, comprimido y agresivo, por ello es ideal para géneros como el rock, el punk o el heavy metal.

El funcionamiento de este efecto consiste en fijar el umbral superior e inferior de la señal de salida, modificando el margen dinámico de salida. Este umbral se podrá modificar, cambiando así el nivel de distorsión de la señal.

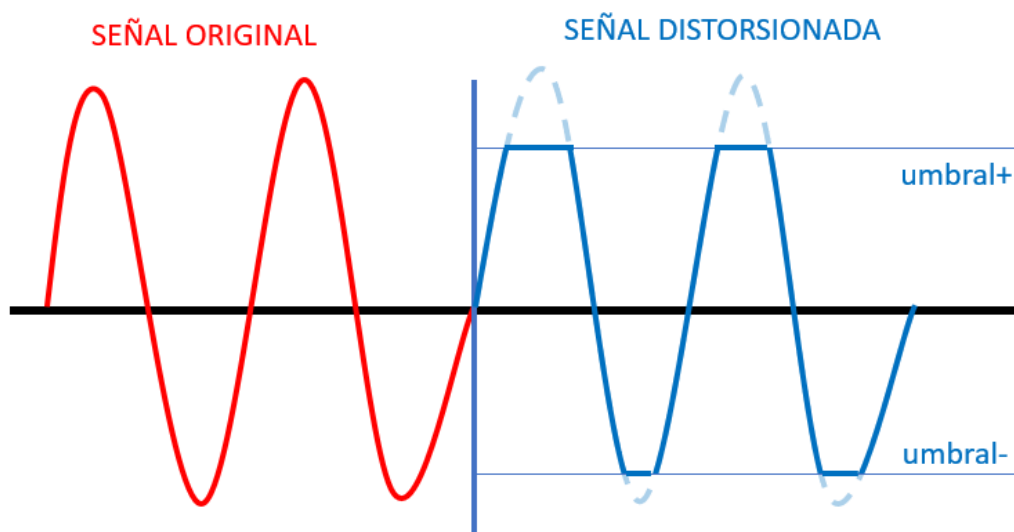


Figura 36: Ejemplo de señal de Overdrive

El código implementado sería el siguiente:

```
void overdrive (uint32_t sample[], uint8_t threshold, uint8_t gain, int n){
    uint32_t uptresh;
    uint32_t downtresh;
    uptresh=4000-(150*(threshold-1));
    downtresh=4095-uptresh;
    sample[n]=((float)sample[n]-2048)*(1+gain/2)+2048;
    if(sample[n]>=uptresh)sample[n]=uptresh;
    if(sample[n]<=downtresh)sample[n]=downtresh;
}
```

Básicamente consta de una función a la que se le introduce la señal muestreada mediante 'sample []' y 'n', un valor de umbral ('*threshol*d') entre 1 y 10 y una ganancia '*gain*' también entre 1 y 10. El '*threshol*d' indicará, en un rango de 4000 a 2200, el umbral al que, si la muestra introducida supera, forzaremos a que la salida sea el valor de dicho umbral, por otro lado, el valor de '*gain*' amplificará la señal hasta un máximo de ganancia de 6. Jugando con estos valores podremos conseguir diferentes tonos en el efecto.

A la muestra se le resta el valor de offset introducido a la entrada, así la ganancia se aplica tanto en los valores negativos como positivos. Una vez aplicada la ganancia, volvemos a añadir el offset.

2.3.3.2. DELAY

La palabra '*delay*' significa retardo, así que este efecto no consiste más que en aplicar un retraso a la señal y sumarlo a la muestra actual que se va a reproducir. Se usa en guitarras sin distorsión y hacen que el sonido sea más completo y continuo.

Para su funcionamiento, se irán almacenando muestras en un vector de hasta 24000 muestras, lo que nos generará un retardo de máximo 500ms, controlado por una variable que indicará cuántas muestras se tienen que almacenar. La Figura 37 muestra el funcionamiento del efecto.

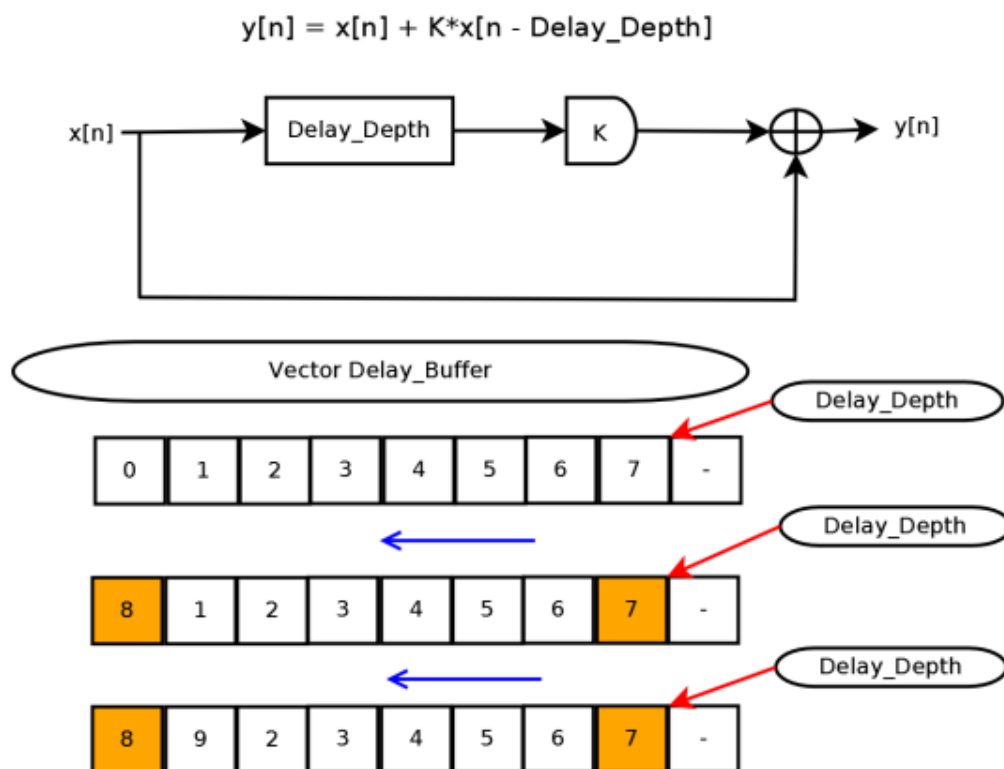


Figura 37: Esquema y funcionamiento del efecto de delay (Ramirez, 2017)

La muestra actual a reproducir se almacenará en el vector '*delay_buffer*' y se le sumará la muestra que almacenó el número de muestras anteriores que indica '*delay_depth*'. Finalmente, el parámetro

'*delay_wet*' multiplicará a la muestra de retardo con un factor dado para controlar el volumen de la mezcla. El código utilizado sería el siguiente:

```
void delay(uint32_t sample[], uint8_t delay_depth, uint8_t delay_wet, int n){
    uint16_t depth;
    float wet_factor;

    wet_factor=2-delay_wet/5;
    depth=2400*delay_depth;

    delay_buffer[delay_counter]=sample[n];
    delay_counter++;
    if(delay_counter==depth){delay_counter=0;}
    sample[n]=(float)sample[n]+((float)delay_buffer[delay_counter]-
2048)/wet_factor;

}
```

El usuario configurará el '*delay_depth*' con números del 1 al 10, que almacenará de 2400 a 24000 muestras. Si nuestra frecuencia de muestreo es de 48kHz, esto quiere decir que el retardo irá desde 50ms a 500ms. Además, podrá configurar el parámetro '*delay_wet*' para modificar el volumen al que suena el sonido retardado.

El valor de 2048 que aparece restando surge de restar el offset introducido en la etapa de entrada para que valores positivos sean restados por los negativos y viceversa. Sin este ajuste, la señal a la salida saturaría todo el tiempo ya que siempre estaríamos sumando valores positivos entre ellos. Como luego tenemos que devolverle al resultado ese offset para que pueda ser leído por el DAC, se añadiría sumando al otro lado de la igualdad y se cancelaría con el valor restado de la muestra actual, por ello sólo se aplica a la muestra retardada.

2.3.3.3. TREMOLO

La definición de la palabra trémolo según Wikipedia, “es un término musical que describe la fluctuación o variación periódica en la intensidad de un sonido, mientras que la altura o frecuencia se mantiene constante” (2023). Dicho de otro modo, el trémolo consiste en variar la amplitud de la onda a un ritmo determinado, sin cambiar la frecuencia del sonido

Este término es comúnmente confundido con el vibrato, el cual consiste en modificar la frecuencia de la onda, no la amplitud, como por ejemplo se confunde con las palancas de las guitarras eléctricas que cambian la longitud de la cuerda, cambiando por tanto la longitud de la onda no la amplitud.

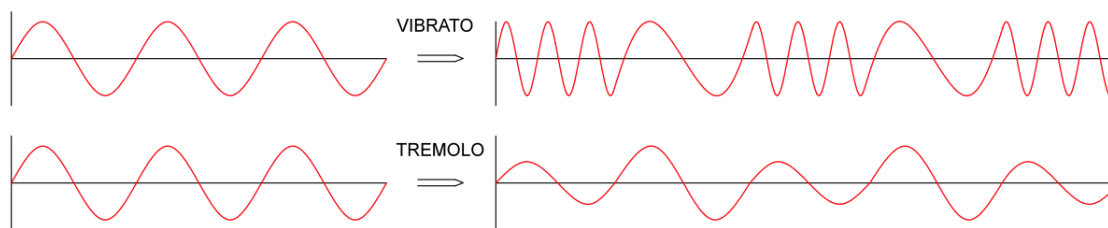


Figura 38: Diferencia entre Vibrato y Tremolo (Espinosa, 2014)

Existen varias formas de producir el efecto de trémolo, algunos modifican la señal según una onda sinusoidal, otros según una onda triangular, de sierra, etc. En nuestro caso, la amplitud de la onda vendrá dada por una onda sinusoidal de frecuencia y profundidad variables por el usuario.

```
void tremolo(uint32_t sample[],uint8_t trem_depth, uint8_t rate, int n){  
    float a;  
    float lfo;  
    a=(float)trem_depth/20;  
    float offset=1-a;  
  
    lfo=a*sin((2*PI*rate)*trem_counter/4800)+offset;  
    trem_counter++;  
    if (trem_counter==4800){trem_counter=0;}  
  
    sample[n]=((float)sample[n]-2048)*lfo+2048;  
}
```

La variable más importante en el código es '*lfo*'. Esta variable generará los valores que determinarán la amplitud de cada muestra según una función seno de frecuencia '*rate*' (dada en Hz). Como el valor de frecuencia más bajo configurable es 1Hz, o lo que es lo mismo, tiene un periodo de 1 segundo, como nuestra frecuencia de muestreo es de 48000 muestras/segundo, tendremos que tomar 48000 valores entre 0 y 1. Estos valores irán avanzando a través de la variable '*trem_counter*'. La función seno sería entonces:

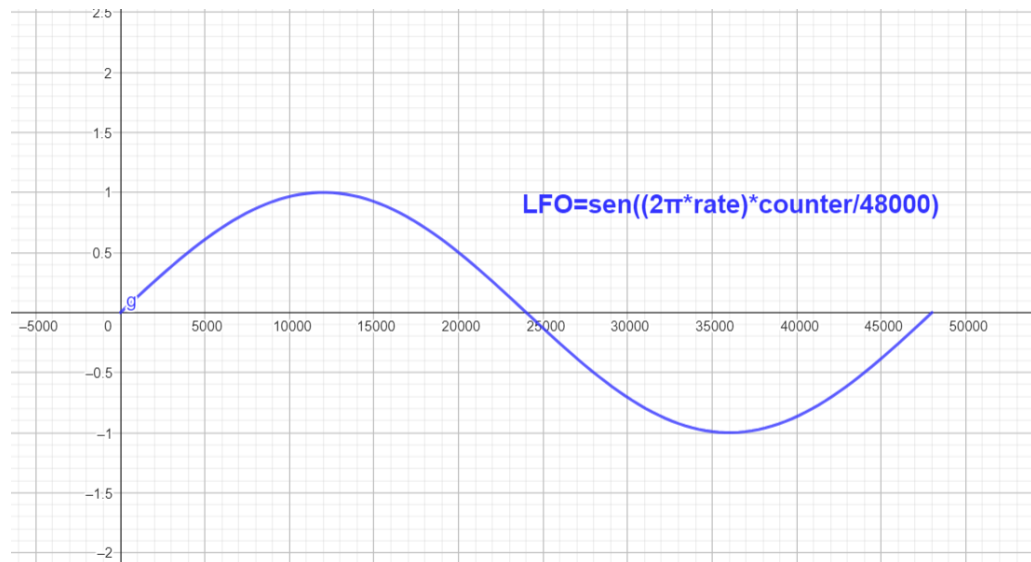


Figura 39: Función seno LFO a 1Hz

Como podemos ver la función se repite cada 48000 muestras a una frecuencia de 1Hz (suponiendo que está configurada así). Esta función otorga valores entre 1 y -1 lo cual no nos sirve ya que lo que queremos es ir modulando el volumen de cada muestra, por lo que necesitaremos transformar la función para que de valores (como máximo) entre 0 y 1. Para conseguir estos valores, multiplicaremos la función anterior por 0.5 y le añadiremos un offset de 0.5, la función entonces quedaría de la siguiente manera:

