UNIVERSIDAD PONTIFICIA COMILLAS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Ingeniero Industrial

PROYECTO FIN DE CARRERA

ESTUDIO DE EFECTOS DE AUDIO PARA GUITARRA, E IMPLANTACIÓN MEDIANTE DSP.

Director: **Cesáreo Fernández Martínez.**
Autor: **Ángel Pérez Rodríguez.**

Feb/2006

ÍNDICE DE MATERIAS:

- 1. Introducción o motivación del proyecto.

La idea del trabajo nace de la gran importancia y actualidad que están cobrando en los últimos años las técnicas de procesamiento digital de señales (técnicas DSP en inglés) ; técnicas que si bien no son excesivamente recientes (los primeros estudios datan de los años 50 y 60), su utilidad sólo pudo hacerse manifiesta a raíz de la aparición en el mercado de procesadores digitales lo suficientemente rápidos y potentes para poder implementar los algoritmos de tratamiento de señales de forma económica y eficazmente rentable. Estos procesadores digitales de señal comenzaron a aparecer en la década de los 80 y reciben el nombre de procesadores digitales de señal o DSPs(Digital Signal Processors en inglés).

De esta manera y en el contexto general anteriormente citado, el presente trabajo centra toda su atención, de entre la extensa gama de posibilidades y aplicaciones posibles, sobre las técnicas de procesamiento digital de señales de audio en tiempo real para la obtención de los efectos más frecuentes con los que modificar el sonido original de un instrumento musical, todo ello con el objetivo de proporcionar al músico un potente vehículo para la expansión de sus capacidades expresivas.

Debe señalarse pues, haciendo un poco de historia, que la búsqueda de nuevos sonidos ha sido una constante a lo largo de los años. Los músicos en su afán por innovar y conseguir nuevos estilos y sonidos con los que marcar su impronta personal, siempre han echado mano de todo el ingenio posible para primero idear el efecto buscado, y después fabricar el dispositivo que les permitiera obtenerlo, siendo las técnicas tradicionalmente usadas puramente analógicas (recuérdese el sonido característico de las distorsiones empleadas en los amplificadores de válvulas , el sonido característico del famoso pedal Fuzz de Jimi Hendrix o los primitivos efectos de Flangers retrasando grabaciones usados por los Beatles en su aclamado álbum blanco).

Pues bien, es la idea motivadora central del presente trabajo, la de mostrar las posibilidades de las citadas técnicas de procesado digital de señales de audio para guitarra, como una solución versátil y potente al problema creativo de la obtención de efectos, así como proporcionar a dicho campo, un rigor ingenieril suficiente.

- 2. Objetivos del proyecto.

- **Estudio general teórico** de los distintos efectos de sonido para guitarra.

- **Modelado y descripción** de dichos efectos en el campo del tratamiento digital de señales.

- **Programación** de los modelos de los efectos de audio en entorno y lenguaje de programación **Matlab**.

- **Programación** de los efectos mediante el uso de **lenguaje C para microprocesadores** con el objetivo de proporcionar un código estructurado para su uso con microprocesadores digitales de señal DSP.

- **Implementación** de los algoritmos en la plataforma **DSP** concreta de Analog Devices **ADSP 21000** y comprobación de resultados.

-3. RESUMEN.

Comenzaremos este apartado exponiendo los principios teóricos básicos, tanto a nivel del mecanismo auditivo humano como entrando ya en más detalle en el campo del sonido digital.

Se comentarán cuestiones generales sobre audio y audio digital, posteriormente se hablará de la tecnología DSP de forma general para pasar a la descripción de los efectos en el marco del procesado digital de señales.Tras este estudio, se profundizará en el análisis de las simulaciones efectuadas en Matlab y se expondrán sus resultados así como las consideraciones a las mismas oportunas, así como los resultados obtenidos en la implementación definitiva sobre el microprocesador.

3.1.PRINCIPIOS DEL MECANISMO AUDITIVO HUMANO.

El sonido es una vibración que se propaga a través del aire, gracias a que las moléculas del aire transmiten dicha vibración hasta que llega a nuestros oídos. Se aplican los mismos principios que cuando se lanza una piedra a un estanque: la perturbación de la piedra provoca que el agua se agite en todas las direcciones hasta que la amplitud de las ondas es tan pequeña, que dejan de percibiese. La Figura-1 muestra las vibraciones físicas de un diapasón que ha sido golpeado. Las vibraciones del diapasón fuerzan a las moléculas de aire a agruparse en regiones de mayor y menor densidad, dando lugar a que la presión del aire aumente o disminuya instantáneamente. El diapasón es un excelente ejemplo de fuente de sonido por dos razones: la primera es que puede observarse el movimiento de vaivén de sus brazos mientras se escuchan los resultados de esta vibración; la segunda es que el diapasón vibra a ***una frecuencia*** (vibraciones por segundo) constante hasta que toda su energía se ha disipado en forma de sonido. Una perturbación que viaja a través del aire se denomina *onda* y la forma que adopta esta se conoce como ***forma de onda***.

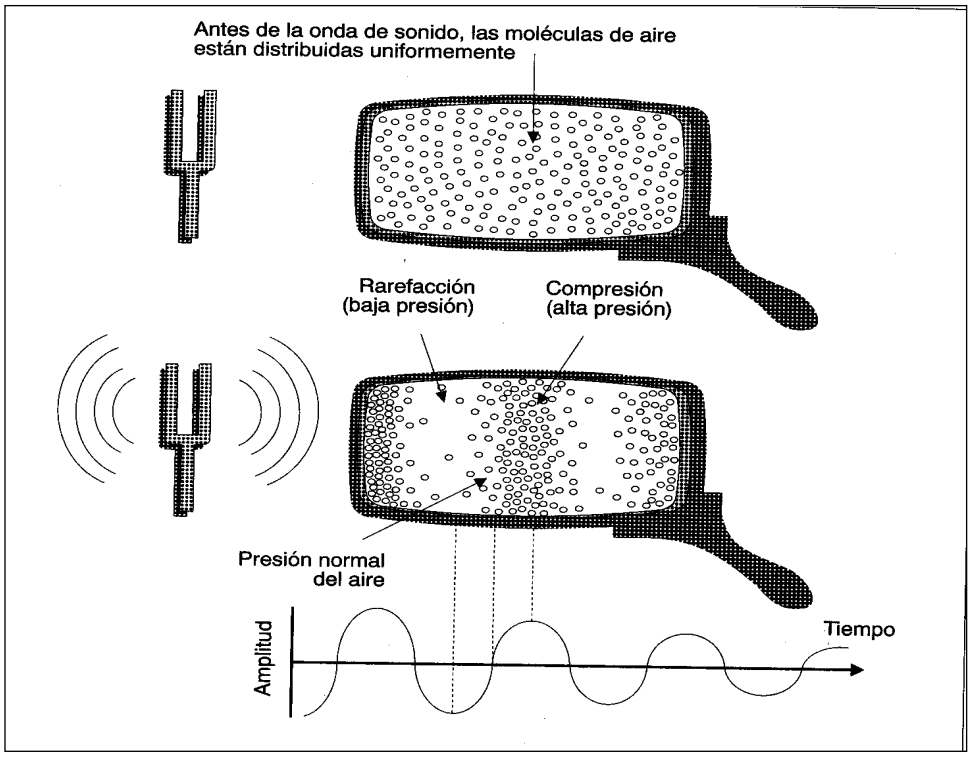


Figura-1 .Diapasón

El oído humano es un órgano tremendamente complejo. La figura 2 ilustra la estructura principal y la mayor parte de los procesos que tienen lugar en el oído humano.

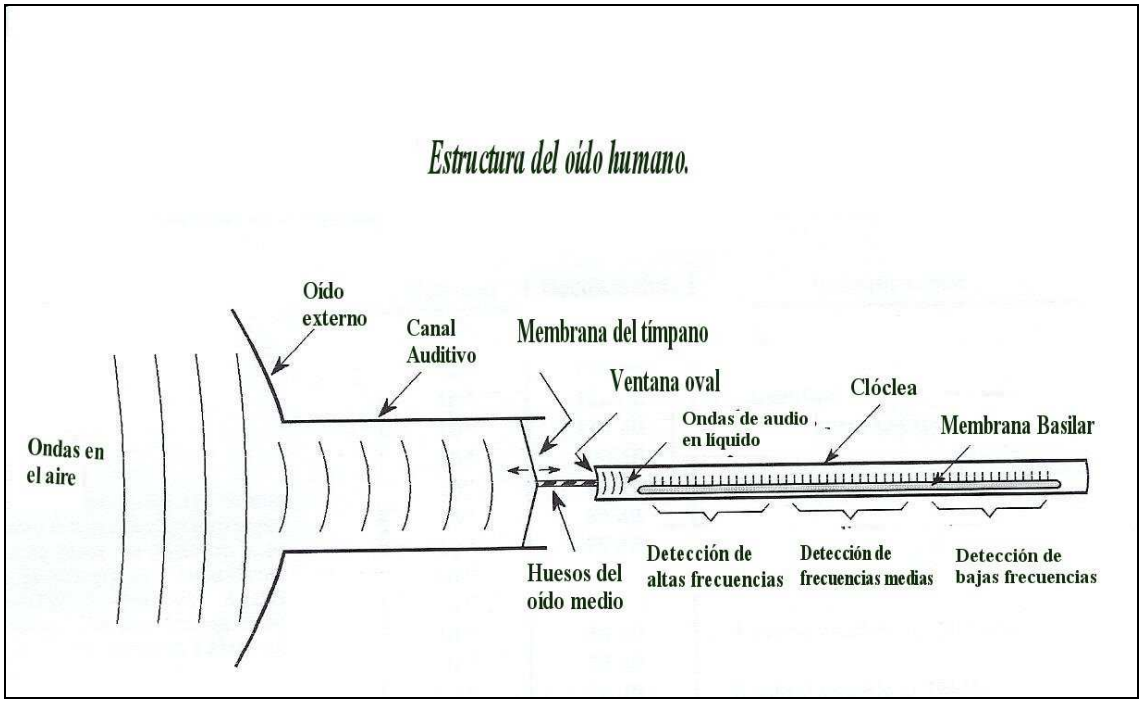


Figura 2 – Estructura del oído humano

El oído externo se compone de dos partes, la visible solapa de piel y el cartílago sujeto a la cabeza, y el canal auditivo, un tubo de aproximadamente 0.5cm de diámetro que penetra unos 3cm en la cabeza. Estas estructuras direccionan de forma directa los sonidos ambientes hacia los sensibles órganos del oído medio e interno localizados y bien protegidos en el interior de los huesos del cráneo. Al final del canal auditivo encontramos una delgada lámina de tejido que llamamos la membrana del tímpano o tambor del oído. Las ondas de sonido que llegan a dicha membrana la golpean haciéndola vibrar. El oído medio se compone de un conjunto de pequeños huesos que transfieren esta vibración a la clóclea (oído interno) donde son convertidas en impulsos neuronales. La clóclea es un tubo lleno de líquido de aproximadamente 2mm de diámetro y 3 cm de longitud . Aunque se muestra recta en la figura 1, la clóclea está arrollada y parece una pequeña concha de caracol. De hecho, la palabra clóclea deriva del la palabra griega caracol.

Cuando una onda de sonido trata de pasar dentro del líquido, solamente una pequeña fracción del sonido es transmitido a través del interfaz, mientras que la energía restante es reflejada. Esto es debido a que el aire tiene poca impedancia mecánica, mientras que el líquido tiene una alta impedancia mecánica. Lo que es lo mismo en términos menos técnicos, requiere más esfuerzo sacudir los brazos en el agua que en el aire. Esta diferencia de impedancias mecánicas produce que la mayoría de los sonidos sean reflejados en un interfaz aire/liquido.

El oído medio es una red adaptadora de impedancia que incrementa la fracción de energía sonora que entra en el líquido del oído interno. Por ejemplo, dígase aquí que los peces no tienen un oído medio pues no tienen la necesidad de oír en el aire. La conversión de impedancia es debida principalmente a la diferencia de área existente entre el oído medio (que recibe el sonido del aire) y la ventana oval(que transmite el sonido al líquido del oído interno; revisar figura 1). La membrana del tímpano tiene un área de aproximadamente 60 mm^2, mientras que la ventana oval tiene un área de unos 4mm^2 . Ya que la presión es igual a la fuerza divida del área , esta diferencia de área incrementa la presión de la onda de sonido unas 15 veces.

Contenida dentro de la clóclea se encuentra la membrana basilar, estructura compuesta por al menos 12.000 células sensoras que constituyen el nervio cloclear. La membrana basilar es más rígida por la parte más cercana a la ventana oval y más flexible a medida que se avanza en dirección opuesta a ella . Es esta característica la que le permite actuar como un analizador espectral de frecuencia. Así, cuando se encuentra expuesta a una señal de alta frecuencia , la membrana basilar resuena donde es más rígida, produciendo la excitación de las células nerviosas más cercanas a la ventana oval . Del mismo modo y como cabe esperar, las bajas frecuencias excitan las células nerviosas del otro extremo de la membrana . Es por ello por lo que nervios específicos del nervio cloclear responden a frecuencias específicas . Esta organización es lo que se denomina **principio de emplazamiento** y es preservado a través de todo el camino auditivo dentro del cerebro.

Otro esquema de codificación para la información sonora es también usado en el mecanismo de audición humano, denominado **principio de descarga**. Así, las células nerviosas transmiten la información generando breves pulsos eléctricos denominados

potenciales activos. Una célula nerviosa de la membrana basilar puede codificar información auditiva produciendo un potencial activo en respuesta a cada ciclo de vibración . Por ejemplo, un sonido de 200 Hertzios puede ser representado por una neurona mediante la producción de de 200 potenciales activos por segundo. De cualquier forma, esto solamente funciona para frecuencias por debajo de aproximadamente 500 Hertzios, tasa máxima a la que las células nerviosas pueden producir potenciales activos. El oído humano soluciona este problema disponiendo varias células nerviosas para realizar esta sencilla tarea por turnos. Por ejemplo, un tono de 3000 Hertzios deberá ser representado por diez células nerviosas disparando alternativamente 300 veces por segundo. Esto extiendo el rango del principio de descarga hasta aproximadamente los 4Khz, por encima del cual únicamente es usado el principio de emplazamiento.

La tabla-1 muestra la relación existente entre la intensidad del sonido y la fuerza percibida. Es común expresar la intensidad del sonido en una escala logarítmica, denominada **decibelio SPL** (Sound Power Level). En esta escala, 0 Db SPL es una onda de sonido de potencia 10^{-16} watios/cm^2, aproximadamente el sonido más débil detectable por el oído humano. Una conversación normal se encuadra a aproximadamente 60dB SPL, mientras que el peligro de daño doloroso para el oído tiene lugar a aproximadamente 140dB SPL.

	Watts/cm^2	Decibelios SPL	Sonido de ejemplo
	10^{-2}	140 dB	Dolor
	10^{-3}	130 dB	
↑	10^{-4}	120 dB	Molestias
	10^{-5}	110 dB	Martillos neumáticos
	10^{-6}	100 dB	
	10^{-7}	90 dB	Límite para el ruido industrial
+ Ruidoso	10^{-8}	80 dB	
	10^{-9}	70 dB	
	10^{-10}	60 dB	Conversación Normal
- Ruidoso	10^{-11}	50 dB	
	10^{-12}	40 dB	Más débil audible a 100Hz
	10^{-13}	30 dB	
↓	10^{-14}	20 dB	Más débil audible a 10KHz
	10^{-15}	10 dB	
	10^{-16}	0 dB	Más débil audible a 3KHz
	10^{-17}	-10 dB	
	10^{-18}	-20 dB	

Tabla 1- Intensidad del sonido y fuerza percibida

La diferencia entre los sonidos más fuertes y más débiles que pueden ser escuchados por el oído humano es de aproximadamente 120 dB, un rango de un millón hablando en términos de amplitud. Los oyentes pueden detectar un cambio en la fuerza cuando la señal es alterada en aproximadamente 1dB(un 12% de cambio en amplitud). En otras palabras, hay solamente 120 niveles de fuerza que pueden ser percibidos por el oído humano, desde el más leve susurro hasta el más ruidoso trueno. La sensibilidad del oído humano es por tanto sorprendente; ¡cuando estamos escuchando sonidos muy débiles el oído vibra menos del diámetro de una simple molécula!

La percepción de la fuerza está relacionada con la intensidad del sonido según exponente de 1/3. Por ejemplo, si se incrementa la fuerza del sonido en un factor de 10, los oyentes notarán que la fuerza del sonido ha aumentado un factor de alrededor de 2. Esto es un problema a tener seriamente en cuenta en materia de aislamiento sonoro , como por ejemplo los ruidos de los vecinos en las casas. Así supongamos que diligentemente se cubre el 99% de la superficie de las paredes con un perfecto material

insonorizante, dejando únicamente el 1% de la superficie sin cubrir debido a las puertas, esquinas, ventanas, etc. Aunque la fuerza del sonido haya sido reducida a un 1% de su valor original , la fuerza percibida habrá caído aproximadamente a $0.01^{1/3}$, esto es 0.2 o 20%.

El rango frecuencial por su parte, distinguible por el oído human se considera está entre 20Hz y 20kHz, pero éste es bastante más sensible a sonidos comprendidos entre 1Khz y 4Khz . Por ejemplo, los oyentes pueden detectar sonidos tan débiles como de 0dB SPL a 3kHz, pero requieren de al menos 40dB SPL a 100Hz(esto es un incremento de amplitud de 100) Así mismo los oyentes pueden diferenciar dos tonos siempre que éstos se diferencien en su frecuencia en más de un 0.3% a 3Khz. Esto se incrementa al 3% para 100 Hz. Por comparación diremos que las teclas adyacentes en un piano se diferencian en aproximadamente un 6% en frecuencia.

La principal ventaja de tener dos oídos es la habilidad de identificar la dirección del sonido. Los oyentes humanos pueden detectar la diferencia entre dos fuentes de sonido situadas tan poco separadas como a tres grados de distancia. Esta información direccional es obtenida de dos formas separadas. Primero, las frecuencias por encima de aproximadamente 1 KHz son fuertemente *sombreadas* por la cabeza. En otras palabras, el oído más próximo a la fuente del sonido recibe una señal más fuerte que el oído del otro lado de la cabeza . El segundo indicio de direccionalidad es que el oído más alejado escucha el sonido ligeramente más tarde que el oído más cercano, debido a la mayor distancia desde la fuente. Así basándonos en la medida de una cabeza de tamaño medio (22cm) y en la velocidad del sonido (340 metros por segundo), una discriminación angular de tres grados requiere una precisión temporal de aproximadamente 30 microsegundos. Ya que este tiempo es el requerido por el principio de descarga este indicio de direccionalidad es empleado predominantemente para frecuencias de menos de 1Khz.

Las dos fuentes de información direccional anteriormente citadas se ven complementadas por la habilidad de poder girar la cabeza y poder observar el cambio de las señales. Una interesante sensación ocurre cuando un oyente se encuentra con exactamente el mismo sonido en ambos oídos, sensación tal como la que se produce

escuchando sonido monoaural en unos cascos. ¡El cerebro concluye con que el sonido proviene del centro de la cabeza del oyente!

Debe señalarse, que mientras el oído humano es capaz de determinar la dirección del sonido, apenas es capaz de identificar con precisión la distancia a la que se encuentra su fuente. Esto es debido a que hay muy pocos indicios intrínsecos a una forma de onda de sonido que puedan facilitar esta información. Esto es así debido a que las ondas de sonido disipan sus altas frecuencias a medida que se propagan grandes distancias. Así mismo la percepción del efecto de eco es una pista muy poco valiosa en cuanto a la percepción de la distancia se refiere, pues para el oído humano proporciona una percepción aproximada del tamaño de una habitación. Consultar [4] para más detalles

3.1.1. CARACTERÍSTICAS Y PERCEPCIÓN DEL SONIDO.

La percepción de un sonido continuo, tal como puede ser una nota procedente de un instrumento musical, es divida clásicamente en tres conceptos fundamentales: Amplitud o fuerza, tono o frecuencia y timbre:

1º. La medida de la amplitud de una onda es importante porque informa de la fuerza, o cantidad de energía, de la misma, que se traduce en la intensidad de lo que oímos, su unidad de medida es el decibelio. Un decibelio, abreviado como dB, es una unidad de medida de la fuerza de la señal y es útil en la comparación de la intensidad de dos sonidos. La sensibilidad del oído humano es extraordinaria, con un rango dinámico o variación en intensidad muy amplio. La mayoría de los oídos humanos pueden capturar el sonido del murmullo de una hoja y, después de haberse sometido a ruidos explosivos como los de un avión, siguen funcionando. Lo que es sorprendente es que la fuerza de la explosión en un avión es al menos 10 millones de veces mayor que el murmullo que una hoja produce con el viento.

2º. El tono o simplemente frecuencia hace referencia a la frecuencia de la componente fundamental del sonido, esto es, la frecuencia con que la onda se va repitiendo a lo largo del tiempo .

3º. El timbre es más complicado, siendo éste determinado por el contenido armónico de la señal en cuestión. La figura 3 ilustra dos formas de onda, cada una de ellas construida a partir de la suma de una componente de 1Khz senoidal con una amplitud de 1 y otra componente senoidal de 3Khz con una amplitud de 0.5. La diferencia entre las dos formas de onda resultantes es que la mostrada en b) tiene la componente de mayor frecuencia invertida previamente a la suma. Dicho de otra manera, el tercer armónico está invertido un total de 180º comparado con el primer armónico. Salvo la diferencia altamente notable en cuanto a forma de onda se refiere, estas dos señales suenan idénticamente igual. Esto es así porque como ya expusimos en el apartado anterior , el mecanismo de escucha humano está basado en la amplitud de las frecuencias y es muy insensible a la fase de las mismas. De esta forma, la forma de un determinado sonido en el dominio del tiempo está relacionada sólo indirectamente con lo que escuchamos, y normalmente es una información ignorada en gran parte por los sistemas de audio.

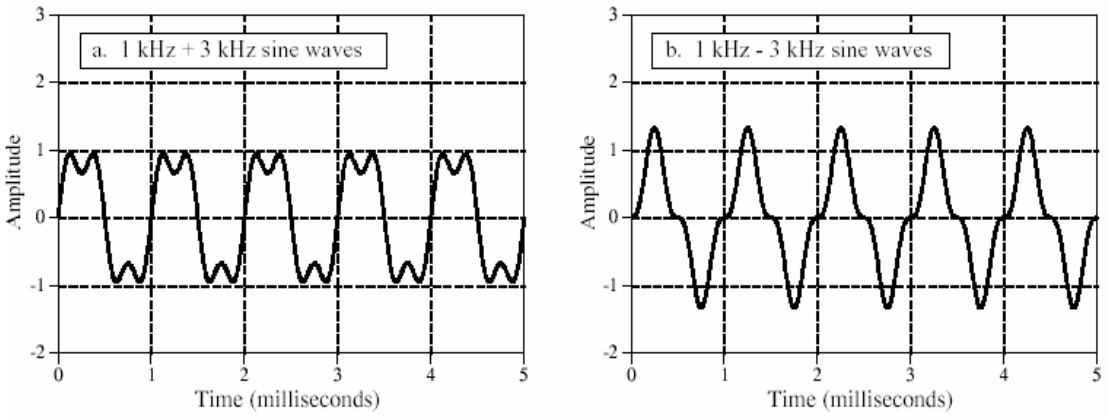


Figura 3 - Detección de fase por el oído humano.

Supóngase que se toca una nota en un violín, por ejemplo A por debajo de media C. Cuando la forma de onda es mostrada en el osciloscopios, se parecerá mucho a la diente de sierra mostrada en la figura 4-a). La figura 4-b) muestra el sonido tal y como el oído lo percibirá , una frecuencia fundamental de 220 Hz, más armónicos en 440, 660, 880 Hz, etc. Si esta nota fuera tocada en otro instrumento la forma de onda se vería realmente diferente, si bien el oído seguiría escuchando una frecuencia de 220Hz más los armónicos correspondientes al nuevo instrumento. Así, los dos instrumentos

producirán la misma frecuencia fundamental para la misma nota, esto es tendrán el mismo **tono.** No obstante, ya que la amplitud relativa de cada armónico será lógicamente distinta, no sonarán idénticos, diremos por tanto que tienen diferente **timbre.**

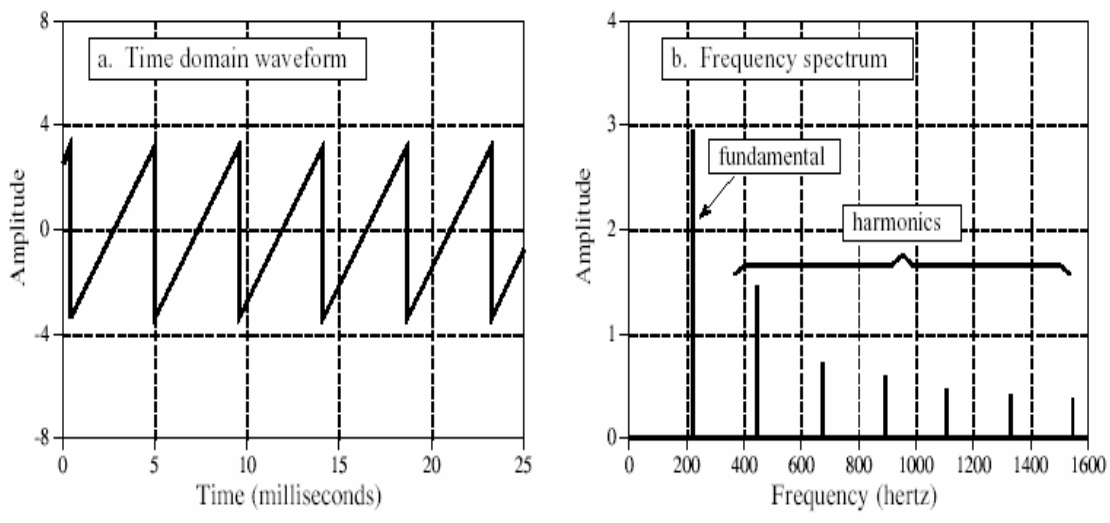


Figura 4- a) violín visto por el osciloscopio. b)Violín visto por el oído.

Retomando la explicación sobre el significado del timbre, hay que continuar recordando que se ha dicho que el timbre es determinado por la forma de onda del sonido. Pues bien, esto es cierto, pero está engañosamente enunciado. La percepción del timbre resulta de la percepción de los armónicos por parte del oído. Mientras el contenido armónico es determinado por la forma de la onda de sonido, la insensibilidad del oído a la fase de la misma hace que la relación sea parcial. Esto es, una forma de onda particular tendrá únicamente un timbre mientras que un timbre particular tendrá un número infinito de posibles formas de onda.

El oído humano está muy acostrumbado a escuchar una frecuencia fundamental más los armónicos. Así si a un oyente se le presenta una combinación de 1Khz con 3Khz, notará inmediatamente que suena agradable y natural. Por el contrario si la composición de senoidales implica una componente de 1Khz y otra de 3,1 Khz, notará un sonido desnaturalizado.

Esta es la base de la escala musical estándar, como se ilustra en la figura5. Golpear la tecla de más a la izquierda del teclado produce una frecuencia fundamental de 27.5 Hz, más armónicos a 55, 110, 220, 440, 880 Hz, etc. Estos armónicos se corresponden con las frecuencias fundamentales producidas por otras teclas del teclado. Específicamente, cada séptima tecla blanca es un armónico de la tecla más a la izquierda. Esto es, la octava tecla desde la más a la izquierda tendrá una fundamental de 55Hz, la quinceava tendrá una fundamental de 110Hz, etc. Siendo armónicos unas de otras, estas teclas suenan similar cuando son tocadas y son armoniosas cuando se tocan al unísono. Por esta razón son todas denominadas A. De la misma forma la tecla blanca inmediatamente a la derecha de cada A se denomina B, y son todas las B siguientes armónicos de la primera. Este patrón se repite para las siete notas: A, B, C, D, E, F, y G.

El término octava quiere decir un factor de dos, en términos de frecuencia. En el piano, una octava comprende ocho teclas blancas. En otras palabras, las frecuencias de las teclas correspondientes del piano, doblan su valor cada siete teclas blancas, y el teclado entero se expande a lo largo de algo más de siete octavas. El rango de la escucha humana está normalmente acotado en el rango de 20Hz a 20Khz, correspondiendo sus límites a aproximadamente media octava a la izquierda y dos octavas a la derecha del teclado del piano. Ya que las octavas se basan en doblar la frecuencia cada cierto número fijo de teclas, ellas son una representación logarítmica de la frecuencia. Esto es importante porque la información auditiva está distribuida generalmente de la misma forma. Por ejemplo, tanta información porta la octava entre 50Hz y 100Hz, como la octava entre 10 Khz y 20 Khz. Así, aunque el piano únicamente cubre aproximadamente el 20% de las frecuencias que el oído humano puede escuchar (4Khz de 20 Khz), puede producir más del 70% de la información auditiva que el ser humano puede percibir(7 de 10 octavas). De esta forma, la frecuencia más alta que el oído humano puede detectar cae desde aproximadamente 20Khz a 10Khz a lo largo de la vida, lo que supone una pérdida tan sólo del 10% de la capacidad auditiva.

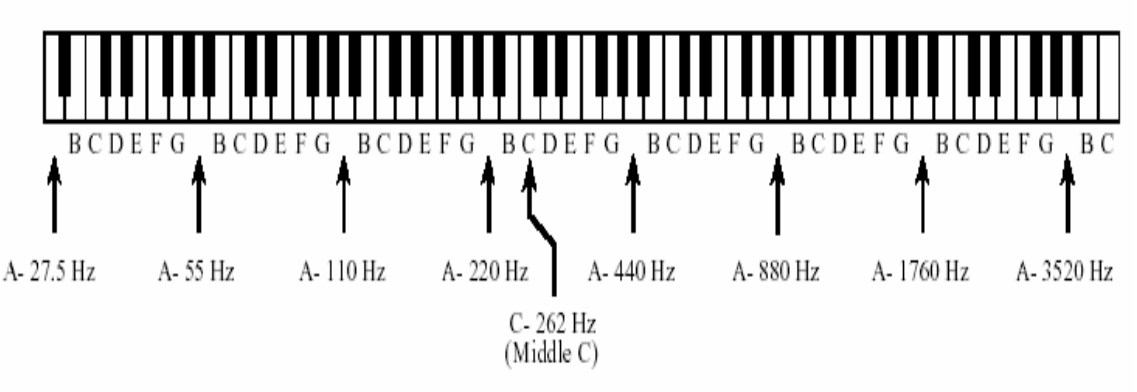


Figura 5 Descripción del teclado del piano.

3.2 FUNDAMENTOS SOBRE AUDIO DIGITAL.

A continuación se exponen los conocimientos básicos necesarios para el procesado digital de las señales de audio sin entrar en demostraciones matemáticas rigurosas ni en largos desarrollos por no ser el objetivo del presente trabajo. De esta manera, para corroborar lo que aquí se expone de forma somera con el objetivo de continuar avanzando hacia el corazón del trabajo, consultar las reseñas bibliográficas relativas a teoría sobre el procesado digital de señales en referencias [1], [2] y [3]

3.2.1. DEFINICIONES Y CONCEPTOS PRELIMINARES.

He aquí algunos términos con frecuencia empleados al hablar de audio digital y su significado:

1. Rango dinámico .

La calidad de los sonidos musicales grabados no es demasiado importante, ya que nunca son comparables a los reales. La razón principal es que el equipo estéreo no puede duplicar el rango dinámico completo de una orquesta o de un concierto de rock. Una orquesta puede alcanzar los 110 dB en su climax y en el punto más suave bajar hasta los 30 dB, dando lugar a un rango dinámico de 80 dB. Este rango es superior al rango dinámico de un sistema estéreo típico y, de hecho, superior a la capacidad de grabación de medios tales como un disco de vinilo y una cinta de audio.

2. Ancho de banda

Profundizamos ahora en aspectos prácticos, como el rango de frecuencia con el que es capaz de trabajar un reproductor CD, nuestro oído o nuestra voz. El ancho de banda es muy importante para disfrutar de la música (como manifiestan las quejas de sonido "de lata" de una radio de bolsillo) y es un criterio básico a la hora de seleccionar un equipo de audio para utilizar con la tarjeta de sonido. Por ejemplo, el ancho de banda teórico de la radio FM es aproximadamente tres veces el ancho de banda de la radio AM, por lo que la FM será capaz de reproducir frecuencias que no entran dentro del campo de trabajo de la AM.

Nota: A menudo el ancho de banda se simboliza mediante un único número cuando la frecuencia baja está bastante próxima a cero. Por ejemplo, el ancho de banda de una voz femenina se sitúa en torno a los 9 kHz, aunque realmente puede estar en el rango que va desde los 200 Hz hasta los 9 kHz.

Existe una medida estándar para definir el ancho de banda: el rango de frecuencias sobre el que la amplitud de la señal no difiere del promedio en más de 3 dB, es decir la diferencia de las frecuencias en la que se produce una caída de 3 dB, ya es el punto donde su amplitud cayó a la mitad, y éste es el mínimo cambio en la fuerza de la señal que puede ser percibido como un cambio real en la intensidad por la mayoría de los oídos.

