

UNIVERSIDAD NACIONAL DE LA MATANZA

Departamento de Ingeniería e Investigaciones Tecnológicas

**Ingeniería en Electrónica**

**Técnicas digitales III**

Pedal Multiefecto para Guitarra Eléctrica

**Integrante:** Rosmirez, Jorge

**Profesores:** Lic. Carlos Maidana

Ing. Guillermo Buranits

Ing. Mauro Cipollone

2020

[1 Descripción general 1](#_Toc40565713)

[2 Introducción teórica 1](#_Toc40565714)

[2.1 Delay 1](#_Toc40565715)

[2.1.1 Estructura básica de un Delay 2](#_Toc40565716)

[2.2 Fuzz 3](#_Toc40565717)

[2.3 Wah-Wah 5](#_Toc40565718)

[3 Descripción técnica del proyecto. 6](#_Toc40565719)

[3.1 Hardware 6](#_Toc40565720)

[3.1.1 Protocolo I2S 6](#_Toc40565721)

[3.1.2 Driver CS43L22 7](#_Toc40565722)

[3.1.3 ADC 9](#_Toc40565723)

[3.1.3.1 Configuración con el Códec de DAC 9](#_Toc40565724)

[3.1.3.2 Transmisión de audio al códec del DAC 9](#_Toc40565725)

[3.1.4 Diagrama en bloques 10](#_Toc40565726)

[3.1.5 Esquemático 10](#_Toc40565727)

[3.1.5.1 Jack 10](#_Toc40565728)

[3.1.5.2 Divisor resistivo 11](#_Toc40565729)

[3.1.5.3 Potenciómetro 11](#_Toc40565730)

[3.1.5.4 Filtro Pasa Banda 11](#_Toc40565731)

[3.1.6 Circuito impreso 12](#_Toc40565732)

[3.1.7 Fotos 13](#_Toc40565733)

[3.2 Software 13](#_Toc40565734)

[3.2.1 Enumeración de rutinas 14](#_Toc40565735)

[3.2.2 Descripción del funcionamiento de cada rutina 14](#_Toc40565736)

[4 Modo de operación 21](#_Toc40565737)

[5 Ensayos 22](#_Toc40565738)

[6 Conclusiones 22](#_Toc40565739)

[7 Proyecto finalizado 24](#_Toc40565740)

[8 Referencias 25](#_Toc40565741)

# Descripción general

El proyecto consiste en la implementación de un pedal multiefecto para guitarra eléctrica. El sistema será puesto en funcionamiento en una placa donde el usuario pueda interactuar con un potenciómetro simulando un pedal de expresión a través de distintos periféricos como botones o potenciómetros. Además, se informará el efecto seteado a partir de luces led.

El proyecto implementará hasta 3 efectos de audio, *Delay*, *Fuzz* y *Wah-Wah*. Pudiendo cambiar entre cada uno de ellos mediante un botón de forma secuencial, mientras que con un potenciómetro se variarán los parámetros característicos del efecto previamente elegido. El sistema será capaz de procesar en tiempo real la señal, aplicando un efecto.

El proyecto no implementará display para informar el efecto, el efecto seleccionado se informará mediante leds (los leds integrados en el board), un color por efecto.

El footswitch permitirá al usuario el efecto conocido como loopback que interconecta la entrada de señal con la salida de manera pasiva, cuando el pulsador este apagado.

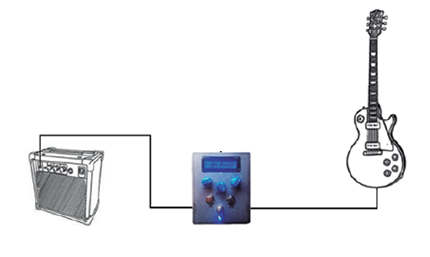


Figura 1

# Introducción teórica

En este apartado se definirán de forma teórica los efectos utilizados en el pedal y la manipulación matemática que sufre una señal de información para modificarla y obtener la señal esperada.

## 2.1 Delay

Como su propio nombre indica, el delay consiste sencillamente en reproducir la señal de entrada junto con una versión retrasada y atenuada de sí misma, simulando así un eco real. Si lo representamos con un diagrama de bloques, como indica la Figura 2.

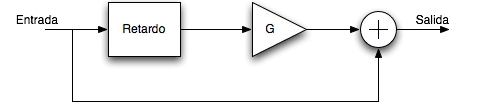


Figura 2

Los retrasos se pueden experimentar en espacios acústicos. Una onda de sonido reflejada por una pared se superpondrá a la onda de sonido en la fuente. Si la pared está lejos, tal como un acantilado, oiremos un eco. Si la pared está cerca de nosotros, notaremos los reflejos a través de una modificación del color del sonido.

Pueden aparecer reflejos repetidos entre límites paralelos. En una habitación, tales reflexiones serán un eco más plano. La distancia entre los límites determina el retraso que se impone a cada una de la onda de sonido reflejada. En un cilindro, se desarrollarán reflexiones sucesivas en ambos extremos. Si el cilindro es largo, escucharemos un patrón iterativo, mientras que, si el cilindro es corto, oiremos un tono agudo.

Los equivalentes de estos fenómenos acústicos se han implementado como unidades de procesamiento de señales. La variación del tono se debe al hecho de que la distancia entre la fuente y nuestros oídos está siendo variada al variar la distancia, para nuestra aplicación, sería el equivalente a variar el retraso del tiempo. Si seguimos variando periódicamente el tiempo de retraso produciremos una variación periódica del tono. Esto es sencillamente un efecto de vibrato.

### 2.1.1 Estructura básica de un Delay

La red que simula un solo retraso se llama Comb Filter[[1]](#footnote-1) (filtro de peine FIR). La entrada

la señal se retrasa por una duración de tiempo dada. El efecto será audible solo cuando

la señal procesada se combina (agrega) a la señal de entrada, que actúa aquí como

una referencia. Este efecto tiene 2 parámetros de ajuste: la cantidad de tiempo de retraso “T” y la amplitud relativa de la señal retrasada a la de la señal de referencia.