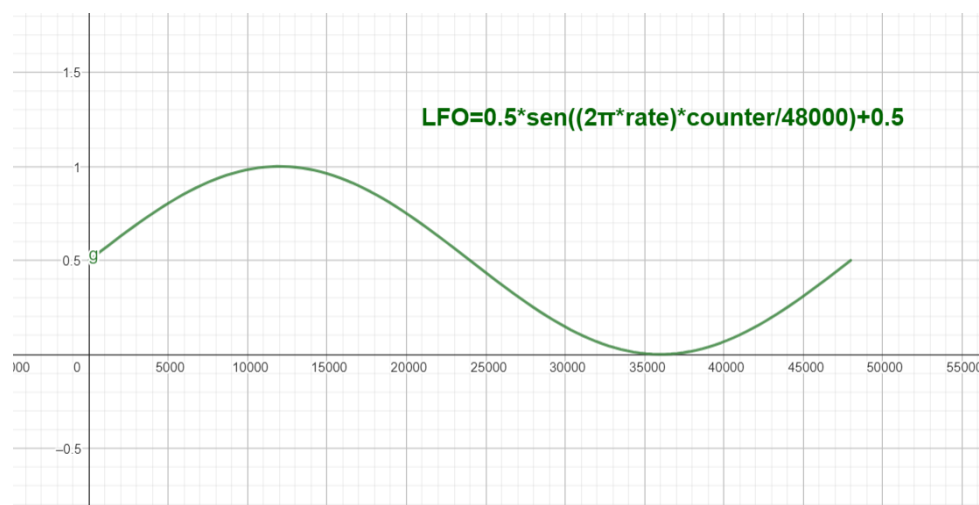


Figura 40: Función seno LFO a 1Hz transformada

Pero si lo dejamos tal cual está, lo que hará el efecto será apagar y encender el sonido, pero tal vez esto no sea lo que nos convenga, sino que se escuche un volumen oscilante. Para ello *'trem_depth'* modificará los valores de *'a'* y de offset de manera que si por ejemplo sólo queremos que el sonido vaya del 30-100% del volumen, un *'trem_depth'* de valor 7 bajará un 70% la amplitud de cada muestra como máximo. Como ejemplo, en la Figura 41 aplicamos el efecto a 1Hz con una profundidad del 70% a una señal sinusoidal de 5Hz:

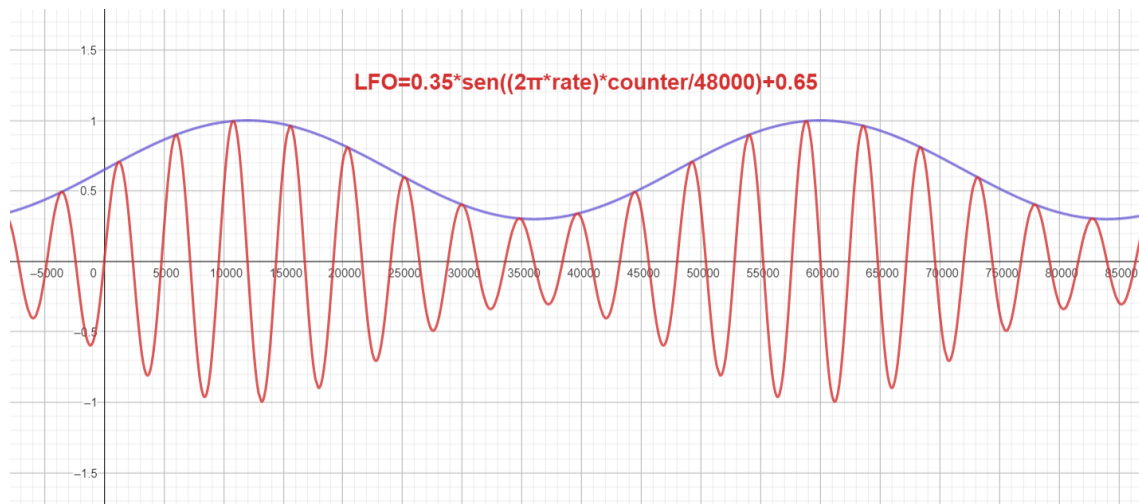


Figura 41: Efecto tremolo a una señal seno de 5Hz

Como podemos ver la señal va oscilando en amplitud, pero no llega a apagarse, se queda en un 30% de la amplitud como mínimo y el 100% como máximo.

Por último, a la muestra leída tenemos que restarle el offset de entrada, ya que si no lo que serían muestras negativas de la señal de guitarra en vez de hacerse más pequeñas las estaríamos haciendo más grandes, dividiendo su valor las acercaremos más al valor 0, el valor de amplitud máxima negativa. Tras la multiplicación de LFO, volvemos a añadir el offset.

(Hack Audio, 2015)

2.3.3.4. CHORUS

Y el último efecto a implementar es el llamado *'Chorus'*. Este efecto consiste en simular que hay varios instrumentos tocando a la vez. Para ello se suele utilizar un *'delay'* de retardo muy pequeño y variable, y en algunos casos se desafina un poco, pues en la vida real es imposible que dos personas toquen la guitarra exactamente al mismo tiempo y sin variaciones en la frecuencia de las notas (esto siempre puede ser casi imperceptible para el oído).

Para programar este efecto, almacenaremos muestras en un buffer con "capacidad variable" (entre comillas, porque realmente lo que hacemos es dejar de escribir y volver al índice 0 en distintos puntos del array). Este retardo variable vendrá dado por una función triangular, o lo que es lo mismo, el *'delay'* irá incrementándose hasta llegar a un valor y luego volviendo a decaer para luego volver a subir y así sucesivamente.

```

void chorus(uint32_t sample[], uint8_t chorus_depth, int n){

    uint16_t c_depth;

    c_depth=chorus_depth*2;
    chorus_buffer[chorus_counter]=sample[n];
    chorus_counter++;

    if(chorus_counter>=chorus_def_depth){
        chorus_counter=0;

        if(triang_up){
            chorus_def_depth=chorus_def_depth+c_depth;

            for(uint8_t i=0; i<c_depth; i++){
                chorus_buffer[c_depth+i]=chorus_buffer[c_depth-1];
            }

            if(chorus_def_depth>=500){triang_up=0;}
        }
        else{
            chorus_def_depth=chorus_def_depth-c_depth;
            if(chorus_def_depth<=200){triang_up=1;}
        }
    }

    sample[n]=sample[n]+chorus_buffer[chorus_counter]-2048;
}

```

Lo primero que haremos será ir almacenando muestras en el buffer de almacenamiento hasta llegar al valor de la variable *'chorus_def_depth'* (por defecto comenzará en 250). Al llegar volveremos a poner el contador a 0 y si estamos en el periodo de subida de la función triangular, añadiremos el valor de *'chorus_depth'* al valor de la variable *'chorus_def_depth'* haciendo que la siguiente vez que se empiecen a almacenar valores en el buffer se genere un retardo mayor. Como los espacios “añadidos” al buffer están vacíos, estos se rellenan a través del bucle “for” con la última muestra tomada, evitando así que queden con valores extraños.

En caso de que nos encontremos en el periodo de decrecimiento del *'delay'*, lo que se hará es ir restando espacio en el buffer, generando un retardo menor. La función tendrá como máximo un retardo de 500 muestras y un mínimo de 200 muestras.

Al final la suma de las muestras retrasadas se hace de la misma forma que en el efecto de *'delay'*, solo que en este caso la amplitud no se ve afectada por ningún parámetro.

2.3.4. DIBUJO DE LAS PANTALLAS DE LA INTERFAZ DE USUARIO

Para dibujar las pantallas que verá el usuario mientras navega por el menú, usaremos la librería anteriormente mencionada. Crearemos un archivo .h para las cabeceras de las funciones de las distintas pantallas llamado “PANTALLAS.h” el cuál vendrá ligado con un fichero .c llamado “PANTALLAS.C”.

Las funciones son:

```
void Pantalla_MENU(void);
```

- Dibujará la pantalla principal de la pedalera



Figura 42: Pantalla de menu

```
void Pantalla_PLAY(uint8_t sel_efecto);
```

-Dibujará la pantalla según el preset seleccionado en ese momento. El preset 1 será rojo, el 2 azul y el 3 verde

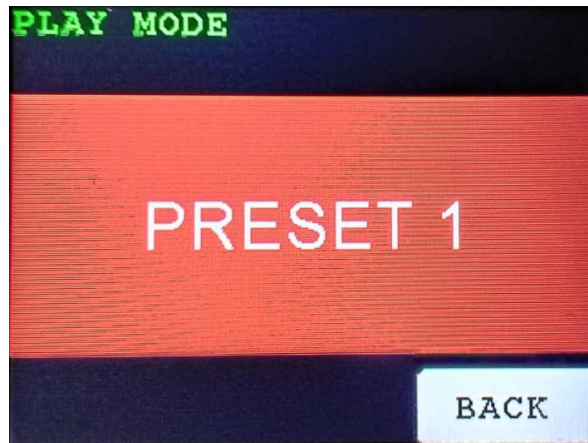


Figura 43: Pantalla del modo PLAY

```
void Pantalla_SETTINGS(void);
```

- Dibujará la pantalla en la que se seleccionará el preset a modificar

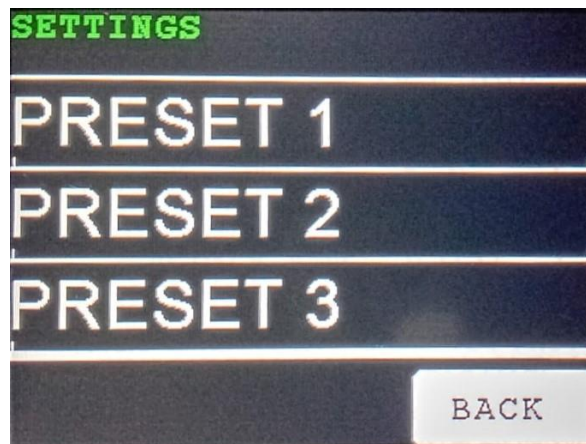


Figura 44: Pantalla de configuración de presets

```
void Pantalla_SETPEDAL_X(uint8_t *ST_odX, uint8_t *ST_tremoloX, uint8_t *ST_delayX, uint8_t *ST_chorusX);
```

- Dibujará la pantalla en la que se verán los pedales activos de cada preset. Hay una función por cada preset.

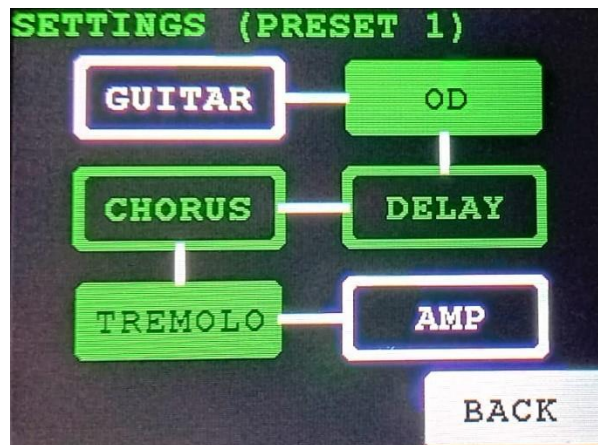


Figura 45: Pantalla de configuración de los efectos del preset

```
void Pantalla_SET_OVERDRIVE(uint8_t *ST_od, uint8_t *treshold);
```

- Dibujará la pantalla con las configuraciones del efecto OVERDRIVE.