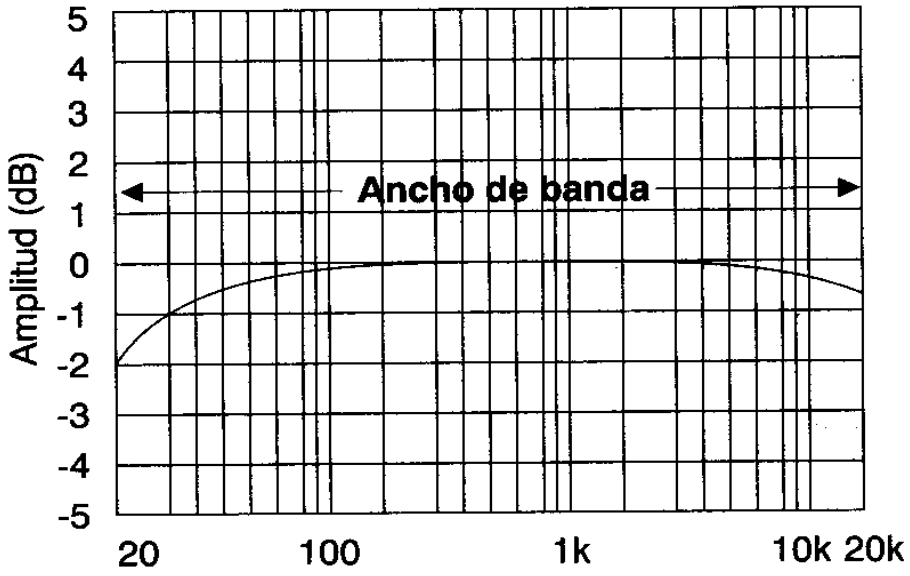


Figura 6.Ancho de banda del sistema

Es importante tener en cuenta que el ancho de banda de un equipo de sonido depende del enlace más débil del canal, que normalmente no es la tarjeta de sonido en caso de estar trabajando con el PC. La calidad del sonido producido por la computadora refleja el esfuerzo de muchas componentes, y la salida no será mejor que la interpretación del miembro menos capacitado de un grupo. En el caso del sistema de sonido de la computadora, una señal debe pasar por muchas fases de transformación de audio y por diferentes dispositivos. Por ejemplo, consideremos el sonido grabado mediante un micrófono y que luego es reproducido. La tarjeta de sonido transforma el sonido recogido del micrófono en una señal eléctrica que, posteriormente, se transforma en audio digital y se almacena en disco. El audio digital del disco es transformado de nuevo en una señal eléctrica y reproducido a través de los cascos o de los altavoces. El ancho de banda efectivo del sistema de sonido está limitado por el dispositivo con el ancho de banda más estrecho de todos los dispositivos que procesan el sonido.

El enlace más débil en grabación suele ser el micrófono, que tiene probablemente un ancho de banda aproximadamente de 12 kHz.

3. Ruido .

Del mismo modo que perturban los ruidos y ecos en una habitación, también puede generarse ruido y distorsión en la tarjeta de sonido, en los altavoces o en el micrófono si estamos trabajando con un PC. El ruido (sonidos aleatorios que transforman y enmascaran el sonido deseado) se mide también en decibelios. Dado que es tan poco probable disponer de un entorno de audio digital en perfecto silencio, lo que interesa realmente es saber la cantidad de ruido en relación con la señal que se introduce en el equipo de sonido, especialmente en la tarjeta de sonido. La fuerza de la música, del habla o de cualquier otro sonido, comparada con la fuerza promedio del ruido, se conoce como relación señal/ruido. A medida que aumenta la relación s/n, es mejor el trabajo realizado en grabación. La relación señal ruido de una tarjeta digital sencilla es del orden nada despreciable de 85 dB. Esto significa que la fuerza de la señal es 85 dB mayor que la fuerza del ruido. Una relación de 70 dB se considera válida para propósitos musicales y una relación de 65 dB s/n está en el límite de aceptación.

4. Sistema de audio en tiempo real.

En ellos la señal de audio es procesada mediante el/los algoritmo/s apropiado/s lo suficientemente rápido para reproducir el resultado sin retardo apreciable por el oyente. La memoria se requerirá, si para generar la señal de salida en el instante actual se necesita información relativa a muestras anteriores. Sistemas de procesado del audio en tiempo real son los procesadores de efectos en tiempo real para instrumentos musicales fabricados a partir de procesadores digitales de señales.

Por el contrario, el procesado no en tiempo real, es un procesado propio del que realiza el PC en los programas de edición musicales, en el cual primero se graba un sonido que se almacena en un archivo en memoria, después se manipula éste convenientemente guardando el resultado en un nuevo archivo o sobrescribiendo el original, para por último reproducirlo cuando se desee.

3.2.2. CONVERSION A/D Y D/A , MUESTREO.

3.2.2.1. GENERAL.

Antes de que el sistema pueda digitalizar, manipular y reproducir sonido, debe transformarse el sonido de una forma analógica audible a una forma digital aceptable por el mismo, mediante un proceso denominado conversión analógica - digital (ADC).

Una vez que los datos de sonido se han convertido en bytes y se han almacenado si fuera necesario en memoria (en caso de necesitar muestras pasadas para generar la salida en tiempo real o en caso de que solamente se pretenda grabar para manipular y reproducir ya no en tiempo real), puede hacerse uso de la potencia de la CPU para transformar este sonido de miles de modos. Con el software adecuado es posible, por ejemplo, añadir reverberación o eco a la música o a la voz. Pueden eliminarse trozos de sonido grabado. Puede ajustarse el tono de la grabación y muchas cosas más. Finalmente, para obtener el resultado, el proceso de conversión digital-analógica (DAC) transforma de nuevo los bytes de sonido ya procesados a una señal eléctrica analógica que emiten los altavoces. La figura 7 muestra el diagrama de bloques de un sistema genérico para procesar señales, aplicable por supuesto al ámbito del procesado digital del audio.

Debe indicarse que con un sistema de estas características se puede efectuar un procesado en tiempo real o bien un procesado propio del que realiza el PC en los programas de edición musicales tal y como se describió anteriormente.

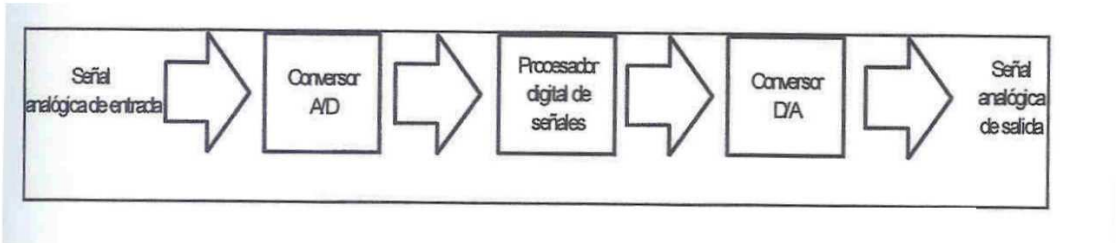


Figura7. Diagrama de un sistema genérico de procesado de señal.

3.2.2.2 COVERSIÓN ANALÓGICA/DIGITAL Y VICEVERSA. .

Comenzaremos con la captura del sonido haciendo uso del micrófono si es que la fuente de sonido no es un instrumento que realiza él mismo la transducción de audio a electricidad como ocurre en el caso de por ejemplo una guitarra acústica o eléctrica por acción de las pastillas. Cuando las ondas de sonido llegan al micrófono, el movimiento mecánico se traduce en una señal eléctrica. Esta señal se denomina señal analógica porque es una señal continua en el tiempo, análoga al sonido original.

CONVERSION ANALOGICA-DIGITAL (ADC).

Podemos dividir este proceso en tres partes fundamentales:

1ª. Dada una señal analógica, se van tomando valores discretos de su amplitud a intervalos de tiempo pequeños, evidentemente será más fiable la reproducción cuantas más muestras por segundo se tomen; a este proceso se le denomina **MUESTREO** y no es más que la conversión de una señal en tiempo continuo en su equivalente en tiempo discreto. Así si $x_a(t)$ es la entrada al muestreador, la salida es $x_a(nT)$ **donde T se denomina intervalo de muestreo**.

2ª. Cuantificación. Este proceso consiste en la conversión de una señal en tiempo discreto con valores continuos en otra en tiempo discreto con valores discretos(señal digital). Esto es, el valor de cada muestra de la señal se representa mediante un valor seleccionado de un Conjunto finito de valores posibles. La diferencia entre la muestra sin cuantificar x(n) y la salida cuantificada $x_q(n)$ se denomina **error de cuantización.**

3ªCodificación. En el proceso de codificación, cada valor discreto $x_q(n)$ se representa mediante una secuencia binaria de b bits. A cada muestra obtenida se le asigna un equivalente binario. Por esta razón y dependiendo de la fidelidad con que queramos trabajar podemos utilizar palabras de 8 o 16 bits pudiendo obtener así 256 o 65536 combinaciones distintas y obtener mayor resolución. Palabras mayores son por supuesto posibles siempre que los conversores las soporten y se posea la suficiente memoria como para almacenar las muestras.

La figura 8 muestra el diagrama de la conversión ADC teniendo en cuenta los procesos anteriores.

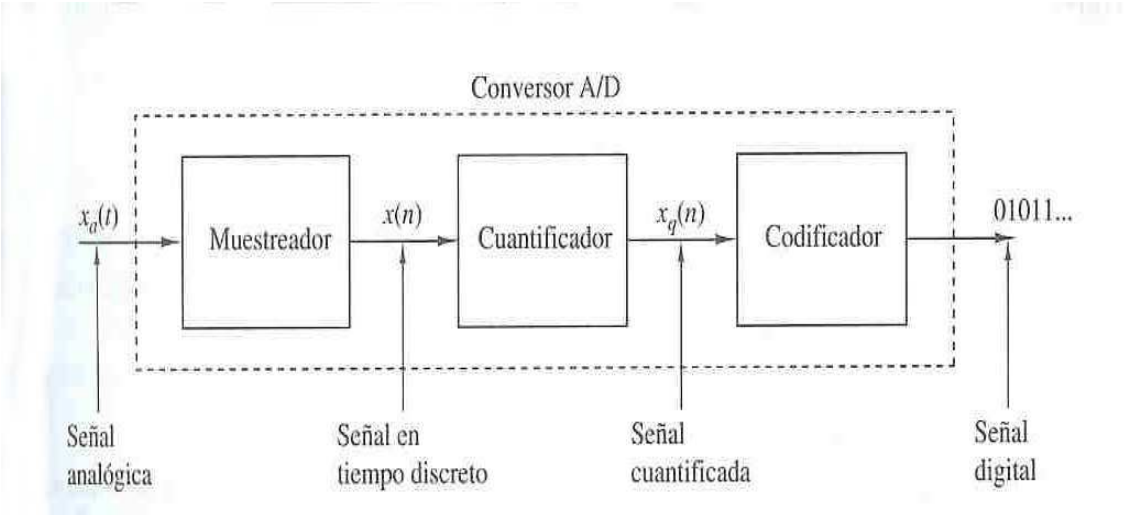


Figura 8 - Procesos en un conversor ADC.

Debe decirse que aunque se modele el conversor ADC con un muestreador seguido de un cuantificador, en la práctica la conversión A/D se efectúa en un único dispositivo que toma $x_a(t)$ y produce un número codificado en binario . Las operaciones de muestreo y cuantificación pueden realizarse en cualquier orden , pero en la práctica , el muestreo siempre tiene lugar antes de la cuantificación.

La figura 9 ilustra como se efectúa la captación de la señal de sonido analógica por un ADC .

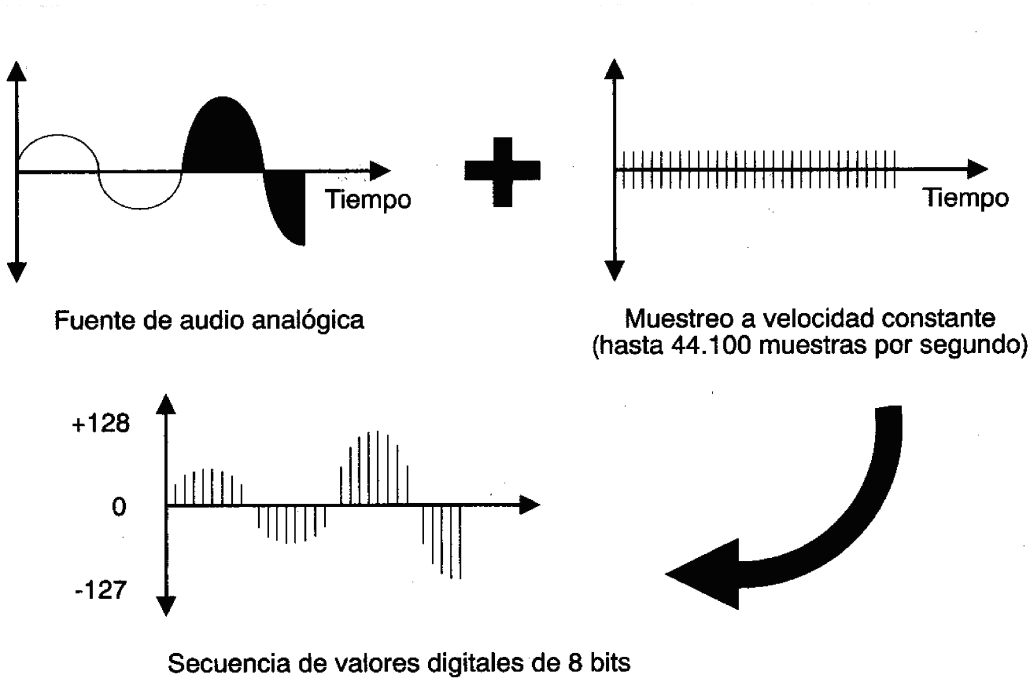


Figura 9- captación de la señal de sonido analógica por un ADC .

CONVERSION DIGITAL-ANALOGICA (DAC): El proceso inverso es mucho menos complejo ya que solo se trata de ir poniendo los valores de las muestras en el mismo orden que fueron tomadas una vez que se hayan procesado numéricamente de acuerdo con el algoritmo adecuado, y los filtros de recomposición a la salida del DAC se encargan de convertir esa señal resultante de valores discretos (digital) en una señal analógica. Debe mencionarse que el muestreo no da lugar a una pérdida de información ni introduce distorsión en la señal si su ancho de banda es finito. En principio , la señal analógica puede reconstruirse a partir de sus muestras, siempre que la tasa de muestreo sea lo bastante alta como para evitar el problema comúnmente denominado aliasing. Por otra parte, la cuantificación es un proceso no invertible o irreversible que resulta en la distorsión de la señal.

3.2.2.3. MUESTREO: VELOCIDAD Y TAMAÑO DE LA MUESTRA.

La exactitud en la réplica de la música original del sonido audio digital depende de la selección de la correcta frecuencia de muestreo y del correcto tamaño de muestra, siendo este último el número de bytes utilizados para almacenar cada muestra.

1º. FRECUENCIA DE MUESTRA(Fs) : La frecuencia de muestra Fs (también denominada frecuencia de muestreo) debe ser lo suficientemente alta para que

los sonidos de alta frecuencia, como el sonido del cristal de una copa de vino o el del arqueo de un violín, puedan recogerse con precisión.

Según el **teorema de Nyquist o teorema del muestreo** (ver reseña bibliográfica [1] para más detalle) es posible repetir con exactitud una forma de onda si la frecuencia de muestreo es como mínimo el doble de la frecuencia de la mayor componente espectral de la señal analógica a muestrear, de forma que se evita la aparición de las muestras ambiguas y repetidas originadas por el fenómeno de aliasing. Así, la frecuencia más alta que puede percibir el oído humano está cercana a los 20 kHz, de modo que la frecuencia de muestreo de 44.1 kHz de las tarjetas de sonido es más que suficiente para grabar fielmente cualquier tipo de sonido audible. Este valor es el utilizado hoy en día por los reproductores de audio CD.

Los archivos de audio digital pueden grabarse seleccionando la frecuencia de muestreo. A medida que aumenta la frecuencia de muestreo, aumenta la calidad del sonido. Por ejemplo, una velocidad de 6.000 Hz (6.000 muestras por segundo) es buena para una voz masculina típica, pero no lo es para una voz femenina típica, que tiene componentes con una frecuencia más alta. Una frecuencia de muestreo de 8.000 Hz proporciona una grabación de la voz femenina de mayor calidad. Así mismo, una señal procedente de una guitarra eléctrica contendrá frecuencias hasta los 10KHz aproximadamente lo que hace que la frecuencia de muestreo debe tomarse a partir de 20Khz.

Es típico en el proceso de muestreo el uso del denominado filtro antialiasing para la señal de entrada analógica, filtro paso bajo muy selectivo con una frecuencia de corte igual a la mitad de la frecuencia de muestreo Fs. Así mismo a la salida del DAC se colocará un paso bajo igual de selectivo y con la misma frecuencia de corte denominado de aliasing como parte del grupo de filtros de reconstrucción.

La siguiente tabla ofrece una lista de algunas tarjetas de sonido Sound Blaster y de sus frecuencias de muestreo a título meramente orientativo:

Tarjeta	Grabación(Khz)	Reproducción (Khz)
Sound Blaster 16	44.1 (mono o estéreo)	44.1(mono o estéreo)
Sound Blaster Pro	22.050 (estéreo)	22.050(estéreo)
	44.100(mono)	44.100 (mono)
Sound Blaster 2.0	15.000 (mono)	44.100 (mono)
	13.000 (mono)	23.000 (mono)

Tabla 2 – Frecuencias de muestreo de tarjetas de audio comerciales.

Existen varias razones para no utilizar las frecuencias de muestreo más altas. En primer lugar, las frecuencias de muestreo altas necesitan gran capacidad de almacenamiento.

2º. TAMAÑO DE MUESTRA: El tamaño de muestra es la otra componente de mayor influencia en la fidelidad del audio digital. Las tarjetas de sonido de 16 bits ofrecen la posibilidad de elegir entre un tamaño de muestra de audio digital de 8 bits (1 byte) o de 16 bits (2 bytes) y los conversores ADC existentes para aplicaciones DSP permiten muestras de 14, 16, 24 e incluso 32 bits.

El tamaño de muestra controla el rango dinámico que puede grabarse. Por ejemplo, las muestras de 8 bits limitan el rango dinámico a 256 pasos (rango de 50 dB). Por el contrario, una muestra de 16 bits tiene un rango dinámico de 65.536 pasos (rango de 90 dB), una mejora sustancial. El oído humano percibe todo un mundo de diferencias entre estos dos tamaños de muestra. Los oídos humanos, que están acostumbrados a detectar sonidos con variaciones de varios órdenes de magnitud en la fuerza, perciben el sonido de 8 bits en un tono apagado o desafinado si se compara con el sonido de audio digital de 16 bits.

3º. COMPROMISOS EN EL MUESTREO Se podría asumir que todo lo que hay que hacer para obtener buen sonido es muestrear a la velocidad límite de 44,1 kHz con muestras de 16 bits (2 bytes). El único problema que aparece es que si se muestrea en estéreo, tomando muestras simultáneamente en los canales izquierdo y derecho a 44,1 kHz, una muestra de sonido de un minuto necesita un espacio para almacenarse de 10,58MB.

Lo aconsejable es usar la frecuencia de muestreo más baja posible. Por ejemplo, supongamos que planeamos procesar la señal procedente de una guitarra eléctrica (aproximadamente 10Khz de máxima componente espectral considerable) de acuerdo con el teorema de Nyquist, la grabación será acertada si la frecuencia de muestreo es de 22 kHz si se quiere obtener una buena calidad.

Cuando se elige la frecuencia de muestreo, también hay que considerar el ancho de banda de todo el sistema. Por ejemplo, no existe ningún problema en la grabación de audio digital a 44,1 kHz si el micrófono utilizado funciona a 12 kHz y la fuente de sonido es una voz masculina grave que no supera los 7 kHz.

3.3. ANÁLISIS DE LA TECNOLOGÍA DE MICROPROCESADORES DIGITALES DE SEÑALES(DSP).

Una vez completados los capítulos dedicados a la introducción teórica con los conceptos generales más importantes, se pasa a describir las posibilidades que los DSP nos brindan para llevar a cabo el procesado de señal.

3.3.1. ALGORITMOS TÍPICOS DE PROCESAMIENTO DIGITAL DE SEÑALES.

El procesamiento digital de señales en general y los procesadores DSP en particular se emplean en una amplia gama de aplicaciones desde sistemas militares de radar a electrónica de consumo. Por supuesto, ningún procesador satisface las necesidades de todas las aplicaciones. Criterios como capacidad, coste, integración, facilidad de desarrollo y consumo de potencia son puntos clave a examinar al diseñar o elegir un DSP concreto para una clase de aplicaciones. En el caso concreto de la aplicación a efectos de audio en tiempo real , resulta necesario el empleo de un DSP moderadamente rápido y con una buena cantidad de memoria interna para la implementación de bufferes circulares.

3.3.2. DIFERENCIAS ENTRE UN DSP Y UN PROCESADOR DE PRÓPÓSITO GENERAL.

Los DSPs se diferencian de los microprocesadores en muchos aspectos. Los microprocesadores se construyen para un conjunto de funciones de propósito general y normalmente ejecutan grandes bloques de software, corno sistemas operativos. Los microprocesadores a menudo no están indicados para operación en tiempo real. A menudo, tienen la libertad de organizar sus cargas de trabajo y elegir su propio curso de acción, esperando acabar un trabajo de impresión por ejemplo antes de responder a un comando de usuario. Y aunque los procesadores tienen algunas capacidades numéricas, en absoluto son lo suficientemente capaces para la mayoría de las aplicaciones DSP. Mientras que un procesador es versátil y apto para muchas aplicaciones, el DSP es un especialista, que realiza a gran velocidad un menor número de funciones. A menudo se utiliza a los DSPs como una especie de controlador incrustado, un procesador que, acompañado del software necesario, se incluye dentro de una parte del equipo y se dedica a un único conjunto de tareas. En los sistemas de ordenadores, los DSPs a menudo se emplean como procesadores incluidos, asistiendo a un microprocesador host de propósito general. He aquí un resumen de las principales diferencias entre ambos:

- Los DSPs está especializados en aplicaciones de procesamiento de señal/control incrustado en tiempo real, no para procesado de propósito general.

- Los DSPs son estrictamente no programables por el usuario (sin gestión de memoria, sin sistema operativo, sin caché, sin variables compartidas, orientados a un único proceso).

- Los DSPs generalmente emplean algún tipo de "arquitectura Harvard" para permitir búsquedas de código y datos simultáneas.

- La mayoría de los DSPs realizan una operación MAC (multiply-accumulate) en un solo ciclo de reloj.

- Los programas DSP a menudo residen en ROM y/o RAM rápidas integradas (aunque a menudo es posible la expansión de bus fuera del chip).

- La mayoría de los DSPs tienen al menos dos RAMs integradas multipuerto para almacenar operandos.

- La gestión de interrupciones de los DSPs es sencilla, rápida y eficiente

- Muchas aplicaciones DSP están escritas en ensamblador, debido a las restricciones de procesamiento en tiempo real (aunque existen compiladores de C para la mayoría de los DSPs).

- Las previsiones de *l/O* para los DSPs a menudo son bastante simples.

- Los anchos de bus de direcciones de los DSPs a menudo son más pequeños quelos de los procesadores de propósito general (el tamaño del código suele ser pequeño).

- Los DSPs de punto fijo utilizan aritmética de saturación (mejor que permitir que el desbordamiento de complemento a 2 tenga lugar).

- Los modos de direccionamiento de los DSPs están orientados a las aplicaciones de procesamiento de señales (buffers circulares, esquemas de acceso de "mariposa").

- Los DSPs a menudo proporcionan apoyo de hardware directo para la implementación de bucles "do". Muchos DSPs emplean una pila de hardware integrada para facilitar el enlace de subrutinas.

- La mayoría de los DSPs cuentan con ROM de programa integrada y RAM para permitir soluciones en un único chip. La mayoría de los DSPs no tienen ADCs o DACs integrados, dichos periféricos se implementan a menudo externamente.

- Conjunto de puntos de referencia utilizados para comparar los DSPs son totalmente diferentes de los usados para comparar procesadores de propósito general *(RISC/CISC)*

3.3.3. BALANCE ENTRE FPGA Y DSP.

Uno de los principales usos de las FPGAs ha sido realizar prototipos rápidos antes de la construcción de un ASIC A medida que la tecnología FPGA ha ido mejorando, gente de la industria y de la investigación ha comenzado a utilizar las FPGAs no sólo como un componente de banco de pruebas, sino también como un componente del núc1eo del sistema.. La flexibilidad y creciente capacidad de las FPGAs modernas son muy atrayentes. Por otro lado, el procesador digital de señal (DSP) ha sido largo tiempo el núcleo de un sistema de procesamiento digital de señales debido a su flexibilidad. Un reciente avance en el desarrollo de los DSPs introdujo el uso de la arquitectura VLIW, produciendo nuevos y prometedores resultados. Con ambas opciones dando atractivos resultados, resulta de interés tener un conocimiento más profundo del equilibrio actual entre ambas posibilidades de implementación.

Para realizar un estudio comparativo es necesario definir algunas propiedades que describan las características de ciertos algoritmos comunes que deben implementarse en ambas plataformas para obtener datos de su eficiencia. Algunas de dichas propiedades pueden ser las siguientes:

· **Tamaño.** Hace referencia al número de nodos necesarios para implementar el algoritmo en una solución hardware o bien la cantidad de código necesaria para una solución software.

· **La libertad temporal** se cuantifica a través de medidas como la proporción de la tasa de muestreo, el número y tipos de nodos en la red crítica, etc. Esta variable mide la cantidad de libertad en tiempo que tiene el diseñador, ya sea en limitaciones de tiempo en una solución hardware o la capacidad de realizar ejecución fuera de orden en software. Puede medirse esta característica como una función de la longitud del camino crítico, así como de los tipos de operaciones en el mismo.

· **La uniformidad** mide la distribución de operaciones y accesos a memoria en el tiempo.

· **La concurrencia** mide la cantidad de operaciones y accesos de interconexión que deben realizarse simultáneamente para una implementación paralela, o por paso de tiempo de entrada para una implementación serie.

· **La temporalidad** es una característica que describe el tiempo de vida de las variables.

· **La localización espacial** es una medida de la cantidad de operación paralela que puede ocurrir en los clústeres hardware, o en las llamadas a procedimientos en software.

· **La regularidad** es una medida de la cantidad de estructuras comunes que puede haber en un algoritmo y que pueden ser mareadas en una estructura normal de nodos en hardware paralelo o en una estructura normal de bucle.

· Para medir las **propiedades cíclicas** de un algoritmo puede medirse su límite de iteración, así como el porcentaje de nodos que actúan en los bucles.

· Las propiedades del **flujo de programa** también son muy importantes para la funcionalidad de un algoritmo, y algo que puede ser fácilmente rebajado en el primer paso de un algoritmo.

Las conclusiones que se obtienen de este tipo de experimentos son dos básicamente. La primera es que el paralelismo en un diseño FPGA tiene un precio en tiempo de diseño, que normalmente será superior al doble del tiempo necesario para implementar el mismo algoritmo sobre un DSP. Por otro lado, el código optimizado a mano puede a menudo aprovecharse del paralelismo inherente a un algoritmo mejor que el código sintetizado, pero en cualquier caso el diseñador puede quedarse rápidamente sin espacio útil incluso en la mayor de las FPGAs. En el caso de que el algoritmo sea realizable en una única FPGA, el diseño FPGA sobrepasará la capacidad del algoritmo DSP en las propiedades previamente mencionadas. En diseños regulares pueden alcanzarse capacidades de orden de magnitud doble y triple de las FPGAS frente a los DSPs. De este modo, para este tipo de algoritmos que puedan implementarse en una

única FPGA este tipo de implementación puede ser preferible si se dispone del tiempo y de los recursos necesarios.

3.3.4. DSP Y EFECTOS DE AUDIO.

Un DSP es la herramienta más útil a la hora de implementar un efecto para guitarra, habiendo elaborado prestigiosas empresas como Yamaha o Creative procesadores de efectos basados en esta tecnología. Así, una vez comentados los aspectos generales sobre audio necesarios y algunas características tecnológicas de los DSP a nivel general y en contraposición con los microprocesadores genéricos, se pasa a describir los efectos de audio para guitarra en el marco del tratamiento diital de señales, lo que nos dejará en condiciones óptimas para pasar a la implementación sobre dicha plataforma, pues todas las descripciones que se realizarán se efectuarán en forma de ecuaciones idóneas para su implementación muestra a muestra en un DSP. Para más detalles se recomienda consultar las referencias [5],[6],[7],[8].

3.4. EFECTOS EN EL DOMINIO DINÁMICO.

Este grupo de efectos se basa en las variaciones que producirán en el volumen o fuerza de la señal a tratar. Estas variaciones obedecerán lógicamente al fin perseguido por el efecto en cuestión. Los más importantes son los siguientes:

- Distorsión.
- Puerta de ruido.
- Compresor
- Expansor.
- Modulador en anillo

3.4.1. DISTORSIÓN .

3.4.1.1. INTRODUCCIÓN .

Se entiende por distorsión en general, la modificación de un sonido original. De esta forma, cualquier efecto de audio podría ser calificado como distorsión. Aparte de esta definición generalista, se conoce como distorsión, aquel efecto que modifica la amplitud de un sonido sin el uso para ello de ninguna información temporal (de esta forma cualquier procesado de la señal en el dominio dinámico puede ser considerado como distorsión). Así, debido a que ninguna información temporal es usada, cada muestra puede ser independientemente modificada.

La distorsión es usada normalmente para simular los efectos de un amplificador que está saturado y se emplea principalmente con guitarras .

3.4.1.2. PRINCIPIOS Y MODELADO.

La distorsión del tipo que producen los amplificadores para guitarra consiste sencillamente en comprobar si la señal de entrada está por encima de *+umbral* o por debajo de *-umbral* y si es así, convertirla inmediatamente en el **nivel de saturación prefijado** (positivo o negativo según el umbral que sea superado)**.** Si dicho umbral no

es superado, la señal original no se modificará. Este modelo es muy sencillo y su acción se muestra en la figura 10 donde umbral= 10 y nivel de saturación=20.

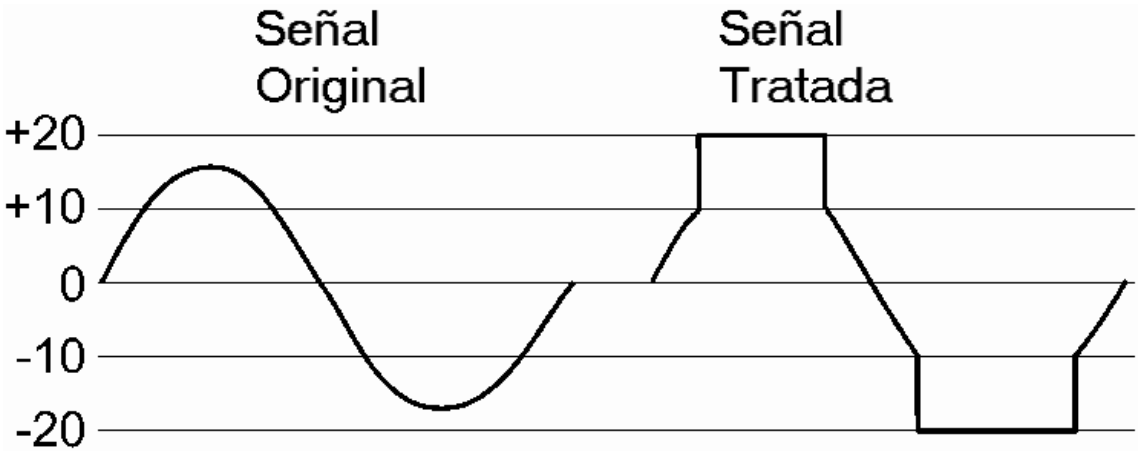


Figura 10- Distorsión Básica.

Un segundo modelo más completo que comprenderá al anterior, es el que consiste tomar un porcentaje de la señal original de entrada que es sumada con un porcentaje del nivel de saturación elegido para generar el efecto. Así, la señal distorsionada es idéntica a la señal original, salvo si se supera el **umbral prefijado.** Cuando la señal supera este umbral, es modificada según la ecuación en diferencia:

Si x(n)>+umbral

$$y(n) = a\,x(n) + b \text{ (nivel de saturación)} \qquad (1)$$

Si x(n)<-umbral:

$$y(n) = a\,x(n) + b \text{ (-nivel de saturación)} \qquad (2)$$

Si no se aplicaron ni (1) ni (2):

$$y(n) = x(n) \qquad (3)$$

Donde ***a*** es la cantidad de señal original que se añade (en tanto por uno) y ***b*** es la cantidad de **nivel prefijado de saturación** que se usa en la mezcla (en tanto por uno también) . Obsérvese que este modelo comprende al anterior para *a= 0* y *b=1*, caso en el que la salida se convierte directamente en el nivel de saturación prefijado.

En la figura 11 se muestra el diagrama de sistema del distorsionador si se supera el umbral prefijado.

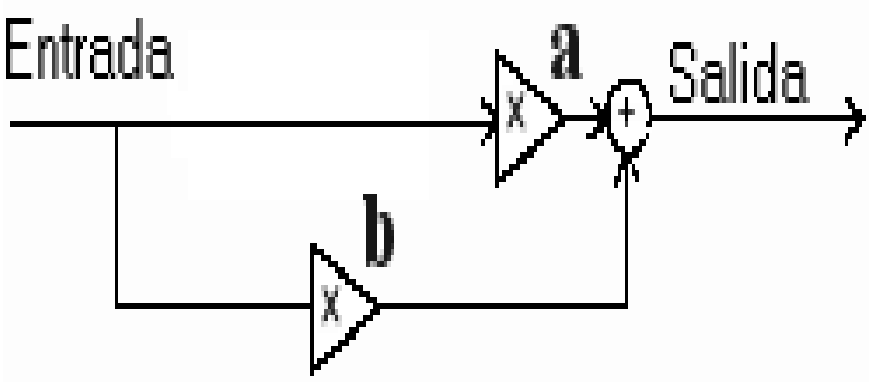


Figura 11. Distorsionador si se supera el umbral.