La ecuación de diferencia y la función de transferencia están dadas por:

Figura 3

Con:

Figura 4

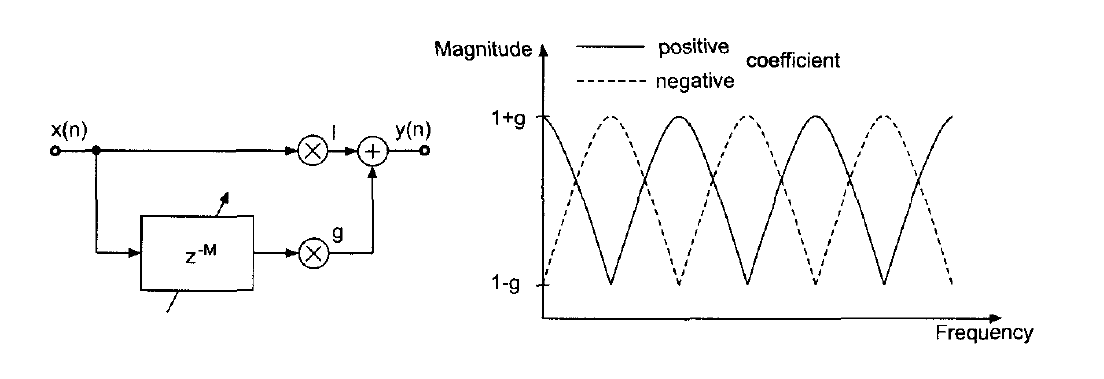


Figura 5

## 2.2 Fuzz

El efecto fuzz básicamente aplica sobre la señal un recorte (clipping) como el que haría un amplificador de válvulas extremadamente saturado. Los picos de la señal de entrada quedan recortados, y los valles aplanados, generando el característico sonido distorsionado.

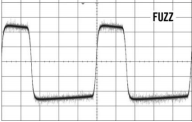
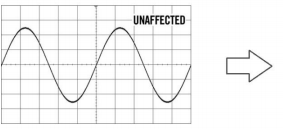


Figura 6

El fuzz es un overdrive [[2]](#footnote-2)exagerado, con una mayor distorsión de la onda que hace que existan más picos en los armónicos. Esto genera un sonido más grueso y denso que se focaliza en las frecuencias bajas y medias, pero no tanto en las más agudas (un fuzz es más grave y menos chillón que una distorsión). Sin duda es un sonido menos nítido y más sobrecargado. De hecho, muchos guitarristas, con tendencia a un sonido más definido, huyen de este efecto, mientras que otros lo prefieren.

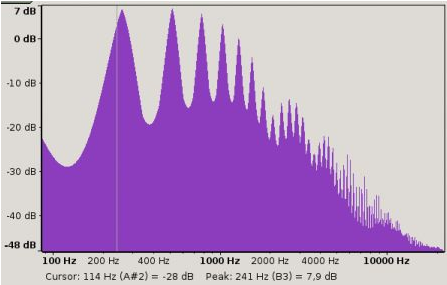


Figura 7

Procesamiento de la señal **Symmetrical Soft Clipping** (Recorte suave simétrico): para simulaciones de sobremarcha, se debe realizar un recorte suave asimétrico de los valores de entrada. Un posible enfoque para una no linealidad de saturación suave viene dado por:

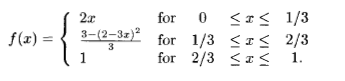


Figura 8

La relación entrada / salida estática se muestra en la figura 8 hasta el umbral de 1/3, la entrada se multiplica por dos y la curva característica se encuentra en su región lineal. entre valores de entrada de 1/3 hasta 2/3, la curva característica produce una compresión suave descrita en el término medio de la ecuación. Por encima de los valores de entrada de 2/3, el valor de salida se establece en uno.

La figura 9 muestra la forma de onda de una simulación con la curva característica descrita anteriormente y una sinusoide de 1Khz.

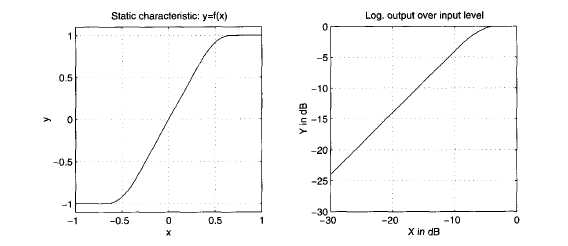


Figura 9

## 2.3 Wah-Wah

Un procesador de señales con control de pie que contiene un paso de banda. Filtro con frecuencia central variable. Moviendo el pedal hacia adelante y hacia atrás cambia la frecuencia central del paso de banda.

El efecto wah-wah es producido principalmente por procesadores de señal controlados por el pie que contienen un filtro de banda ancha con frecuencia de resonancia central variable y un ancho de banda pequeño. Al mover el pedal hacia adelante y hacia atrás cambia la frecuencia de corte del centro de paso de banda. El efecto "wah-wah" se mezcla con la señal directa como se muestra en la figura 10.

Este efecto conduce a una formación del espectro similar al habla y produce un discurso como el sonido "wah-wah".

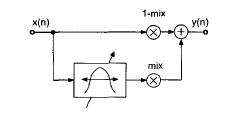


Figura 10 WahWah: Filtro pasa banda variable en el tiempo

Si la variación de la frecuencia central es controlada por la señal de entrada, se usa un oscilador de baja frecuencia para cambiar la frecuencia central. Tal efecto se llama filtro auto-wah. Si el efecto se combina con una variación de amplitud de baja frecuencia, que produce un trémolo, el efecto se denomina filtro de tremolo-wah. Reemplazar el retardo de la unidad en el filtro de paso de banda por un M tap delay lleva al filtro M-fold wah-wah, que se muestra en la figura 8. Los filtros de paso de banda M son

se extienden por todo el espectro y cambian simultáneamente su frecuencia central.