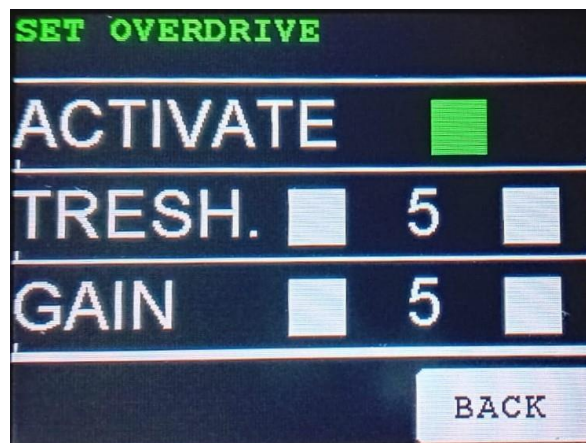


Figura 46: Pantalla de configuración del efecto Overdrive

```
void Pantalla_SET_DELAY(uint8_t *ST_delay, uint8_t *delay_depth);
```

- Dibujará la pantalla con las configuraciones del efecto DELAY.

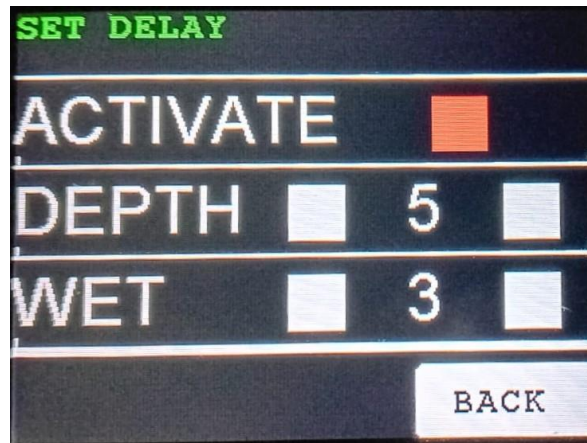


Figura 47: Pantalla de configuración del efecto Delay

```
void Pantalla_SET_CHORUS(uint8_t *ST_chorus, uint8_t *chorus_depth);
```

- Dibujará la pantalla con las configuraciones del efecto CHORUS.

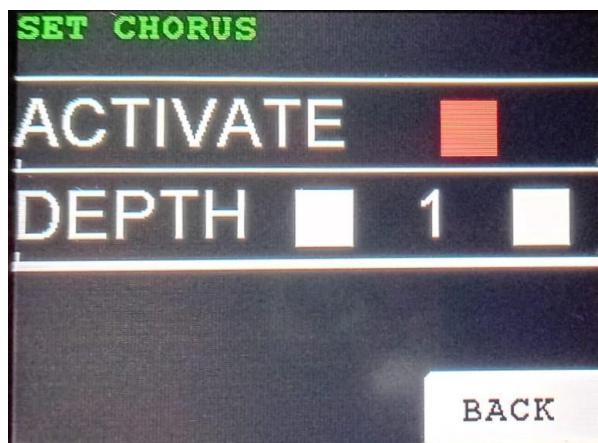


Figura 48: Pantalla de configuración del efecto Chorus

```
void Pantalla_SET_TREMOLO(uint8_t *ST_tremolo, uint8_t *trem_depth, uint8_t *trem_rate);
```

- Dibujará la pantalla con las configuraciones del efecto TREMOLO.

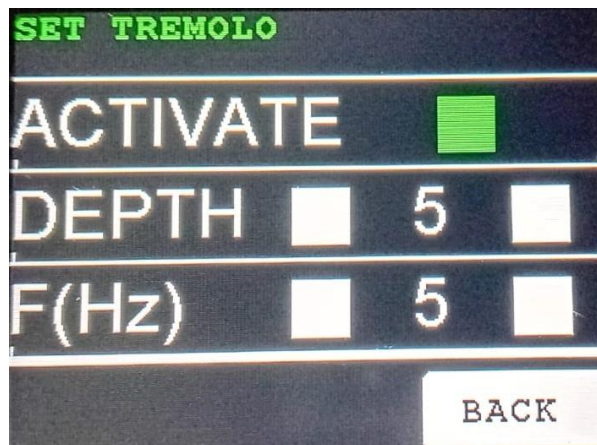


Figura 49: Pantalla de configuración del efecto Tremolo

2.3.5. DIAGRAMA DE FUNCIONAMIENTO

El funcionamiento es bastante sencillo y habrá 2 modos generales de funcionamiento de la pedalera: Modo para tocar y modo de configuración de los presets. A estos dos modos les precede una pantalla de bienvenida y otra de menú, en la que mediante el uso de la pantalla táctil podremos elegir en qué modo estaremos.

Para la lectura de la pantalla táctil usaremos la función “*void TSC2046_ReadXY(void)*” contenida en el fichero “tsc2046.c” que forma parte del paquete de bibliotecas usado para la pantalla LCD. Esta función, almacena las coordenadas x e y en valores de 12 bits en una estructura, que nosotros luego mediante dos variables de tipo float convertiremos a píxeles.

Este es un ejemplo de lectura de la pantalla:

```
if(TSC2046_PENIRQ() == 0){
    TSC2046_ReadXY();
    Xpos= Ts.YPosition/13;
    Ypos= Ts.XPosition/17;
}
```

TSC2046_PENIRQ() es una etiqueta para la función *HAL_GPIO_ReadPin(TSC2046_PENIRQ_GPIO_Port, TSC2046_PENIRQ_Pin);*

que leerá el pin dedicado a activarse (con lógica inversa) cuando la pantalla detecte que se está tocando el táctil, por lo que hasta que no esté igual a 0, no se leerá el táctil. Una vez que tengamos las coordenadas en píxeles en las variables ‘Xpos’ y ‘Ypos’, podremos hacer comprobaciones para saber qué botones se han tocado y las instrucciones a realizar en consecuencia.

El diagrama de flujo del menú principal, con las impresiones por pantalla de los botones para entrar en los modos y haciendo uso de la pantalla táctil tal y como se ha explicado sería:

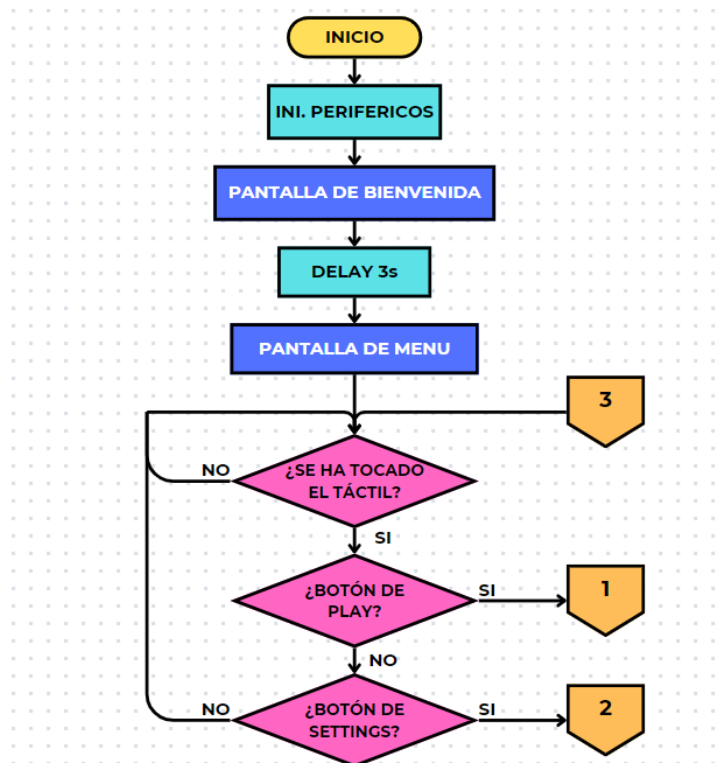


Figura 50: Diagrama de flujo del menú principal

2.3.5.1. MODO “PLAY”

En el modo PLAY, básicamente entrarán 3 funciones distintas: una en la que se imprimirá por pantalla el preset con el que se está tocando además de elegir las variables de los efectos que se van a tener en cuenta para la modificación de la señal, y 2 interrupciones, una será la causada por las funciones *void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc)* y *void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)* y otra interrupción lanzada según los ‘footswitch’ que cambiarán el preset activo.

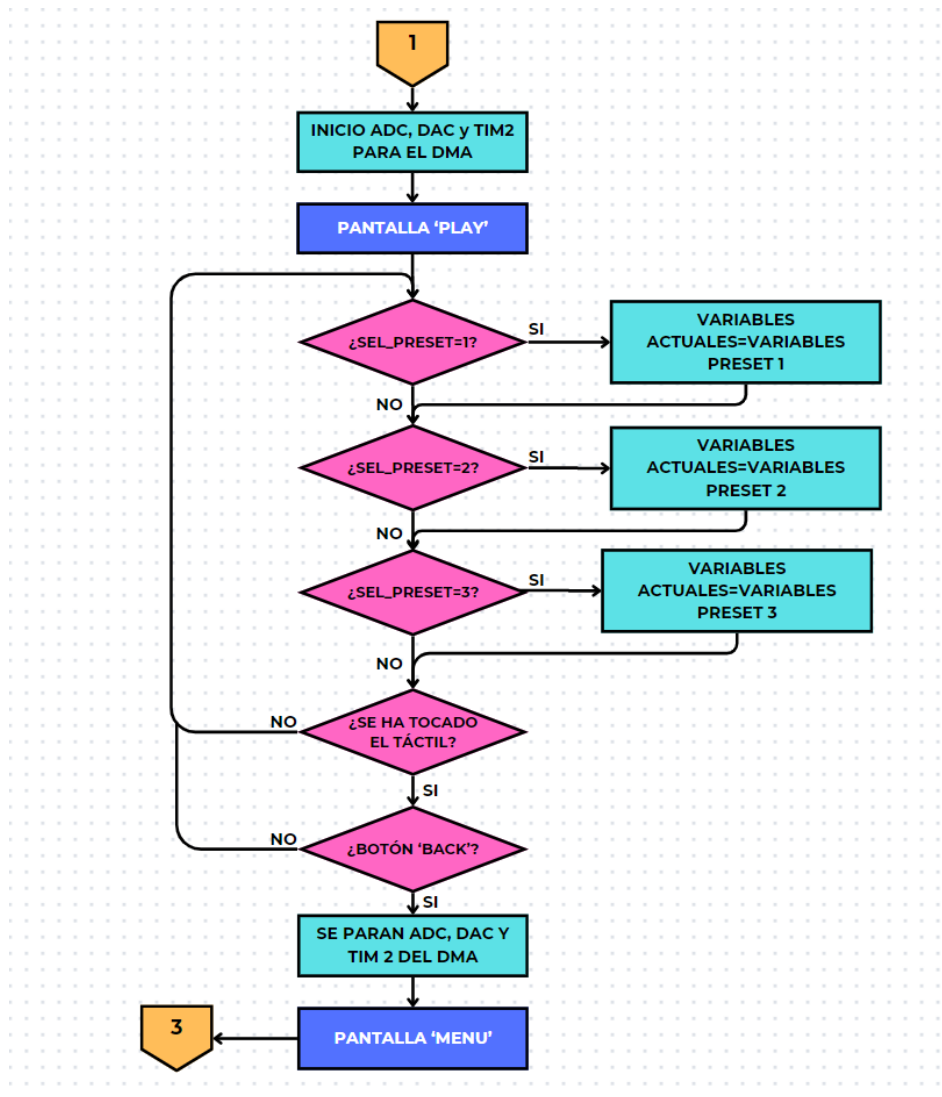


Figura 51: Diagrama de flujo del modo PLAY

Cuando entremos en el modo 'PLAY' lo primero que haremos será activar el ADC, el DAC y el TIM2 para el funcionamiento del DMA y se dibujará la pantalla con el preset 1(en el caso de que sea la primera vez desde que encendimos el dispositivo). Luego el sistema estará leyendo constantemente si ha habido algún cambio de preset o si se ha pulsado de mientras el botón de volver al menú de inicio en el táctil. Mientras tanto, cada vez que el DMA complete medio buffer de lectura del ADC, se ejecutará una de las funciones 'callback' según lo explicado en el punto 2.3.1. Las funciones funcionarán de la siguiente manera:

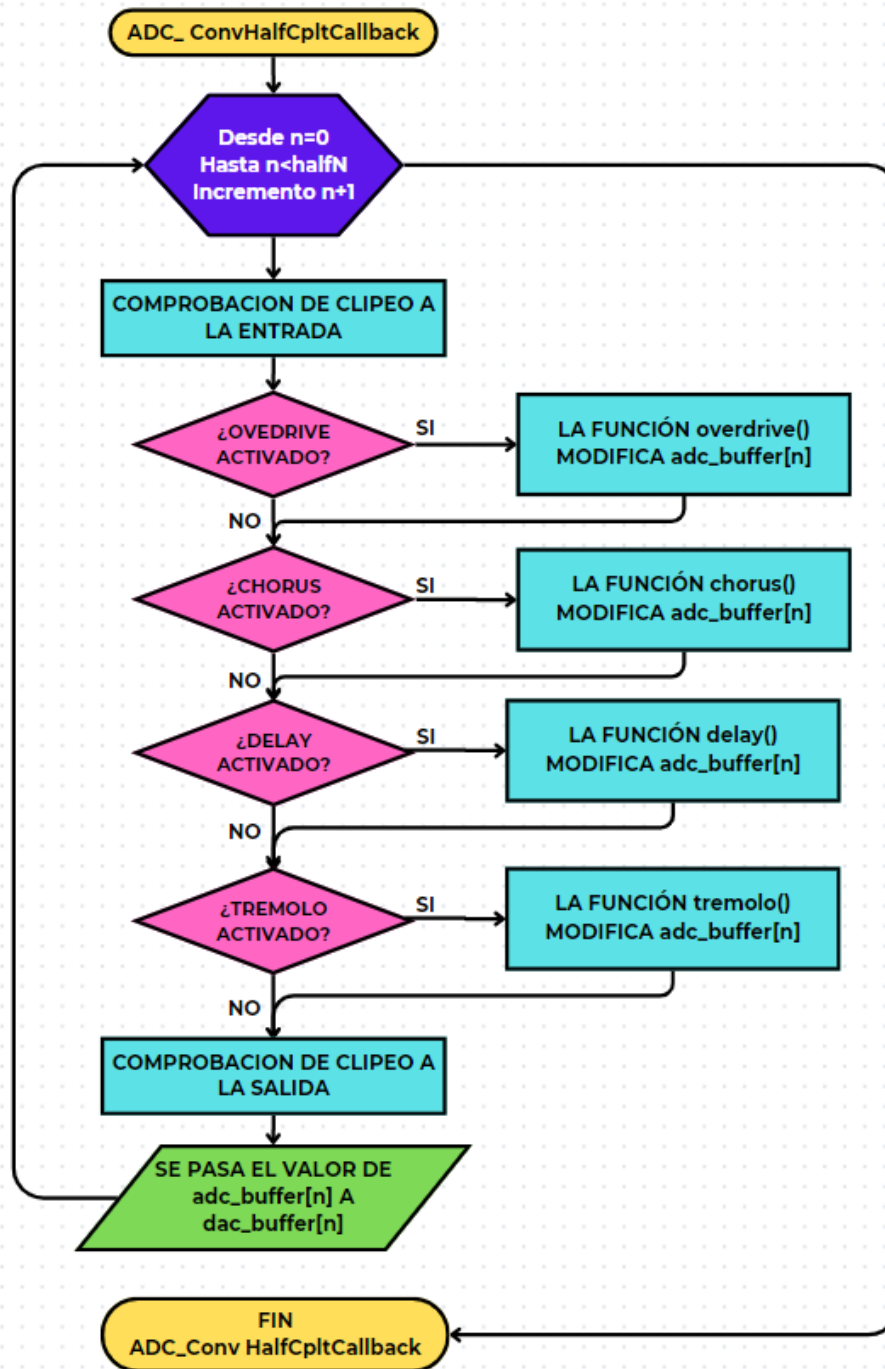


Figura 52: Diagrama de flujo de la función ADC_ConvHalfCpltCallback

Cuando hayamos escrito la mitad del buffer del ADC, se ejecutará la función *void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc)*. Esta función se basa en un bucle 'for' que recorrerá el buffer hasta llegar a la mitad de este (*halfN=256*) e irá modificando cada una de las muestras que han sido recogidas antes de pasarlas al buffer del DAC para su salida. Pero antes de la modificación de las muestras es importante comprobar, que la señal a la entrada no está saturando (o 'clipeando') y provocando que se distorsione, por ello se introduce un trozo de código

antes de la ejecución de los efectos que encenderá o apagará un led rojo según si la muestra llega a los límites de 0 o 3.3V, o lo que es lo mismo, valores de 0 o 4095 (en el código se apura un poco y se comprueban los valores 5 y 4090). El código es el siguiente:

```
if(clip_flag==0){
    if(adc_buffer[n]>=4090 || adc_buffer[n]<=5){ clip_flag=1;}
    else {clip_flag=0;}
}
if(clip_flag==1){
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11, GPIO_PIN_SET);
    clip_counter++;
    if(clip_counter>=4800){
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_11,GPIO_PIN_RESET);
        clip_counter=0;
        clip_flag=0;
    }
}
```

Básicamente lo que hacemos es usar una variable a modo de '*flag*' de activación para que cuando haya una sola muestra que alcance los límites, esté el clip encendido durante 4800 veces que se ejecute la función. Además, de esta manera, cuando se ejecute la función para la otra mitad del buffer del ADC, también siga manteniéndose encendido. Si no hiciéramos esto, el led estaría tan poco tiempo encendido que sería indistinguible para el ojo y se haría el efecto de introducir un PWM a un led, haciendo que se encienda sólo a un porcentaje de su totalidad. A la salida, se hace una segunda comprobación de la muestra tras las transformaciones de los efectos, ya que estos podrían hacer que también saturara en el DAC y generaría distorsiones en el sonido.

Para la función *HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)* es exactamente todo lo mismo, sólo que el bucle 'for' del principio recorrerá el buffer desde la mitad (*halfN=256*) hasta el final (*N=512*).

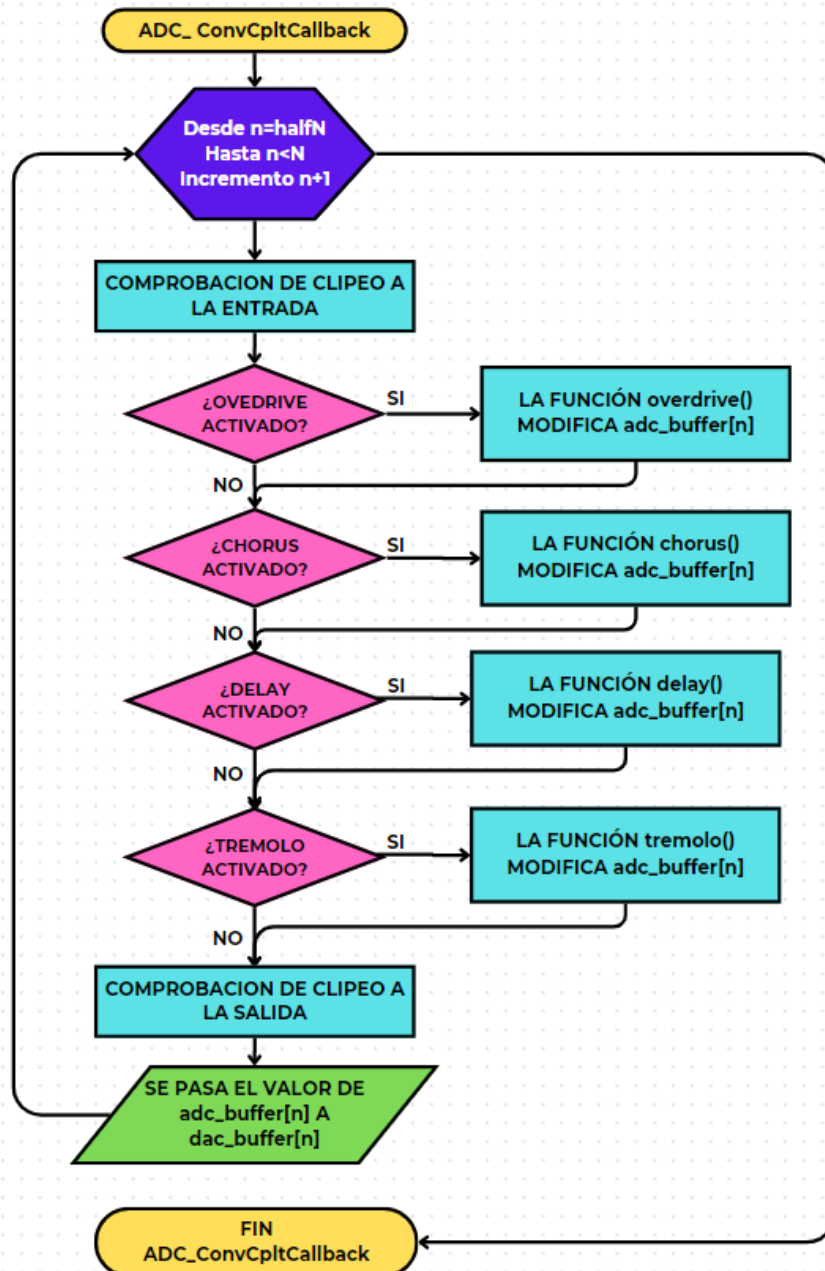


Figura 53:Diagrama de flujo de la función ADC_ConvCpltCallback

La otra interrupción se generará cuando pisemos alguno de los *'footswitch'* para cambiar de preset.

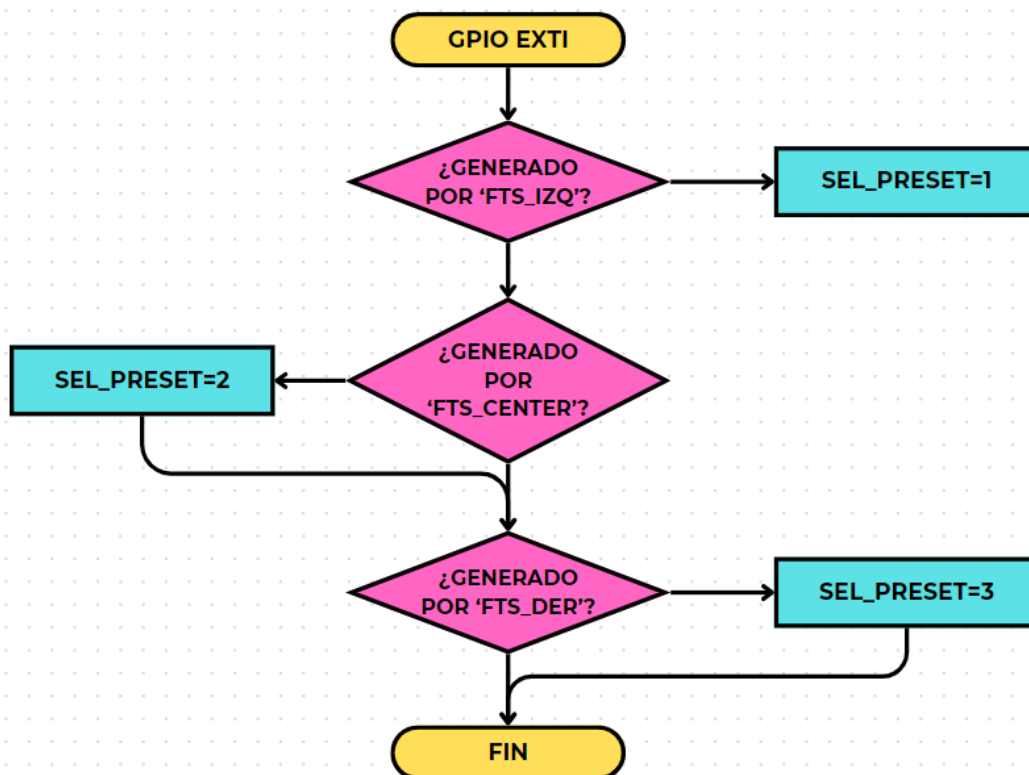


Figura 54:Diagrama de flujo de las interrupciones externas

La función se ejecutará cuando se detecte un flanco de subida en alguno de los puertos conectados a los *'footswitch'*. Simplemente la función tendrá como parámetro cuál de los *'footswitch'* ha sido y al comprobarlo, pondremos la variable *'sel_preset'* con el número de corresponda para que se actualicen las variables actuales en el modo *'PLAY'* y se imprima la pantalla del preset seleccionado.