3.4.1.3. IMPLEMENTACIÓN.

Obsérvese que la implementación en microprocesador del modelo anterior para el distorsionador es realmente sencilla, reduciéndose todo a una comparación con *+umbral*, otra con *–umbral* , un par de productos numéricos y una adición por muestra en caso de que algún umbral se superase. Los parámetros a introducir por el usuario para controlar el efecto son :

- umbral.
- saturación.
- a = la cantidad de señal original que se añade (en tanto por uno)
- b= cantidad de nivel prefijado de saturación que se añade (en tanto por uno)

3.4.2. PUERTA DE RUIDO (NOISE GATE).

3.4.2.1 PRINCIPIOS Y MODELADO.

Otra forma de distorsión es la denominada Puerta de Ruido. Así este efecto consiste en ignorar todas las partes de una forma de onda por debajo de un **umbral** especificado. Esto permite únicamente a los sonidos más fuertes, o a las partes más fuertes de un sonido, pasar a través de la unidad de distorsión. Un efecto interesante se consigue para aquellos sonidos cuya forma de onda cruza frecuentemente el umbral prefijado, tal y como se aprecia en la figura12

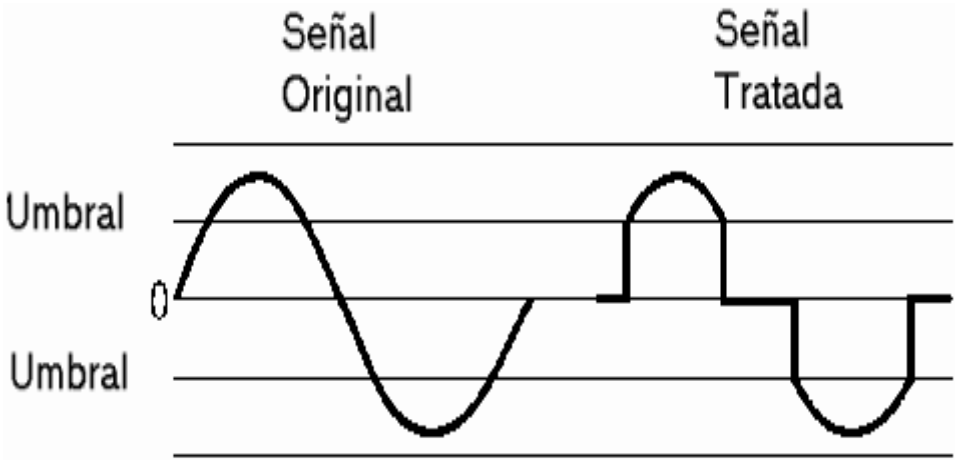


Figura 12. Puerta de Ruido. Señal que cruza el umbral frecuentemente.

En la figura 13 se observa como los sonidos débiles se suprimirán por completo.

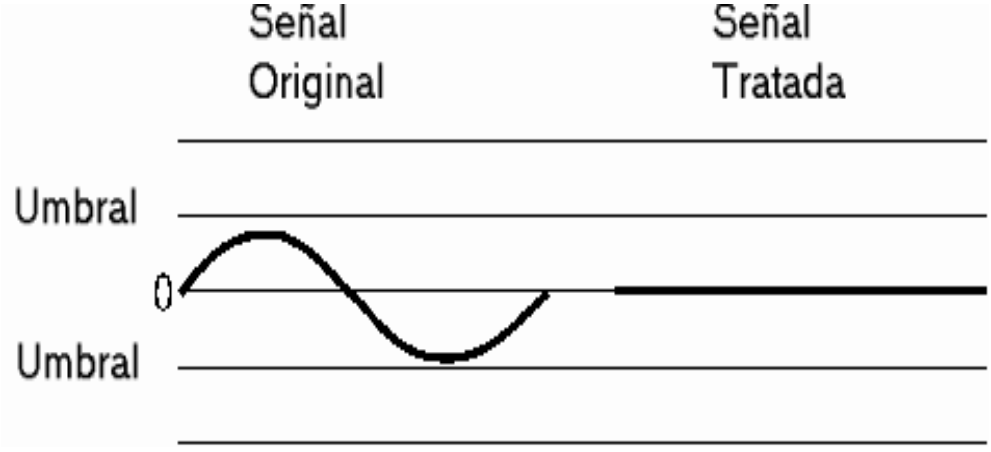


Figura13. Puerta de ruido. Señal débil eliminada completamente.

De esta forma la puerta de ruido queda caracterizada:

Si

+umbral>x(n)>-umbral (4)

entonces:

y(n) = 0 (5)

Si la condición (4) no se cumpliera:

y(n) = x(n) (6)

Esta atenuación total por debajo del umbral se emplea para eliminar bajos niveles de ruido en partes silenciosas de una grabación. Este efecto es usado más bien como herramienta de edición y refinamiento de grabaciones, aunque como ya se dijo anteriormente, se consigue un resultado interesante desde el punto de vista creativo para el músico, con sonidos que crucen el umbral prefijado frecuentemente tal y como se muestra en la figura 12.

3.4.2.2. IMPLEMENTACIÓN.

De nuevo tenemos un algoritmo realmente sencillo de implementar en un microprocesador, pues el procesado por muestra se reducirá a una comparación con + umbral, otra con –umbral y en caso de que alguno de ellos no se supere, se anula la señal de salida. Si esto no es así, la señal queda intacta a la salida y no se opera sobre ella. El único parámetro de control por parte del usuario será **umbral.**

Es importante señalar, que la puerta de ruido queda comprendida como caso particular del expansor, que se explicará más adelante. No obstante por claridad, se exponen separadamente.

3.4.3. COMPRESOR.

3.4.3.1 INTRODUCCIÓN.

Como su nombre indica, la compresión reduce el rango dinámico de una señal. Se usa extensivamente en trabajos de producción, reducción de ruido y para grabaciones en directo, pero debe usarse con cuidado. Comúnmente se ha dicho que los compresores hacen los sonidos fuertes, débiles y los débiles más fuertes, pero esto en realidad es correcto sólo a medias.

3.4.3.2. PRINCIPIOS Y MODELADO.

Un compresor es un dispositivo de ganancia variable, donde el nivel de ganancia usada dependerá del nivel alcanzado por la señal de entrada Así, la ganancia será reducida cuando la señal de entrada esté por encima de un **umbral** fijado y no será modificada cuando esté por debajo.

La figura 14 muestra el diagrama de sistema del compresor:

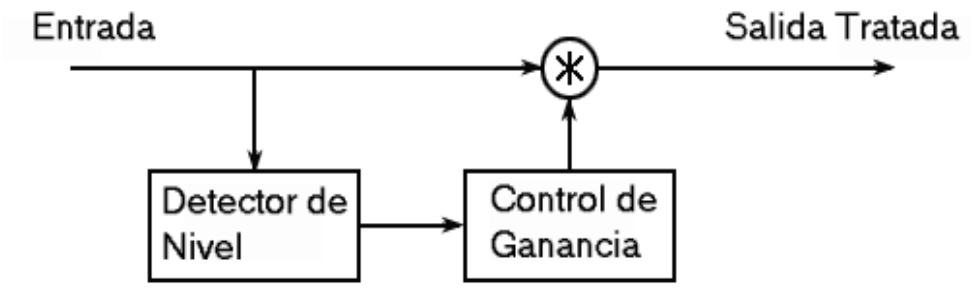


Figura14- Diagrama de sistema del compresor.

La relación entre la entrada y la salida del compresor queda descrita en términos de fuerza del sonido (dB) en la figura 15 donde se muestra su función de transferencia. Los 45º de pendiente corresponden a una ganancia unidad, y el sistema es completamente transparente aquí. Así , el compresor cambiará la pendiente haciéndola menor que 1 e igual a la **nueva pendiente** prefijada en la función de transferencia, si la intensidad de la entrada supera un valor igualmente prefijado de nominado **umbral.**

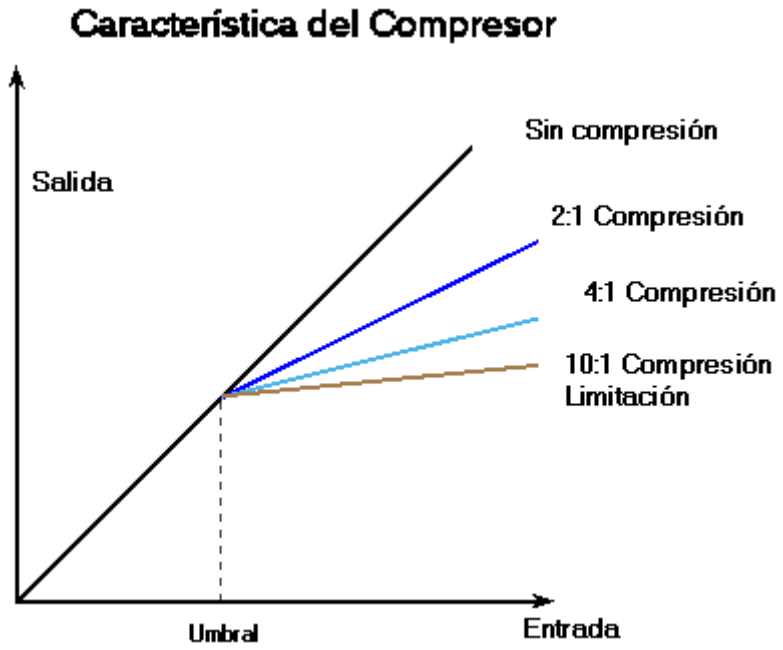


Figura15- Función de transferencia del compresor (en dBs).

Obsérvese que llevando la compresión al límite, esto es fijando una pendiente de cero para cuando el umbral es superado, se obtiene un **Limitador** , con el que se obtendrá un efecto de distorsión aprovechable desde el punto de vista creativo.

Así, observando la figura 15, se puede apreciar que un compresor hace más débiles las señales fuertes pero no hace las señales débiles más fuertes (aunque pueda percibirse de esta manera). De esta forma, la mayoría de los compresores tienen un control de ganancia adicional independiente al automático expuesto, que sirve para prevenir una compresión excesiva que pueda causar que el volumen del instrumento sea insuficiente. Pues bien, es esta ganancia extra la que produce el efecto de amplificar los sonidos más débiles .

Otro parámetro característico es la cantidad de tiempo que el compresor tarda en responder una vez que se ha superado el umbral para la entrada; esto se denomina **tiempo de ataque**, y es usualmente alrededor de 100mseg. Igualmente, cuando la entrada está por encima del nivel umbral y cae por debajo de él, al compresor le llevará algo de tiempo el incrementar la ganancia a 1 de nuevo. Este es el denominado **tiempo**

de relajación, que es generalmente mayor que el tiempo de ataque (alrededor de 1 o 2 segundos). En la figura 16 se muestra cómo trabaja el efecto con todos los parámetros.

A veces puede ser deseable tener un tiempo de ataque o de relajación muy corto lo que ocasiona un cambio en la ganancia muy brusco, que puede ser incluso oído como lo que se conoce como ‘ breathing’ o ‘pumpimg’. Cuando el nivel de sonido cae por debajo del umbral, la ganancia se incrementa a uno y la señal de entrada está ahora más próxima al nivel de ruido del sistema pudiendo ser audible.

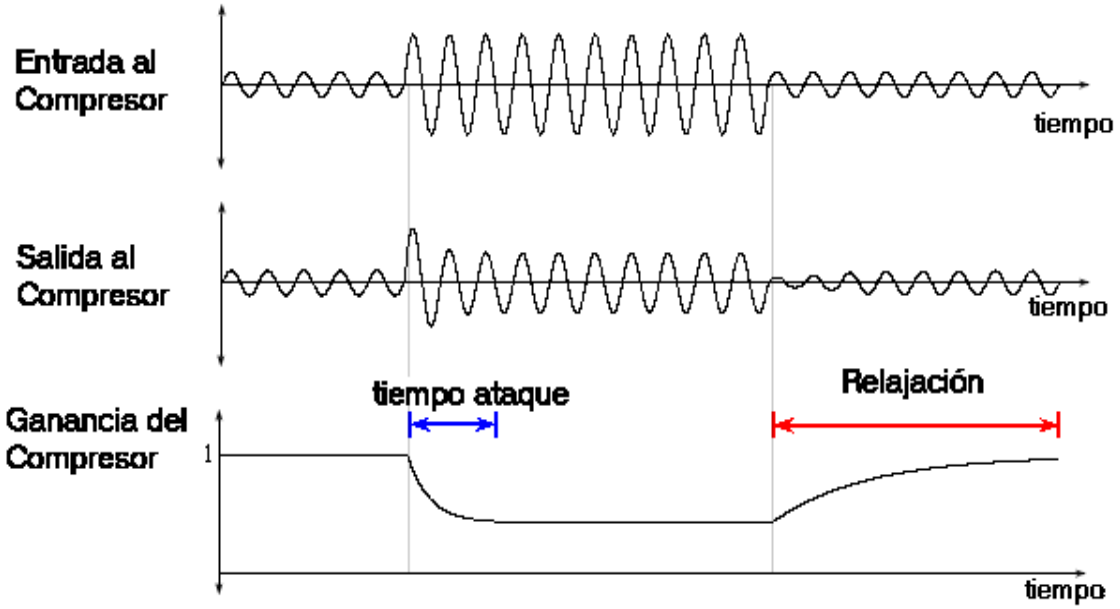


Figura 16- Compresor funcionando con todos sus parámetros característicos.

El compresor queda caracterizado digitalmente en su modelo más simple, esto es teniendo sólo en cuenta umbral y nueva pendiente como sigue:

Si x(n) >+umbral

$$y(n) = (+umbral) + m^* [x(n)-(+umbral)] \quad (7)$$

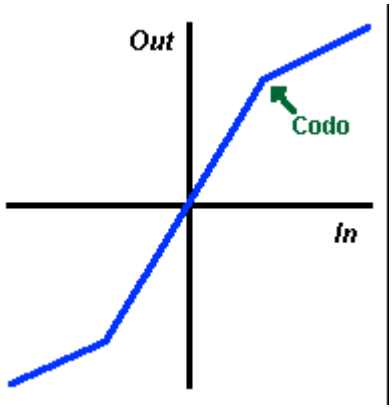
Si x(n)<-umbral

$$y(n) = (-umbral) + m^* [x(n)-(-umbral)] \quad (8)$$

Si no se aplicó (7) ni (8)

$$y(n) = x(n) \quad (9)$$

Donde debe hacerse notar que m es la nueva pendiente y es menor que 1. La diferenciación entre *+umbral* y *-umbral* se hace debido a que trabajamos con la función de transferencia, pero en términos numéricos relativos al valor de las muestras tal y como se hacía en los dos efectos anteriores debido a la simetría de las señales a tratar. Esto se muestra en la figura 17 .



Los umbrales deberán darse en tanto por uno del fondo de escala del ADC.

Figura 17- Función de transferencia del compresor (en términos numéricos relativos al valor de las muestras).

Un uso frecuente de la compresión es el de incrementar el 'sustain' de un instrumento. Así, el compresor tratará de mantener un nivel constante en la salida amplificando la señal de salida y manteniendo constante su nivel. De este modo , después de que una cuerda haya sido pulsada, el voltaje producido por las pastillas y muestreado por el sistema digital va atenuándose gradualmente hasta desaparecer. Un poco de compresión evitará que el instrumento cambie radicalmente después de haber sido pulsada la cuerda en cuestión, lo que es percibido como un incremento del 'sustain' del instrumento. Un tiempo de relajación mayor que la propia cadencia del instrumento preservará el sonido del mismo. Nótese que esta aplicación tiene un doble filo. Se puede ajustar el compresor para obtener el máximo 'sustain' posible, pero en el proceso se está eliminando la posibilidad por parte del guitarrista de acentuar ciertas frases o notas de la forma en que podría hacerlo sin ningún tipo de compresión. Desde luego, el músico puede experimentar con valores extremos en los parámetros del compresor para obtener sonidos inusuales.

Igualmente, se podrán simular distorsiones particulares , sustituyendo la función de transferencia lineal por otra específica a partir de un determinado umbral, con lo que la unidad pasaría a ser un dispositivo de distorsión multifuncional.

3.4.3.3. ÚLTIMAS NOTAS.

Se ha estado hablando de compresores que procesan la señal que está siendo usada en el proceso de detección de nivel. Pero en algunos casos, es preferible tener el nivel de una señal controlado por una señal diferente- así cuando el nivel de una señal es alto, la otra señal es atenuada, como se muestra en la figura 18 . Esto es lo que se llama 'ducking'. El ejemplo más común de uso de este nuevo tipo de compresor es el uso que hacen de él los DJ de radio. Así, mientras que la música suena, el hablar por el micrófono causará que el nivel de la música caiga de manera que sea más fácil entender lo que el DJ está hablando.

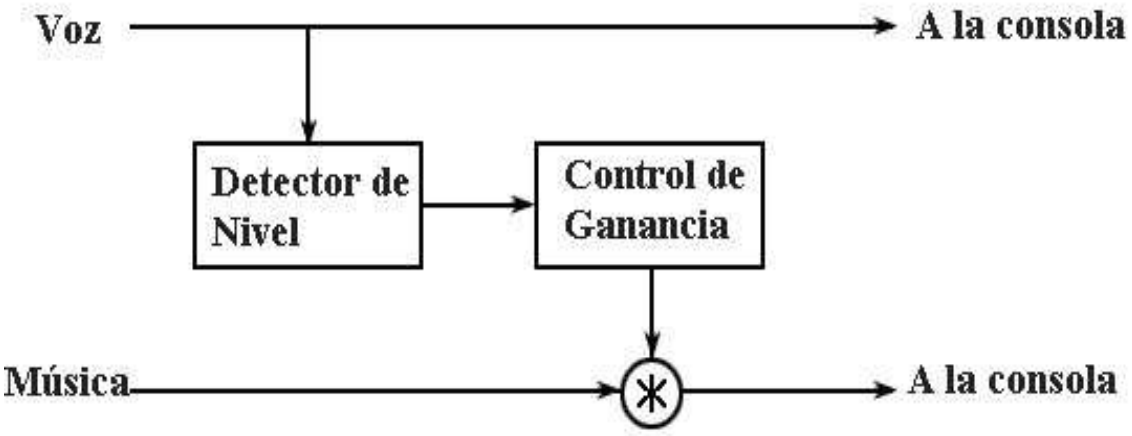


Figura18- Compresor 'ducking'.

3.4.3.4. IMPLEMENTACIÓN.

Implementar un compresor que contemple los parámetros básicos de umbral y nueva pendiente a aplicar si se supera el umbral, es sencillo en el microprocesador.

Como se muestra en las ecuaciones (7), (8) y (9) el procesado por muestra consistirá en hacer un par de comparaciones y en función del resultado de las mismas, bien no hacer nada, o efectuar un producto y dos adiciones correspondientes a la ecuación de la recta a aplicar en cada caso . Los parámetros de control sobre los que podrá interactuar el usuario será:

Umbral.

Nueva pendiente a aplicar por encima del umbral

Modelos más complejos pueden ser desarrollados para tener en cuenta parámetros como el tiempo de ataque y el tiempo de relajación para obtener un efecto más realista .

3.4.4. EXPANSOR.

3.4.4.1 INTRODUCCIÓN.

El expansor es un tipo de efecto basado en el rango dinámico. Como su nombre indica, expande el rango dinámico de la señal de manera que las señales débiles son atenuadas mientras que las porciones más fuertes no son ni atenuadas ni amplificadas. Este comportamiento es opuesto al del compresor. La puerta de ruido ya descrita anteriormente no es otra cosa que la expansión llevada al extremo, de manera que la atenuación de las señales débiles sea total, dejando en su lugar únicamente silencio.

3.4.4.2. PRINCIPIOS Y MODELADO.

Al igual que el compresor, el expansor es esencialmente un amplificador con ganancia variable. La ganancia nunca es mayor que uno, y es controlada por el nivel de la señal de entrada. Cuando el nivel de la señal de entrada es alto, el expansor presenta ganancia unidad, y cuando el nivel de ésta cae, la ganancia decrece, haciendo a la señal de entrada aparecer a la salida atenuada. El diagrama de sistema que presenta este dispositivo es idéntico al del compresor (figura 14).

Razonando de forma análoga a como con el compresor, la relación entre la entrada y la salida del expansor queda descrita en términos de fuerza del sonido (dB) en la figura 19 donde se muestra su función de transferencia. Los 45º de pendiente corresponden a una ganancia unidad, y el sistema es completamente transparente aquí. Así, el compresor cambiará la pendiente haciéndola menor que 1 e igual a la **nueva pendiente** prefijada en la función de transferencia, si la intensidad de la entrada está por debajo de un valor prefijado denominado **umbral.** Este decremento de la ganancia expande el rango dinámico de la señal, de ahí el nombre de expansor.

Cuando un expansor es llevado a su extremo, esto es la ganancia por debajo del umbral prefijado es prácticamente cero, se obtiene la **puerta de ruido** o **Noise Gate** . En este caso la señal es casi completamente eliminada o eliminada del todo. Como ya se describió previamente la puerta de ruido actúa como una especie de interruptor sonoro. Cuando la señal es lo suficientemente fuerte, el interruptor está encendido y la salida es igual a la entrada en cada momento, pero cuando la señal de entrada tiene un valor pequeño, la salida del expansor es cero. El parámetro fundamental en una puerta de ruido es por tanto el umbral que se use para por debajo de él hacer la atenuación completa de la señal.

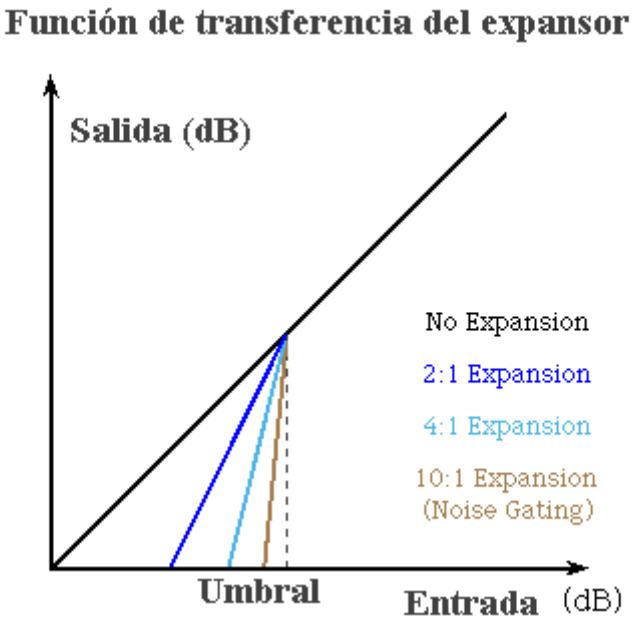


Figura 19- Función de transferencia del expansor(en dBs).

Al igual que ocurría con el compresor , el expansor se caracterizará por sus tiempos de ataque y relajación respectivamente . **El tiempo de ataque** es el tiempo requerido por el expansor para restaurar la ganancia a uno, una vez que la señal de entrada ha superado el umbral prefijado. Análogamente, el tiempo que tarda el expansor en reducir su ganancia una vez que la entrada cae por debajo del umbral se **denomina tiempo de relajación** . Ambos tiempos proporcionan al expansor un cambio más suave en la ganancia. La figura 20 muestra como opera un expansor con todos sus parámetros.

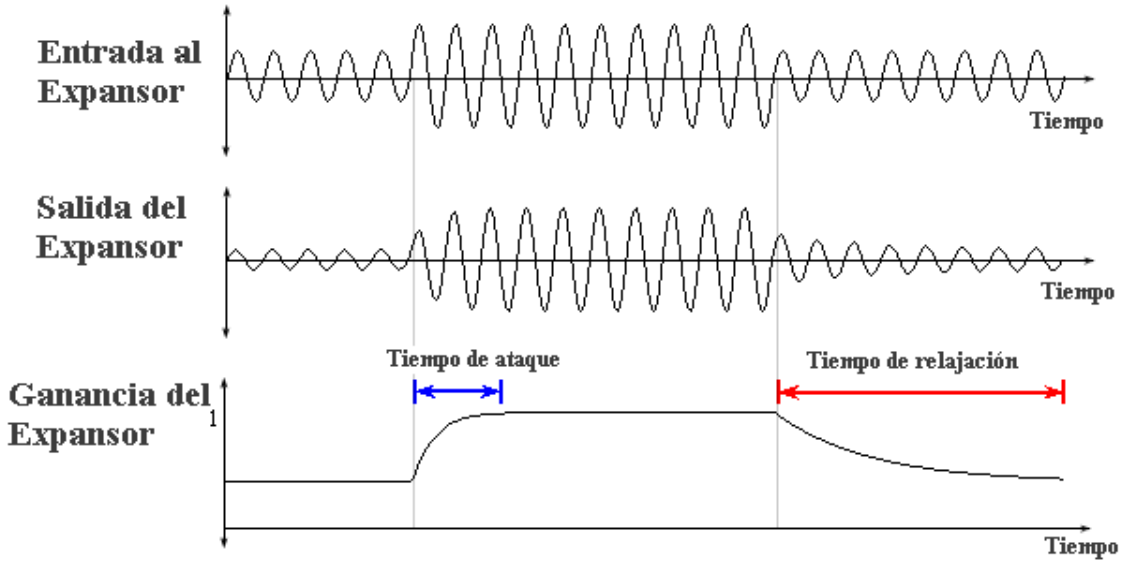


Figura 20- Expansor funcionando con todos sus parámetros característicos.

El expansor queda caracterizado digitalmente en su modelo más simple, esto es teniendo sólo en cuenta umbral y nueva pendiente como sigue:

Si +umbral>x(n) >-umbral y x(n) >0

$$y(n) = m^* x(n) \qquad (10)$$

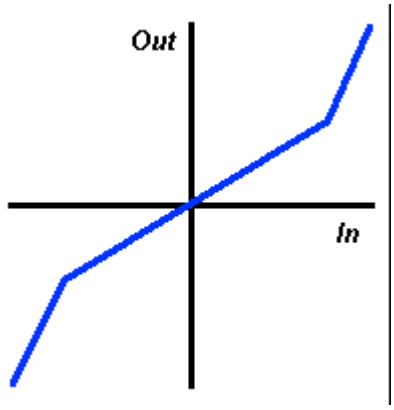
Si +umbral>x(n) >-umbral y x(n) <0

$$y(n) = m^* x(n) \qquad (11)$$

Si no se aplicó (10) ni (11)

$$y(n) = x(n) \quad (12)$$

Donde debe hacerse notar que m es la nueva pendiente y es menor que 1. La diferenciación entre *+umbral* y *-umbral* se hace debido a que trabajamos con la función de transferencia pero en términos numéricos relativos al valor de las muestras tal y como se hacía en los efectos anteriores debido a la simetría de las señales a tratar. Esto se muestra en la figura 21 .



Los umbrales deberán darse en tanto por uno del fondo de escala del ADC.

Figura 21- Función de transferencia del expansor (en términos numéricos relativos al valor de las muestras).

En cuanto a las aplicaciones de los expansores se refiere, éstos son usados para aumentar el rango dinámico en grabaciones procedentes de discos de vinilo o casetes, soportes que poseen un rango dinámico muy limitado.

Pero la aplicación más característica de los expansores es probablemente la reducción de ruido. Ayudan a reducir el efecto de realimentación indeseado. Las puertas de ruido son comúnmente utilizadas para eliminar ruido que puede ser sin ellas amplificado y oído cuando el músico está tocando su instrumento. Es importante señalar que el umbral debe ser lo suficientemente alto como para que el ruido ambiente caiga por debajo de él, pero no tan alto que el sonido del instrumento se vea perjudicado.

3.4.4.3. IMPLEMENTACIÓN.

Implementar un expansor que contemple los parámetros básicos de umbral y nueva pendiente a aplicar si no se supera el umbral, es sencillo en el microprocesador. Como se muestra en las ecuaciones (10), (11) y (12) el procesado por muestra consistirá en hacer un par de comparaciones y en función del resultado de las mismas, bien no hacer nada, bien efectuar un producto correspondiente a la ecuación de la recta . Los parámetros de control sobre los que podrá interactuar el usuario será:

Umbral.

Nueva pendiente a aplicar por debajo del umbral.

Modelos más complejos pueden ser desarrollados para tener en cuenta parámetros como el tiempo de ataque y el tiempo de relajación para obtener un efecto más realista .

3.4.5. MODULADOR EN ANILLO.

3.4.5.1. INTRODUCCIÓN.

Un modulador en anillo es un dispositivo que puede ser usado para crear sonidos curiosos e inusuales a partir de la salida de un instrumento. Efectivamente, toma dos señales (cada una de una frecuencia) y produce una señal que contiene la suma y la resta de esas frecuencias (bandas laterales). Estas frecuencias serán típicamente no armónicas, de manera que el modulador en anillo puede crear algunos sonidos muy disonantes. Por esta razón, la modulación en anillo no es un efecto ampliamente extendido.

3.4.5.2. PRINCIPIOS Y MODELADO.

Modulación significa que se está cambiando algún aspecto de un tono, tal como la amplitud, la frecuencia o la fase. En un modulador en anillo, se usa la modulación de amplitud (concretamente de portadora suprimida), la cual se modela simplemente multiplicando dos señales tal y como se muestra en la figura 22.

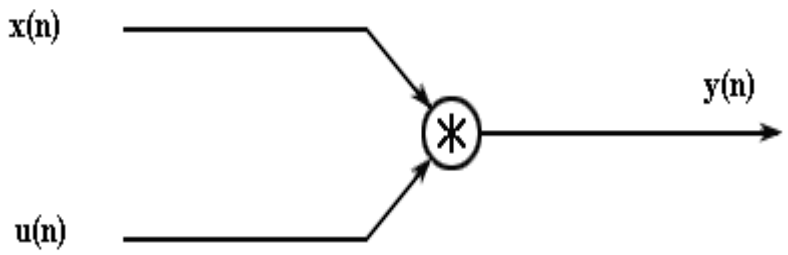


Figura22 - Diagrama de sistema del modulador en anillo.

Esta multiplicación da como resultado, una nueva señal que contiene frecuencias diferentes que las propias de las señales originales. Más específicamente, la salida contiene notas a la suma y a la diferencia de las frecuencias de las dos señales involucradas. Estas nuevas frecuencias que aparecen reciben el nombre de **bandas laterales,** siendo la suma de las frecuencias de las señales, la banda lateral superior y la diferencia la banda lateral inferior.

El modulador en anillo no es un armonizador, en general, las notas producidas por el modulador no están relacionadas con las entradas por ningún intervalo musical o relación armónica. Así, el sonido producido como resultado es muy disonante e incluso chillón. Este sonido es descrito como un sonido de aspecto ‘ gong’ , pues las campanas normalmente contienen fuertes componentes no armónicas cuando suenan.

La figura 23 muestra lo que sucede con las frecuencias de las señales involucradas en el modulador. De este modo, se muestra un segmento de una onda senoidal de 400 Hz, otro de 600Hz y el producto de ambas. Observando la señal resultado del producto, se aprecia la componente de alta frecuencia (600+400=1000 Hz) añadida a la componente de frecuencia más pequeña (600-400=200Hz). Esta componente de 200 Hz, es la que da a la forma de onda resultante, la suave inclinación y ascenso que se aprecia en la figura. La señal resultante es el tono suma de las dos bandas laterales (1000Hz + 200 Hz).

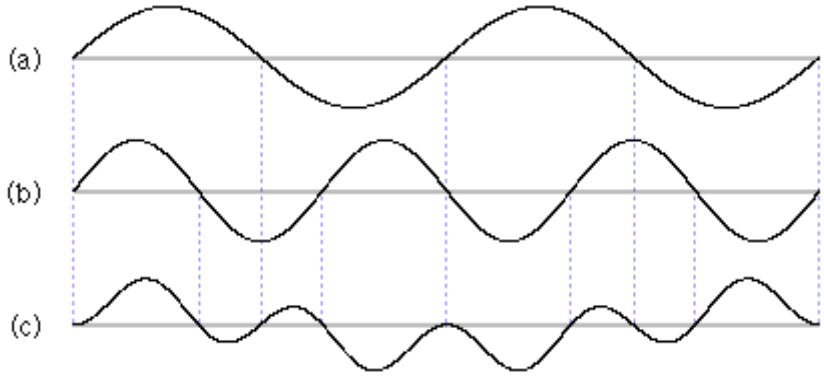


Figura 23- a) 400Hz, b) 600Hz, c) componentes de 200Hz y 100Hz sumadas.

Muchos de los moduladores en anillo existentes tan sólo tienen una entrada, destinada a la conexión de un instrumento. La otra señal es creada normalmente por algún oscilador interno (la cual es considerada la portadora . Esta señal no aparece en la salida, de ahí el nombre de portadora suprimida). En algunos casos se podrá seleccionar la frecuencia del oscilador y en otros, la frecuencia se fijará a algún valor, o limitada a un grupo de valores fijos. Pero no hay ninguna razón por la cual ambas señales no puedan ser instrumentos o ambas osciladores.