Cuando se aplica una señal de entrada de ruido blanco a un filtro wah-wah M-fold, un espectrograma de la señal de salida que se muestra en la figura 2.26 ilustra la mejora periódica del espectro de salida. La Tabla 2.5 contiene varias configuraciones de parámetros para diferentes efectos.



Figura 11: Filtro Wah-Wah

# Descripción técnica del proyecto.

## 3.1 Hardware

En esta sección se detallará el funcionamiento interno de los protocolos y drivers implementados para el proyecto, dado que no fue necesario la inclusión de hardware adicional, ya que con la placa STM32F411 se ha podido cubrir todas las necesidades asociadas al proyecto. Con la salvedad de un filtro pasa bajo pasivo de orden 2.

### 3.1.1 Protocolo I2S

Multimedia -> I2s3 -> GPIO settings

Para que se escuche un sonido libre de glitches, es necesario que el I2S (para el DAC) y el ADC trabajen a frecuencias lo más cercanas posible.

A continuación, se describirán los pines que necesita la comunicación I2S

**MCK:** (main clock) es el clock principal que contiene la mayor frecuencia, este es enviado por el master y recibido por el/los esclavos, en este caso el master es el micro STM32F411 y el esclavo es el CS43L22 (driver), su frecuencia es N\*WS; siendo N un numero estandarizado (ej 128, 256, 512) *Ver sección 7.4.2, página 39 manual*.

**SCK**: (system clock) es el BitClock (menor frecuencia que el primero) frecuencia de C/Bit, su frecuencia es Fs\*(#de bits por canal) \*2.

**WS:** también se lo llama LRCK (left right clock) este sería el Fs que se menciona en el "SCK", cuando está en estado alto se realiza la conversión del canal izquierdo y cuando está en estado bajo en el canal derecho.

**SD:** (Serial Control Data) Son los datos.

Para comunicarse con el DAC "CS43L22", debe generarle el Master Clock para que pueda filtrar correctamente ya que se trata de un DAC delta-sigma.

Según la hoja de datos, cuando el Master Clock del I2S es generado, La Fs se calcula con la siguiente fórmula:

FS = I2SxCLK / ((32\*2) \*((2\*DIV) +ODD) \*4)

Por otro lado, I2SxCLK se calcula como:

I2SxCLK = 1MHz \* PLLI2SN / PLLI2SR

Para una Fs=32KHz recomienda:

- I2SxCLK=106.5MHz (PLLI2SN=213 PLLI2SR=2)

- DIV=6

- ODD=1

Este seteo genera una Fs=32001.20117Hz (con un error del 0.0038% respecto a los 32KHz)

### 3.1.2 Driver CS43L22

Connectivity -> I2C1

Necesita primero de I2C para configurar frecuencia, canales, volumen, mute, etc (está en el manual), luego se comunica por I2S para enviar los datos.

* **I2S**

- El clock del periférico I2S es 66.666667 MHz.

- Ese clock es dividido x 8 en un divisor interno dando 8.333333 MHz que es utilizada como el MCK.

- La velocidad (MCLK/LRCK Ratio) está seteada a 256, lo que quiere decir que MCLK/LRCK = 256 => LRCK = 32.552 KHz (para adaptarse al estándar de 32KHz).

(LRCK = WS)

- como el I2S está configurado para 16 bits, SCK = WS\*16 => SCK = 520.833 KHz

* **ADC**

- El ADC utiliza PCLK como fuente de clock. El PLCK está seteado a 12.5MHz

- Divisor interno = PCLK2/2

- 12 bits

- 2 canales (para el audio y el pote)

- Sampling time = 84 ciclos.

- Todo eso da una Frecuencia de sampling de:

fs = (CLK/(divisor\*canales)) / (bits+SamplingTime) = 32.552KHz

De esta manera ambos trabajan a la misma frecuencia

La hoja de datos define el periodo de sampleo como:

Tconv = (Sampling\_time + resolution) [ADCCLK cycles]

siendo ADCCLK configurable mediante el STM32CubeMX

Teniendo en cuenta que se dispone de 1 solo ADC y se requieren medir 2 entradas (ADC y potenciómetro), se deben configurar 2 canales para medir secuencialmente.

La configuración más cercana al Fs del I2S es la siguiente:

- ADCCLK = 1,34375 [MHz] (PLCK2 = 8.0625 [MHz], Prescaler\_ADC = 6)

- Resolución = 12 [bits]

- Sampling\_time (ADC) = 15 [ciclos]

- Sampling\_time (potenciómetro) = 3 [ciclos]

Tconv (ADC) = 15 + 12 = 27 [cycles]

Tconv (potenciómetro) = 3 + 12 = 15 [cycles]

Fs = ADCCLK / (Sampling\_time (ADC) + Sampling\_time (potenciómetro))

Fs = 1,34375 [MHz] / (27 + 15)

Fs = 31.994,04761

Error = 0,0224% (respecto a la Fs del I2S)

Después como el error no era aceptable lo dispara a través de un timer de 96 MHz y periodo de 1500 cuentas

96Mhz/1500= 64khz (porque son 2 canales. izquierda y derecha, es decir, no es de potenciómetro y audio, porque se miden juntos)

error=Fs=1.20117Hz /32k= 0,0037% (una décima parte de error de lo que tenía antes)