2.3.5.2. MODO "SETTINGS"

El modo SETTINGS nos servirá para configurar los presets de manera independiente. Usando la pantalla táctil, podremos elegir qué efectos queremos tener activos y modificar las variables que afectan al funcionamiento de los mismos.

Cuando entremos en el modo SETTINGS nos encontraremos un nuevo menú en el que seleccionaremos el preset a editar. Simplemente esperaremos a tocar la pantalla táctil y ver cual se ha seleccionado o si se ha presionado el botón de vuelta al menú principal.

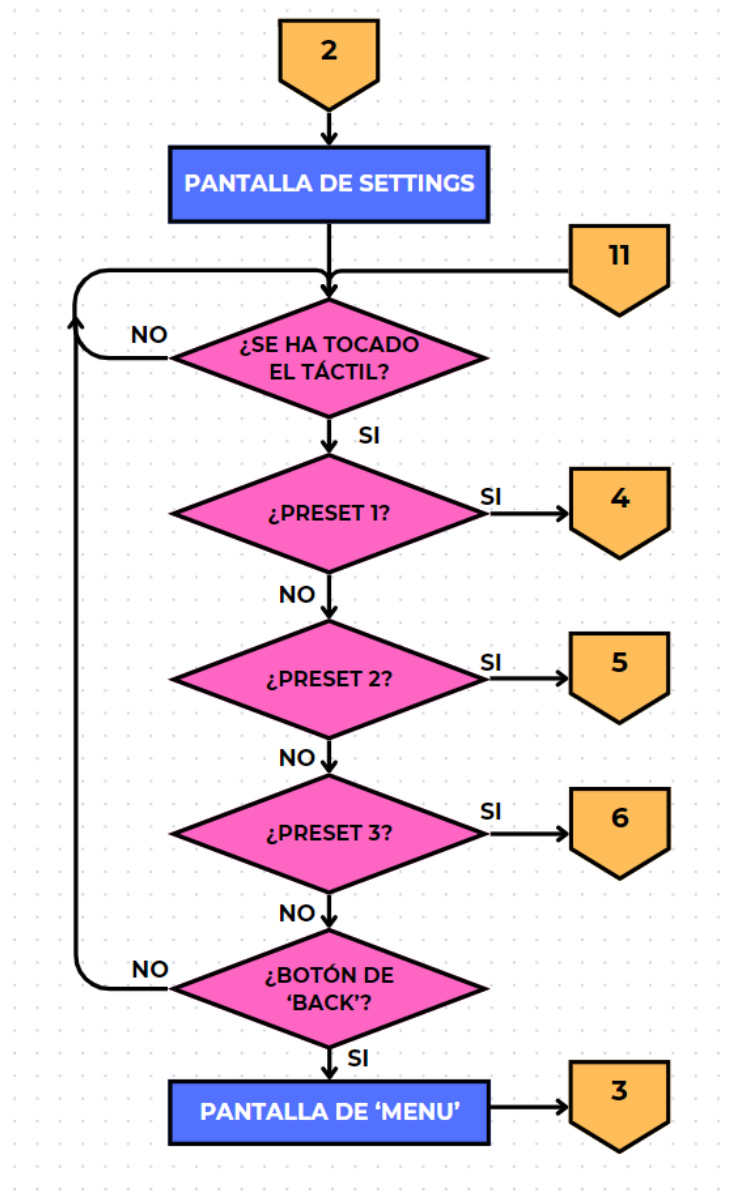


Figura 55: Diagrama de flujo de la selección del preset a configurar

Una vez entremos en la configuración de uno de los presets, nos encontraremos con una representación de lo que sería la cadena de efectos desde la guitarra hasta el amplificador, en la que podremos ver qué efectos están actualmente activos. Igual que en la pantalla anterior, comprobaremos si se ha tocado el táctil y si se ha hecho en alguno de los efectos, pasaremos a la pantalla de configuración del mismo o si se ha presionado el botón de volver, volveremos a la pantalla de selección del preset a editar.

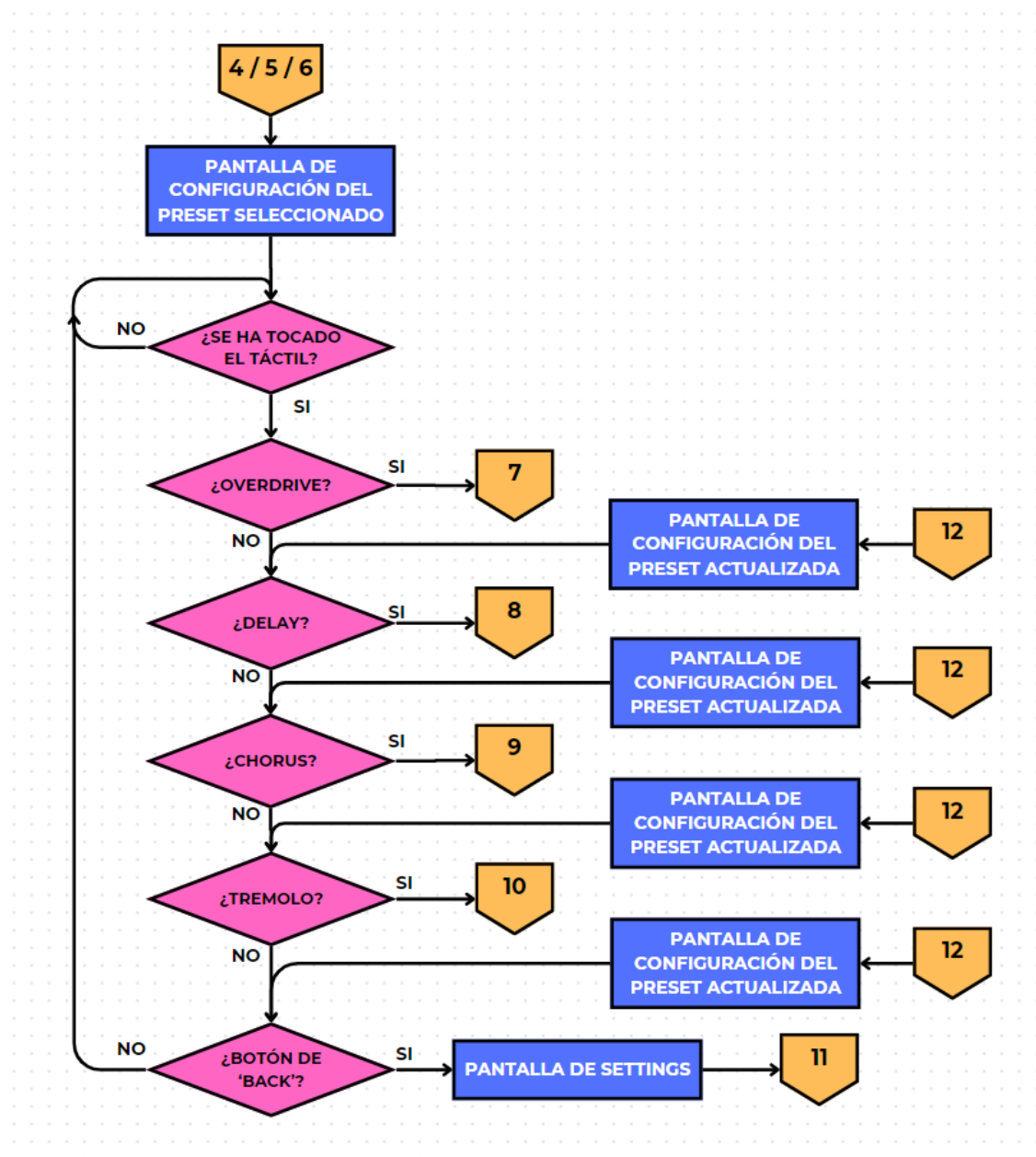


Figura 56: Diagrama de flujo del preset elegido para configurar

Una vez hayamos seleccionado el efecto que queremos editar nos aparecerá una pantalla en la que veremos si el efecto está activo o no y la/las variable/s que podremos modificar tocando los botones a la derecha e izquierda de la misma. Esperaremos a que haya un toque con el táctil y actualizaremos las variables necesarias según el botón utilizado, o por si el contrario se ha pulsado el botón de volver, volveremos a la pantalla del preset. Si hemos activado o desactivado algún efecto, al volver podremos verlo reflejado en la cadena de efectos.

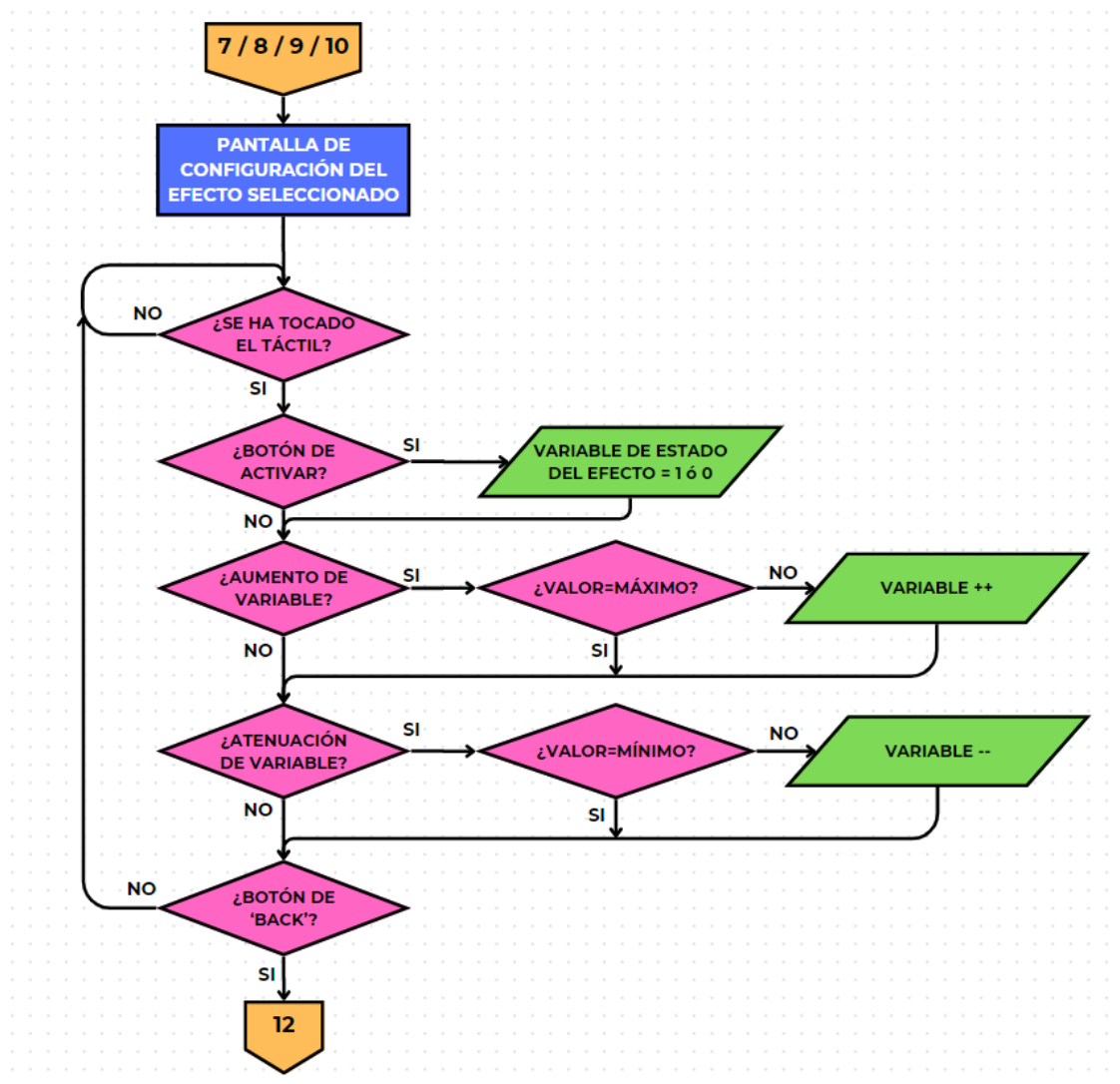


Figura 57: Diagrama de flujo de la configuración del efecto a modificar

2.4. DISEÑO DEL ENCAPSULADO

Para el diseño del encapsulado de este prototipo que contendrá todo el circuito de la pedalera y lo protegerá contra los golpes, se ha optado por usar la fabricación aditiva con impresoras 3D. Para ello, se ha usado el programa de diseño 3D “Catia V5” de Dassault Systems.