Desde luego, el sonido de un instrumento es considerado como la suma de varias ondas senoidales diferentes (teorema de Fourier) , no un tono puro. De esta forma, el oscilador usado podría también generar una onda compleja. El número de tonos crecería rápidamente, pues cada componente produciría la suma y diferencia con todas las señales senoidales parciales contenidas en la otra señal, creando un sonido muy complejo y presumiblemente muy asonante.

La ecuación del sistema mostrado en la figura 22 es muy sencilla:

$$y(n) = x(n) * u(n) \qquad (13)$$

donde *u(n)* es la señal procedente del oscilador en el sample n. Este oscilador tendrá una frecuencia normalmente ajustable por el usuario para obtener distintos matices del efecto.

Usar la salida de un modulador en anillo sin más podría resultar inconveniente dependiendo de los gustos del oyente. Así, el sonido puede suavizarse un poco mezclando la salida del modulador con el instrumento original, lo cual dará al instrumento un timbre peculiar más suavizado.. La ecuación (13) quedaría ahora como:

$$y(n) = x(n) * Mix + RinGain*[x(n) * u(n)] \quad (14)$$

donde *Mix* controla la cantidad de señal del instrumento original se mezcla con la salida del modulador en anillo y RinGain la cantidad de señal tratada por el modulador que interviene en la mezcla final.

3.4.5.3. IMPLEMENTACIÓN.

Crear un modulador en anillo basado en un procesador digital es tan simple como realizar tres multiplicaciones y una adición cada intervalo de muestreo si se va a implementar el modelo completo de la ecuación (14). El oscilador interno deberá implementarse como subrutina, de la cual se extraerá un valor de la portadora para cada intervalo de muestreo.

No obstante, se deben hacer consideraciones acerca de las señales usadas porque el aliasing puede crear una salida con ruido. Así, debe recordarse que la máxima frecuencia de la salida es la suma de las componentes más altas de cada señal usada. De este modo, y particularizando para el caso concreto de una guitarra eléctrica o acústica, se conoce que éstas sólo tendrán componentes apreciables hasta aproximadamente los 10 Khz. Es por ello por lo que habrá que prevenir un valor de la frecuencia de muestreo Fs para que cumpla el teorema del muestreo, sabiendo que se producirán componentes de peor caso a (*10.000 + FrecuenciaMax del modulador) Hz.* Recuérdese que el teorema de las mínimas muestras establecía que la frecuencia de muestreo debía ser al menos el doble de la componente de mayor frecuencia de la señal procesada que se pretende reconstruir. Un buen valor para la frecuencia Fs es el adoptado de 22Khz, que permitiría usar frecuencias de oscilación para el modulador en anillo de hasta 1000Hz

3.5. EFECTOS BASADOS EN EL USO DE RETARDOS.

Este tipo de efectos están basados en el uso de muestras y/o salidas pasadas para generar la salida en el momento actual. Es necesario por tanto el uso de memorias, concretamente de tablas circulares o colas para su implementación digital. Son de un gran interés para el músico pues los resultados creativos son muy buenos. A continuación se describen los más usados.

3.5.1. DELAY.

3.5.1.1. INTRODUCCIÓN.

El delay es uno de los efectos más simples de los que usan retardos pero es tremendamente valioso usado adecuadamente. Así, un poco de delay puede "dar vida" a mezclas apagadas, ampliar el sonido de un instrumento e incluso permite tocar un sólo sobre uno mismo. La unidad de delay es también un elemento constructivo básico para la elaboración de otros efectos más complejos como reverb, chorus y flanger.

3.5.1.2. PRINCIPIOS Y MODELADO.

En su formulación más simple, la unidad de delay toma una señal de audio y la reproduce de nuevo después del **tiempo de delay,** escalada y mezclada con la señal actual**.** Este tiempo de retraso puede oscilar en el rango de varios milisegundos (efecto parecido a un coro) hasta varios segundos (produciéndose una repetición del sonido bien definida y apreciable de forma separada a la reproducción del sonido original). La figura 24 presenta el diagrama de sistema de la unidad de delay en su modelo básico.

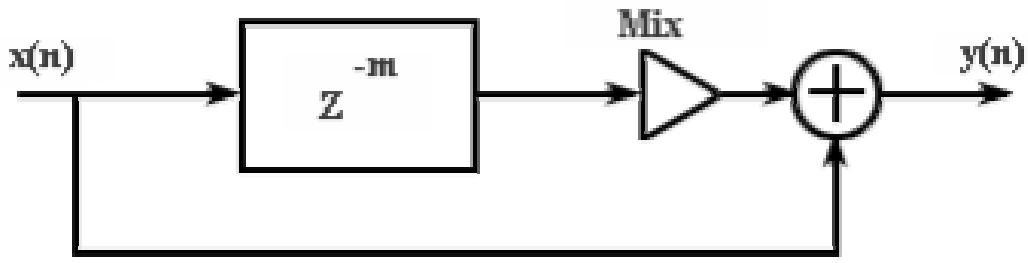


Figura 24- Unidad de delay básica.

Este modelo produce únicamente una copia del sonido original, de modo que este dispositivo es también denominado **de eco discreto.** La ecuación en diferencia que define al modelo es:

$$y(n) = x(n) + Mix\ x(n-m) \qquad (15)$$

Donde ***Mix*** controla la cantidad de eco que se percibe y ***m*** representa el número de samples de retraso para el eco. Recuérdese que el tiempo correspondiente a m samples de retraso es $t_{delay} = m * Ts$, donde *Ts* es el período de muestreo.

Este modelo presenta dos utilidades fundamentales:

1º. Como slapback delay. Se debe fijar un tiempo de delay muy pequeño (entre 40 y 120 milisegundos). Con ello se consigue un efecto de coro poco elaborado, como si dos instrumentos estuvieran tocando al unísono. Para que el efecto sea enteramente realista, el tiempo de delay deberá ser variable, para producir una modulación en tono que diferencie a los dos instrumentos que tocan a la vez (ver chorus más adelante).

2º Eco discreto. Tenemos este efecto cuando el retardo es de 100-120 milisegundos en adelante. Tiempos de retardo de 1 segundo o mayores pueden usarse para tocar encima de notas pasadas, de manera que el tiempo de delay se haga corresponder con el compás de una canción, de forma que las copias retrasadas del sonido entren en dicho compás. Delays de 1 segundo o mayores dan por tanto, la posibilidad al músico de tocar sobre sí mismo y desarrollar armonías de otro modo imposibles.

Sólo tener un sencillo eco es más bien un efecto pobre, así que muchas unidades de delay incorporan realimentación (regeneración), que toma la salida del delay y la envía de vuelta a la entrada. Ahora, el dispositivo tiene la habilidad de repetir el sonido una y otra vez (efecto parecido a la reverberación), de forma que las copias del original se van atenuando cada vez que se repiten, si la ganancia de realimentación es menor que uno, condición que muchas unidades imponen por diseño para garantizar la estabilidad. Con este lazo de realimentación, el sonido es teóricamente repetido eternamente (al menos hasta que se apague la unidad). No obstante, después de un punto se volverá tan tenue que estará por debajo del ruido ambiente y será inaudible.

Para sintetizar esta nueva unidad, dos modelos equivalentes son posibles, tal y como se muestran en la figura 25-a) y 25-b):

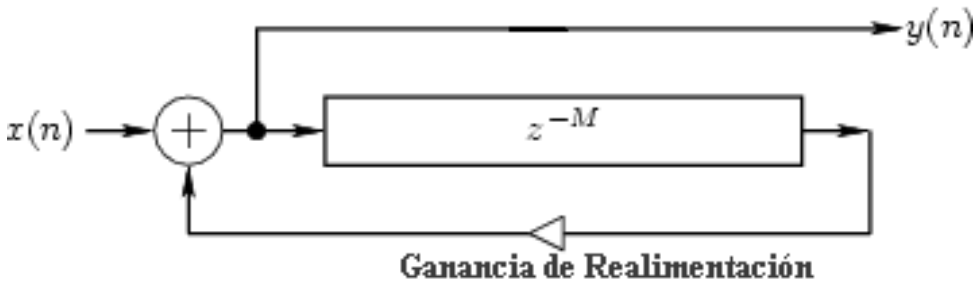


Figura 25-a) Delay realimentado 1.

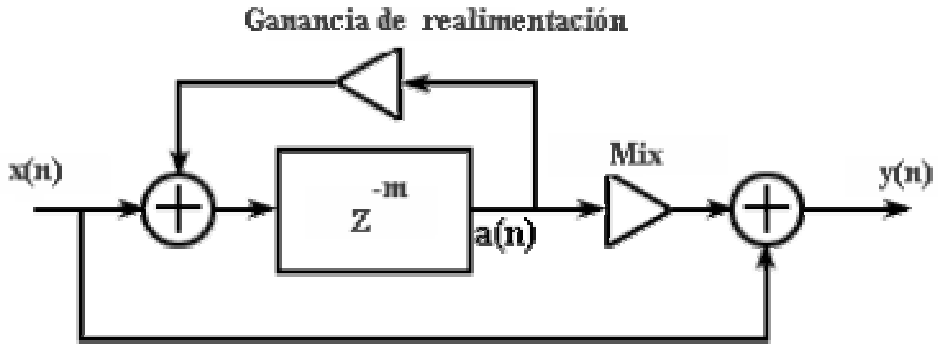


Figura 25-b) Delay realimentado 2.

Ambos modelos son equivalentes y se pueden reducir a la ecuación en diferencia del modelo 1:

$$y(n)= x(n)+ g\ y(n-m) \qquad (16)$$

Con *g= La ganancia de realimentación* para el primer modelo y *g=[mix*(x(n-m)+a(n-m)*ganancia de realimentación)]/[x(n-m)+a(n-m)*mix]* para el segundo, donde *m* es el retraso de la copia en samples y *Mix* controla la cantidad de eco que se percibe.

A la vista de (16) se aprecia que estamos tratando con un filtro recursivo (pues se usan salidas pasadas) para cuya estabilidad la ganancia de realimentación debe ser menor que uno tal y como se indicaba en la introducción teórica .

La diferencia entre ambos modelos reside en los distintos valores que toma la constante g, y se resume en que el modelo de delay realimentado 2 comprende al dispositivo de eco inicial sin más que hacer *Ganancia de realimentación =0* y permite más posibilidades sonoras mediante las combinaciones de valores de *Mix* y *Ganancia de realimentación* , mientras que el modelo de delay realimentado 1 es una unidad que no comprende al modelo básico y que sólo podrá funcionar como reverberador, pues si Ganancia de realimentación es igual a 0, el dispositivo simplemente se reduce a *y(n) = x(n).*

Así, el modelo realimentado 1 se usa para la simulación de reverberación sencilla, que consiste en repeticiones sucesivamente atenuadas de un sonido original y uniformemente espaciadas en el tiempo tal y como se muestra en la respuesta al impulso unitario para este filtro en la figura 26, para *g=Ganancia de realimentación = 0.9* y *m=1* .

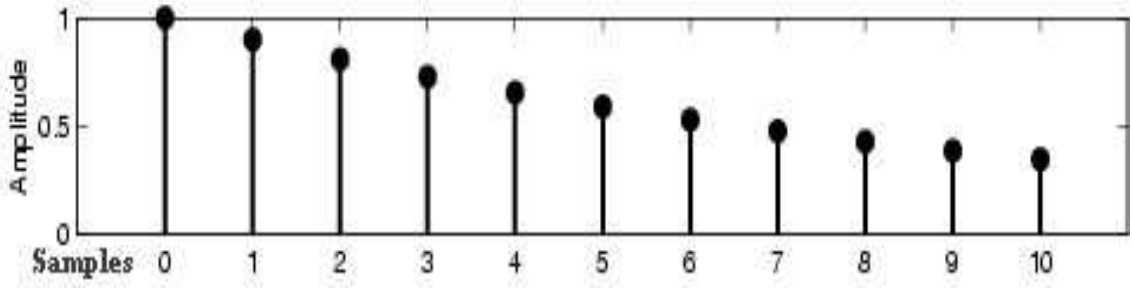


Figura 26 – Respuesta al impulso unitario de la unidad de delay realimentada para g=0.9 y m=1.

Por su parte , el modelo realimentado 2 se usa para unidades en las que se quiera tener la posibilidad de no sólo producir un eco discreto, sino además se quiera simular reverberación de forma análoga a la producida por el modelo 1. De esta forma, el reverberador quedará activado si *Ganancia de realimentación≠0*

3.5.1.3. OTRAS NOTAS.

Se reseñan a continuación otros tipos de delays que pueden ser de interés.

1º. Multi-Tap Delay. En algunos casos se puede desear mayor flexibilidad en una unidad de delay, y un multi-tap la ofrece. Este tipo de dispositivo es interesante porque permite crear patrones más complejos de ecos que pueden añadir una cualidad rítmica al instrumento.

En las líneas de delay tradicionales, la salida es tomada después de que la señal haya sido retrasada el tiempo total de delay prefijado, pero se puede también tomar la salida de forma que la señal haya sido retrasada tan sólo una porción del tiempo total de delay. Tomar la salida de puntos de dentro de la línea de retraso es lo que se denomina 'tapping' , como si fuera un grifo en una cañería que permite obtener agua en varios puntos a lo largo de la misma. Los distintos puntos se etiquetan normalmente con el número de punto y 'tap' . Así, los 'taps' no deseados pueden ser eliminados fijando el factor de ganancia del tap en cuestión a cero. La cantidad de delay entre varios 'taps' puede ser diferente.

Por tanto, una unidad multi-tap (TDL, tapped delay line) es una línea de retraso con al menos un 'tap', de manera que en dicha línea se puede extraer una señal en determinados puntos la misma. Así, cada 'tap' implementa un delay más corto dentro de una línea de retraso mayor. Un ejemplo de multi-tap delay de un único 'tap' se muestra en la figura 27, donde b_{M1} es la ganancia del único 'tap'.

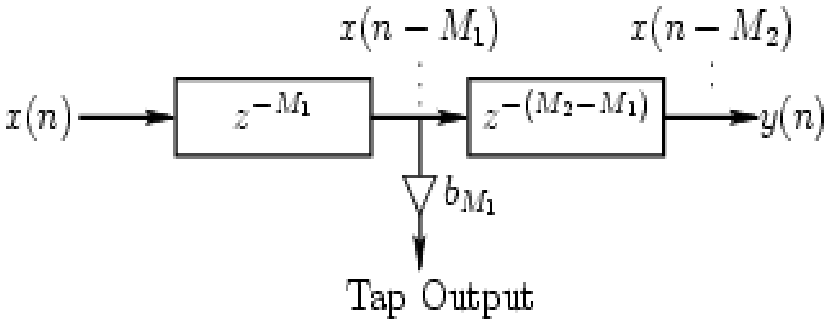


Figura 27- TDL de un único 'tap'.

Las unidades multi-tap simulan eficientemente múltiples ecos. Es por ello por lo que se usan extensivamente en la reverberación artificial más realista.

2º. Ping-Pon delays. Como su nombre indica, estas unidades producen un sonido balanceante, típicamente rebotando entre los canales derecho e izquierdo del estéreo. La figura 28 muestra el diagrama de sistema de una unidad de estas características.

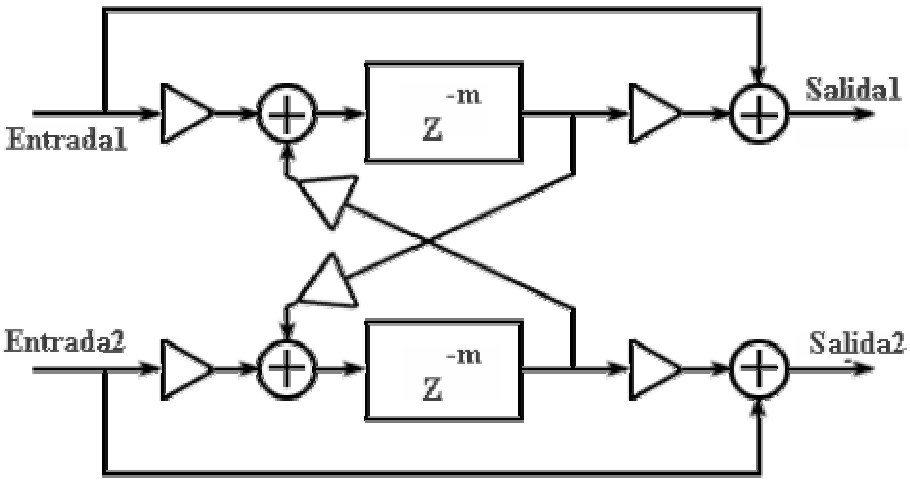


Figura 28- Ping-Pon Delay.

3.5.1.4. IMPLEMENTACIÓN.

Debido al accesible precio de las memorias, las unidades digitales de delay son ahora mismo rentables. El algoritmo a realizar es relativamente sencillo. Así en cada período de muestreo se lee un valor previamente almacenado en la memoria y se guarda en dicha memoria el valor actual de la señal de entrada en otra posición . El próximo período de muestreo se efectúa la misma operación pero leyendo de la posición siguiente a la que se leyó anteriormente y escribiendo en la siguiente a la que se escribió antes. De esta forma, cuando se alcance el final de la memoria, se realiza el mismo proceso pero volviendo a la primera posición de la misma.

En el ámbito del procesado de la señal, esto es lo que se denomina una tabla circular o cola circular, y es una estructura muy eficiente para este segundo gran bloque de efectos. Así, el microprocesador manejará típicamente un puntero de lectura y un puntero de escritura que irá marcando la posición en la que se almacenará la muestra actual o la salida actual según el modelo implementado y que servirá de 'pivote' para desde él, retrasar el puntero de lectura los samples correspondientes al tiempo de retraso requerido en cada período de muestreo. Estos punteros se incrementarán en cada período de muestreo. En caso de implementar un multi-tap delay se requerirán algunos punteros de lectura adicionales.

Los parámetros a controlar por el usuario serán:

- Tiempo de retardo.

- Mix.

3.5.2. CHORUS.

3.5.2.1. INTRODUCCIÓN.

El efecto de chorus consigue hacer que un único instrumento suene como si en realidad hubiera dos instrumentos sonando al unísono. Añade por tanto, algo de grosor al sonido, que se describe comúnmente como 'exhuberancia' o 'riqueza'.

3.5.2.2. PRINCIPIOS Y MODELADO.

El modelo que se encuentra detrás del efecto, no es realmente espectacular o un truco complicado. Así pues, planteémonos qué sucede si dos personas tocan dos instrumentos juntos o bien cantan juntos. No lograrán una precisa sincronización, y al sumarse los sonidos procedentes de ambos instrumentos, se detectarán algunos retardos entre un instrumento y el otro. Además, los tonos de ambos instrumentos diferirán algo. Pues bien, estos serán los efectos que el chorus tratará de simular tal y como se explica a continuación.

Así, el leve retraso puede ser fácilmente modelado con una línea de delay, con el modelo básico sin realimentación ya conocido.

Por otra parte, crear el efecto de leve desafinación, no parece tan simple a primera vista, pero puede lograrse transformando la línea de delay básica en una línea de retraso variable. De este modo, la longitud del retraso cambiará con el tiempo para crear la modulación de tono.

Para comprender como el tono es modificado, pensemos que el bloque de delay es un dispositivo de grabación. Éste almacena una copia exacta de la señal de entrada conforme llega , como si fuera un casette y después lleva a la salida la misma copia un tiempo después. Para incrementar la cantidad de retraso, se precisa un segmento mayor de la señal que esté almacenada en la unidad antes de que se reproduzca. Para lograr esto, leemos la señal retrasada de la línea de la línea de retraso a una tasa menor que la de escritura (la tasa de escritura- grabación no se altera). Así, leer a una tasa menor es como arrastrar los dedos en las ruedas del casette, que como se sabe, produce una bajada del tono.

En la otra cara de la moneda, para reducir el tiempo de delay, podemos leer a una tasa mayor que la escritura, efecto similar al de aumentar la velocidad del casette, cuyo resultado es un aumento del tono en la grabación.

Si ambos procesos son efectuados lo suficientemente rápido y secuencialmente, tenemos las ligeras variaciones de tono que se persiguen para el chorus.

Pasemos pues, al modelado del efecto. Así, mezclando esta copia retrasada y modulada en tono con la original, tenemos el efecto de chorus. El diagrama de sistema que modela este efecto se muestra en la figura 29.

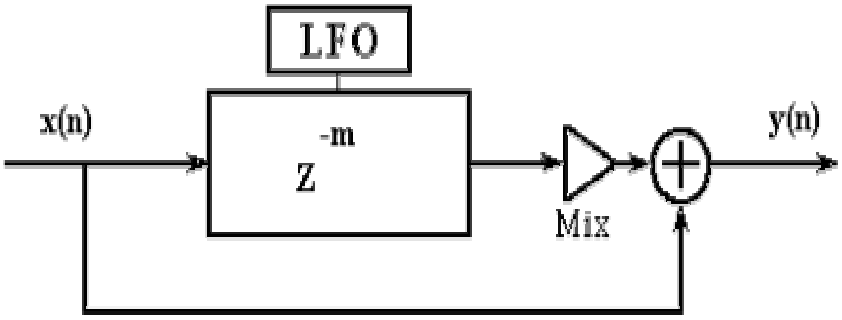


Figura 29 – Diagrama de sistema del efecto de chorus.

Obsérvese que la base del modelo, es la misma estructura del delay básico sin realimentación, pero con un tiempo de retraso típicamente entre 20-30 milisegundos, para lograr la superposición del sonido consigo mismo. El modelo de delay básico no produce las variaciones de tono que el efecto requiere pues el tiempo de delay es fijo, por lo que es necesario añadir un segundo elemento.

Este segundo elemento que completa el modelo de chorus es el encargado de variar el tiempo de retardo para conseguir la modulación del tono. En general, cualquier forma de onda periódica se podría usar, tal como una senoidal. Esta forma de onda debe cambiar lentamente (3Hz o menor frecuencia) y nos referiremos a ella como un LFO (oscilador de baja frecuencia).

Se puede controlar el efecto, cambiando la frecuencia de la forma onda, su amplitud y su forma. De esta manera queda completada la explicación del modelo de la figura 29. La ecuación en diferencia que caracteriza al modelo es:

$$y(n) = x(n) + Mix \; x(n-m) \qquad (17)$$

Donde ***Mix*** controla la cantidad de coro que se percibe y ***m*** representa el número de samples de retraso para el eco. Recuérdese que el tiempo correspondiente a m samples de retraso es $t_{delay} = m * Ts$, donde *Ts* es el período de muestreo. Este tiempo de retraso es el que varía ahora de acuerdo con el LFO, de forma que podemos escribir:

$$m = f(Base, Barrido, Frecuencia, Forma)_{LFO} \quad (18)$$

Es posible establecer variaciones en el modelado del efecto, como por ejemplo, en vez de usar un LFO se podría usar un generador de tiempos de retraso, variables y aleatorios.

3.5.2.2.1. PARÁMETROS COMÚNES.

1º. Delay. Este parámetro controla la cantidad de retraso usado, más específicamente en realidad controla el mínimo tiempo de retraso que se usa.

2º Profundidad de Barrido. Este parámetro controla cuánto varía el retraso total en el tiempo. Usualmente se expresa en milisegundos. Así:

Delay+ Profundidad de Barrido = Máximo delay usado (19)

Este control puede ser visto también como la amplitud pico a pico del LFO. En la figura 30 se ilustra la relación entre los dos parámetros anteriores y su significado.

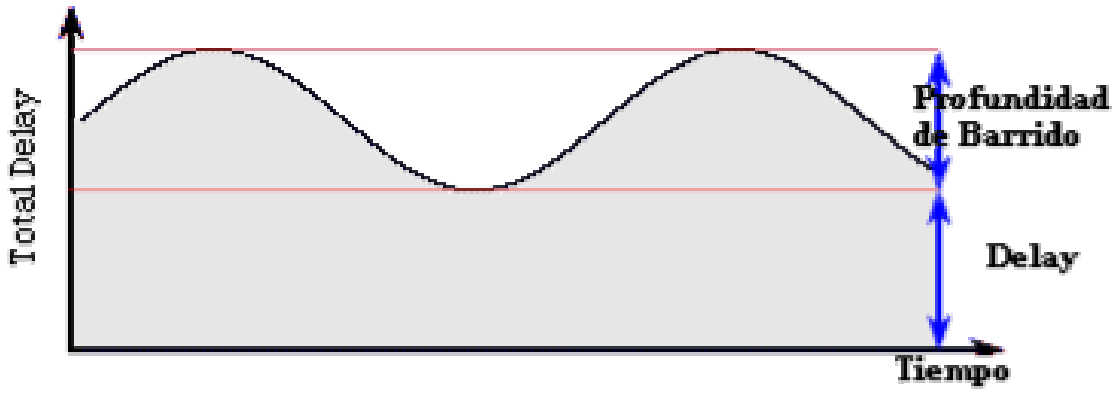


Figura 30- Caracterización del LFO.

La profundidad de barrido también incrementa la modulación de tono.

3º. Forma de onda del LFO. La forma de onda del LFO muestra como cambia el tiempo de retraso. Cuando la forma de onda alcanza su máximo, entonces el retraso es mayor.

Cuando la forma de onda del LFO se incrementa (aumenta el retraso) el tono se rebaja (revisar el ejemplo del casette) y cuando se decrementa (menor tiempo de retardo) el tono se incrementa. Las formas de onda más usadas para el LFO se muestran en la figura 31, siendo la comúnmente más empleada para el chorus, la senoidal.

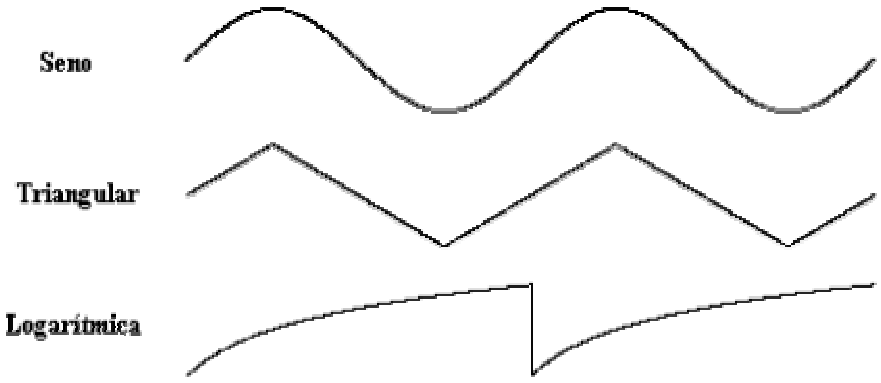


Figura 31 – Formas de onda para el LFO.

Para comprender como afecta la profundidad de barrido y la forma de de la onda del LFO a la modulación de tono debe tenerse en cuenta que la cantidad de modulación de tono introducida por el chorus, se relaciona con cómo cambia la forma de onda del LFO. Las partes más abruptas de la forma de onda producen una gran cantidad de modulación de tono, mientras que las partes relativamente llanas tienen un efecto muy tenue o casi nulo en la modulación de tono.

Podemos usar esta forma de razonar para comprender cómo la profundidad de barrido varía el tono. Si se incrementa la profundidad de barrido se está estirando verticalmente la forma de onda, lo que la hace más abrupta, de ahí que el tono se altere más. En cuanto a la forma de la onda usada en el LFO diremos:

- El seno es una función muy suave que está continuamente cambiando, así que el tono está también continuamente cambiando. Es la opción más usada en el chorus.

- La triangular por su parte, sólo produce dos tonos apreciables, porque la pendiente sólo tiene dos valores diferentes y el cambio entre los dos tonos es repentino.

- La logarítmica es suave en la mayor parte de su ciclo, pero presenta un salto al final del mismo. Debido a que la pendiente al principio y al final es diferente, existe un cambio abrupto también.

4º. Frecuencia de oscilación. Este parámetro es realmente intuitivo. Se refiere a la frecuencia de repetición de la forma de onda periódica usada por el LFO.

Siguiendo una línea de razonamiento análoga al caso de la forma y la profundidad de barrido, se llega a la conclusión de que incrementando la frecuencia se consigue una compresión en el tiempo de la forma de onda del LFO, lo que ocasiona una mayor modulación de tono.

3.5.2.3. OTRAS NOTAS.

Todo lo expuesto hasta el momento se refiere a una sencilla voz en chorus, ello significa que sólo tenemos una copia de la entrada. Pero no hay razón por la cual no se puedan sacar múltiples copias del sonido, modelando la situación con más de dos instrumentos en coro. Algunas unidades de chorus permiten hacer esto e incluso elegir cuántas voces usar.

Típicamente, un chorus multivoz usa un LFO único para todas las voces, pero cada voz tiene una fase diferente. Esto quiere decir que en un punto del tiempo cada voz está en un punto distinto de su forma de onda, de manera que tendrían distintos tiempos de delay. Si todas las voces estuvieran en fase, tendría el mismo efecto de una simple voz con el nivel incrementado. También es posible que cada voz tenga su propio LFO e incluso frecuencia. El modelo para un chorus multivoz se muestra en la figura 32.

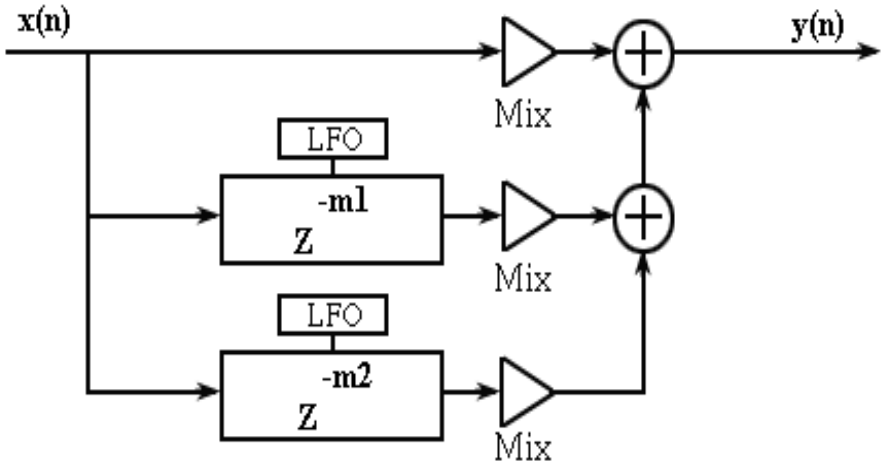


Figura 32- Chorus Multivoz(2 voces).

3.5.2.4. IMPLEMENTACIÓN.

Las unidades de delay básicas son fáciles de implementar en el dominio digital tal y como se describió anteriormente. De cualquier forma, un delay con tiempo de retraso variable según el modelo de chorus básico, conlleva más trabajo. Así, además de sintetizar el oscilador, el cambio en el tiempo de retraso requerirá tiempos que no son múltiplos enteros del periodo de muestreo. Así, es frecuente el efectuar una estimación del valor de la señal entre dos de los valores que tenemos almacenados. Con estos valores contiguos, podemos tomar una estimación del valor deseado, esto es una interpolación. La posibilidad más extendida es la de conectar estos dos valores por una línea recta y efectuar una interpolación lineal.

Este método es simple, pero muy efectivo a la hora de anular el 'zipper noise ' que se produciría sin interpolación. No obstante, otros métodos más complejos de interpolación suavizan aún más este ruido perjudicial y podrían ser usados para optimizar aún más la calidad del sonido.

Los parámetros típicos de control por parte del usuario serán:

- Mix.
- Todos los parámetros del LFO: Delay, profundidad de Barrido, frecuencia y forma de onda.

3.5.3. FLANGER.

3.5.3.1. INTRODUCCIÓN.

El efecto de flanger produce un sonido muy característico, al que mucha gente se refiere con el término inglés ‘ whoosing’ , o lo que es lo mismo, un sonido muy similar al del vuelo de un avión de motor a reacción volando sobre la cabeza del oyente.

El efecto de flanger se caracteriza porque crea un grupo de muescas igualmente espaciadas en el espectro de audio que se desplazarán armónicamente por él filtrando la señal de forma que se produce tal efecto.

3.5.3.2.PRINCIPIOS Y MODELADO.

Se cree que fue descubierto por accidente, se dice que los Beatles lo descubrieron mientras grababan. Así, un magnetofón estaba siendo usado como unidad de delay y alguien tocó el borde de una bobina de la cinta cambiando el tono. Con alguna mezcla más de señales, el efecto característico del flanger fue creado. El borde de una cinta magnética se designa en inglés por la palabra ‘flange’, de ahí el nombre del efecto.

El modelado del efecto de flanger es muy similar al del chorus, pero con tiempos de retardo menores y con el uso de una forma de onda típicamente triangular para el LFO.

Así, este efecto es modelado mezclando una señal con una copia levemente retrasada de sí misma, donde la longitud del retardo está constantemente cambiando. Por tanto, para la construcción del modelo definitivo, partimos del modelo de delay básico sin realimentar. Esta unidad no es otra cosa que un filtro digital en peine (no se había comentado hasta ahora por carecer de interés hasta ahora. Ver descripción de filtros peine más adelante) que produce una serie de muescas en la respuesta frecuencial. Para producirlas, basta con mezclar la muestra retardada con la entrada actual. ¿Por qué?, pues porque para algunas frecuencias el retraso de fase introducido

será de 180 grados, lo que es equivalente a añadir exactamente la entrada pero negativa. Así cuando se mezcla esta señal con la entrada, las frecuencias que experimentaron el retraso de 180 grados se cancelarán exactamente con las componentes originales de la entrada para tales frecuencias (**interferencia destructiva)**, originando las muescas para dichas frecuencias. El caso contrario se dará cuando el desfase producido sea de 360º en cuyo caso las señales original y retrasada se sumarán doblando su valor **(interferencia constructiva)**.