**Clock Configurado:**

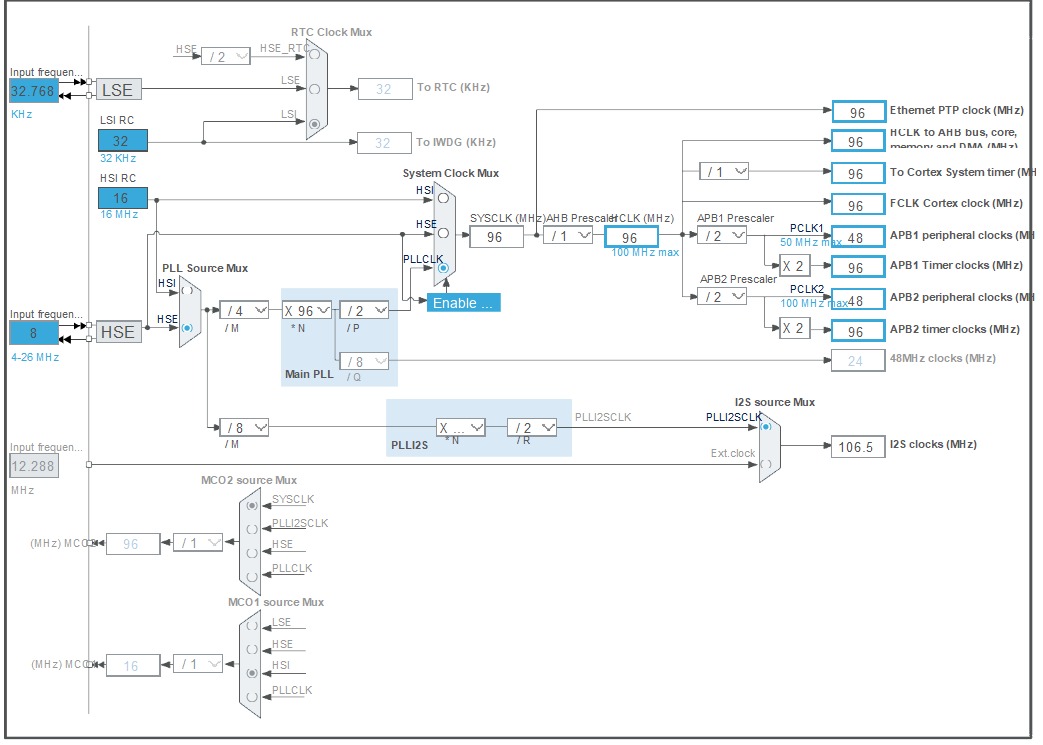


Figura 12

**Pulsadores para el usuario:**

*PA0:* (Pulsador interno para modificar efecto)

*PD9:*(Pulsador Interno para hacer un reset)

### 3.1.3 ADC

*PA3*: (Control de expresión de efectos)

*PB0*: (Entrada de Audio)

#### 3.1.3.1 Configuración con el Códec de DAC

*I2C1\_SCL:* PB6

*I2C1\_SDA:* PB9

#### 3.1.3.2 Transmisión de audio al códec del DAC

*I2S3\_WS***:** PA4

I2S3\_*MCK***:** PC7

I2S3\_SCK**:** PC10

I2S3\_SD: PC12

### 3.1.4 Diagrama en bloques

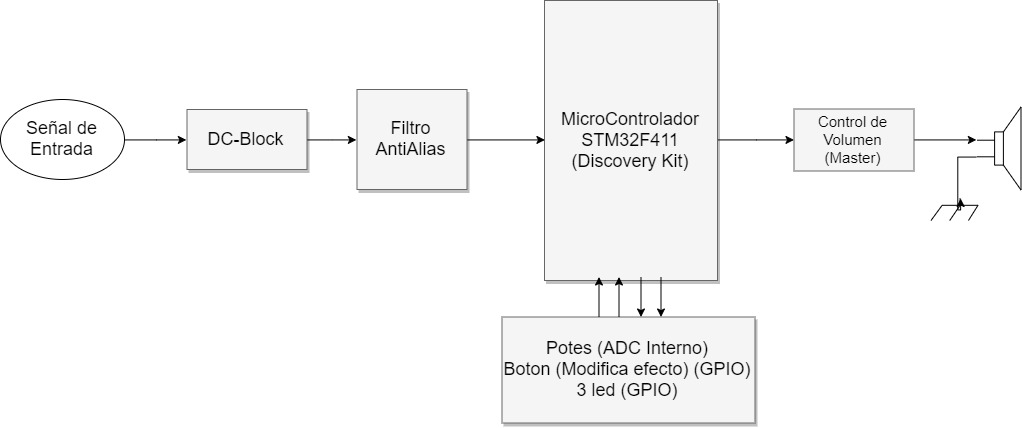


Figura 13

El conjunto de este diagrama en bloques, explica el funcionamiento de un pedal multiefecto. El mismo tiene una adaptación en la señal de entrada para poder ingresar al ADC del microcontrolador, seguido de un filtro AntiAlias, para eliminar el contenido armónico no deseado. El microcontrolador es el encargado de realizar el procesamiento digital de la señal, agregando los efectos (LoopBack, Delay, Fuzz y Wah-wah).

### 3.1.5 Esquemático

En esta sección se hará la descripción de forma separada de cada uno de los bloques que conforman el sistema.

#### 3.1.5.1 Jack

Comenzaremos con la descripción de este conector, ss el conector de audio analógico​ de señales analógicas, que se utiliza para Inyectar la señal de entrada. La misma es un conector monocanal dado que la aplicación es para instrumentos.

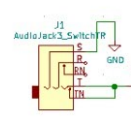


Figura 14

#### 3.1.5.2 Divisor resistivo

Este divisor se implementó para que el punto medio de la señal inyectada esté en 0 (en el centro), además se incluyó un capacitor para bloquear la componente continua de la señal.

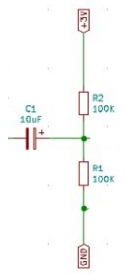


Figura 15

#### **3.1.5.3** Potenciómetro

Este potenciómetro se implementó para realizar las expresiones de los efectos.

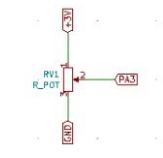


Figura 16

#### 3.1.5.4 Filtro Pasa Banda

Se experimentó la necesidad de incluir un filtro pasa bajo de orden 2, con frecuencia de corte en 8khz, dado que el ADC interno esta seteado a 32Khz, al tener este valor la performance del audio no es buena, incluyendo un nivel de ruido notable.