En el diseño se ha optado por usar 2 piezas que se ensamblan una sobre la otra y van atornilladas lateralmente a una pieza inferior que se ocupa de que la PCB y la placa de desarrollo reposen, así como de soporte para los ‘footswitch’. Las piezas superiores tendrán huecos para que el usuario pueda acceder a los elementos necesarios para el uso de la pedalera, como son la pantalla, los Jack de entrada/salida, los potenciómetros, los ‘footswitch’, la entrada de alimentación tipo C y para que quede a la vista el LED de clipeo. La entrada, salida y potenciómetros, así como el LED estarán debidamente etiquetados. Además, el logo del modelo de la pedalera podrá verse en la superficie.

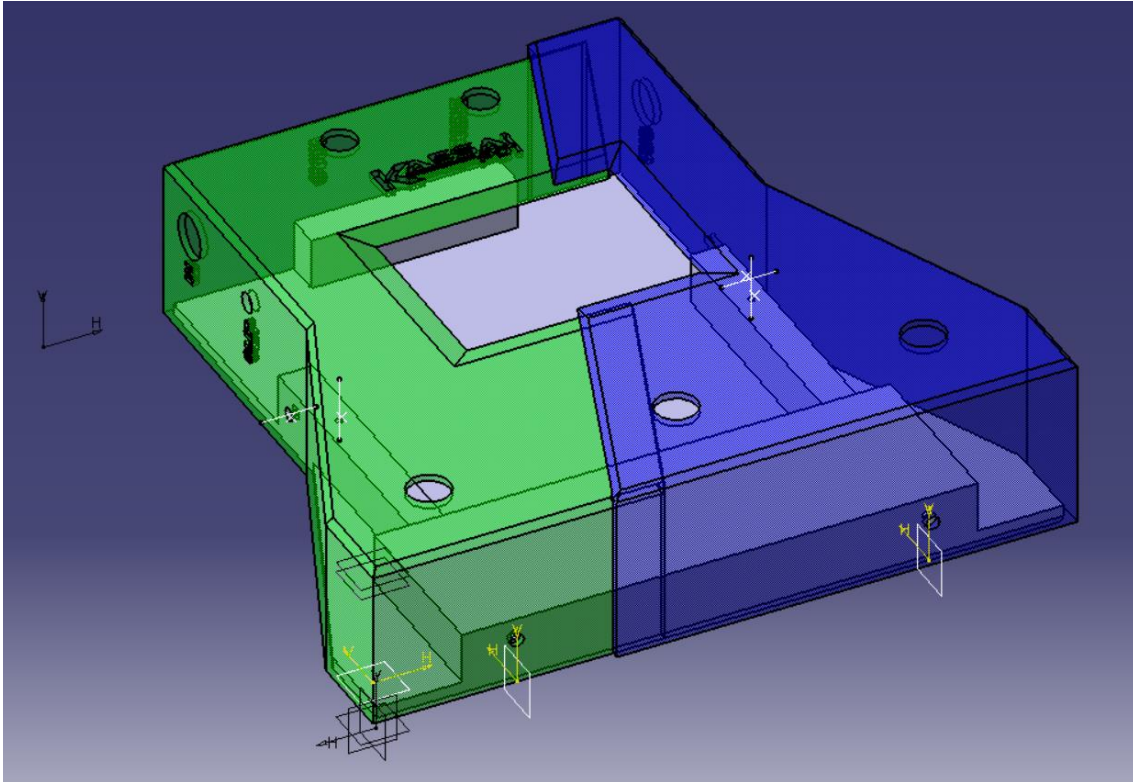


Figura 58: Diseño de la carcasa 3D

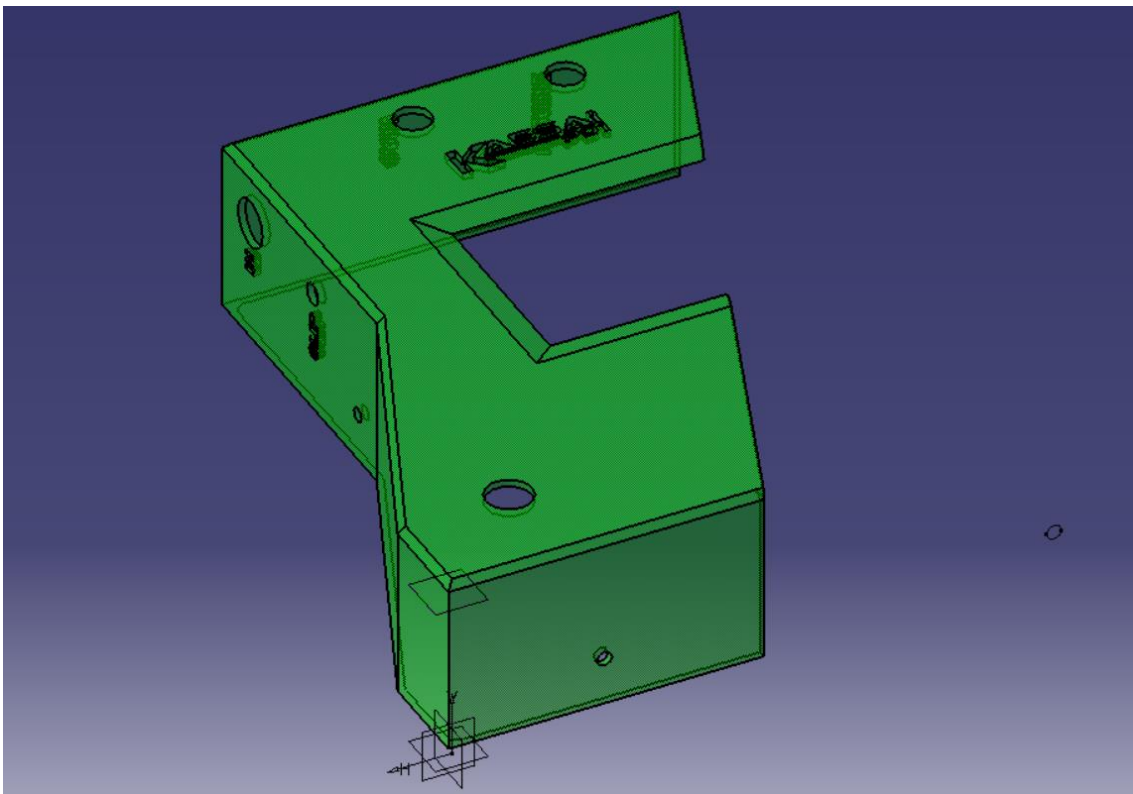


Figura 59: Diseño 3D de la pieza superior 1

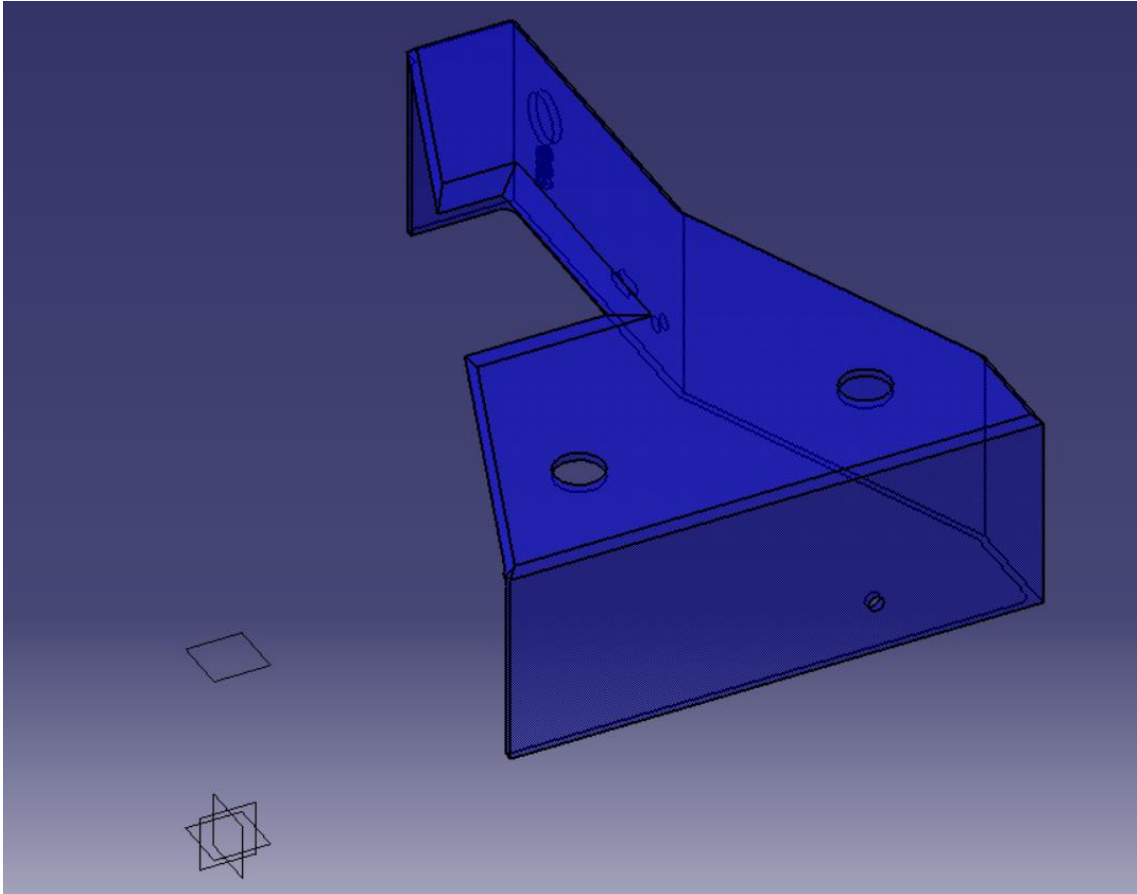


Figura 60: Diseño 3D de la pieza superior 2

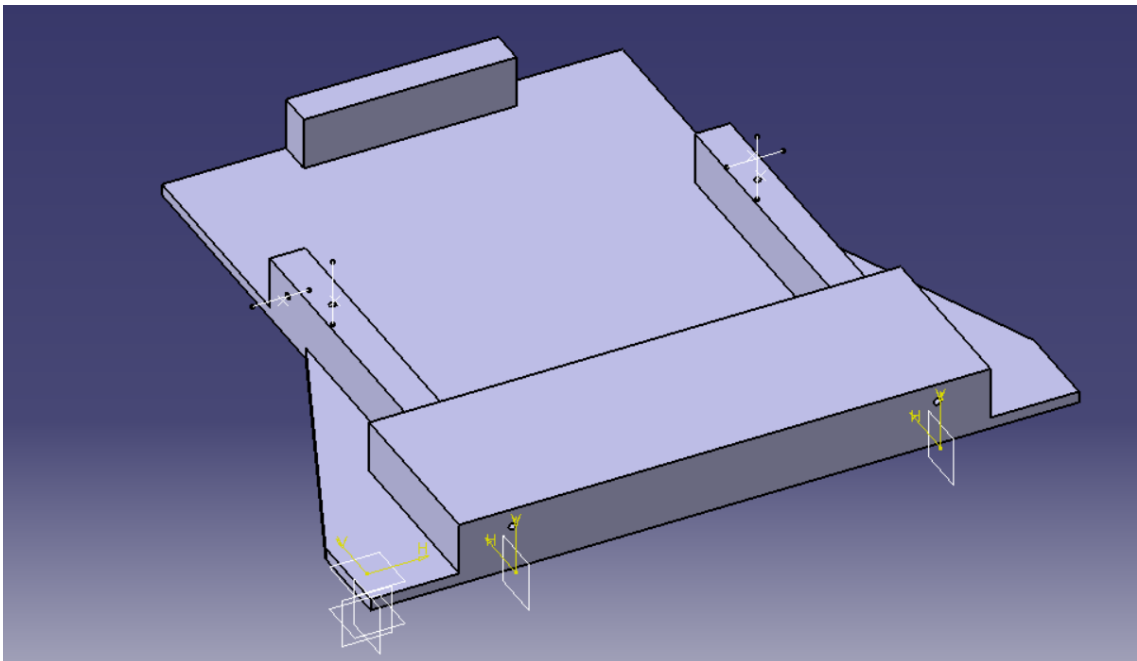


Figura 61: Diseño 3D de la pieza inferior

Para la impresión de estas piezas se ha utilizado la máquina una impresora 3D Creatbot F430, situada en el laboratorio de robótica de la Escuela Superior de Ingeniería de la Universidad de Cádiz. El material utilizado para la impresión es filamento de PLA de 1.75mm.