Pues bien, con un retraso tan corto como el característico del flanger, ni se escucha un eco , ni una superposición de sonidos. Lo que sucede es que las muescas propias de la respuesta en frecuencia de este filtro están muy espaciadas, produciendo un efecto de filtrado frecuencial apreciable (pues con tiempos de retardo mayores el filtrado de la señal juega un papel secundario desde el punto de vista de la percepción del sonido tratado).

El efecto completo se consigue cuando se varía este retraso en el rango típico de **1 milisegundo a 10 milisegundos**, lo que da lugar a la compresión y expansión de las muescas de la respuesta en frecuencia del modelo, movimiento éste que es precisamente lo que caracteriza al sonido del flanger. El la figura 33 podemos ver el diagrama de sistema correspondiente al modelo definitivo de flanger.

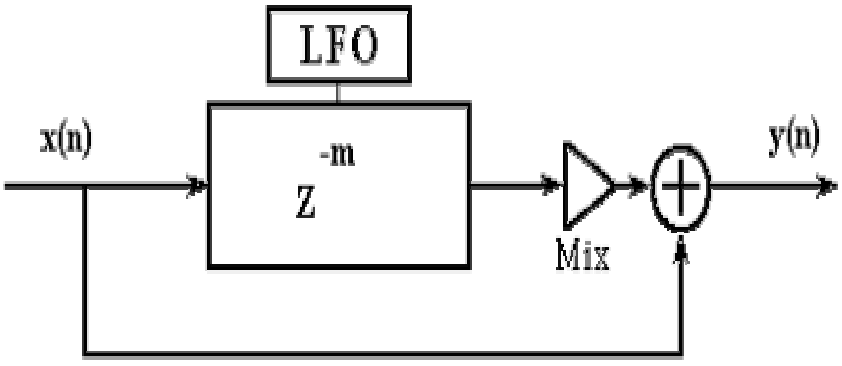


Figura 33- Flanger Básico.

En la figura 34 se muestran los extremos de máxima compresión de la característica frecuencial (para el mayor retardo) y el de mínima compresión (para el

menor retardo) que ilustran el movimiento de la característica frecuencial del cual hablábamos.

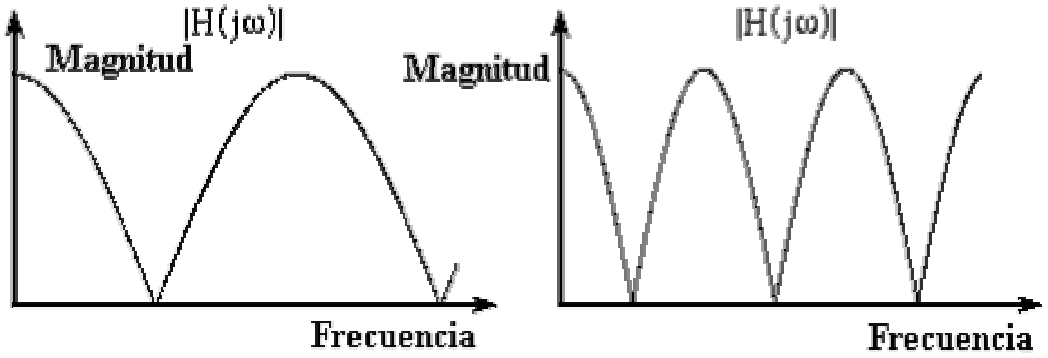


Figura 34- Movimientos de la característica frecuencial propios del Flanger.

El sonido característico del flanger resulta pues cuando las muescas de la respuesta van recorriendo el eje de frecuencia a lo largo del tiempo como resultado de la variación del retraso con un LFO a frecuencia de aproximadamente 0,5 Hz. Así, cuando el retraso se incrementa, las muescas se comprimen hacia las bajas frecuencias y cuando éste disminuye, dichas muescas se expanden hacia las altas frecuencias. La forma de onda del LFO será triangular preferiblemente o senoidal. Como ocurría en el chorus, el cambio del tiempo de retraso del flanger produce modulación de tono en la copia del sonido original; copia que luego será mezclada con la señal sin alterar.

Este movimiento de la figura 34 se ilustra en la animación del archivo de muestra **flanger2.mov.**

La ecuación en diferencia que caracteriza al modelo es:

$$y(n) = x(n) + Mix * x(n-m) \qquad (20)$$

Donde ***Mix*** controla la cantidad de efecto que se percibe y ***m*** representa el número de samples de retraso para la copia. Recuérdese que el tiempo correspondiente a m samples de retraso es $t_{delay} = m^* Ts$, donde *Ts* es el período de muestreo. Este tiempo de retraso es el que varía de acuerdo con el LFO, de forma que podemos escribir:

$$m = f(Base, Barrido, Frecuencia, Forma)_{LFO} \qquad (21)$$

3.5.3.2.1. PARÁMETROS COMÚNES.

1º. Mix. Cuando este parámetro es cero, la respuesta en frecuencia es plana (ganancia unidad), pero a medida que se incrementa acercándose a uno, las muescas empiezan a aparecer, extendiéndose hacia cero, alcanzándolo cuando *Mix=1*. Incluso cuando las muescas no se extienden hasta cero, el efecto de flanger es audible, siendo máximo cuando mix es igual a la unidad.

2º. Delay. Especifica el mínimo retraso usada en la copia de la señal de entrada. Razonando sobre la respuesta en frecuencia, este valor determina cómo de lejos (derecha en el eje de frecuencias) irá la primera muesca (recuérdese que a menor tiempo de retraso, menor compresión de las muescas).

3º. Profundidad de Barrido. Este parámetro determina cómo de amplio es el barrido en términos del tiempo de retraso, esto es la amplitud de pico a pico del LFO. Es el delay máximo adicional que se le añade al base para buscar la copia retrasada según la ecuación

Delay+ Profundidad de Barrido = Máximo delay usado (22)

Así, este parámetro fija el delay máximo, que a su vez determina cómo de cerca en el eje de frecuencias (recuérdese que a mayor retraso mayor compresión de las muescas) llegará la primera muesca.

Un valor pequeño para este parámetro mantendrá una pequeña variación en el tiempo de retraso, causando una compresión/expansión de la respuesta en frecuencia menos acusada, mientras que un valor elevado causará que las muescas se compriman y expandan a lo largo de un área mayor. Así mismo, a medida que la profundidad de barrido se incrementa, la modulación en el tono se vuelve más notable. La figura 30 ilustra la relación entre los parámetros 2º y 3º y es perfectamente válida aquí.

4º. Forma de onda del LFO. Algunas unidades de flanger permiten elegir la forma de onda usada por el LFO. Esta forma de onda determina como varía el retraso

para generar el efecto. La más usada y característica por el matiz que añade al efecto es la triangular, aunque puede usarse también la senoidal .

5º. Frecuencia. Controla la frecuencia de repetición de la forma de onda usada por el LFO. Afecta directamente a la modulación de tono producida, a mayor frecuencia, mayor modulación de tono en la copia.

6º Realimentación. Algunas unidades permiten tomar una porción de la salida del flanger y llevarla a la entrada. Una gran cantidad de realimentación puede crear un sonido muy metálico e intenso, con eco.

Es importante señalar que si este parámetro es mayor que uno , el sistema puede volverse inestable y resultar peligroso para la integridad del equipo.

De esta forma, incluyendo este lazo, el Flanger completo queda modelado de acuerdo con la figura 35. Para desactivar el lazo, basta con fijar *Ganancia de realimentación=0*

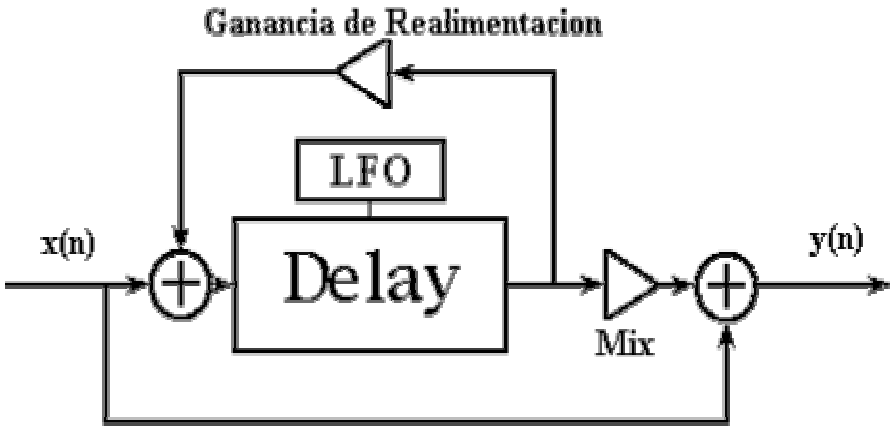


Figura 35- Modelo de Flanger completo.

3.5.3.3. OTRAS NOTAS.

Para instrumentos no de percusión que generan un tono y armónicos superiores, las muescas que el flanger produce podrían en teoría coincidir exactamente con la fundamental y los armónicos superiores, anulando a ambos y suprimiendo el instrumento.

En la práctica un instrumento no desaparecerá pero puede darse una seria modulación en amplitud. Por esta razón es mejor usar flangers en instrumentos de percusión. También destacar que el flanger se usa más para retocar mezclas finales que para un sólo instrumento, aunque se consiguen muy buenos resultados con guitarras eléctricas.

3.5.3.4. IMPLEMENTACIÓN.

Todo lo dicho en el apartado sobre la implementación del chorus es aplicable aquí, incluida la necesidad de interpolar. La única diferencia será la implementación del lazo de realimentación. Los parámetros típicos de control por parte del usuario serán:

- Mix.

- Ganancia de Realimentación.

- Todos los parámetros del LFO: Delay, profundidad de Barrido, frecuencia y forma de onda.

De esta forma, en una única unidad general según el modelo de la figura 35, se pueden implementar ambos efectos. La actuación de uno u otro efecto dependerá de los parámetros introducidos. Así, se tendrá chorus si Ganancia de *Realimentación=0 , delay =20* milisegundos y *barrido =10* milisegundos (parámetros típicos) y un Flanger si *delay=1* milisegundo y *barrido=9* milisegundos para cualquier *Ganancia de Realimentación menor que uno*.

3.5.4. PHASER.

3.5.4.1. INTRODUCCIÓN.

El phaser consigue su sonido característico creando una o más muescas en el dominio de le frecuencia que eliminan los sonidos correspondientes a las frecuencias donde se encuentran dichas muescas. Así, el phaser usa el efecto de filtrado producido por las muescas en el dominio de la frecuencia al igual que el flanger. De hecho el flanger es un tipo especial de phaser. Estas muescas son creadas simplemente filtrando la señal y mezclando la salida del filtro con la señal de entrada.

Los filtros pueden ser diseñados de manera que podamos controlar independientemente la localización de cada muesca, el número de muescas e incluso la amplitud de las mismas. Esto puede desembocar en muchas e interesantes posibilidades sonoras.

3.5.4.2. PRINCIPIOS Y MODELADO.

Las muescas necesarias para conseguir el efecto de phaser son normalmente implementadas usando un grupo especial de filtros llamados Allpass filters. Como su propio nombre indica, el Allpass deja pasar igual todas las frecuencias (Ganancia unidad). De esta manera, si se introduce una señal senoidal en un Allpass, se verá a la salida otra senoidal, de la misma amplitud que la de entrada.

Para completar el phaser, sólo añadimos a la salida del filtro la señal de entrada, tal y como se muestra en la figura 36.

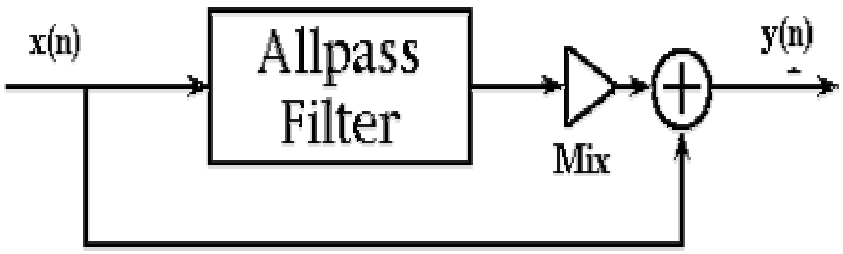


Figura 36. Modelo básico de phaser.

La ecuación para este modelo:

$$y(n)= [y´(n)* Mix] + x(n) \quad (23)$$

Donde *y´(n)* es la salida del Allpass y *Mix* controla la cantidad de efecto que se percibe. Todo lo expuesto es sin duda interesante, pero si el filtro deja pasar todas las frecuencias por igual, ¿cómo altera al sonido entonces ? ¿ y cómo aparecen las muescas? Bien, hay otra característica del filtro que no se ha mencionado aún que es su característica en fase.

El significado de la respuesta en fase se explica a continuación, junto con el efecto del filtro en ella. Digamos que se tiene una caja negra de la cual no se conoce la circuitería interna. Se puede probar introduciendo en la entrada un generador senoidal y observando la salida junto con la entrada procedente del oscilador en un osciloscopio de doble canal.

De este modo, dos características importantes serán observables suponiendo que la caja negra sea un filtro allpass. La primera es la amplitud relativa de ambas señales. Así, para el allpass, esta relación es la unidad, pues ambas amplitudes serán iguales. La otra característica importante es la alineación relativa de ambas señales en el tiempo. La diferencia temporal entre ambas señales, es el retraso de fase introducido por el Allpass. De esta forma, el Allpass está alterando o cambiando la fase de la señal de entrada, de ahí precisamente el nombre de phaser (cambiador se fase). El retraso de fase se mide típicamente en fracciones del período.

Todos los filtros tienen una respuesta en fase que cambia con la frecuencia. Un caso interesante es una respuesta en fase lineal. En este caso, doblar la frecuencia supondrá doblar el retraso. Este tipo de respuesta mantiene alineadas a todas las componentes frecuenciales en el tiempo. Esto es justamente lo que hace una línea de retraso sencilla como la que formaba parte del Delay básico. En concreto, el Allpass, es un filtro **con una respuesta en fase no lineal**, que retrasa la señal, pero no todas las frecuencias por igual.

Ahora, se explicarán cómo se crean las muescas. Para producirlas, basta con mezclar la salida del filtro allpass con la entrada. ¿Por qué?, pues porque para algunas frecuencias el retraso de fase introducido por el filtrado será de 180 grados, lo que es equivalente a añadir exactamente la entrada pero negativa. Así cuando se mezcla esta señal con la entrada, las frecuencias que experimentaron el retraso de 180 grados ser cancelarán exactamente con las componentes originales de la entrada para tales frecuencias, originando la/s muesca/s para dicha/s frecuencia/s. El caso contrario se dará cuando el desfase producido sea de 360º en cuyo caso las señales original y retrasada se sumarán doblando su valor.

Retomando el caso de fase lineal a título orientativo, la respuesta en fase alcanza los valores de -180 grados y múltiplos negativos de 360 a frecuencia igualmente espaciadas. De esta forma, cuando se mezcla la copia retrasada de la señal con la original, se producirán muescas en frecuencias uniformemente espaciadas (caso del flanger), dependiendo este espaciado únicamente del tiempo de retraso.

Tal y como se dijo antes, usar un filtro Allpass supondrá usar una característica de fase no lineal, tal y como se muestra en la figura 37-b), de manera que se puede distorsionar la respuesta en fase para producir una muesca a cualquier frecuencia sin más que añadir filtros allpass en cascada de forma que las características de fase se sumarán. Esto es posible ya que una propiedad fundamental de estos filtros es la de que una combinación serie de allpass es un nuevo allpass, cuya ganancia será la unidad para cada frecuencia y cuya respuesta en fase será la suma de cada respuesta en fase.

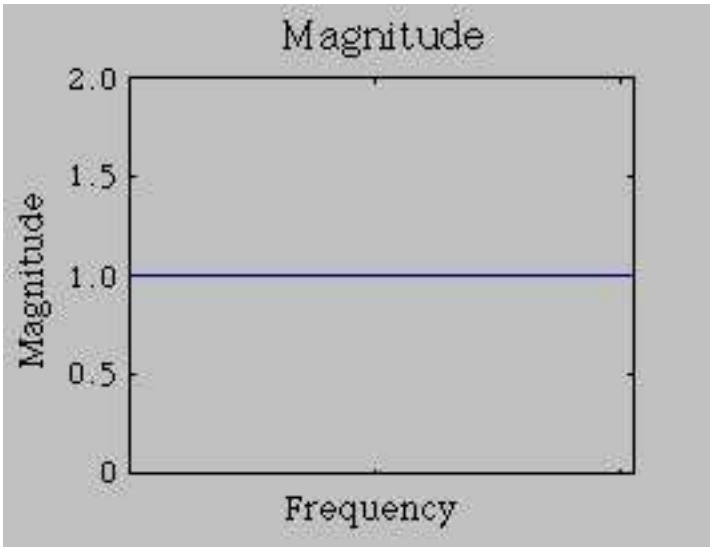


Figura 37- a). Característica de amplitud para un filtro allpass.

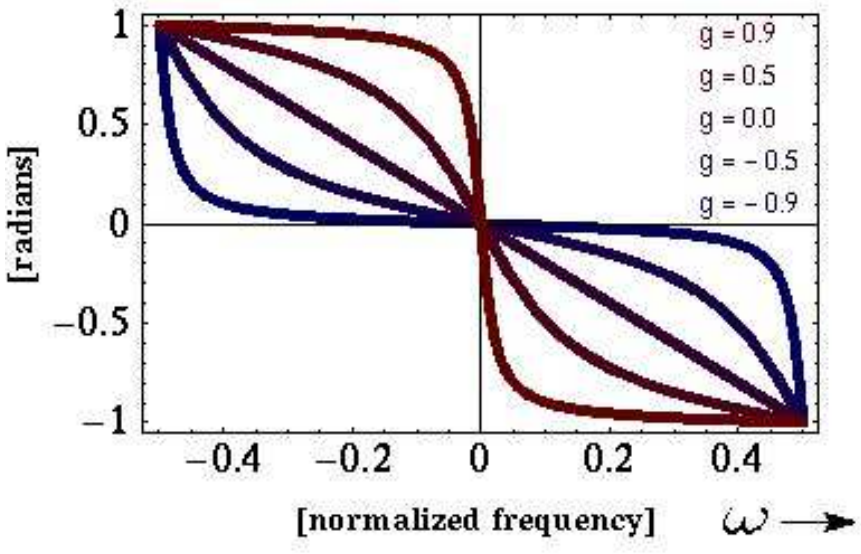


Figura 37- b). Característica de fase para un filtro Allpass.

El diagrama de sistema del filtro allpass se presenta en la figura 38.

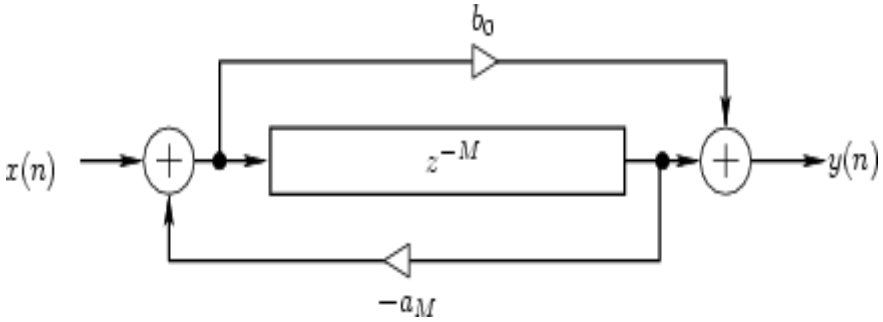


Figura 38. Filtro Allpass.

Siendo su ecuación en diferencia

$$y'(n) = b_0x(n) + x(n-M) - a_My'(n-M) \qquad (24)$$

La expresión (24) no es muy intuitiva debido a que su obtención se efectúa a partir de considerar el Allpass compuesto por dos filtros peine (ver apartado siguiente para detalles sobre el análisis). Baste decir que el filtro **Allpass se obtiene cuando $b_0=a_M=g$.**

Para terminar de caracterizar al filtro, se muestra en la figura 39 su respuesta al impulso unitario h(n) y la relación entre ésta y el factor g.

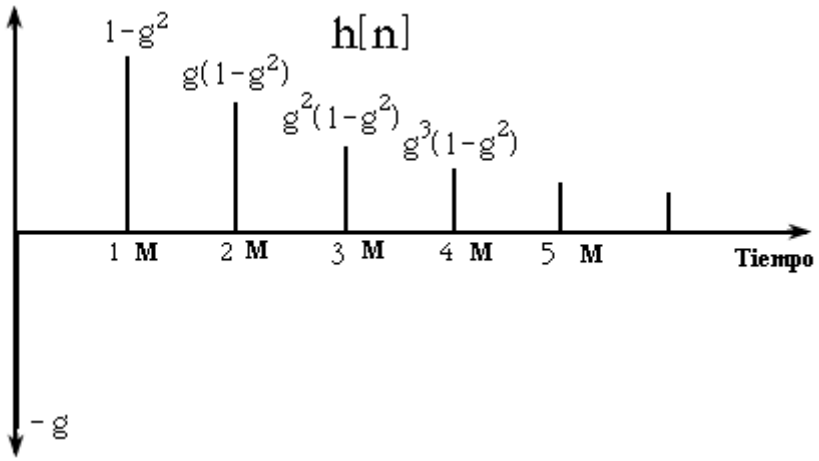


Figura 39. Respuesta al impulso unitario del filtro Allpass. Relación de ésta con el factor g.

De este modo, en función de las necesidades del usuario, se pueden usar filtros adicionales encadenados para crear más muescas y producir un efecto más pronunciado. Por ejemplo, el pedal MXR de Ibáñez contiene cuatro etapas para crear su sonido, pues cómo se deduce a partir de la característica en fase de la figura 37-b), una etapa no producirá una muesca completa, pues la respuesta en fase de un allpass se aproxima a 180 grados para frecuencias muy altas, de manera que múltiples etapas se requerirán para crear una sola muesca.

Un punto aún por discutir es el de que para generar el efecto de phaser, además de lo expuesto hasta ahora sobre la generación de muescas en la respuesta en amplitud, es frecuente hacer que éstas se muevan con el tiempo. Así para el chorus y el flanger se usaba un LFO para controlar el tiempo de retraso variable. Pero en el caso del phaser es mejor hacer cambiar las frecuencias de las muescas exponencialmente. Así, cuando las muescas se mueven hacia las altas frecuencias se deberá doblar la frecuencia de la muesca a cada paso, dando pasos más y más largos y una vez alcanzado el punto más alto, se van dividiendo entre dos las frecuencias de las muescas en el movimiento inverso.

3.5.4.2.1. PARÁMETROS COMUNES

1º. Mix/Profundidad. Este parámetro controla la cantidad de señal filtrada que es mezclada con la original. Se denomina también profundidad porque a medida que aumenta, las muescas aumentan su profundidad también. Así cuando es igual a la unidad, las muescas tienen la máxima profundidad alcanzando el cero.

2º. Profundidad de barrido. Con este parámetro se controla cómo de lejos movemos las muescas en el eje de frecuencia.

3º. Realimentación . El efecto característico del phaser puede hacerse mucho más intenso usando realimentación. De esta forma, se añade una parte de la salida del allpass a la entrada, tal y como se muestra en la figura 40 .

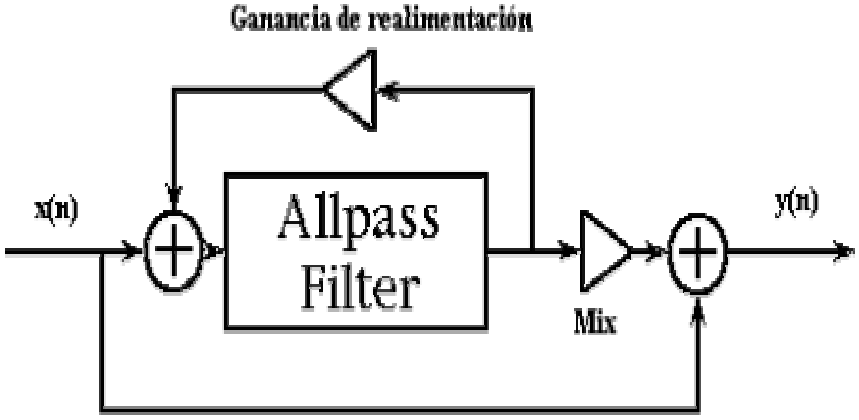


Figura 40- Phaser realimentado.

4º. Frecuencia. Este parámetro simplemente controla la velocidad con la que las muescas se mueven arriba y abajo en el eje de frecuencia. La frecuencia controla por tanto, cuántas veces las muescas barren arriba y abajo el eje por segundo. La velocidad a la cual las muescas se mueven por el espectro, está determinada por el control de frecuencia y la profundidad de barrido .

3.5.4.3. IMPLEMENTACIÓN.

Para implementar un phaser en el microprocesador sin necesitar una gran cantidad de recursos, es frecuente usar el modelo elemental, donde está implícito el uso del allpass de la figura 38. Al igual que se hacía para el resto de los efectos, se usará una tabla circular para guardar y extraer las muestras. Así mismo, no se incluye el lazo de realimentación ni el movimiento de las muescas mediante ningún tipo de LFO porque el modelo constituye un modelo muy básico de una sola etapa que no distribuiría por el espectro tan siquiera una muesca.

Los parámetros de control serán:

- Ganancia G.
- Tiempo de retraso.
- Mix.

Algoritmos más complejos requerirán varias etapas de filtros allpass o incluso filtros allpass de orden superior , con el objeto de obtener un mayor número de muescas en el espectro de audio. Es en este contexto donde tiene sentido usar un oscilador para controlar el tiempo de retardo, de manera que el movimiento de las muescas (ahora varias) sea perceptible. Estos nuevos modelos y algoritmos precisarán de mayores y más profundas consideraciones para su diseño y no serán objeto del Presente Proyecto fin de Carrera por su extensión y profundidad.

3.5.5. REVERBERACIÓN.

3.5.5.1. INTRODUCCIÓN.

La reverberación es probablemente uno de los efectos más utilizados en la música. Muchas personas no comprenden la importancia de la reverberación y sin embargo escuchan reverberación de forma natural continuamente a su alrededor y sin la ayuda de ningún tipo de procesador especial.

3.5.5.2. PRINCIPIOS Y MODELADO.

La reverberación es el resultado de varias reflexiones del sonido que ocurren dentro de una habitación. Así, desde cualquier fuente de sonido, como por ejemplo los altavoces de un equipo de música, hay un camino directo que la música recorre para alcanzar los oídos. El sonido puede también tomar un camino más largo a través del rebote en las paredes o el techo, previamente a su llegada a los oídos, tal y como se ilustra en la figura 41.

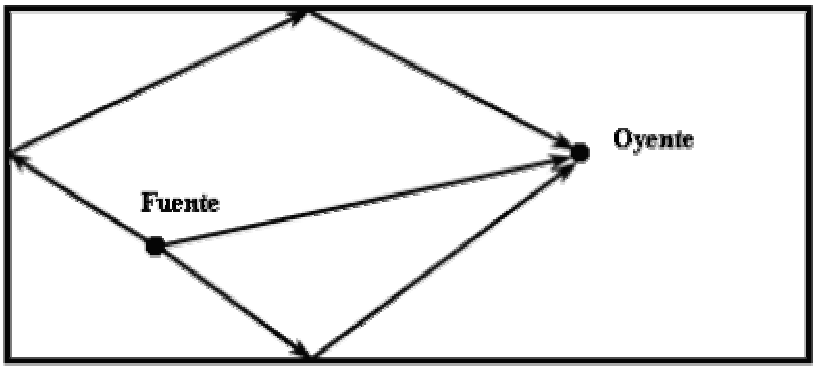


Figura 41. Camino del sonido.

La onda de sonido reflejada llegará al oído con cierto retardo respecto a la onda directa, pues viaja una mayor distancia. Además, es generalmente más débil, pues las paredes y otras superficies de la habitación , así como el propio aire, absorben parte del sonido. Por supuesto que la onda reflejada puede rebotar varias veces en distintas paredes antes de llegar a nuestros oídos. Esta serie de atenuaciones y retardos es lo que se denomina reverberación y es el efecto que produce la sensación de espacio en una habitación.

Es muy tentador e incorrecto decir que la reverberación es una serie de ecos. El eco generalmente implica una versión retardada del sonido, la cual se escucha con un retardo mayor o igual a 200 milisegundos. En la reverberación, cada onda de sonido retardada llega en un período de tiempo tan corto que la onda reflejada no es percibida como una copia del original. Así, a pesar de no poder distinguir a cada onda reflejada, si podemos escuchar el efecto generado por toda la serie de reflexiones.

Aparentemente un modelo realimentado de delay como el ya explicado podría producir reverberación. No obstante, una unidad de este tipo puede causar un efecto similar, pero no tiene en cuenta una serie de características fundamentales para realizar un modelado realista de la reverberación.

En primer lugar, no se podrá modelar el hecho de que la tasa de llegadas de las ondas reflejadas que llega al oyente varía con el tiempo, pues este modelo sólo puede simular reflexiones con un intervalo fijo entre ellas (ver respuesta al impulso unitario del delay realimentado).

Más concretamente, en una simulación realista de la reverberación, deben existir reflexiones tempranas y reflexiones tardías. Así, tras la llegada de la onda directa, aparece un corto espacio de tiempo en el que las reflexiones se caracterizan por ser bien definidas y direccionadas. Estas reflexiones están directamente relacionadas con la forma y tamaño de la habitación, así como con la posición de la fuente del sonido y del oyente. **Son las denominadas reflexiones tempranas.** Luego de estas reflexiones, la tasa de ondas reflejadas que llegan al oído aumenta considerablemente. En este segundo período, las reflexiones son más aleatorias y difícilmente relacionables con las características físicas de la habitación. Esto es lo que se denomina **reverberación difusa o reflexiones tardías.** Se cree que este segundo tipo de reflexión es el factor primario al diseñar el tamaño de una habitación y decae exponencialmente en las buenas salas de conciertos. La figura 42 muestra la respuesta al impulso unitario de un simulador de reverberación realista.

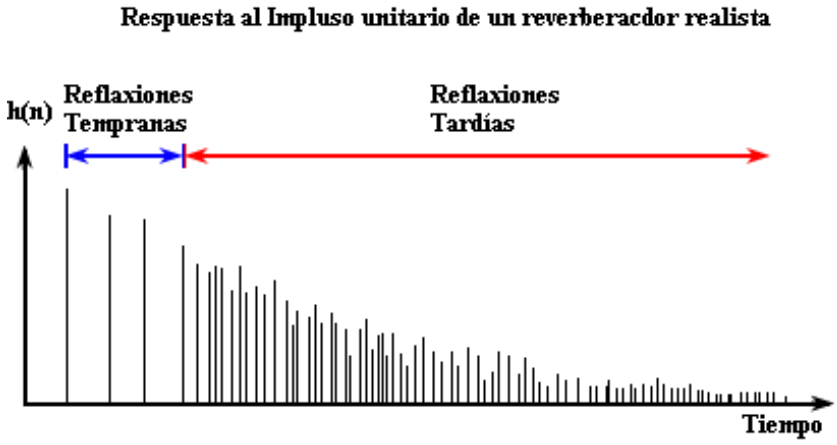


Figura 42. Respuesta al impulso unitario de un reverberados realista.

Otra característica importante de la reverberación realista es la correlación de las señales que llegan al oído. Para darle al oyente una verdadera sensación de ambiente espacioso, el sonido que llega a cada oído debe ser en cierta forma incoherente. Esta es una de las razones por la que una sala de conciertos tiene techos altos. Así, al usar techos altos, las primeras reflexiones que llegan al oído serán las procedentes del rebote de las paredes, y como éstas están situadas a diferentes distancias una de otra respecto al oyente, el sonido que llega a cada oído lo hará en momentos distintos. Las reflexiones procedentes del techo, por su parte, llegarán a la vez pero después de las procedentes de las paredes.

Una medida que es utilizada para caracterizar la reverberación de una habitación es el denominado **tiempo de reverberación** . Técnicamente, es el tiempo que le toma al nivel de presión del sonido o intensidad del mismo en decaer 60 dB de su valor original. Tiempos de reverberación largos indican que la energía del sonido se mantiene dentro de la habitación por mayor tiempo, antes de ser absorbida. Las salas de conciertos tienen un tiempo de reverberación entre 1,5 y 2 segundos. El tiempo de reverberación se encuentra fuertemente ligado a dos factores, **las superficies de la habitación** y **el tamaño de la misma.**

Así, las superficies de la habitación determinan cuánta energía se pierde en cada reflexión. De este modo, los materiales altamente reflectantes como ladrillos y ventanas o cerámica, incrementan el tiempo de reflexión, pues no absorben mucha energía debido a que son muy rígidos. Por su parte, los materiales absorbentes tales como cortinas, reducirán el tiempo de reverberación. Hay que señalar aquí que la capacidad de absorción de los materiales varía además con la frecuencia.

Por otro lado, las habitaciones grandes tienen mayores tiempos de reverberación, ya que en promedio las ondas de sonido viajan distancias más largas entre reflexiones. Así mismo, el propio aire de la habitación atenúa las ondas de sonido (sobre todo las altas frecuencias), reduciendo el tiempo de reverberación. Esta atenuación varía además con la humedad y la temperatura. Es frecuente por ello, incorporar filtros paso bajo en los simuladores de reverberación realista.