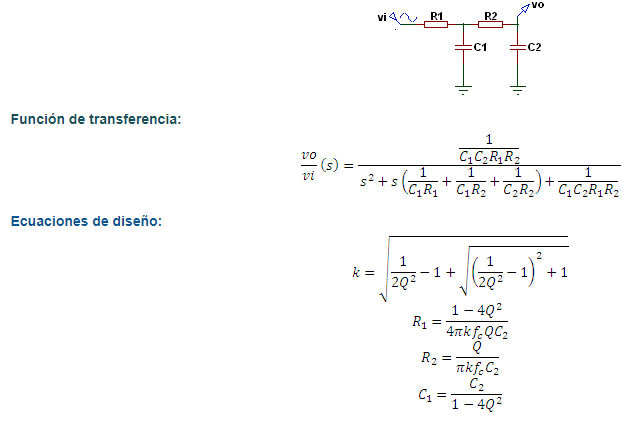


Figura 17

**Calculo del Filtro pasa bajos Pasivo:**

Frecuencia de corte Fc= 8 Khz  
Factor de calidad Q= 0.33

**Componentes del filtro Pasa bajos:**

R1= 6288 [Ω]  
R2=4853 [Ω]  
C1= 1.77 [nF]  
C2= 1 [nF]

### 3.1.6 Circuito impreso

Este proyecto incluye un hardware adicional muy sencillo ya que solo integra el Jack para inyectar la señal de entrada, un filtro pasa bajos y un potenciómetro oficiando como pedal de expresión.

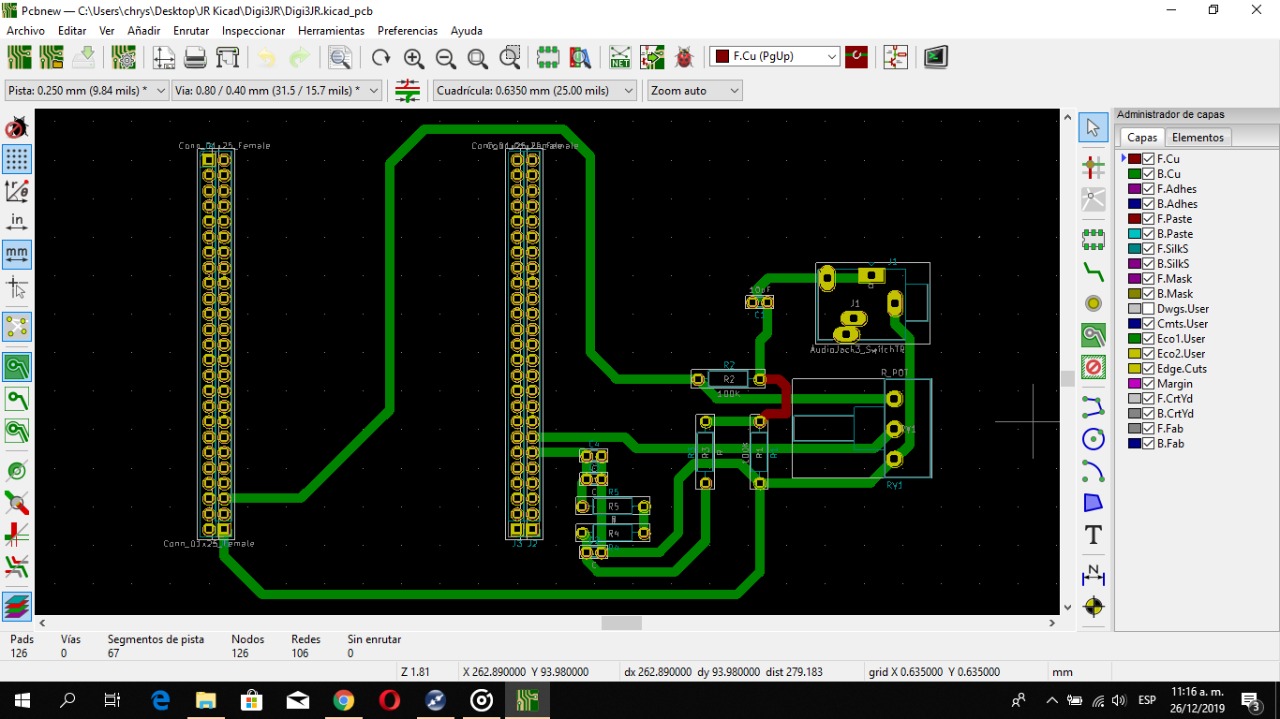


Figura 18

### 3.1.7 Fotos

Modelo 3D:

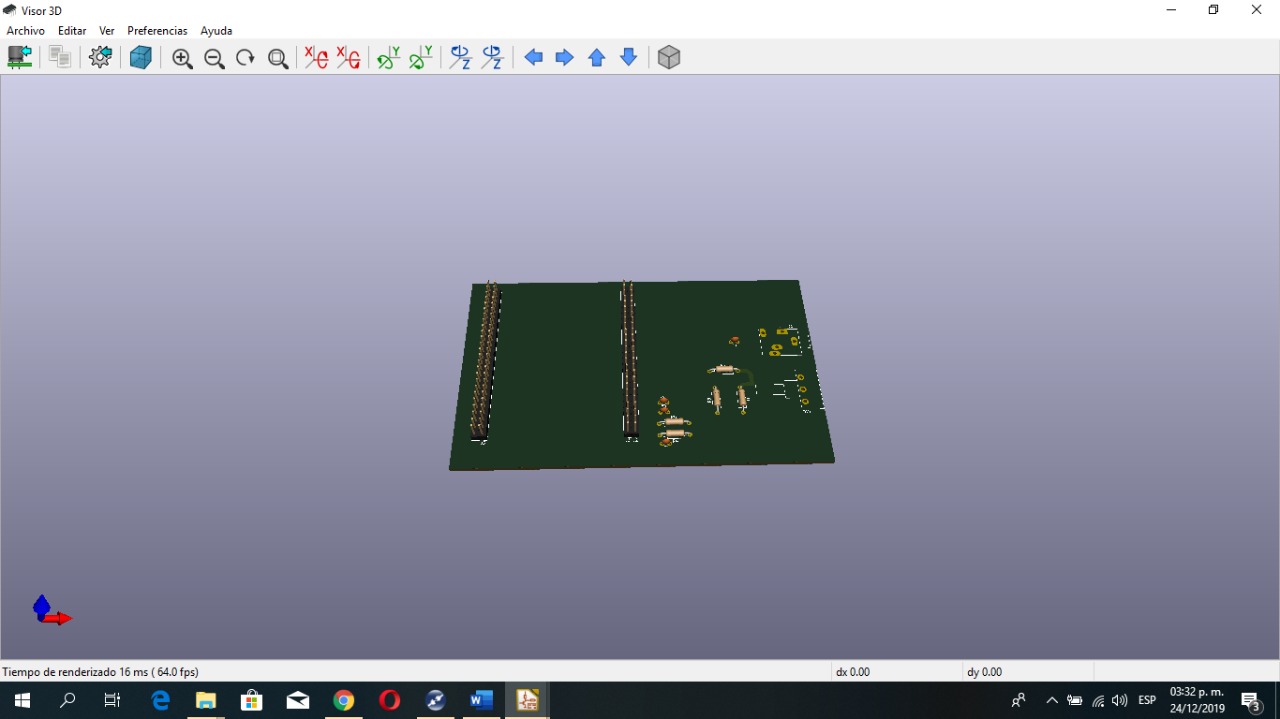


Figura 19

## 3.2 Software

En esta sección se deberá realizar la descripción del funcionamiento del software corriendo en el microcontrolador. Téngase presente que esta es la sección más importante del informe, es por esto que se desea una descripción lo más detallada posible.

No se deben pegar tramos de código en este informe, la descripción de las distintas funciones del código se debe realizar a través de diagramas de estados finitos y/o diagramas de flujo. El código propiamente dicho debe adjuntarse en el anexo y debe encontrarse comentado en forma prolija y detallada para facilitar la corrección del mismo. A continuación, se observan los subtítulos a incluir en este desarrollo.

### 3.2.1 Enumeración de rutinas

En esta sección simplemente se desea que se enumeren todas las rutinas incluidas en el programa y la función que cumplen en el mismo en forma resumida. Por ejemplo:

* Rutina “main”:
* Subrutina “HAL\_Init ();”
* Subrutina “MX\_GPIO\_Init();”
* Subrutina “MX\_DMA\_Init();”
* Subrutina “MX\_I2C1\_Init();”
* Subrutina “MX\_I2S3\_Init();”
* Subrutina “MX\_TIM2\_Init();”
* Subrutina “MX\_ADC1\_Init();”
* Subrutina “MX\_TIM2\_Init();”
* Subrutina “main\_loop ();
* Subrutina “FSM ();”
* Subrutina “main\_loop\_start ();”
* Subrutina “main\_loop\_end ();”
* Subrutina “void main\_loopback ()”
* Subrutina “void main\_eco ()”
* Subrutina “float fuzz(float, float)”
* Subrutina “void wahwah (float, float, int)”

### 3.2.2 Descripción del funcionamiento de cada rutina

En esta sección se debe realizar la descripción del funcionamiento de las distintas rutinas a través de diagramas de estados finitos y/o diagramas de flujo (siendo de preferencia la primera forma). No es necesaria en esta sección la presentación de un diagrama para cada rutina, para aquellas rutinas que sean triviales (por ejemplo: la inicialización de un periférico o una subrutina de interrupción que solo actualice contadores o flags, etc.) pueden explicarse sus acciones resumidamente en forma escrita.

* Rutina “main”: < Descripción >.

En el main\_init inicializo y disparo los periféricos necesarios para este proyecto (timer2, timer3, DAC CS43L22, DMA para I2S y ADC, inicializo Buffers y filtros)

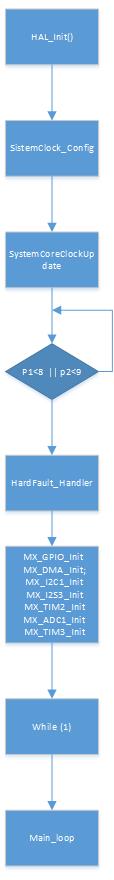


Figura 20

* Subrutina “HAL\_Init();”

Esta subrutina restablece todos los periféricos, inicializa la interfaz de Flash y el Systick. (Systick se utilizará para generar retrasos).

* Subrutina “MX\_GPIO \_Init();”

Inicializa las entradas digitales de entrada y salida de proposito general (boton y leds)

* Subrutina “MX\_DMA\_Init();”:

inicializa DMA para comunicar con ADC y I2S

* Subrutina “MX\_I2C1\_Init();”

Inicializa I2C para configurar DAC

* Subrutina “MX\_I2S3\_Init();”

Inicializa I2S para enviar audio al DAC

* Subrutina “MX\_TIM2\_Init();”

Inicializa timer para el blink del led “I'm alive”

* Subrutina “MX\_ADC1\_Init();”

Inicializa ADC

* Subrutina “main\_loop ();”

Dentro de main Loop esta Finite State Machine (FSM) con los 4 efectos 2 estados por efecto, uno para cuando se pulsa el botón y el otro cuando se suelta

* Subrutina “FSM ();”:

Reemplaza tener que copiar toda la máquina de estado

* Subrutina “main\_loop\_start ();”
* Subrutina “main\_loop\_end ();”

Realizo cálculos con el valor del potenciómetro para obtener el número de “paso” necesario para el efecto Delay.

Estados del DMA, se usa para cuando ocurre la colisión de los clock.