Tras la impresión 3D de las diferentes piezas, éstas han pasado por un proceso de lijado manual para pulir las pequeñas imperfecciones generadas en la impresión y posteriormente se han pintado con pintura acrílica negra para dar un color más uniforme. Como decoración, se le ha aplicado pintura de color verde neón a una de las piezas usando una esponja como pincel, para darle esa textura característica.

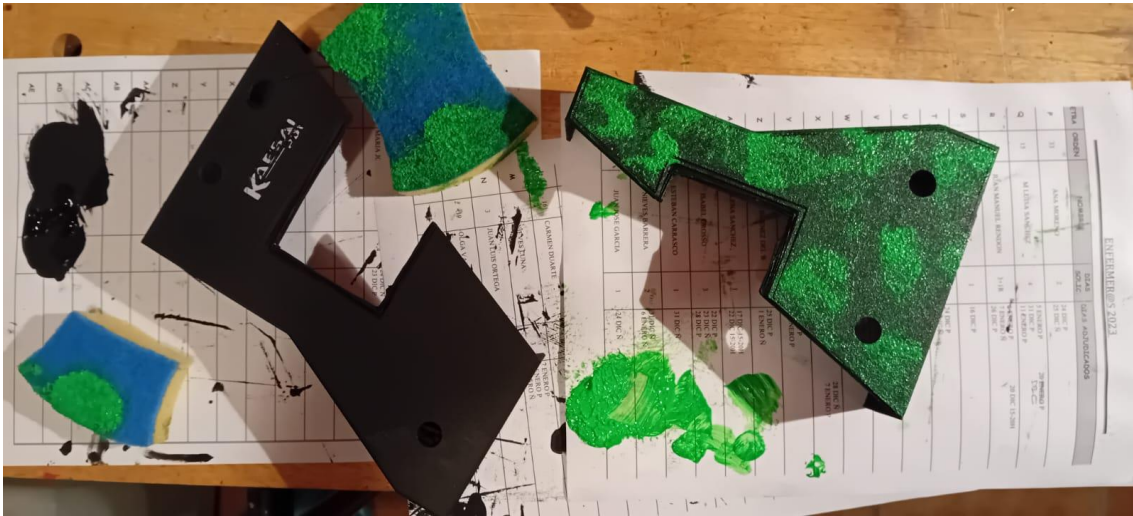


Figura 62. Proceso de pintado de las piezas superiores

Además, a la pieza inferior, se le ha acoplado una lámina de goma-eva en la parte de abajo y 4 patas de goma para evitar el roce del plástico directamente con el suelo, lo cual desgastaría la pieza.



Figura 63: Mejora de la pieza inferior

Por último, sólo faltaría ensamblar todas las piezas, conectar y asegurar los 'footswitch' y asegurar los Jack de entrada y salida.

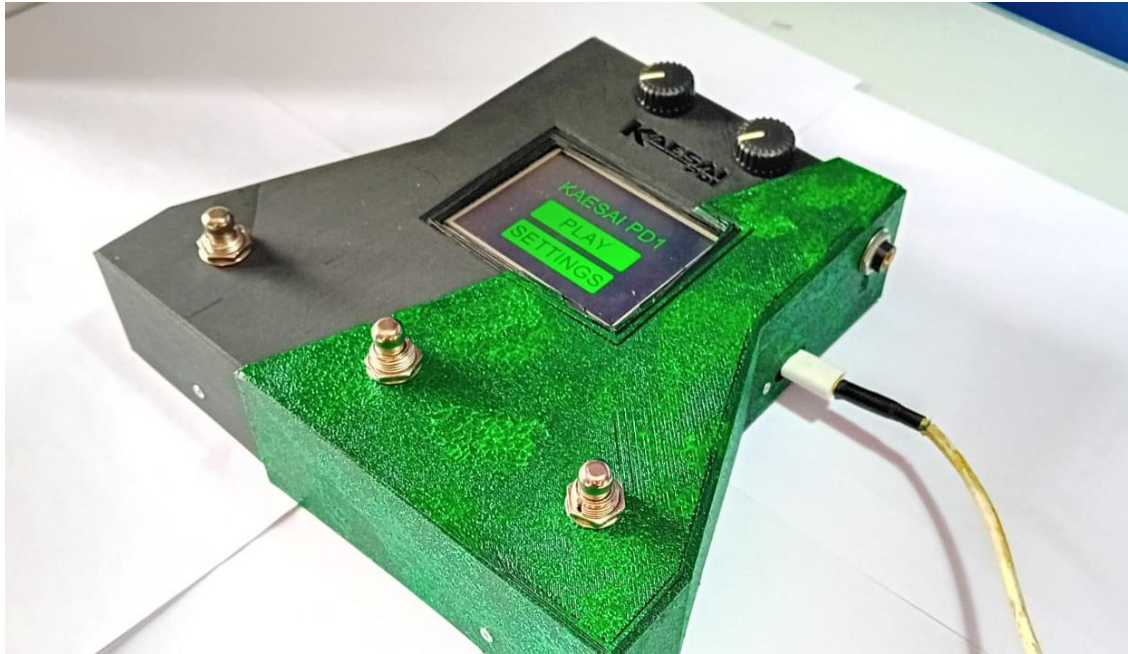


Figura 64: Pedalera ensamblada

3. PRUEBAS

Para la realización de pruebas y comprobar que los efectos funcionan correctamente hemos conectado un generador de funciones a la entrada de la pedalera y un osciloscopio TENMA 72-8725. A la entrada introduciremos una señal sinusoidal de 150mV de pico y 450Hz, centrada en 0V, e iremos aplicando los diferentes efectos y observando su respuesta a la salida. El primer efecto a comprobar será el de 'Overdrive'.

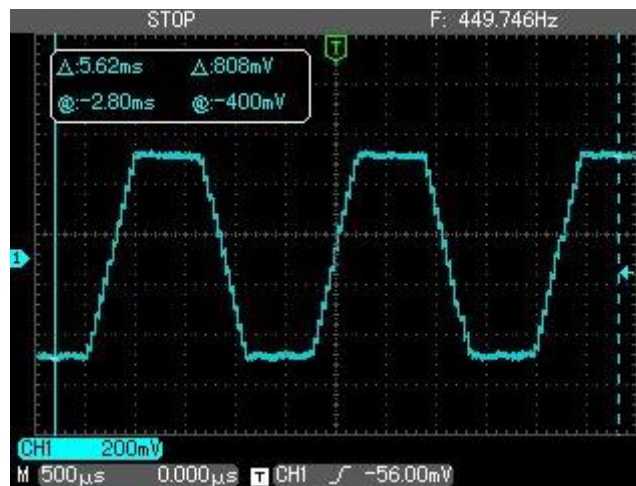


Figura 65: Prueba del efecto Overdrive

Tal y como debería de suceder, la señal al llegar a los umbrales superior e inferior “saturan” virtualmente la señal y provocan una distorsión de la señal que genera ese sonido duro y característico. A continuación, el efecto de 'Delay'.

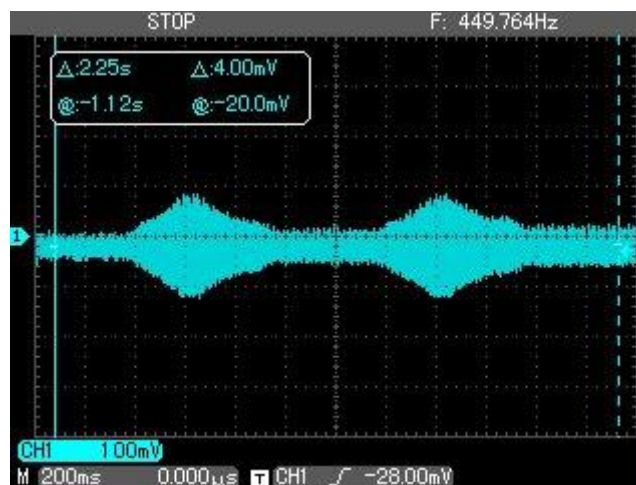


Figura 66: Prueba del efecto Delay

Para que este caso pudiéramos verlo con la mayor claridad posible en el osciloscopio, hemos configurado el efecto para que tenga el mayor volumen de mezcla y el mayor tiempo de retardo posible (500ms). Hemos aumentado momentáneamente la amplitud de la señal para crear un pico visible en el osciloscopio y comprobar que ese mismo pico se reproduce más tarde. El siguiente efecto sería el 'Chorus'.

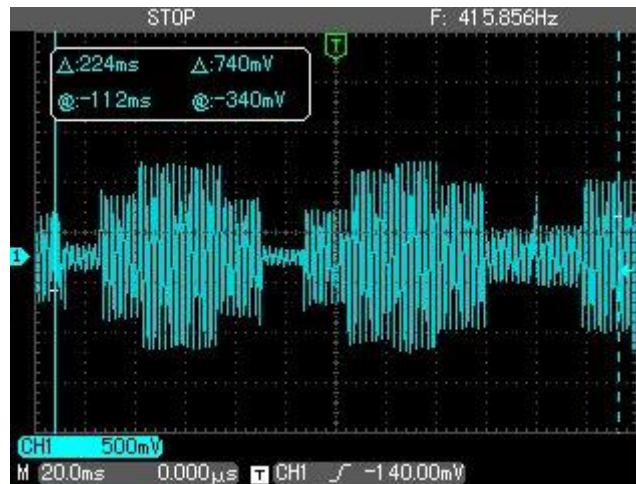


Figura 67: Prueba del efecto Chorus

Este efecto es difícil de analizar de manera visual, ya que es un *delay* muy corto que va aumentando y disminuyendo. Es lo que nos sugieren esos "montículos" que aparecen en la señal, pero en este caso, la mejor manera de comprobar su funcionamiento es escuchar y comprobar que suena tal y como queremos. Y el último efecto a probar sería el 'Tremolo'.

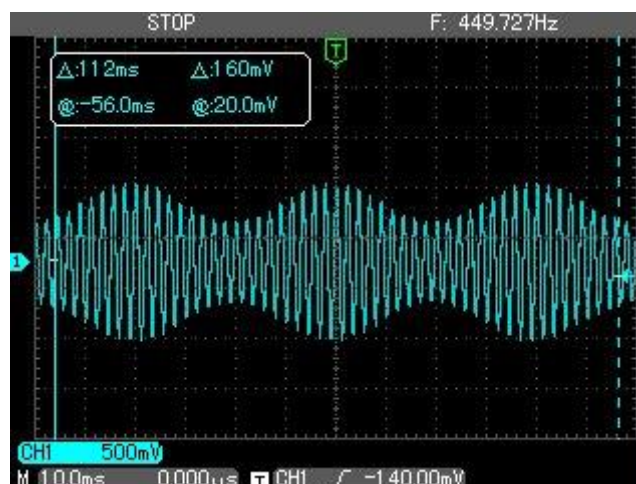


Figura 68: Prueba del efecto Tremolo

El 'Tremolo' es simplemente la modulación de la amplitud de la señal de acuerdo a una onda sinusoidal, justo como podemos ver en la imagen. En este caso, la profundidad del efecto está puesta

a la mitad de su capacidad, haciendo que el sonido sólo fluctúe en un 50% de su amplitud. Por último, probaremos una combinación de efectos para comprobar que son secuenciables, para ello hemos elegido 'Overdrive' y 'Tremolo', ya que son los que se ven de manera más clara en el osciloscopio y no afectan de la misma manera a la señal.