Centrando ya nuestra atención sobre el modelado de las reflexiones tempranas y tardías, se debe señalar que las primeras están relativamente dispersas y abarcan poco tiempo y se implementan frecuentemente usando líneas de retraso multi-tap. Al final de la líneas multi-tap se puede incluir un filtro paso-bajo para filtrar la señal y simular el efecto de absorción del aire. A continuación se simularán las reflexiones tardías. La figura 43 muestra este esquema.

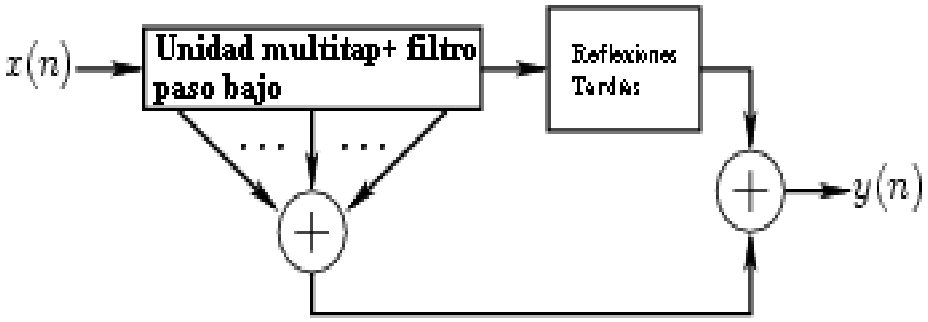


Figura 43- Esquema genérico del reverberador realista.

En cuanto a las reflexiones tardías se refiere, debe indicarse que desde un punto de vista perceptual, las principales cualidades de una buena respuesta al impulso unitario en cuanto a reflexiones tardías se refiere, se caracterizará por:

1º. Una suave cadencia .

2º. Una suave, aunque no demasiado regular, respuesta en frecuencia.

Proporcionar una cadencia de la repuesta al impulso de forma exponencial no es problema, ya que todos los sistemas lineales estables poseen una respuesta al impulso exponencialmente decreciente. La tarea más difícil es hacerla decaer de forma suave. En general, una cadencia suave resultará cuando la densidad del eco es suficientemente alta.

Una suave respuesta en frecuencia no debe exhibir largos huecos o impulsos. Ello se consigue igualmente, cuando la densidad del eco es suficientemente alta.

De esta forma, para conseguir la deseada densidad de eco se puede usar una cadena serie de filtros allpass pues como se apreciaba en su respuesta al impulso unitario , estos filtros tienen la propiedad de afectar a la fase de la señal, permitiendo dar forma a los desvanecimientos.

Teniendo en cuenta todo lo expuesto, en la figura 44 se presenta el modelo completo del reverberador realista, donde se ha sustituido la cadena serie de allpass por un filtro Butterworth que cumple la doble función de modelar la atenuación de las altas frecuencias debida al aire, e introducir el cambio de fase característico de los allpass, a fin de conseguir la densidad de eco buscada . Este modelo también es denominado reberberador de Schroeder (ver bibliografía [5] y [7] específicamente).

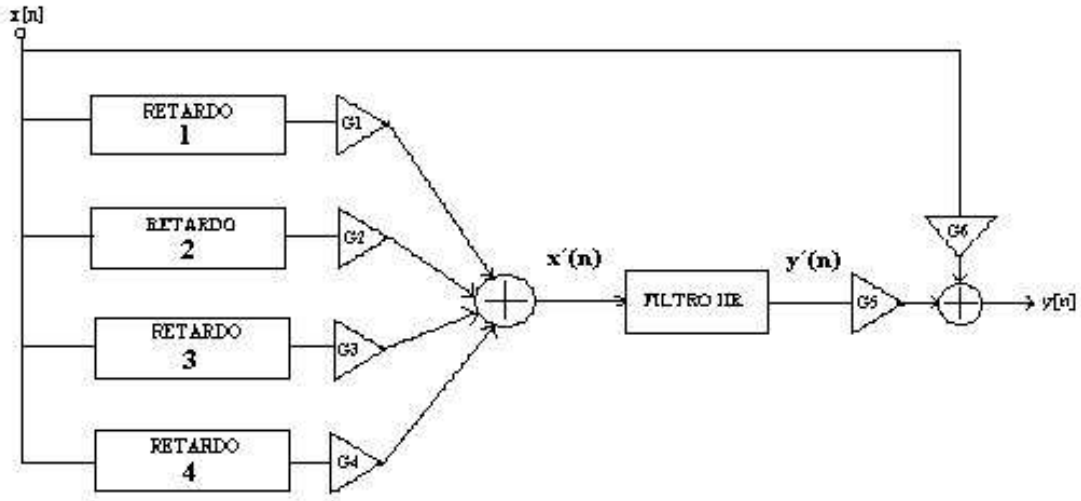


Figura 44. Modelo de reverberador realista.

El filtro IIR es un pasabajos de tercer orden de Butterworth . Su **función de transferencia** es la siguiente:

$$y'[n] = 0.4\, y'[n-1] - 0.2499\, y'[n-2] + 0.0441\, y'[n-3] + 0.5814\, x'[n-1] + 0.2142\, x'[n-2] \quad (25)$$

Es importante señalar que modelos más avanzados pueden desarrollarse para habitaciones con geometrías concretas, posición de la fuente específica, etc, a partir del trabajo directo con funciones de transferencia. Así pues, el diseño de reverberaciones

específicas es todo un campo de desarrollo en el campo dedicado a la fabricación y diseño de procesadores de efectos. No es la pretensión de este Proyecto Fin de Carrera profundizar más en el diseño de los reverberadores, por ser éste el objeto de otro trabajo en sí mismo, sino facilitar las nociones fundamentales y caracterizar un modelo válido para la posterior implementación.

Ahora bien, cabría preguntarse ¿cuál es la utilidad de añadir reverberación a los instrumentos o a los sonidos grabados?. Pues muchas veces cuando se está escuchando música o se está tocando un instrumento, se está haciendo en ambientes con muy poca o muy pobre reverberación. Por ejemplo, dentro de un coche o en un local con malas condiciones de reverberación, donde no es posible recrear el sonido majestuoso de una orquesta. Así, una señal sin nada o muy poca reverberación puede sonar poco natural. Ya que no siempre es posible escuchar música, o hacerla, en una sala de conciertos, es por lo que se le agrega reverberación a las grabaciones y a los instrumentos.

3.5.5.2.1. OTROS TIPOS DE REVERBERACIÓN.

1º. Gated Reverb. Una reverberación de este tipo es creada truncando simplemente la respuesta al impulso del reverberador clásico, lo cual consistirá en sólo permitir al sonido el efectuar un número concreto de reflexiones.

La cantidad de tiempo que transcurre hasta que la respuesta queda truncada se denomina **Gate time** , como se ilustra en la figura 45. Algunas unidades permiten además obtener una cadencia más gradual en el truncamiento, en vez de un silencio abrupto.

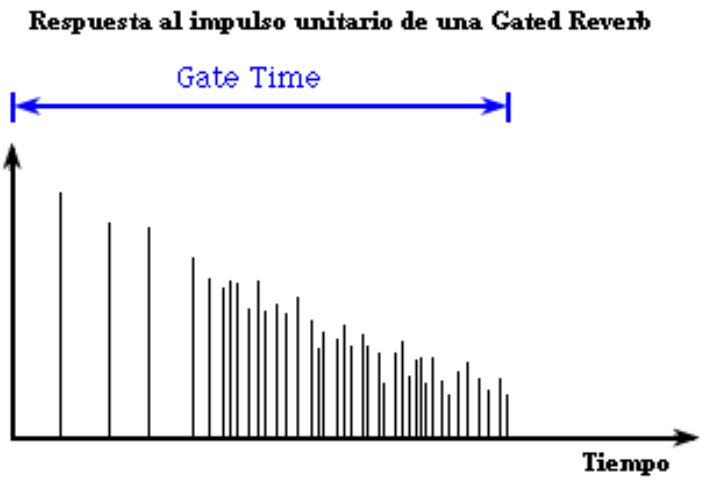


Figura 45. Respuesta al impulso unitario de una gated reverb.

2º. Reversed Reverb. Este tipo de reverberación se consigue girando la respuesta al impulso unitario de la figura 45. Así, en vez de simular reflexiones que se van atenuando gradualmente hasta desaparecer, se consiguen reflexiones en las cuales el sonido se vuelve más fuerte con el tiempo y después se corta abruptamente. La cantidad de tiempo que el sonido en la cual el sonido se va armando se denomina **reverse time** o al igual que antes **gate time,** pues no es más que una gated reverb pero invertida en el tiempo. En la figura 46 podemos ver su respuesta al impulso unitario.

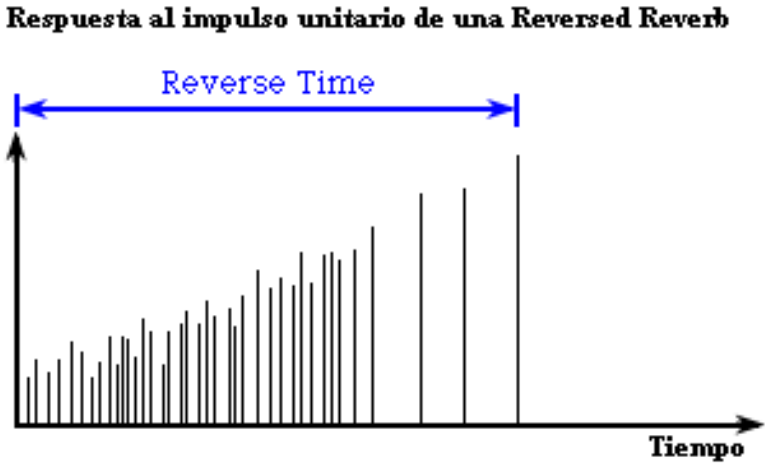


Figura 46. Respuesta al impulso unitario de una reversed reverb.

Esta forma de reverberación tiene un sonido muy parecido al que se consigue con el slapback delay ya explicado, porque termina de repente. No obstante, si se escucha detenidamente, se puede apreciar el sonido tomando forma.

3.5.5.2.2. PARÁMETROS COMUNES EN REVERBERADORES COMERCIALES.

1º. Predelay. Se define como la cantidad de tiempo que las primeras reflexiones tardan en ser oídas. En algunas ocasiones se puede definir como el tiempo que tardan en aparecer las reflexiones tardías. Para la simulación de ambientes reales usando los dos tipos de predelays, el predelay para las primeras reflexiones debe ser siempre menor que el correspondiente a las tardías como se aprecia en la figura 47.

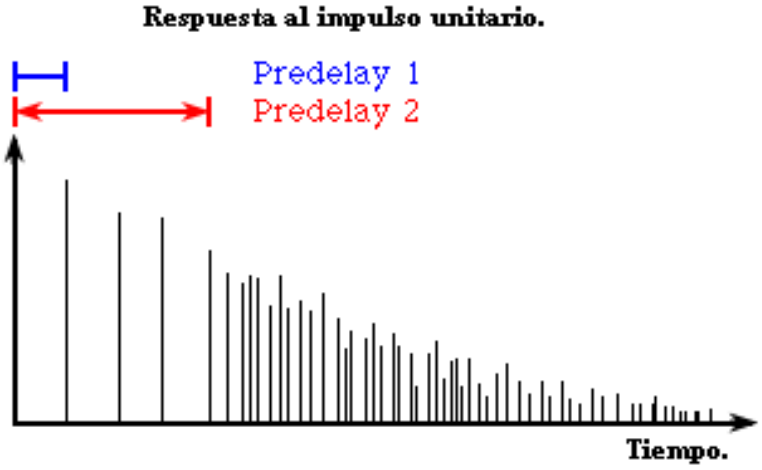


Figura 47. Significado del predelay.

2º. Reverb Decay. Este parámetro indica durante cuánto tiempo la reverberación puede ser oída después de que la entrada se detenga. Se fija típicamente en milisegundos y no es otra cosa que el **tiempo de reverberación** explicado al comienzo.

3º. Gate Reverb. Se aplica a las gated reverb. Es el tiempo que se permite sonar a la unidad. Es válido también para las reversed reverbs.

4º. Gate decay time. Algunas unidades gated reverb permiten también manejar este parámetro. Así, con él se controla la forma en que la gated reverb corta. Un valor pequeño conllevará que la unidad corte abruptamente. Valores mayores para este parámetro, ocasionan que a la unidad se le conceda más tiempo para extinguirse gradualmente las reflexiones.

5º. Gate Threshold. En vez de aplicar una gated reverb a una señal completa, se podría muy bien sólo aplicarla dependiendo de los niveles de señal de entrada. Típicamente, por encima de este **umbral** usamos una reverberación normal sin truncamiento alguno, pero cuando la señal cae por debajo de él, la "puerta" cierra y el número de reflexiones se reduce. Se abrirá de nuevo la "puerta" cuando la señal supere de nuevo el umbral gate threshold.

3.5.5.3. OTRAS NOTAS.

Se expondrán ahora brevemente algunos conceptos de acústica complementarios para comprender mejor la reverberación. Hablaremos pues de **campos directos y reverberante.**

En acústica se habla de campo directo y reverberante del sonido en una habitación. Si el sonido directo que proviene de la fuente es más intenso que las reflexiones, entonces estamos en el **campo directo.** Si por el contrario la intensidad del sonido debida a las reflexiones es mayor que la del sonido directo, estamos en el **campo reverberante**. El punto donde coinciden ambos campos (misma intensidad en las reflexiones y en el sonido directo) se denomina **distancia crítica.**

El campo reverberante es extremadamente importante. De hecho, la mayoría del tiempo se está en él y sin su existencia cualquier representación musical sería imposible de seguir.

Como es sabido, tratar de hablar a un grupo de personas en campo abierto requiere que se hable más alto que cuando se hace en una habitación. La reverberación de una habitación ayuda a mantener la energía del sonido localizada en la misma, incrementando el nivel de presión de sonora y distribuyendo el sonido por ella. Por el contrario en espacio abierto, muchas de las superficies reflectantes no están presentes y la mayoría de la energía se pierde.

El campo reverberante es también importante para la música. Primero, ayuda a escuchar todos los instrumentos como una unidad. Además, muchos instrumentos, como

el violín, no irradian todas las frecuencias de manera igual en todas las direcciones. Únicamente en el campo directo sonará diferente e incluso algo desagradable.

3.5.5.4. IMPLEMENTACIÓN.

La reverberación es un efecto muy adecuado para su implementación en procesadores digitales de señal DSP. De este modo el modelo de la figura 44 es ideal para su implementación directa sobre el microprocesador.

La unidad multitap se implementará usando una tabla circular al igual que se hacía en todos los efectos anteriores. Con varios punteros se extraerán las muestras correspondientes a los diferentes tiempos de retraso. El filtro Butterworth se implementará digitalmente de forma directa a partir de la ecuación (25). Se deberán guardar también dos muestras procedentes de la suma de la línea multi-tap, que denotaremos por x´(n). Igualmente se guardarán tres muestras correspondientes a la salida del filtro IIR que denotaremos como y´(n).

Por último, señalar que el usuario controlará los parámetros correspondientes a los cuatro tiempos de retardo de la línea multi-tap, Las cuatro ganancias de cada tap y las dos ganancias G5 y G6 que controlan cuánta señal tratada y original se mezcla respectivamente.

3.6. PROCESADO EN EL DOMINIO DE LA FRECUENCIA.

Al hablar de procesado en el dominio de la frecuencia nos estamos refiriendo a la manipulación selectiva y apropiada a la aplicación que se desee realizar, de las distintas componentes en frecuencia (armónicos) que componen una señal de audio. De esta forma, en esta sección se estudiarán los métodos más comunes de ecualización, proponiéndose por último la manera de realizar el procesado de forma digital.

3.6.1. ECUALIZACIÓN.

3.6.1.1. INTRODUCCIÓN.

La ecualización es el proceso de incrementar o atenuar ciertas componentes frecuenciales de una señal. El nombre de ecualización proviene la aplicación que trata de obtener una respuesta en frecuencia plana. Por ejemplo, cuando se toca un instrumento en una sala que no realce suficientemente los matices de alta frecuencia, se deberán aplicar filtros ecualizadores para reponer esta pérdida. La ecualización es una herramienta muy importante en grabación y para tocar en directo con el objetivo de mantener la calidad de un instrumento.

3.6.1.2. CONTROLES DE TONO.

El sistema ecualizador más común es probablemente el control de tono que puede encontrarse en la mayoría de los sistemas estéreo e incluso integrados en los controles básicos del instrumento. Así, estos controles proporcionan una rápida y fácil manera de ajustar el sonido para que éste encaje con los gustos del oyente o intérprete y compense las perturbaciones introducidas por el ambiente.

Frecuentemente se encontrarán los controles nombrados como **'bajo'** y **'treble'**. Cada uno de estos controles maneja un tipo especial de filtro denominado **shelving filter** o de forma más concreta un filtro paso bajo shelving y un paso alto tipo shelving también. .

La respuesta frecuencial característica de este tipo de filtros se presenta en la figura 48. Así, una ganancia mayor que la unidad amplificará la señal, mientras que una ganancia menor que uno la atenuará para la banda de frecuencias en cuestión.

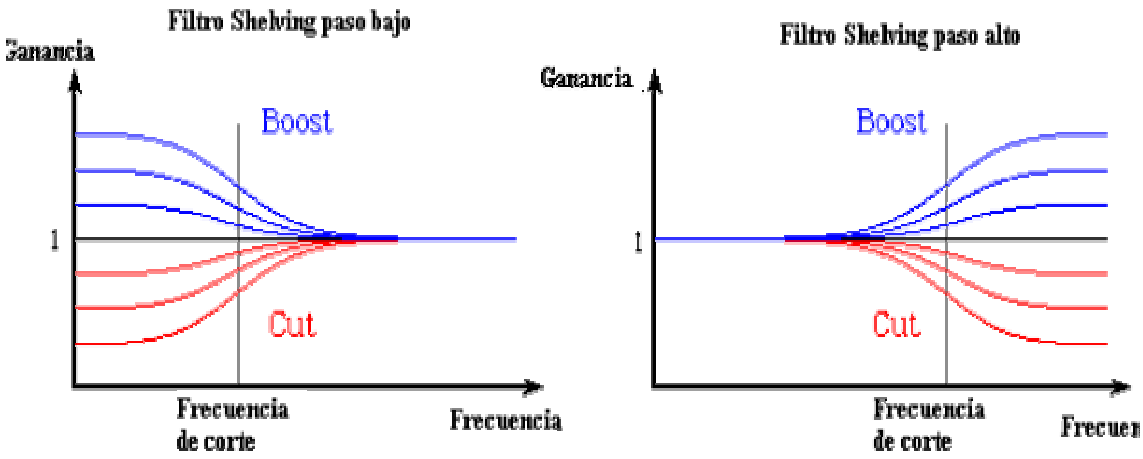


Figura 48. Respuesta en frecuencia de los Filtros shelving.

De este modo, en la mayoría de aplicaciones, los filtros paso bajo y paso alto tratan de eliminar totalmente una porción del espectro. Por ejemplo, es sabido que un filtro paso bajo convencional intentará eliminar las altas frecuencias por encima de su frecuencia de corte. No obstante, los filtros tipo shelving no tratan de eliminar nada, únicamente se pretende amplificar o atenuar una porción del espectro dejando el resto del mismo intacto (ganancia unidad).

La frecuencia para la cual la respuesta realiza la transición entre los dos niveles de ganancia se denomina **frecuencia de corte .** Se puede diseñar un control de tono que permita al usuario cambiar tanto la frecuencia de corte como el nivel de atenuación o amplificación deseado, aunque normalmente, en los dispositivos comerciales no se podrá variar la frecuencia de corte de su valor prefijado.

Además de los controles de bajos y altos anteriores, también es común encontrar **controles de medios** , tales como los ecualizadores de tres bandas que se encuentran comúnmente en las mesas mezcladoras. Como es de suponer, este control afecta a las frecuencias intermedias. Esto es lo que se denomina **peking filter o paso banda** . De nuevo , el tipo shelving no trata de aislar ciertas frecuencias, y sí de amplificar o atenuar una pequeña porción del espectro de audio sin modificar el resto de frecuencias. Este tipo de filtros normalmente no tiene una frecuencia de corte definida, y se caracteriza por otras dos características:

1º. La frecuencia para la que el filtro posee la máxima (amplificando) o mínima (si está atenuando) ganancia, se denomina **frecuencia central.**

2º. **El ancho de banda** del filtro se define como el rango de frecuencias afectadas por su efecto de amplificación o atenuación. Generalmente se podrá únicamente controlar la amplificación /atenuación, pues la frecuencia central y el ancho de banda serán fijos. La figura 49 muestra la respuesta en frecuencia típica de este tipo de filtro.

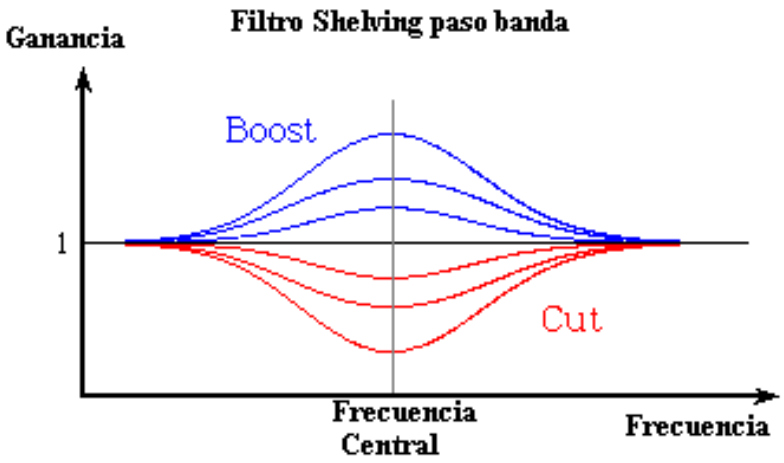


Figura 49. Filtro shelving paso banda.

Los controles de tono son muy simples ya que sólo incorporan dos o tres filtros (bajo, treble y a veces medios). Debido a esta simplicidad, estos filtros normalmente se conectan en serie, de forma que la salida filtrada del primero es la entrada del segundo y la salida de éste excita al tercero.

3.6.1.3.ECUALIZADORES GRÁFICOS.

Los ecualizadores gráficos están un paso más avanzados que los controles de tono en cuanto a flexibilidad y posibilidades de control se refiere, siendo su funcionamiento sencillo de comprender. Así, un ecualizador gráfico es simplemente un conjunto de filtros paso banda comunes, cada uno con una frecuencia central fijada que no puede cambiarse. El único control que se tiene disponible es la ganancia a aplicar en cada banda frecuencial. Esta ganancia se ajustará mediante interruptores deslizantes.

Este interfaz es muy intuitivo ya que la respuesta frecuencial del ecualizador puede ser rápidamente visualizada mirando las posiciones de los deslizadores. De este modo, dichos deslizadores son una representación gráfica de la respuesta fecuencial, de ahí el nombre de ecualizador gráfico.

El uso primordial de este tipo de ecualizadores no es otro que el de reforzar el sonido. Por ejemplo, cuando se toca en un local, es deseable tener una respuesta frecuencial plana (o razonablemente plana en un cierto rango de frecuencias) para el instrumento o la banda. No obstante, la propia resonancia de la habitación y de los altavoces a gran volumen pueden "colorear" el sonido. Con un ecualizador gráfico que cubra la mayor parte del espectro de audio se podrá contrarrestar algo esta coloración, de forma que aunque se toque en una habitación o lugar distinto cada día, el sonido de los instrumentistas sea consistente en cada interpretación.

Actualmente, la implementación de un ecualizador gráfico es diferente a la del control de tonos. Así en éste último, los controles de bajo y medios ('treble') amplificaban o atenuaban ciertas bandas de frecuencias mientras dejaban el resto del espectro inalterado, y se conectaban en serie. Un ecualizador gráfico, por su parte, usa **un banco de filtros paso banda comunes**, diseñados para aislar completamente ciertas bandas de frecuencias. Así, con el objetivo de tener control sobre todo el espectro de audio, los filtros necesitan ser conectados en paralelo. De esta manera, cada filtro en el ecualizador gráfico tiene la misma entrada. El trabajo de cada filtro consistirá en permitir pasar únicamente una pequeña banda convenientemente amplificada o atenuada según las necesidades del músico. Así, una vez que la señal pasa por cada filtro, el usuario puede manipular cada banda independientemente modificando su ganancia, mediante los típicos deslizadores en el frontal del ecualizador.

En la figura 50 se observa la característica frecuencial de un filtro paso banda de los empleados por el ecualizador gráfico. En la figura 51 , por su parte, se muestra el diagrama de bloques de un ecualizador gráfico tal y como se ha descrito anteriormente.

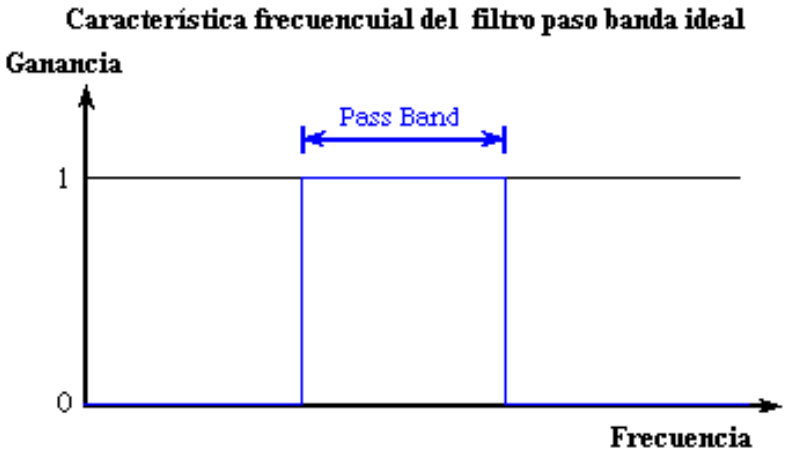


Figura 50. Característica frecuencial del filtro paso banda ideal.

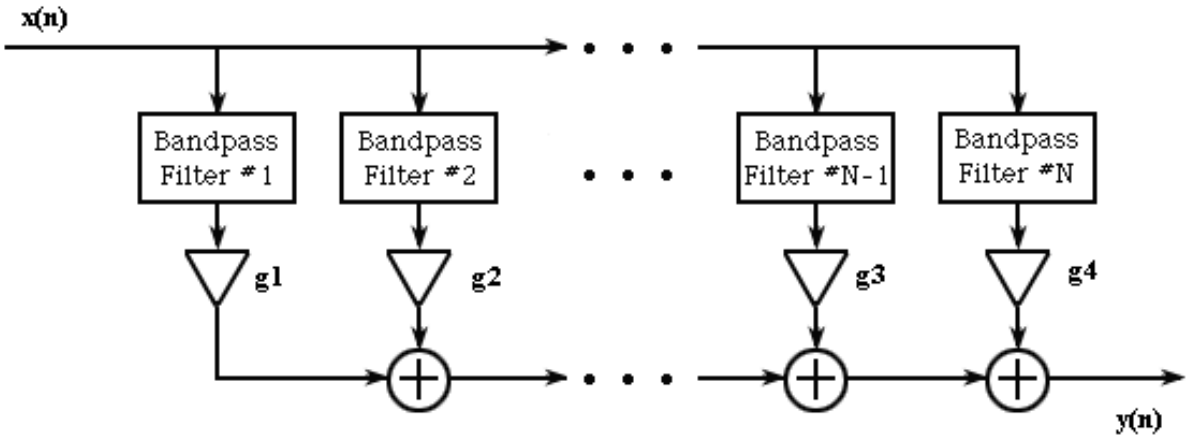


Figura 51. Ecualizador gráfico.

Es importante señalar llegados a este punto, que el conexionado en paralelo de los filtros paso banda del ecualizador gráfico se debe a la intención de reducir al máximo los dañinos efectos que pueden causar sobre el tratamiento de la señal, la interconexión serie de filtros. Nos referimos a que un filtro tiene, como es sabido, una respuesta en fase también. Así, mientras que el cambio de fase en la señal es deseable para la obtención de algunos efectos (como el phaser), en aplicaciones de ecualización es un fenómeno totalmente indeseable. De esta manera, por cada filtro añadido en serie, la respuesta en fase de cada filtro se añade a la del siguiente y así sucesivamente. Si se tienen únicamente dos o tres filtros encadenados en serie, como es el caso del control de tonos, el resultado será satisfactorio, pero si se pretende implementar un control más exigente, como por ejemplo un ecualizador de 15 o 30 bandas, la distorsión de fase causada por la interconexión serie es intolerable.

Por último, se debe indicar que las frecuencias centrales de los ecualizadores gráficos están normalmente espaciadas en octavas y no linealmente espaciadas. Por ejemplo, se pueden encontrar ecualizadores gráficos con 1/3 o 1/6 de octava de esparcimiento. Una octava supone un factor de 2 , de manera que un esparcimiento de una octava supondrá que las frecuencias centrales comiencen a 100 Hz, 200,400,800 Hz y así sucesivamente. Un ecualizador con 1/3 de octava de esparcimiento, se basa en un factor de $2^{1/3}$ que es aproximadamente 1,26. De esta forma, si la primera frecuencia central está en 100 Hz, el ecualizador con espaciado de 1/3 de octava tendrá las frecuencias centrales de los filtros paso banda que lo componen en 126,159,200 Hz …etc. A este respecto hay que señalar que existe un estándar ISO sobre las frecuencias centrales que es preferible usar.

3.6.1.4.ECUALIZADORES PARAMÉTRICOS.

El ecualizador paramétrico constituye el último paso en cuanto a flexibilidad se refiere, pero por el contrario requiere un uso más cuidadoso para resultar efectivo. Así, un ecualizador paramétrico sencillo permite ajustar no sólo la cantidad de amplificación o atenuación, sino también la frecuencia de control el ancho de banda del mismo.

Con experiencia en su uso se puede aplicar para ayudar a un instrumento a ganar presencia y consistencia dentro de la mezcla final o se puede emplear para cancelar la típica e indeseada realimentación , de manera que el ecualizador se ajuste precisamente a la frecuencia a la que se está produciendo dicha realimentación, funcionando como un filtro muesca para anularla. Para minimizar los efectos del filtro en el resto del sonido, se puede usar una pequeña banda de acción.

Debe observarse aquí, que el fenómeno indeseado de la realimentación también se podría controlar por medio de ecualizadores gráficos, si bien las bandas predeterminadas de éstos podrían ser demasiado amplias, con lo que se verían afectadas más frecuencias de las deseadas. En la figura 52 se muestra la acción de un amplificador paramétrico en términos de su respuesta frecuencial.

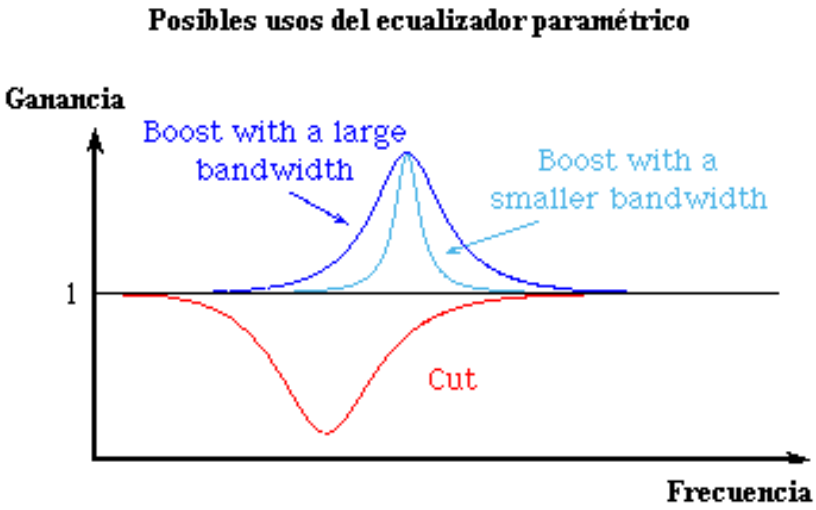


Figura 52 . Posibles usos del ecualizador paramétrico.

3.6.1.5. OTRAS NOTAS.

1º. Presencia. Muchos amplificadores tienen un control denominado 'presencia ' como parte de los controles de tono. Esto es simplemente una amplificación adicional en mitad de las altas frecuencias, de 2 a 6 Khz. Este control altera el sonido de un instrumento en una grabación para dar la impresión de que éste está siendo tocado en la habitación junto con el oyente. Permite, por tanto, al instrumento en cuestión sobresalir más en la mezcla.

2º. Crossover de los altavoces. Aunque los crossover de los altavoces no son realmente un tipo de ecualizador, si están relacionados con éstos. Así, es muy difícil diseñar un altavoz y su carcasa de forma que el conjunto compuesto por ambos luzca una perfecta respuesta plana en el espectro audible. De esta manera, si se estrecha el rango frecuencial un poco, el diseño se facilita pero para cubrir el rango de audio completo, se necesita más de un altavoz.

Es por ello por lo que la mayoría de los equipos de alta fidelidad usan más de un altavoz dentro de lo que nosotros percibimos desde fuera como uno sólo. Normalmente se usarán dos o tres. El más grande es para las bajas frecuencias y recibe el nombre de ' woofer' , el más pequeño, 'tweeter' , está diseñado para respetar las altas frecuencias. El tercero es para las frecuencias medias.

El crossover de los altavoces es por tanto una red de filtros que toma la señal de audio entrante y la fracciona en las componentes frecuenciales que cada altavoz puede manejar (aislando completamente las bandas a manejar). El diagrama de un crossover se muestra en la figura 53.