* DMA\_INIT: es el estado inicial
* DMA\_ADC\_READY: cambia este estado cada vez que termina de convertir el ADC
* DMA\_LOGIC\_BUSY: Cambia este estado cuando comienza la lógica del programa, este estado permite detectar cuando el tiempo de procesamiento sea insuficiente
* DMA\_DAC\_READY: Cuando termina de enviar los datos por I2S

El tiempo de procesamiento queda confinado entre DMA\_ADC\_READY y DMA\_DAC\_READY

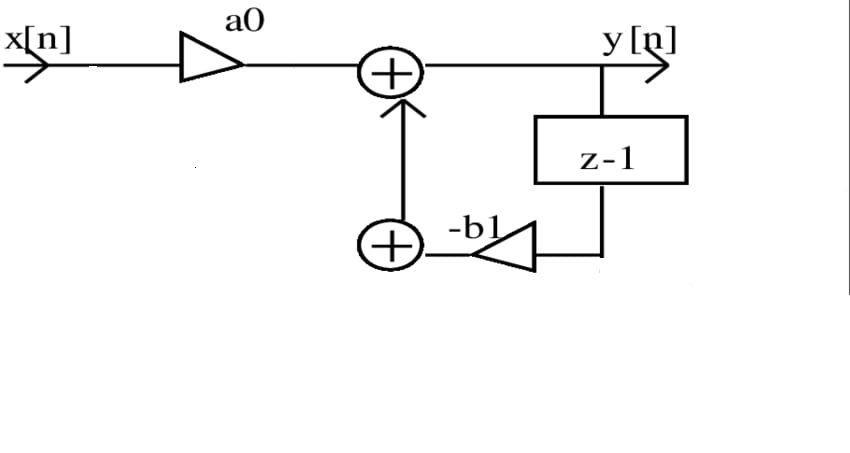
Si el tiempo de procesamiento (tiempo entre LOGIC\_BUSY y LOGIC\_READY) es mayor, en ese momento, se produjo la colisión entre las dos frecuencias de sampling la del ADC y la del I2S. En ese caso se debe omitir la transmisión de bloques para ADC y el I2S hasta que el tiempo de procesamiento vuelva a la normalidad.

* Subrutina “void main\_loopback ()”

Ya que el ADC convierte 2 canales en el buffer del DMA van a llegar las muestras de ambos canales intercalados por los tanto hay que hacer un procesamiento para separarlos y poder procesarlos y enviarlos al i2s. el loopback solo se encarga de realizar este manejo

* Subrutina “void main\_eco ()”

Figura 21



Partiendo de un vector llamado all\_in de N muestras. Voy llenando las posiciones de ese vector según van llegando las muestras, entonces tengo un puntero a la posición actual.

Ejemplo:

all\_in:

[1]

[2]

[3]

[4] <- actual

[5]

...

[N]

El puntero "actual" va a ir avanzando hacia abajo conforme vayan llegando nuevas muestras.

Luego tengo otro puntero que apunta al mismo vector pero unas posiciones más atrasadas.

Ej:

all\_in:

[1]

[2] <-atrasado

[3]

[4] <- actual

[5]

...

[N]

Luego yo voy a ir sacando las muestras al DAC sumando "actual"+"atrasado" que es lo que le da el efecto de delay (eco). El puntero "actual" y "atrasado" se van a ir moviendo a la misma velocidad para q la frecuencia del eco no se vea afectada, y la distancia entre el puntero "actual" y "atrasado" es lo que le da el tiempo de retardo. El máximo retardo se da cuando "atrasado" está en la posición más atrasada en el vector;

all\_in:

[1]

[2]

[3]

[4] <- actual

[5] <- atrasado

...

[N]

En ese caso, el retraso es de (N-1)/fs segundos

Retraso= Tam Vector/Fs

* Subrutina “float fuzz(float, float)”

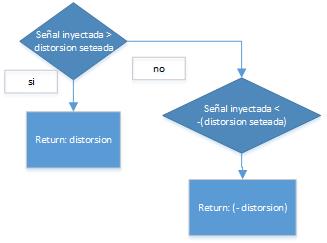


Figura 22

Cuando el nivel de la señal inyectada es mayor al nivel seteado en el potenciómetro (distorsión seteada) devuelve el valor del nivel seteado con el potenciómetro, en caso contrario, si el nivel de la señal inyectada es menor al nivel de la distorsión seteada por el potenciómetro negada, devuelve el valor de la distorsión negada. En este proyecto omitió el comportamiento de la curva alineal (mostrado en la introducción teoría) por la dificultad de implementación. La característica del efecto mencionado se implementó como un fuzz con distorsión por recorte.

* Subrutina “void wahwah (float, float, int)”:

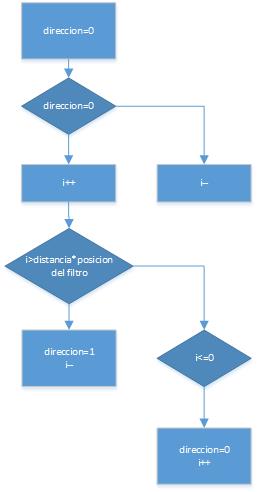


Figura 23

En este proyecto el efecto citado se ha implementado utilizando 20 filtros pasa banda, cada uno de estos filtros pasa banda tiene una frecuencia central distinta, entonces conforme sucede el tiempo se recorren los filtros (de derecha a izquierda, es decir, cuando dirección=0 y viceversa) se activa y desactivan los filtros en forma secuencial. Con el potenciómetro fijo la velocidad de variación de activación y desactivación de los filtros, percibiendo de esta forma el efecto deseado.

# Modo de operación

El modo de operación es muy sencillo, en este producto podemos encontrar tres efectos para el usuario (Delay, Fuzz y WahWah) como mencionamos al principio. El usuario se informará del efecto que esta seteado a partir de los Leds que incluye el board de la placa con la siguiente estructura.