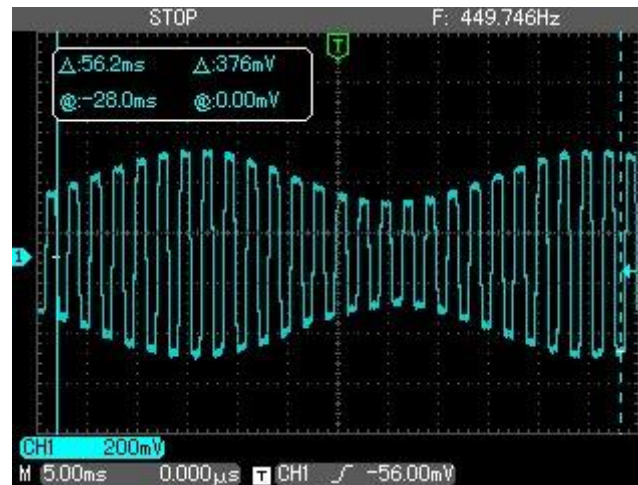


Figura 69: Prueba de la combinación de Overdrive y Tremolo

Tal y como podemos observar, el 'Tremolo' modula la señal en un 50% al igual que en la prueba del efecto en solitario, pero en este caso vemos que la señal modulada tiene el recorte generado por el 'Overdrive'. Este resultado, también nos explica la importancia del orden de la cadena de efectos, ya que si primero tuviéramos el 'Tremolo' y después el 'Overdrive', primero modularíamos la amplitud de la señal y luego según el umbral configurado, la señal sería distorsionada o no. En ese caso, en vez de escuchar una señal distorsionada con una amplitud modulada, tendríamos una señal modulada que va distorsionando y dejando de distorsionar. Realmente para poder sacar el máximo rendimiento y la mezcla perfecta de la actuación de los dos efectos, la primera es la mejor de las opciones, sin embargo, en la música no hay nada correcto o incorrecto, sólo hay cosas más usadas y otras que menos. Es posible que el sonido generado a través de la segunda opción sea válido y de utilidad para la aplicación que necesita el artista, y tal vez nadie antes lo haya probado o haya conseguido darle un sentido a lo que genera, esta clase de cosas son las que hacen que la música evolucione y se originen géneros nuevos.

También es necesario comentar, se han hecho pruebas a nivel de sonido para ajustar ciertos parámetros de los efectos, como por ejemplo el nivel de ganancia máximo del 'Overdrive' o la frecuencia del 'Tremolo'.

4. PRESUPUESTO

A continuación, el desglose del presupuesto de los materiales utilizados en el proyecto para la fabricación del prototipo de la pedalera:

MATERIAL	CANTIDAD	PRECIO	IMPORTE
Placa de desarrollo STM32F407VET6	1	13,60 €	13,60 €
ST-Link V2	1	4,76 €	4,76 €
Pantalla LCD TFT ILI9341	1	10,88 €	10,88 €
Footswitch momentáneo SPST-NO, tacto suave. SCI	3	4,50 €	13,50 €
Jack 6,35mm hembra 3 pines PCB	2	0,83 €	1,66 €
LM833D STMicroelectronics	2	0,75 €	1,50 €
Adaptador USB Tipo C	1	2,45 €	2,45 €
Potenciómetros 100K	2	1,62 €	3,24 €
Perilla de potenciómetro	2	1,26 €	2,52 €
Conectores terminales PCB KF301-5.0-2P	3	0,08 €	0,23 €
LED 5mm rojo	1	0,10 €	0,10 €
Módulo inversor de voltaje 5V	1	3,37 €	3,37 €
Resistencias de varios valores	14	0,10 €	1,40 €
Condensadores de varios valores	3	0,30 €	0,90 €
Conectores JST	4	0,48 €	1,92 €
Placa de cobre laminado para PCB	1	1,90 €	1,90 €
Impresión de carcasa 3D	1	5,00 €	5,00 €
Pintura	1	1,20 €	1,20 €
TOTAL			70,13 €

Este precio puede parecer elevado, pero hay que contar con que todos los componentes se han adquirido al por menor en cantidades muy muy pequeñas, lo que hace que el precio de los componentes sea bastante elevado. Además, se ha usado una placa de desarrollo completa, si fuera a comercializarse en masa, sería interesante desarrollar una PCB donde sólo contáramos con el chip del microcontrolador, el ADC y el DAC, ahorrando en costes y en muchísimo espacio.

5. MEJORAS PROPUESTAS

La primera mejora y que más a simple vista se aprecia es el tamaño. La pedalera podía ser bastante más pequeña, pero el acceso a la alimentación microUSB de la placa de desarrollo, extendida a través de un adaptador a USB tipo C en el lateral hace que la pedalera sea más alta que ancha. Si se hiciera una PCB dedicada, con componentes mucho más pequeños, con diseño de una fuente de alimentación propia y sin el uso de la placa de desarrollo (sólo el chip microcontrolador), podríamos obtener una pedalera mucho más pequeña y fácil para su transporte.

La segunda mejora más importante sería el diseño y construcción de la carcasa. Ya hemos hablado del tamaño, pero un dispositivo el cuál va a recibir muchos pisotones, golpes en el transporte y en definitiva desgaste, no conviene que esté hecho con plástico de grosor de milímetros. Lo mejor para esta clase de pedales de instrumento, es usar algún tipo de metal duro y resistente que aguante cualquier tipo de golpe, además de procurar estar defendido contra derrames de líquidos y/o lluvia. En definitiva, el encapsulado debe ser total, y usar impresión 3D de filamento convencional no es lo ideal, mucho mejor acero o aluminio.

También sería interesante poder disponer de un disco duro o almacenamiento que guarde los presets realizados por el usuario, ya que cada vez que se reinicia la pedalera, los valores de las variables de configuración son borrados.

Por último, los efectos diseñados en este proyecto tienen un carácter muy básico, aunque de sonido sorprendentemente bueno para su simplicidad. Programar un efecto el cuál imite a la perfección cualquier pedal analógico real podría suponer un proyecto por sí sólo, nos tendríamos que introducir en análisis en espectros de frecuencia, ganancias, filtros, etc. Y el objetivo de este proyecto es diseñar una pedalera multiefectos desde 0 y construirla, con mejores o peores efectos, pero que fuera funcional para cualquier guitarrista. También la placa de desarrollo y el lenguaje C empleado tiene sus limitaciones, con conocimientos de Java o Python, lo mejor tal vez hubiera sido emplear un a 'Raspberry Pi' e intentar implementar '*plugins*' digitales VST de carácter gratuito los cuáles están mucho más trabajados para sonar como pedales reales analógicos.

6. CONCLUSIONES FINALES

Este proyecto ha sido en general muy completo ya que hemos podido combinar para el desarrollo de un prototipo 3 áreas distintas de la ingeniería:

En primer lugar, usamos la electrónica, para el diseño de la PCB y los circuitos de adaptación de la señal al microcontrolador.

Luego usamos la informática, para el desarrollo del DSP y los menús que hacen funcionar la pedalera.

Y por último, usamos el diseño 3D y el uso de fabricación aditiva para la carcasa.

Ha sido un proceso total de diseño, con el que nos hemos introducido en el inmenso mundo del DSP en audio y hemos obtenido un resultado más que útil para el uso de cualquier guitarrista. Como hemos dicho, este diseño es un prototipo, pero la capacidad de mejora es enorme, ya que, aunque válido, el uso de la tarjeta de desarrollo conlleva algunas limitaciones, no sólo en rendimiento y capacidad, si no también sobre todo en el tamaño de dispositivo.

En definitiva, este prototipo es un proyecto perfecto para estudiar el proceso de diseño de un producto útil, funcional y atractivo con el que iniciarse en el mundo de la electrónica musical y el DSP.

7. BIBLIOGRAFÍA

ACELEX. (n.d.). *ILI9341*.

Aerosol laca protectora aislante A/1. (n.d.). Dieltron.Com. Retrieved May 19, 2024, from <https://www.dieltron.com/aislarco1-tasovision-aislarco1-laca-protectora-aislante-bote-335-10575.html>

Arduino. (2019). *Arduino DUE*. <https://arduino.cl/wp-content/uploads/2019/09/arduino-due.png>

Doxrud, J. (2019). *Anatomía de la guitarra eléctrica: las pastillas (pickups)*.

<https://images.squarespace-cdn.com/content/v1/554a5733e4b02559210e4177/1564253661869-IZDFQ3XVLRauOp2BN7NB/Captura+de+Pantalla+2019-07-27+a+la%28s%29+14.53.58.png?format=1000w>

ElectroSmash. (2017). *Electro Smash: How to Start Programming pedalSHIELD*.

<https://www.electrosmash.com/forum/software-pedalshield/21-how-to-start-programming-pedalshield?lang=en>

Espinosa, J. A. (2014). *Ondas de vibrato y tremolo*. [https://1.bp.blogspot.com/-](https://1.bp.blogspot.com/-FztAIRqOnIM/VEYpLW51ZRI/AAAAAAAAARA/yqT88E3LPi8/s1600/Ondas.png)

[FztAIRqOnIM/VEYpLW51ZRI/AAAAAAAAARA/yqT88E3LPi8/s1600/Ondas.png](https://1.bp.blogspot.com/-FztAIRqOnIM/VEYpLW51ZRI/AAAAAAAAARA/yqT88E3LPi8/s1600/Ondas.png)

Hack Audio. (2015, September 3). *Tremolo*. Hack Audio. <https://www.hackaudio.com/digital-signal-processing/combining-signals/tremolo/>

Hudak, Z. (2018). *STM32F407VET6*.

https://os.mbed.com/media/uploads/hudakz/stm32407vet6_view01.jpg

Manualguitarraelectrica.com. (2020, July 12). *SALIDA de las PASTILLAS para Guitarra Eléctrica - Ganancia*. Manual Guitarra Eléctrica. <https://manualguitarraelectrica.com/salida-pastillas-guitarra-electrica/>

National Instruments. (2023, March 30). *Adquirir una Señal Analógica: Ancho de Banda, Teorema de Muestreo de Nyquist y Aliasing*. NI. <https://www.ni.com/es-es/shop/data-acquisition/measurement-fundamentals-main-page/analog-fundamentals/acquiring-an-analog-signal--bandwidth--nyquist-sampling-theorem-.html>

- Practice, E. I. (2022). STM32G4 & real time DSP: Part 5 ADC to DAC with DSP, multiplication, addition, and time delays [Video]. In *YouTube*.
<https://www.youtube.com/watch?v=zA3PVZzaUiU&list=PLwa4R2HaZEqihzreDJynbcYGl c-mXn7F2&index=6>
- R, H. (n.d.). *Library for 240x320 graphical display TFT. Driver ILI9341*. Projects with PIC18F Microcontrollers. Retrieved August 21, 2023, from
<http://pic18fx.blogspot.com/2014/12/library-for-240x320-graphical-display.html>
- Ramirez, E. (2017). *Esquema y funcionamiento del efecto de delay*.
- STMicroelectronics. (n.d.). *STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs*.
- Thomann. (2018). *Cadena de pedales de guitarra*. <https://www.thomann.de/blog/es/pedales-guitarra-basicos-una-guia-principiantes/>
- Wikimedia, C. de los proyectos. (2023, June 1). *Trémolo*. Wikipedia.
<https://es.wikipedia.org/wiki/Tr%C3%A9molo>
- Wikipedia, C. de los proyectos. (2023a, May 26). *Efectos de guitarra*. Wikipedia.
https://es.wikipedia.org/wiki/Efectos_de_guitarra
- Wikipedia, C. de los proyectos. (2023b, May 31). *Guitarra eléctrica*. Wikipedia.
https://es.wikipedia.org/wiki/Guitarra_el%C3%A9ctrica