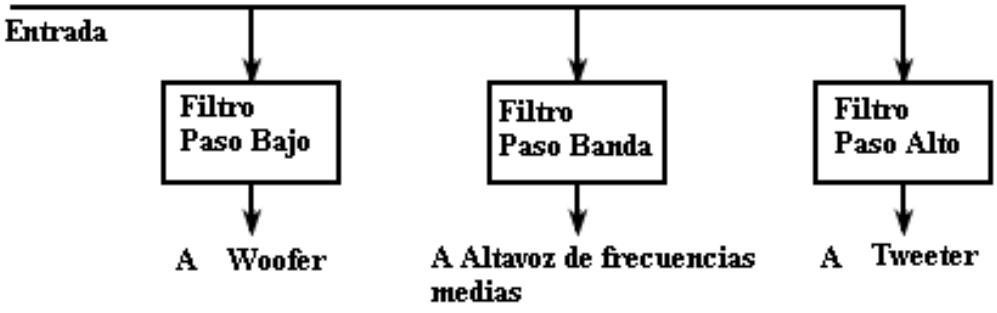


Figura 53. Crossover de los altavoces.

3.6.1.6. IMPLEMENTACIÓN.

La implementación de los ecualizadores ha venido siendo tradicionalmente analógica y se incluye típicamente en los amplificadores, siendo un circuito tradicionalmente usado, la red activa de control de tono Baxandall, cuyo esquema se presenta en la figura 54.

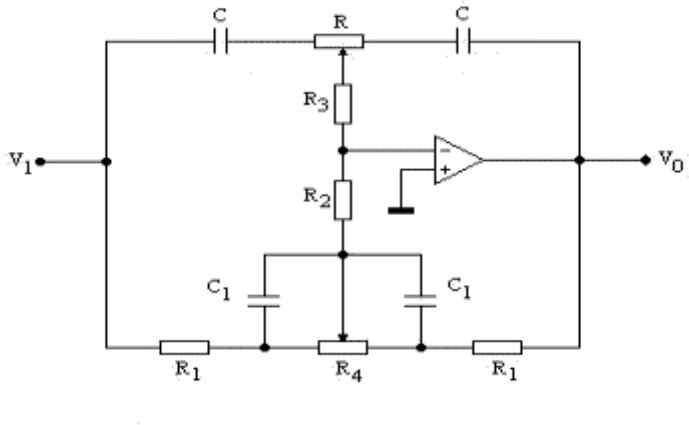


Figura 54. Red de control de tono activa Baxandall.

Este sencillo control de tono es frecuentemente montado en instrumentos musicales, siendo la señal procedente del instrumento a tratar v_1 y la tratada v_0 . Esta red proporciona el control de los bajos por acción del potenciómetro R4 y el control de los altos por acción del potenciómetro R. Ambos potenciómetros proporcionan el máximo refuerzo en sus bandas de trabajo cuando se encuentran en la posición extrema de la izquierda.

Dejando de lado la implementación netamente analógica, diremos que para realizar la ecualización de una señal digitalmente de forma eficiente, lo ideal es calcular la distribución espectral de la señal mediante la DFT (Discrete Fourier Transform) o FFT (Fast Fourier Transform) y almacenar sus componentes armónicas en un vector en la memoria. A continuación , estas componentes se manipularán aritméticamente siendo amplificadas o atenuadas convenientemente según el propósito del filtrado, para después almacenar de nuevo el vector ya modificado en memoria. Por último, tan sólo resta aplicar al vector manipulado que constituye la señal filtrada en el dominio de la frecuencia, la transformación inversa DFT^{-1} o FFT^{-1} para reconstruir en el dominio del tiempo la señal ya filtrada. La figura 55 muestra el diagrama de las operaciones a seguir para el procesado en frecuencia en el dominio digital.

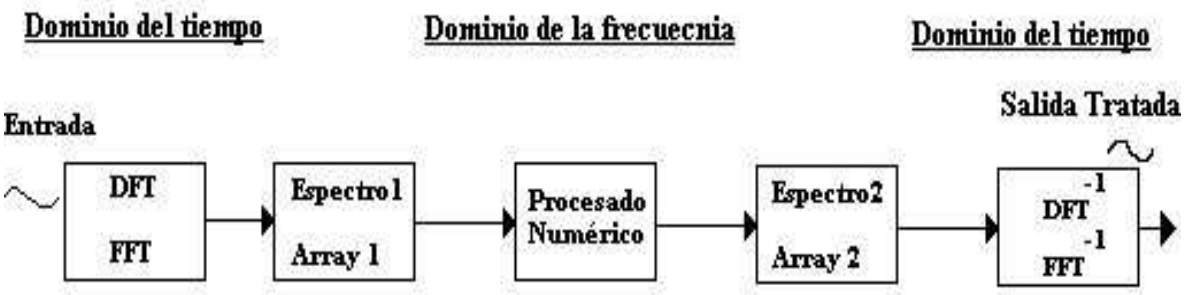


Figura 55. Esquema para el procesado en el dominio de la frecuencia en un DSP.

En función de las necesidades del usuario, el procesado numérico deberá variar. Es importante señalar que en los microprocesadores DSP el cálculo de la DFT y FFT no es excesivamente costoso, pues éstos están especialmente pensados para que la implementación de tales algoritmos sea lo más eficiente posible.

3.7. CARACTERIZACIÓN EXHAUSTIVA DE LOS FILTROS DIGITALES EMPLEADOS EN EL MODELADO DE LOS EFECTOS DE AUDIO.

Es el propósito de esta sección el complementar todo lo expuesto en el apartado anterior sobre modelado digital de los efectos de audio más empleados, proporcionando un conocimiento más profundo de los filtros usados, al margen del efecto sonoro para el cual sean usados. El motivo de la exposición por separado de estos contenidos, es el de no aturdir al lector con explicaciones y desarrollos matemáticos tediosos durante la comprensión del modelado de los distintos efectos. Para detalles ver [2],[3],[8] y [9].

3.7.1. LÍNEA DE DELAY.

3.7.1.1. FUNDAMENTOS.

La línea de delay sencilla es un boque constructivo fundamental para modelar efectos más complejos que un simple eco.

Su función es introducir un tiempo de retardo entre la entrada y la salida tal y como muestra la figura 56.

Figura 56. Línea de delay .

Donde se nota la señal de entrada por *x(n), con n=0,1,2,..*, y la longitud del retraso será de *M* samples. Así la señal de salida vendrá dada por la ecuación:

$$y(n) = x(n-M) \qquad \text{para } n=0,1,2,\ldots \qquad (26)$$

Donde por definición $x(n) \triangleq 0$ para n<0.

Antes de la era digital, las líneas de retardo eran caras e imprecisas en su forma analógica. En el dominio digital, por el contrario, un retardo de N samples se implementará trivialmente mediante el uso de tablas circulares en la memoria de un procesador, y un retraso no entero se conseguirá usando técnicas de interpolación, como la ya mencionada interpolación lineal empleada por ejemplo en chorus y flanger para conseguir la estimación correspondiente a un tiempo de retraso que no es múltiplo entero del período de muestreo.

3.7.1.2. RESPUESTA FRECUENCIAL.

A partir de la ecuación (26), se toma transformada z en ambos miembros, se aplica la propiedad de traslación y se llega a :

$$H(z) = z^{-M} \qquad \text{con } z = e^{j\omega T} \qquad (27)$$

Lo que supone una característica frecuencial en amplitud plana:

$$|H(z)| = 1 \qquad \text{para todo } \omega \qquad (28)$$

Y una característica en fase lineal que mantendrá a todas las componentes frecuenciales con el mismo tiempo de retraso.

$$\Phi(\omega) \angle H(z) = -\omega MT \qquad (28)$$

3.7.1.3.ÚLTIMAS NOTAS.

La línea de retraso o delay es como dijimos antes, un bloque constructivo esencial y se usará primordialmente para mezclar la señal retrasada a su salida, debidamente escalada, con la actual sin retrasar. Esto dará lugar a la aparición de las famosas muescas en el espectro y a los denominados filtros peine.

3.7.2. FILTRO PEINE CON REALIMENTACÓN DIRECTA.

3.7.2.1 FUNDAMENTOS.

Este tipo de filtro debe su nombre al lazo de realimentación directa con que cuenta y su diagrama genérico se muestra en la figura 57.

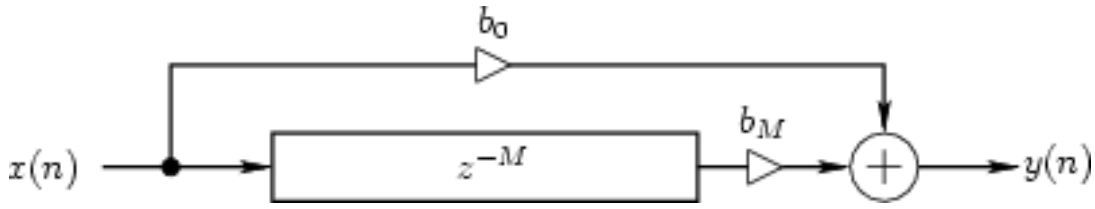


Figura 57. Filtro peine con realimentación directa.

La ecuación en diferencia será

$$y(n) = b_0\, x(n)+b_M\, x(n-M) \qquad (29)$$

Este filtro pude implementar, tal y como ya se ha expuesto, el simulador de eco, fijando $b_0 = 1$ y $b_M = g$. Así, este filtro es el modelo computacional de un simple eco discreto.

3.7.2.2. RESPUESTA FRECUENCIAL.

Este tipo de filtro toma su nombre de las muescas en forma de peine que aparecen al mezclar la salida de la línea de delay con la señal x(n). De esta forma, partiendo de (29), teniendo en cuenta *que* $b_0 = 1$ y $b_M = g$ y aplicando transformada z , así como las propiedades de traslación y linealidad de la misma se llegará la función de transferencia

$$H(z)= b_0 +b_M z^{-M} \qquad (30)$$

a partir de la cual, sustituyendo $z= e^{j\omega T}$ y tomando módulo , obtendremos

$$G(\omega) \triangleq |H(e^{j\omega T})| = |b_0 + b_M e^{-j\omega MT}| \qquad -\pi \leq \omega \leq \pi \quad (31)$$

Es esta ecuación la que se representa en la figura 58, para *M=5 , b_0 = 1, b_M=0.1, 0.5 y 0.9.*

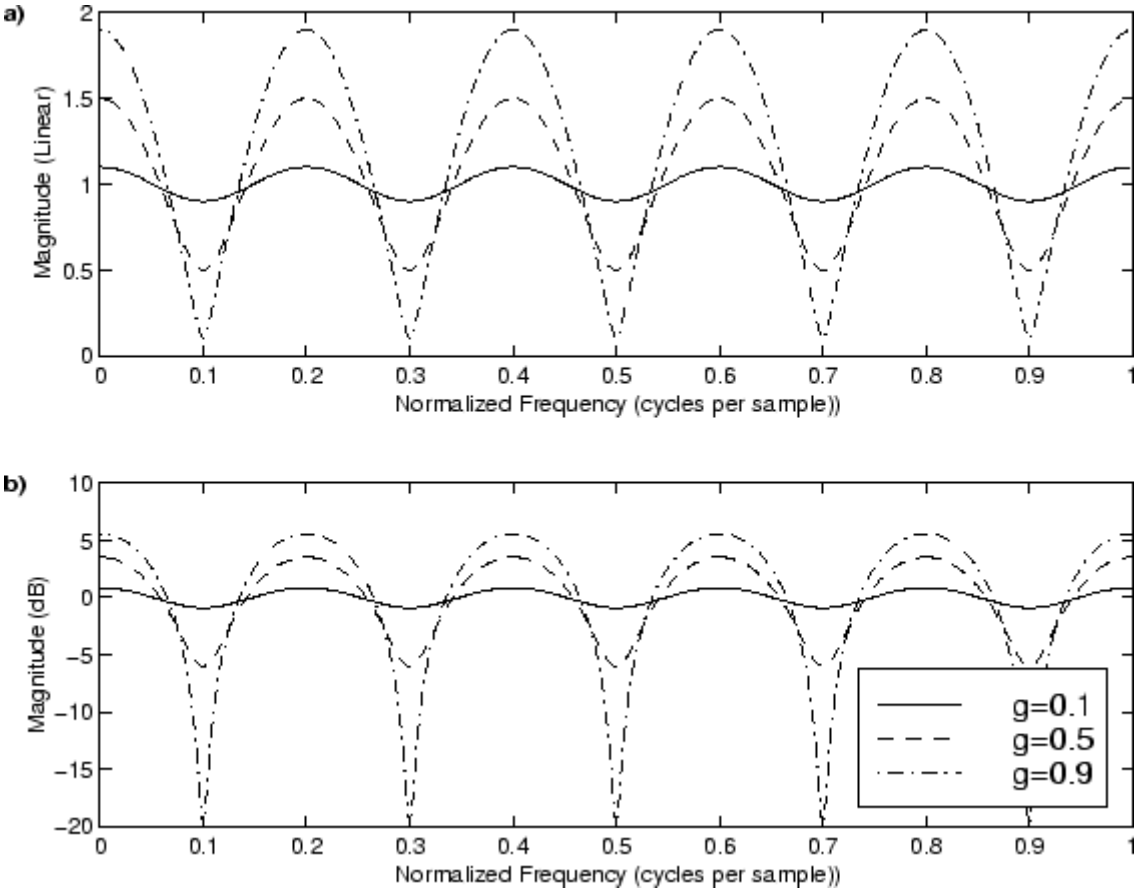


Figura 58. Respuesta frecuencial en amplitud del filtro peine. A) Escala lineal. B) Escala logarítmica.

Cuando $b_M = 1$, obtenemos el resultado simplificado

$$G(\omega) = |1 + e^{-j\omega MT}| = |e^{-j\omega M/2}|\,|e^{j\omega MT/2} + e^{-j\omega MT/2}| = 2|\cos(\omega MT/2) \qquad (32)$$

Sobre la figura 58 se deben hacer una serie de aclaraciones importantes. Así, si M aumenta, aumenta el tiempo de retraso, lo que ocasiona que las muescas de la respuesta se compriman hacia la izquierda. Si M disminuye, disminuirá consecuentemente el tiempo de retraso, lo que ocasionará el efecto contrario, esto es las muescas de la respuesta se estirarán hacia la derecha ocasionando un efecto de filtrado

mucho más apreciable y usado en efectos ya descritos, como el flanger o el phaser que aprovechan la compresión y expansión de las muescas en un rango de tiempos de retraso con extremos muy pequeños.

3.7.3. FILTRO PEINE CON REALIMENTACÓN HACIA ATRÁS.

3.7.3.1. FUNDAMENTOS.

Este tipo de filtro toma su nombre del lazo de realimentación hacia atrás que puede verse en la figura 59.

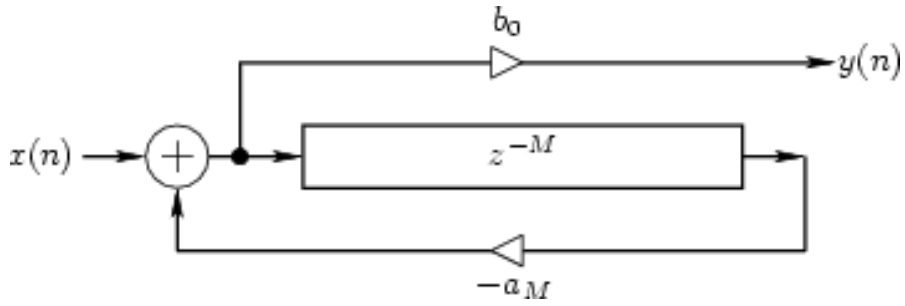


Figura 59. Filtro peine con realimentación hacia atrás.

Siendo su ecuación en diferencia

$$y(n)=b_0\, x(n)-a_M\, y(n-M) \qquad (33)$$

Este filtro es el modelo computacional de una serie de ecos exponencialmente decadentes y uniformemente espaciados en el tiempo para el caso de $b_0=1$ y $-a_M=g$

$$y(n)= x(n)-g\, y(n-M) \qquad (34)$$

Este filtro es recursivo, lo que motivará la necesidad de hablar acerca de su estabilidad. Así, el coeficiente de realimentación g decaerá ser menor que 1 en magnitud. De otra forma, cada eco sería más fuerte que el anterior, produciendo una serie de infinitas y crecientes repeticiones del sonido.

3.7.3.2. RESPUESTA FRECUENCIAL.

A partir de la ecuación (34) , tomando transformada z y aplicando las propiedades fundamentales de la misma se llega a la función de transferencia.

$$H(z) = \frac{1}{1 - g \cdot z^{-M}} \tag{35}$$

donde tomando módulo obtenemos la respuesta frecuencial en amplitud.

$$H(e^{j\omega}) = \frac{1}{1 - g \cdot e^{-j\omega M}} \tag{36}$$

Esta expresión es la que se muestra en la figura 60, para *M=5, g=0.1, 0.5 y 0.9.*

Para el caso especial en que *g=1* se tiene

$$G(\omega) = \frac{1}{2\,|\sin(\omega M / 2)|} \tag{37}$$

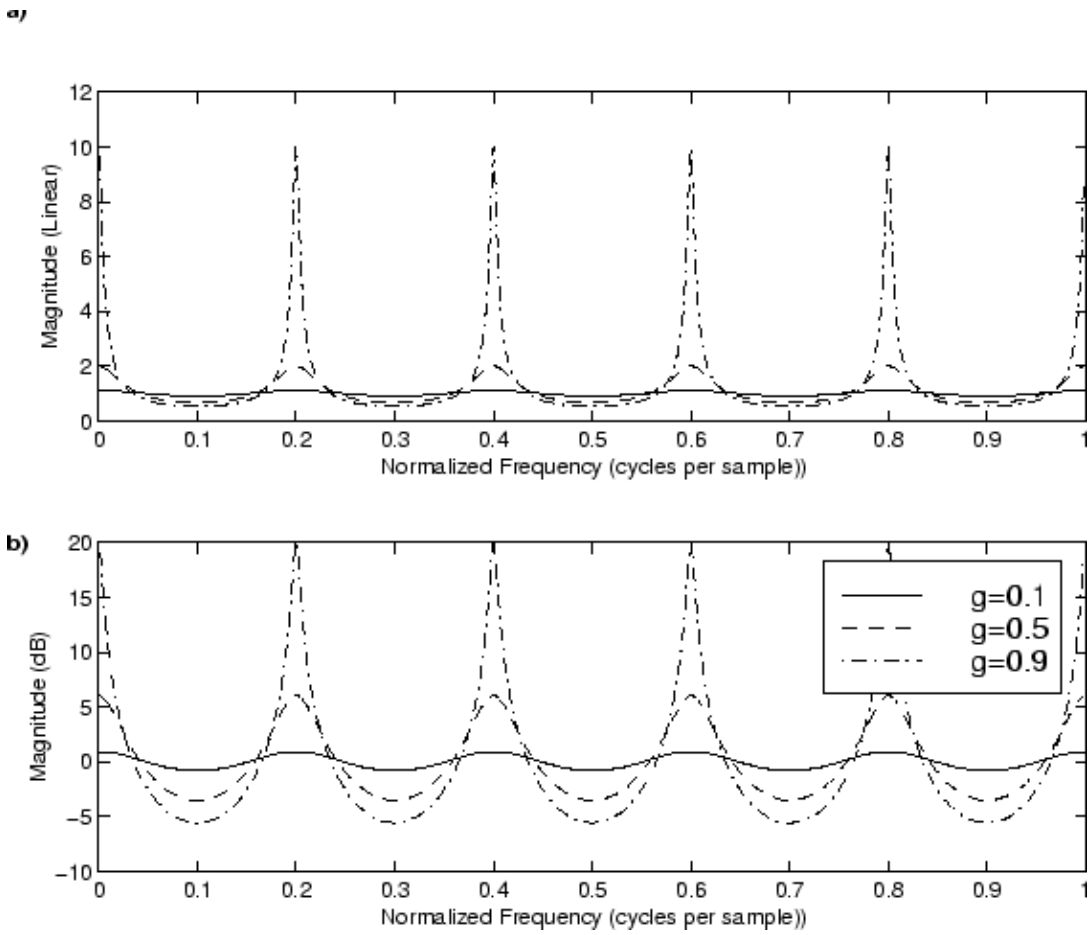


Figura 60. Respuesta en amplitud del filtro peine con realimentación hacia detrás. A) Escala lineal. B) Escala logarítmica.

3.7.3.3. EQUIVALENCIA ENTRE MODELOS REALIMENTADOS HACIA DETRÁS.

El modelo que se presentó en la sección pertinente era otro modelo de filtro realimentado hacia detrás. En su momento se dijo que era equivalente al modelo principal. Veamos esto.

Aquel modelo presentaba las ecuaciones

$$y(n) = x(n) + a(n) \quad Mix \tag{38}$$

$$a(n) = x(n-M) + g\, a(n-M) \tag{39}$$

Siendo g = La ganancia de realimentación.

Pues bien, se compararán ambas ecuaciones anteriores con la ecuación del modelo principal. Observamos que la ecuación (38) difiere de la (39)
en a(n). Tratemos de relacionar por tanto a(n) con y(n-M) en el nuevo modelo para tratar de reducirlo de alguna forma al principal. Así, en el nuevo modelo:

$$y(n-M) = x(n-M) + a(n-M)\,Mix \qquad (40)$$

De esta forma dividiendo (39) entre (40), tenemos:

$$a(n)/y(n-M) = g' = [x(n-M)+a(n-M)g]/[x(n-M)+a(n-M)Mix] \qquad (41)$$

Por tanto:

$$a(n) = g'\ y(n-M) \qquad (42)$$

Sustituyendo (42) en (38) obtenemos:

$$y(n) = x(n) + g'Mix\ y(n-M)$$

Donde si consideramos :

$$g'' = g'\,Mix = [Mix(x(n-M)+a(n-M)g)]/[x(n-M)+\ a(n-\ M)mix] \qquad (43)$$

Tenemos por fin:

$$y(n)= x(n) + g''\ y(n-M) \qquad (44)$$

De donde se concluye con que el nuevo modelo realimentado es equivalente al modelo principal descrito. Lo que sucede es que ahora, el control del filtro es más exigente, pues en g´´ intervienen tanto Mix como g. De cualquier forma, siempre deberá cumplirse que $g''<1$ para garantizar la estabilidad del sistema. Debe observarse que cuando $g=Mix$, $g'=1$ y $g''=Mix=g$ siendo los dos filtros exactamente iguales.

3.7.4. FILTRO ALLPASS.

3.7.4.1. PRINCIPIOS.

El filtro allpass es un bloque importante en el procesado digital de señales de audio. Se llama allpass porque todas las frecuencias son dejadas pasar intactas, o lo que es lo mismo, la respuesta en amplitud del filtro es la unidad para todas las frecuencias.

En cuanto a la fase se refiere , retraso que introduce el allpass será no lineal, de forma que no se retrasarán todas las frecuencias por igual. Para realizar su análisis debe saberse que este tipo de filtro está compuesto por un filtro peine con realimentación directa en serie con un filtro peine con realimentación hacia detrás. De esta forma, el allpass se obtendrá cuando el coeficiente de alimentación directo es negativo con respecto al de realimentación hacia detrás.

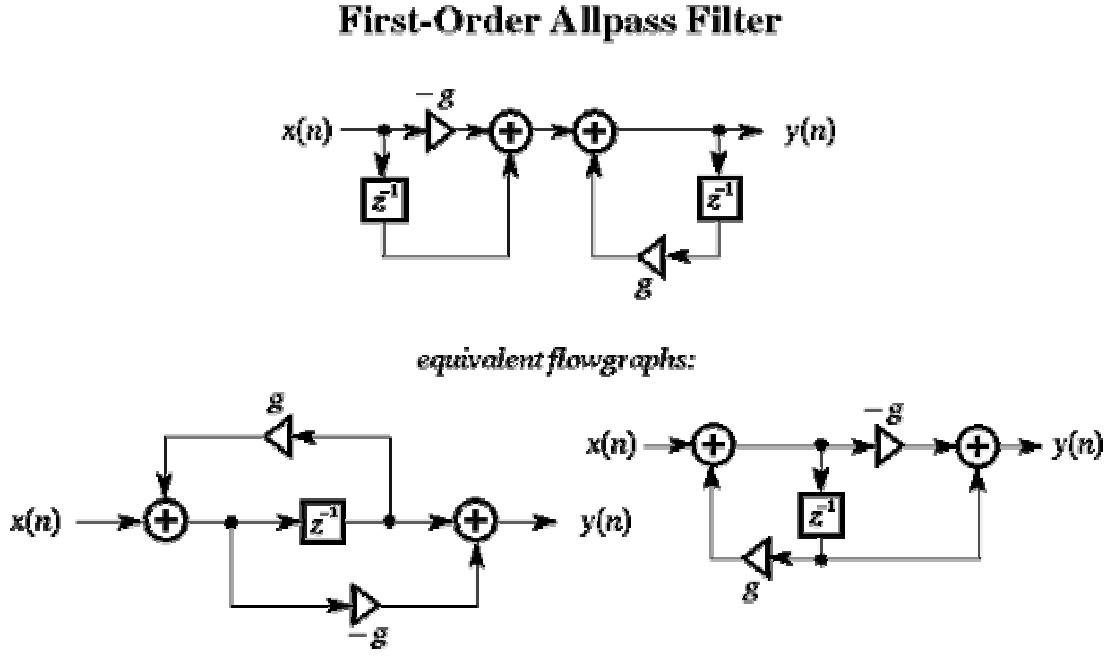


Figura 61. Modelos Allpass equivalentes.

Mediante la observación de primer diagrama de la figura 61 podemos escribir.

$$v(n) = -g\ x(n) + x(n-M) \qquad (45)$$

$$y(n) = v(n)+g\, y(n-M) \tag{46}$$

Sustituyendo (45) en (46) tenemos la ecuación ya conocida para el allpass.

$$Y(n) = -g\, x(n) + x(n-M) + g\, y(n-M) \tag{47}$$

3.7.4.2. RESPUESTA FRECUENCIAL.

Partiendo de la ecuación (47), tomando transformada z y aplicando las propiedades fundamentales de la misma se llega a la función de transferencia

$$H(z) = \frac{b_0 + z^{-M}}{1 + a_M z^{-M}}. \tag{48}$$

Tomando módulo obtenemos la ya mencionada respuesta plana.

$$\left|H(e^{j\omega})\right| = \left|\frac{\overline{a} + e^{-j\omega MT}}{1 + ae^{-j\omega MT}}\right| = \left|\frac{\overline{a} + e^{-j\omega MT}}{e^{j\omega MT} + a}\right| = \left|\frac{\overline{a + e^{j\omega MT}}}{a + e^{j\omega MT}}\right| = 1. \tag{49}$$

Tomando módulo, obtenemos la conocida respuesta en fase no lineal y ya mostrada .

$$H(e^{j\omega}) = -\omega - \tan^{-1}\frac{r\sin(\omega-\theta)}{1-r\cos(\omega-\theta)} \tag{50}$$

3.8. IMPLEMENTACIÓN Y SIMULACIONES.

3.8.1.SIMULACIÓN Y PROGRAMACIÓN EN MATLAB.

Llegados a este punto, cabe plantear la pregunta , ¿Cómo comprobar la validez de los efectos implementados? Antes de proceder a la final implementación en microprocesador se realizó la programación de los algoritmos en Matlab 6.5 y se analizaron y comprobaron los resultados obtenidos, con el objetivo de comprobar que estos se ajustaban a lo esperado y ya desarrollado en apartados anteriores.

De este modo el programa desarrollado es un selector de efectos que permite seleccionar el efecto deseado de entre el total de modelos estudiados y representa gráficamente los resultados de interés que nos permitirán ir comprobando la validez de los algoritmos.

Así, antes de entrar en el efecto en cuestión , que se seleccionará numéricamente, el programa nos pedirá los parámetros generales de la simulación: nos pedirá el valor de la frecuencia de la señal que se va a simular, esto es la señal procedente del instrumento musical, considerando un tono senoidal puro. A continuación nos pide la amplitud de la misma, así como el tiempo total que se desea que dure la simulación. Acto seguido nos pide el numero de puntos a representar por periodo de la señal senoidal de trabajo, parámetro relacionado directamente con la frecuencia de muestreo del simulado sistema discretizador, así si por ejemplo usamos una senoidal de 100hz y queremos 100ptos por período , la frecuencia de muestreo será por tanto 100 veces mayor, esto 10000Hz.

Una vez fijados los parámetros básicos de la simulación, el programa nos pide que seleccionemos el efecto que se desea probar para los parámetros genéricos ya introducidos. Veamos pues uno a uno cómo trabajan los distintos efectos a partir de los parámetros propios de cada uno, que serán solicitados por el programa una vez seleccionado el efecto en cuestión.

En la figura 61 se muestra como procesa el algoritmo de **distorsión1** una señal de 100 Hz con 25 puntos por periodo(muestro de 2500Hz) y amplitud 1 para un tiempo de simulación total de 0,05 segundos.

El programa informa que se ha seleccionado dicho efecto y pide los parámetros del mismo: nivel de saturación. En el ejemplo presentado, dicho parámetro se fijo al 0.5 de la amplitud total

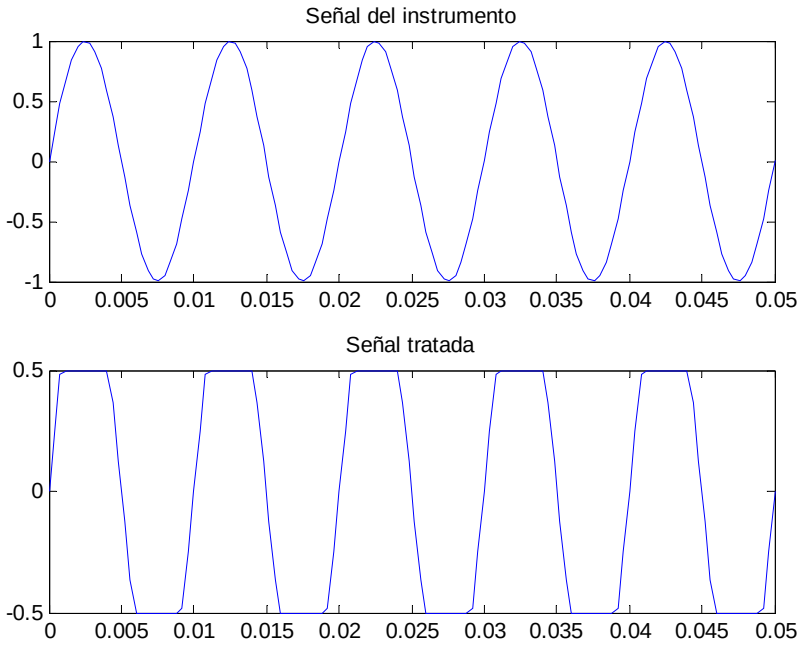


Figura 61. Procesado efectuado por el algoritmo de distorsión Básico para umbral de 0.5.

Se observa que el efecto trabaja correctamente, recortando la onda al 0.5 de su amplitud.

Un resultado equivalente se obtendrá para el **segundo algoritmo de distorsión**, si bien al tener éste dos parámetros de control más permite una mayor flexibilidad y la obtención de formas de onda más complejas. En la figura 62 tenemos este algoritmo trabajando para un umbral de 0.5 y con los coeficientes de mezcla a 1.

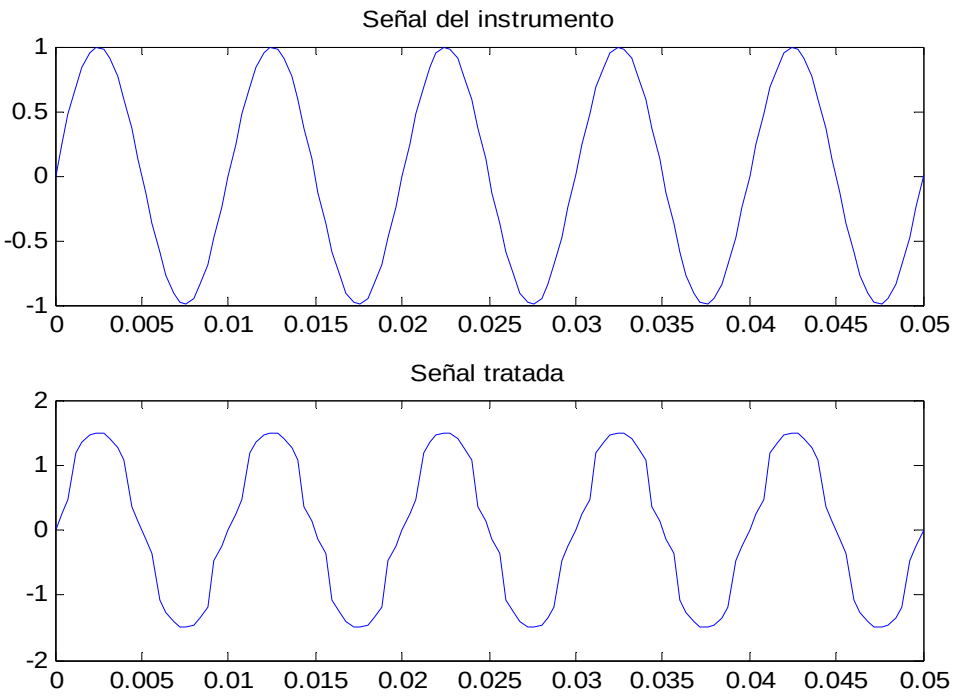


Figura 62. Algoritmo de distorsión2 para umbral de 0.5 y coeficientes de mezcla 1.