* LoopBack -> **Verde**
* Delay (eco) -> **Naranja**
* Fuzz -> **Azul**
* Wah-Wah -> **Rojo**

Podemos navegar por estos efectos a través del botón incluido en el board de la placa (de color azul) para elegirlos de forma secuencial, con el botón de color negro que también está incluido el board, reseteamos y volvemos al By Pass.

El desarrollo incluye un potenciómetro, para manejar la expresión de los efectos.

# Ensayos

Para este proyecto no se realizaron ensayos de mediciones ni de ningún tipo.

# Conclusiones

Entre las experiencias relevadas en el desarrollo del proyecto puedo mencionar, la intención de implementar el Audio Weaver (software de desarrollo que permite de forma más rápida agregar funciones de audio con una modalidad de diagrama en bloques).

## 6.1 Audio Weaver

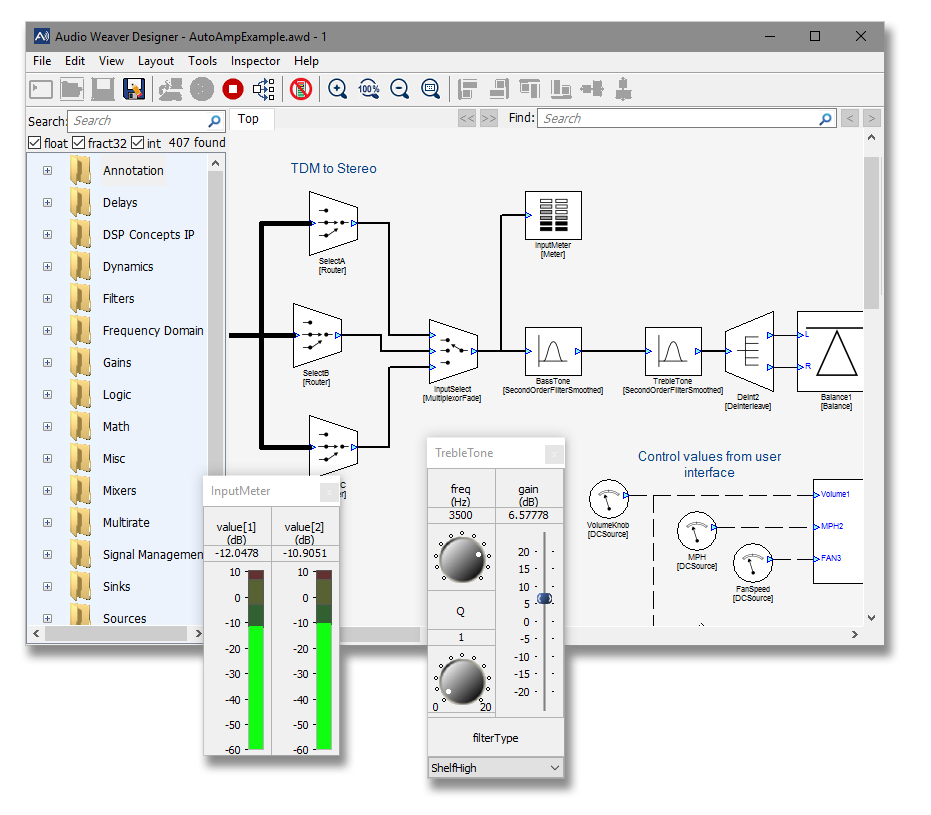


Figura 24

Software de desarrollo que permite de forma más rápida agregar funciones de audio con una modalidad de diagrama en bloques.

Tras haber investigado sobre el software por recomendación de la cátedra. A modo de comienzo para la familiarización de la herramienta, pude realizar a través de una conexión USB, la implementación de algunos efectos, como, por ejemplo, delay, el limitante que me encontré con esta herramienta fue la imposibilidad de “bajar” el código que genera el Audio Weaver a la placa, para poder implementar un Pedal multiefecto completamente autónomo como requiere la catedra, a causa de la dependencia de la conexión USB.

Por este motivo me vi obligado a desistir en la implementación de la herramienta sugerida, Adicionalmente, en página oficial del proveedor (que se puede visitar a través de este [LINK](https://dspconcepts.com/solutions/audio-weaver)), no explica ni provee información adicional para la implementación que requiero.

Además, me vi imposibilitado de implementar un Control de Master Volumen entre el DAC y el Jack de la Placa, dada la limitación física de no poder añadirlo ya que ambos elementos pertenecen al Board de la placa. Por lo tanto, implementé el Footswitch por Software para una simplificación del proyecto.

Otra dificultad sorteada en el proyecto fue la necesidad de incluir un filtro Antialias sobre exigido (Frecuencia de corte: 8[KHz]), para eliminar el contenido armónico no deseado, dado que encontré que la performance del ADC que contiene el board de la placa no es lo suficientemente exigente como para realizar el procesamiento de una señal de audio y percibirla con una fidelidad considerable.

# Proyecto finalizado



Figura 25



Figura 26

# Referencias

Listado enumerado en el cual se detallen todos los documentos de los cuales se han auxiliado los alumnos para realizar el proyecto (por ejemplo, libros, notas de aplicación, hojas de datos, etc.). Por ejemplo:

1. Hoja de dato: CS43L22–Low Power, Stereo DAC w/Headphone & Speak Amps– Cirrus Logic.
2. Libro: Digital Audio Effects – Autor: Udo Zolzer – Primera edición - Editorial: John Wiley& Sons.
3. Link: DSP Concepts–Audio Weaver

<https://dspconcepts.com/solutions/audio-weaver>

1. Comb Filter: se produce al sumarle a la señal original una versión retrasada en el tiempo de sí misma, causando así [interferencia](https://es.wikipedia.org/wiki/Interferencia) [constructiva](https://es.wikipedia.org/wiki/Interferencia_constructiva) y [destructiva](https://es.wikipedia.org/wiki/Interferencia_destructiva). [↑](#footnote-ref-1)
2. Efecto empleado para guitarra eléctrica [↑](#footnote-ref-2)