La figura 63 muestra el funcionamiento del algoritmo de puerta de ruido **NoiseGate** para un umbral igualmente de 0.5 que será el parámetro que pedirá el programa una vez nos avise que es ése el algoritmo seleccionado.

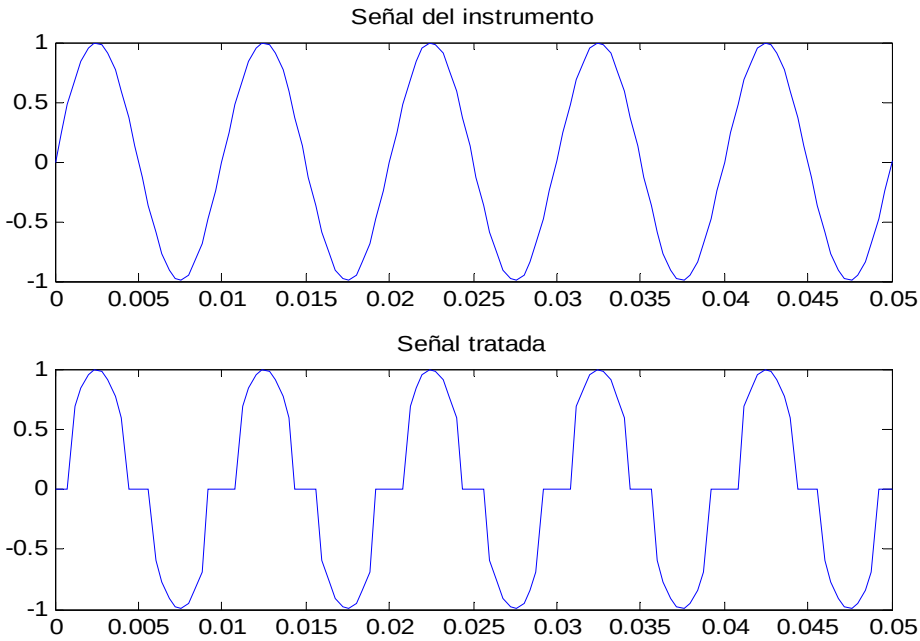


Figura 63. Noise gate con umbral 0.5.

Se observa pues que el algoritmo trabaja correctamente pues anula los sonidos por debajo del nivel umbral prefijado

Otras combinaciones son por supuesto posibles, de este modo, si el umbral fuera más elevado, se suprimiría una mayor porción de señal o incluso toda ella íntegramente. En la otra cara de la moneda, si el umbral es muy pequeño mayores porciones de la forma de onda pasarán por la puerta de ruido. Se comprobaron todas estas combinaciones mediante el procedimiento conocido, con resultados satisfactorios.

Las figuras 64 y 65 muestran los resultados obtenidos para el procesado efectuado por un el **compresor** y como se ha modificado la función de transferencia del sistema por su acción: umbral de 0.5 y ratio de 0.2 (casi limitación. Efecto exagerado) .

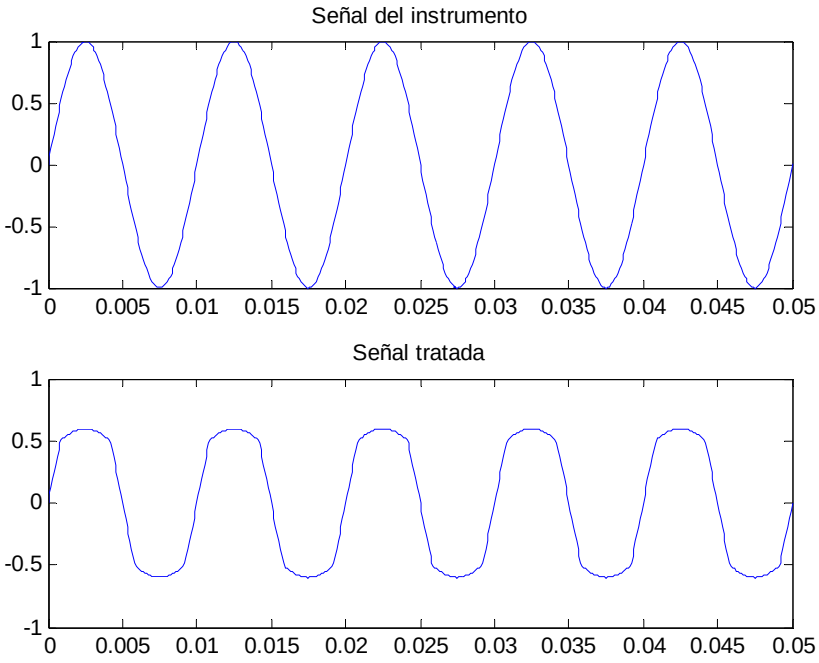


Figura 64. Compresor funcionando con umbral =0.5 y ratio =0.2

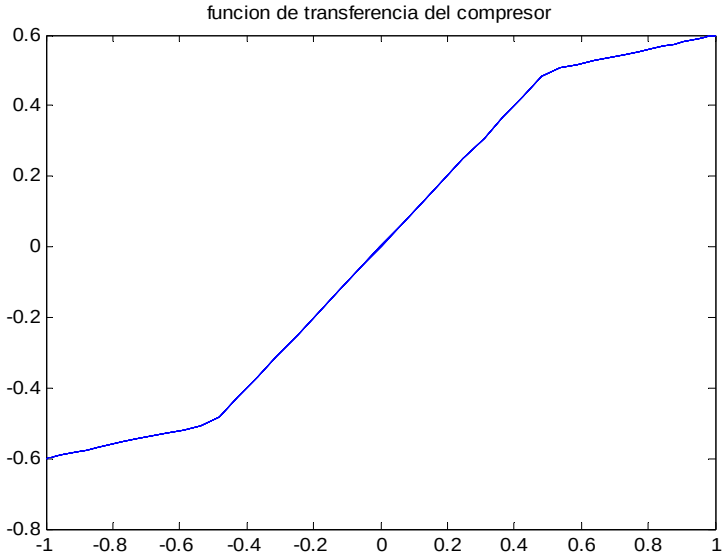


Figura 65.Función de transferencia modificada por efecto.

Así mismo el **expansor** se probó para un umbral de 0.5 con ratio 0.2, de manera que en las figuras 66 y 67 se pueden ver igualmente las formas de onda y la modificación de la función de transferencia.

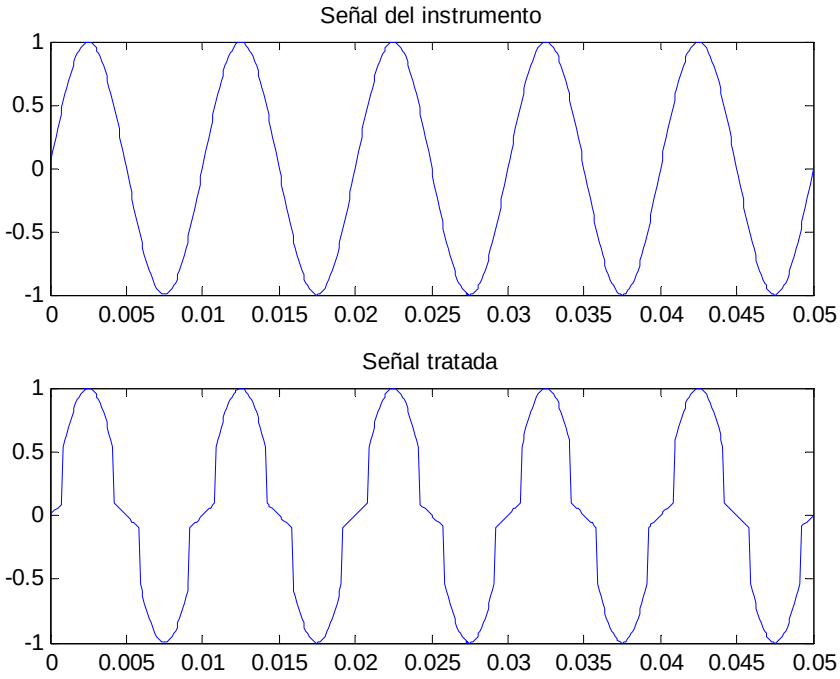


Figura 66. Expansor trabajando a 0.5 con ratio 0.2

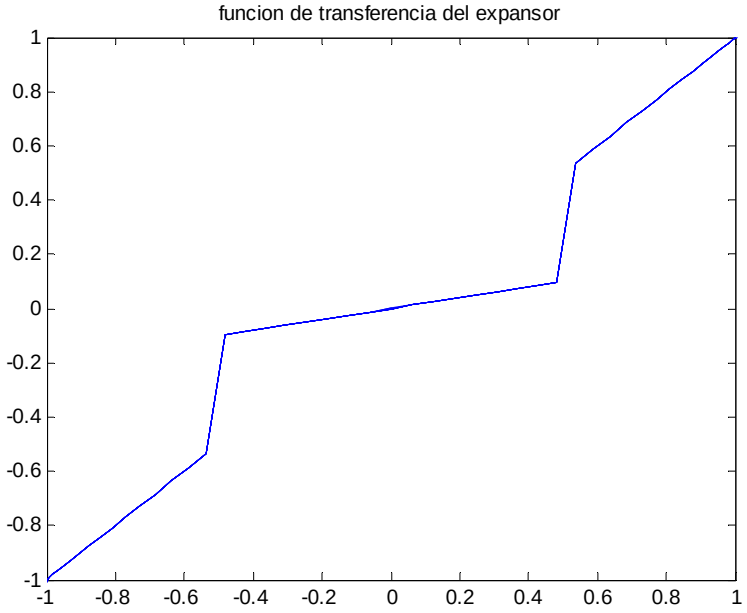


Figura 67. Modificación de la función de transferencia del sistema por el efecto de expansión.

Se puede observar por tanto, el funcionamiento del compresor, como la forma de onda es “achatada por encima del umbral”.

Se observa a partir de las gráficas 66 y 67, como la forma de onda se 'estira' notablemente (se expande) en cuanto al rango dinámico se refiere. Obsérvese que ahora lo que se hace es reducir la pendiente de la función de transferencia **por debajo** del umbral según el ratio introducido, siendo el extremo de la máxima expansión el NoiseGate o eliminación total de los sonidos por debajo del umbral.

En la figura 68 se muestra la experiencia realizada con **el modulador en anillo**, experiencia similar a la que se exponía a nivel teórico De este modo, con una senoidal en el generador interno del efecto de 60 Hz y con el tono senoidal ya conocido pero ahora de 40 Hz, el modulador multiplica ambas señales produciendo la señal suma de las bandas laterales, esto es 60-40=20Hz y 60+40=100 Hz. La experiencia se ha verificado anulando el camino de mezcla de la señal original con el modulador para ver directamente la salida de éste, con el objetivo de obtener un resultado más claro.

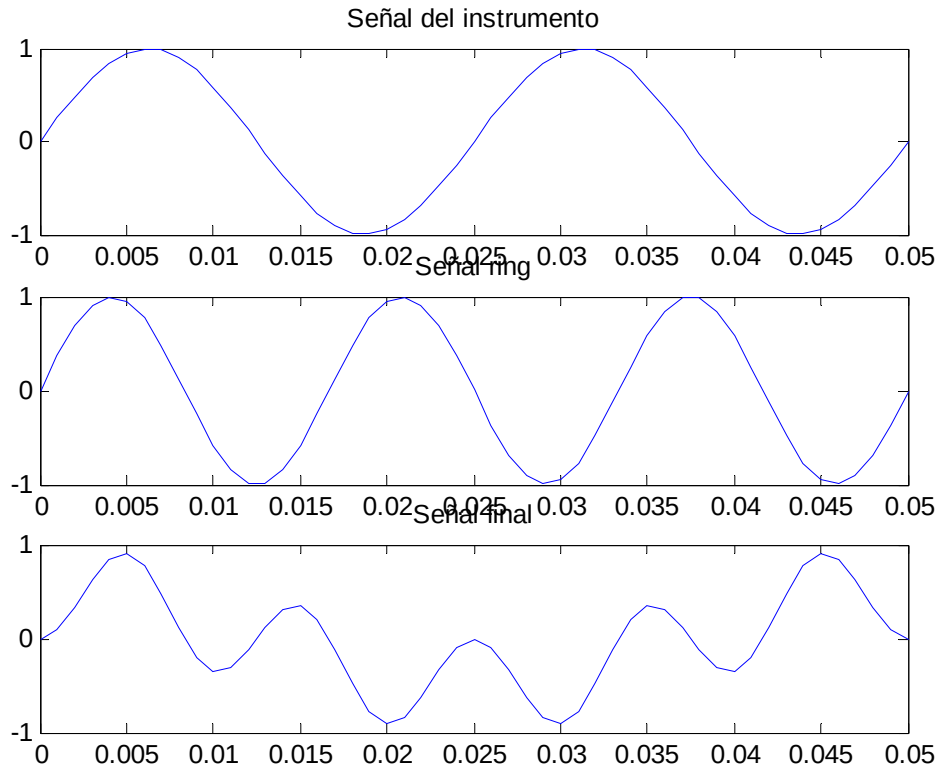


Figura 68. Modulador en anillo trabajando con un oscilador interno de 60Hz para modular una señal de entrada de 40 Hz.

Obsérvese a la vista del resultado como éste es similar al esperado y mostrado en teoría. El modulador responde perfectamente a los cambios de parámetros y la activación del camino de mezcla de la señal original con la salida del modulador

funciona correctamente. Es importante hacer notar que aunque se contempla como variable la amplitud de la moduladora procedente del oscilador interno, el estudio verifica que los resultados serán los esperados para una onda moduladora de amplitud la unidad.

A continuación se repasarán las conclusiones obtenidas para los efectos basados en el **uso de retardos**, según el mismo método de comprobación y simulación, si bien debe aclararse que se usa un solo período de la señal de prueba del instrumento con el objetivo de ver más claramente el resultado generado por el algoritmo.

Así, la figura 69 muestra el algoritmo **de delay básico** en funcionamiento y sin realimentación, produciendo una copia de la forma de onda original a un tiempo suficientemente alto como para que no se mezcle con el la original, y atenuada con respecto a ésta según Mix=0.5. El tiempo de retardo es de 0.015 pues la onda de prueba es de 100Hz(período de 0.01) y con este retardo se verá separada. Mix controla la ganancia de la repetición.

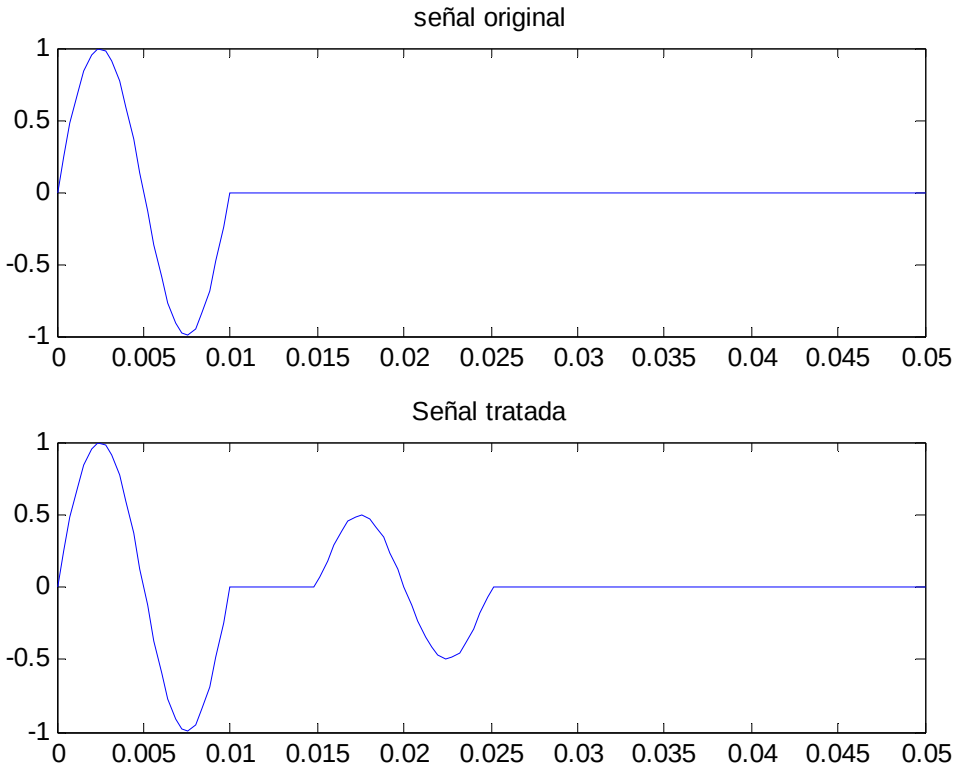


Figura 69. Delay básico sin realimentar. Retraso de 0.015 segundos en señal de 100Hz atenuación de la repetición de 0.5.

Se comprobó así mismo que los controles ganancia y tiempo de retraso responden perfectamente a lo esperado. Debe comentarse también que existe la posibilidad de introducir un tiempo de retardo que no sea múltiplo del período de muestreo por lo que se ha incluido en éste y en todos los efectos que usan tiempos de retrardo la posibilidad de interpolar linealmente entre las dos muestras más próximas, cuando sede el caso de que el tiempo de muestreo introducido no lleve a una muestra en concreto, sino algún lugar entre otras dos. El programa solicita el activar o no esta prestación, si no se activa , coge directamente la muestra más cercana. Se comprobó no obstante que para pruebas con tonos senoidales sencillos no hay mucha diferencia entre métodos, si bien es de prever que cuando las señales se compliquen el no usar interpolación podría generar distorsiones no deseadas.

La figura 70 muestra la misma experiencia que la realizada en la figura 69 , pero ahora **con realimentació**n de valor 0.3 (poca realimentación), para el modelo de delay realimentado.

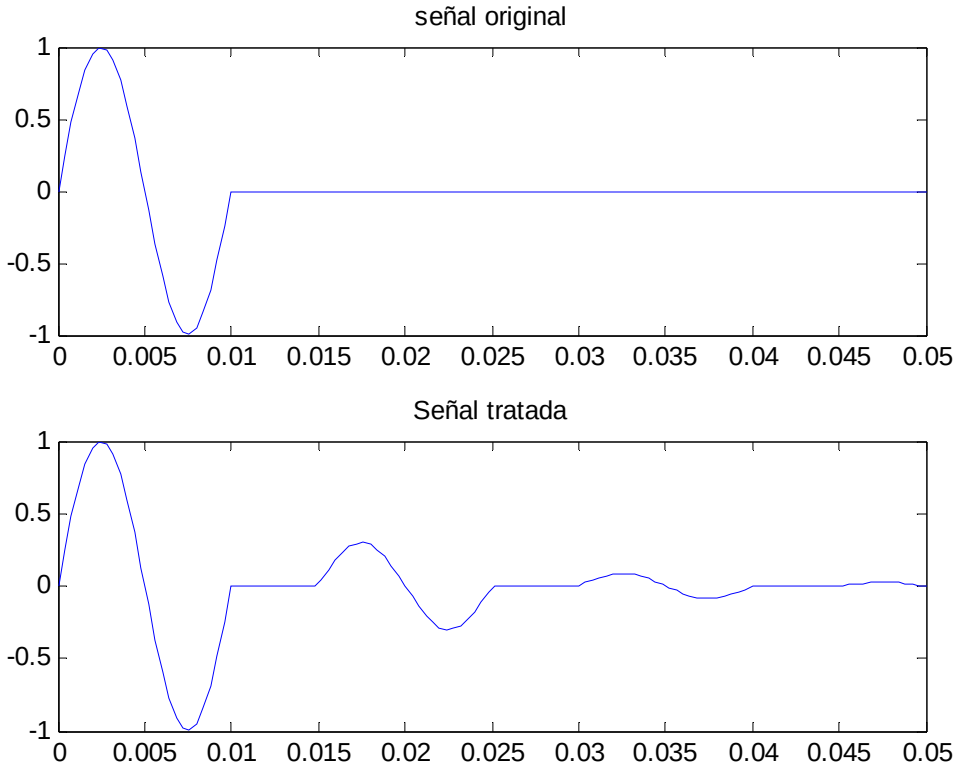


Figura70. Delay Realimentado. Retraso de 0.015seg , FeedBack = 0.3.

Se observa que lo obtenido responde a las repeticiones sucesivas progresivamente atenuadas del sonido original que es justo lo que se esperaba del efecto.

Los efectos de **chorus y flanger** por su parte están basados en el algoritmo de delay básico y realimentado respectivamente, los cuales ya han sido comprobados. No obstante incorporan un oscilador de baja frecuencia o LFO, que será el bloque añadido a los algoritmos base ya conocidos y probados. Para asegurar pues, desde el punto de vista funcional la correcta operación de estos efectos, se ha probado el correcto funcionamiento del LFO empleado junto con el algoritmo completo, de ahí que las figuras incluyan tanto las formas de onda del instrumento como al LFO, en una triple ventana.

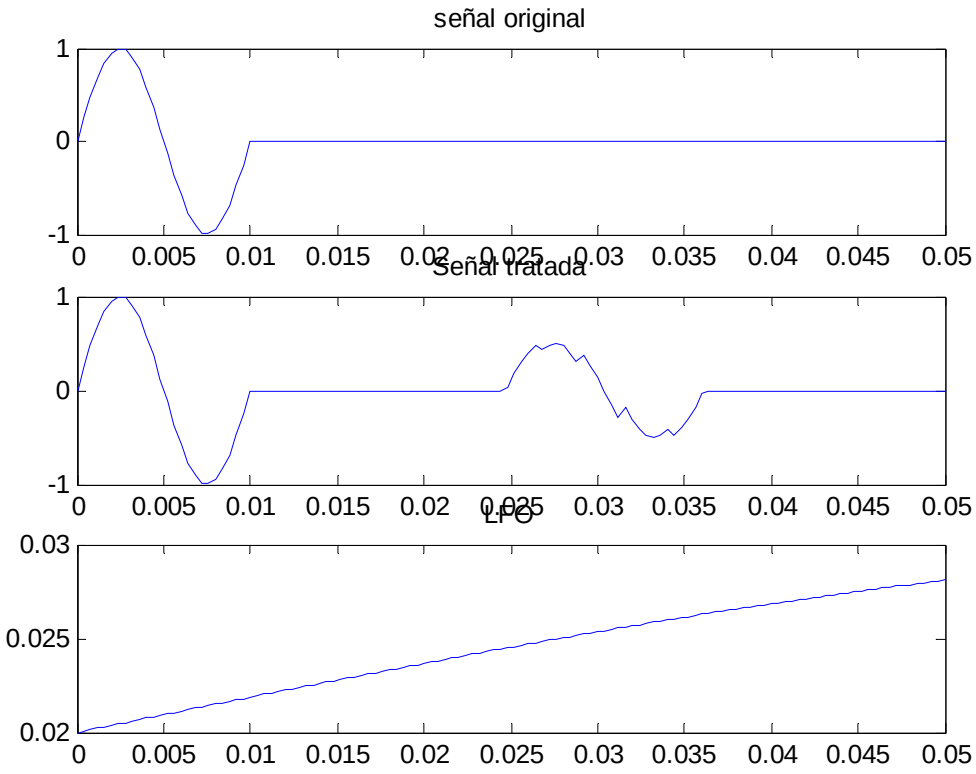


Figura 71. Efecto de Chorus con retraso base a 0.01seg y barrido de retraso de 0.02, ganancia de 0.5 . LFO de 3Hz y ganancia de la repetición de 0.5

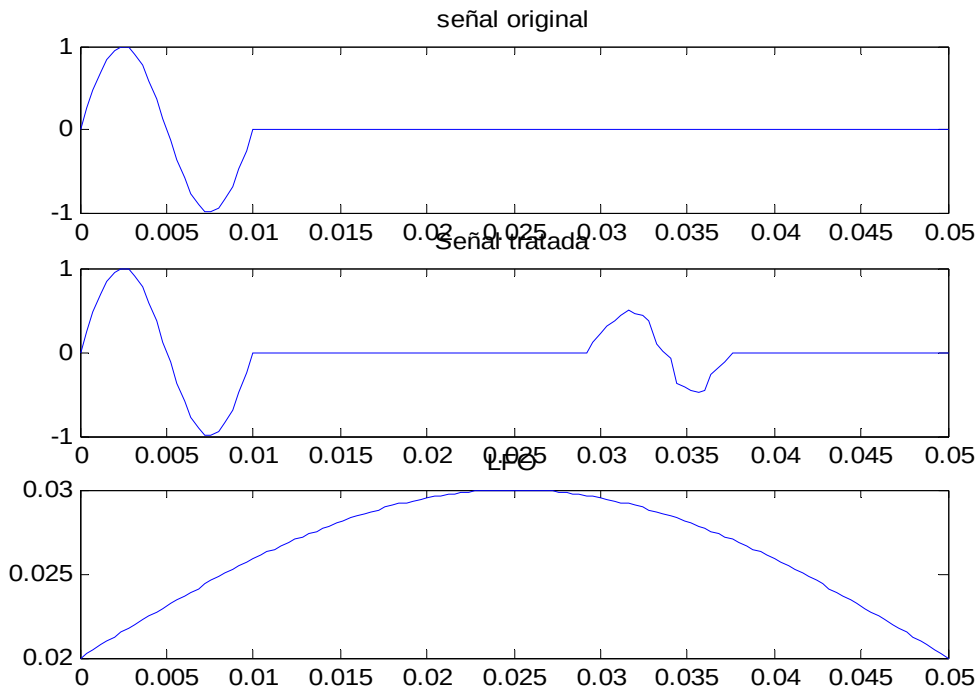


Figura 72. Efecto de Chorus con retraso base a 0.01seg y barrido de retraso de 0.02, ganancia de 0.5 . LFO de 10Hz y ganancia de la repetición de 0.5

Las dos figuras anteriores ponen de manifiesto el correcto funcionamiento del LFO y la influencia de la frecuencia del mismo sobre el resultado. Así, se observa que el resultado vibra en ambos casos, si bien la distorsión es mayor para una frecuencia mayor del LFO. Así mismo, si a igualdad de frecuencia, se hubieran introducido un barrido de delay más grande en uno que en otro, la distorsión de vibración habría sido mayor en el caso del mayor barrido.

En las dos figuras siguientes se hace lo mismo con **el Flanger**, si bien este algoritmo incorpora la posibilidad de emplear un LFO senoidal o triangular, y añade la realimentación.

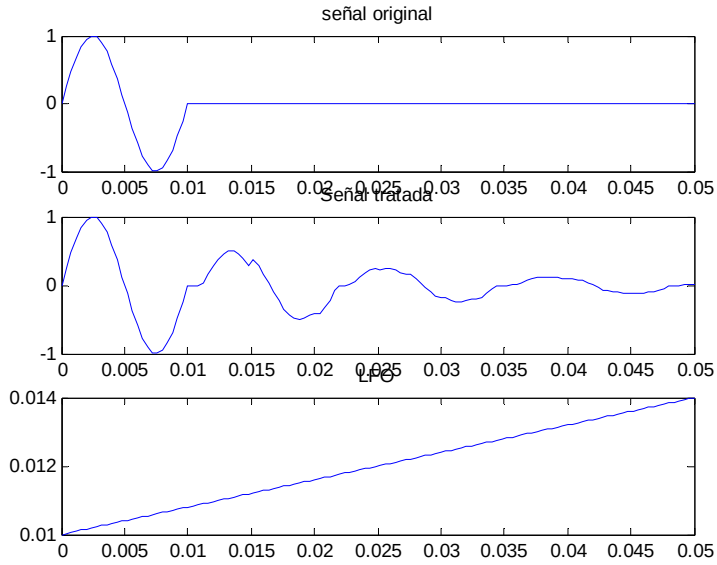


Figura 73. Efecto de Flanger con retraso base a 0.01seg y barrido de retraso de 0.02, ganancia de 0.5 LFO de 2Hz y ganancia de la repetición de 0.5

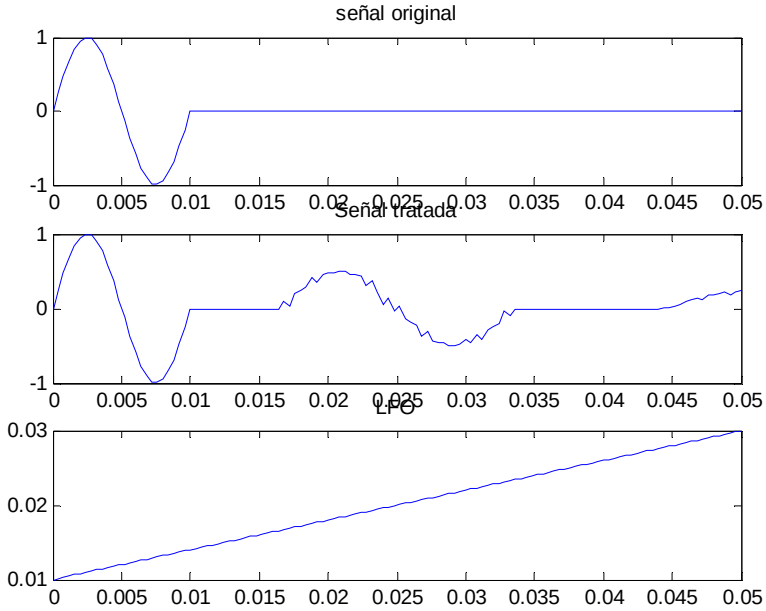


Figura 74. Efecto de Flanger con retraso base a 0.01seg y barrido de retraso de 0.02, ganancia de 0.5 LFO de 10Hz y ganancia de la repetición de 0.5

Al igual que antes, al aumentar la frecuencia del LFO, aumenta la distorsión por vibración, además como se usa una onda triangular, los retardos varían más abruptamente, de ahí el mayor espaciado entre repeticiones. El algoritmo opera correctamente.

La reverberberación realista, se comprobará observando los 4 taps por separado y luego el filtro Butterworth, así se tendrá para la línea tap cuatro ecos a distintos tiempos y atenuados según sus respectivos factores de escala. El filtro Butterworth se ha comprobado mediante la introducción de valores numéricos y observando su operativa con resultados satisfactorios. La figura 75 muestra los 4 taps trabajando con retrasos de 0.01, 0.02, 0.03 y 0.04 posiciones, lo que nos permitirá ver 4 ecos de forma continua tras el sonido original. Las atenuaciones serán 0.5, 0.4, 0.3 y 0.2 respectivamente, de ahí que las copias decrezcan suavemente. La señal original se mezcla sin escalar con los ecos, de ahí que el primer tono corresponda al tono senoidal original sin modificar.

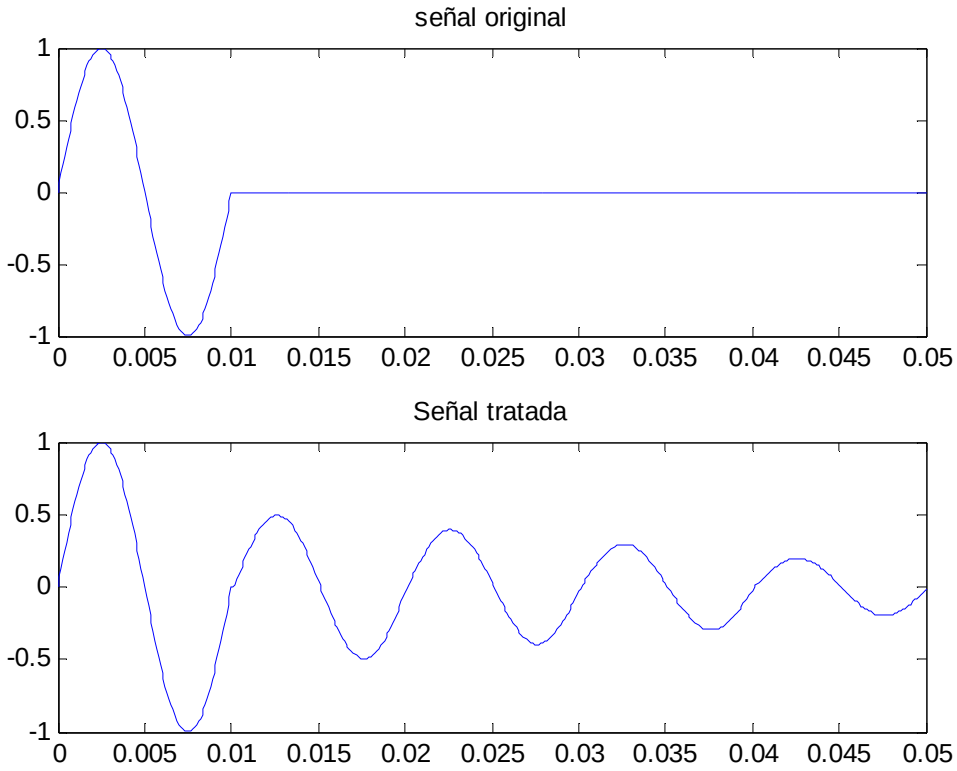


Figura 75. 4 taps trabajando con retardos de 0.01, 0.02, 0.03 y 0.04 segundos con atenuaciones de 0.5, 0.4, 0.3 y 0.2 respectivamente.

El algoritmo responde a lo esperado, teniendo como ventaja el control independiente de cada uno de los taps, lo que permitiría simular la reverberación de forma más realista que un simple delay realimentado.

Finalmente, para la comprobación del correcto funcionamiento **del Phaser**, se simuló el bloque allpass por separado, esperando un resultado adecuado y conforme a la respuesta al impulso unitario para dicho bloque. En la figura 76 se muestra la experiencia realizada con el bloque Allpass para una ganancia g=0.5 y un retraso de 0.01 segundos para la señal de prueba de 100Hz.

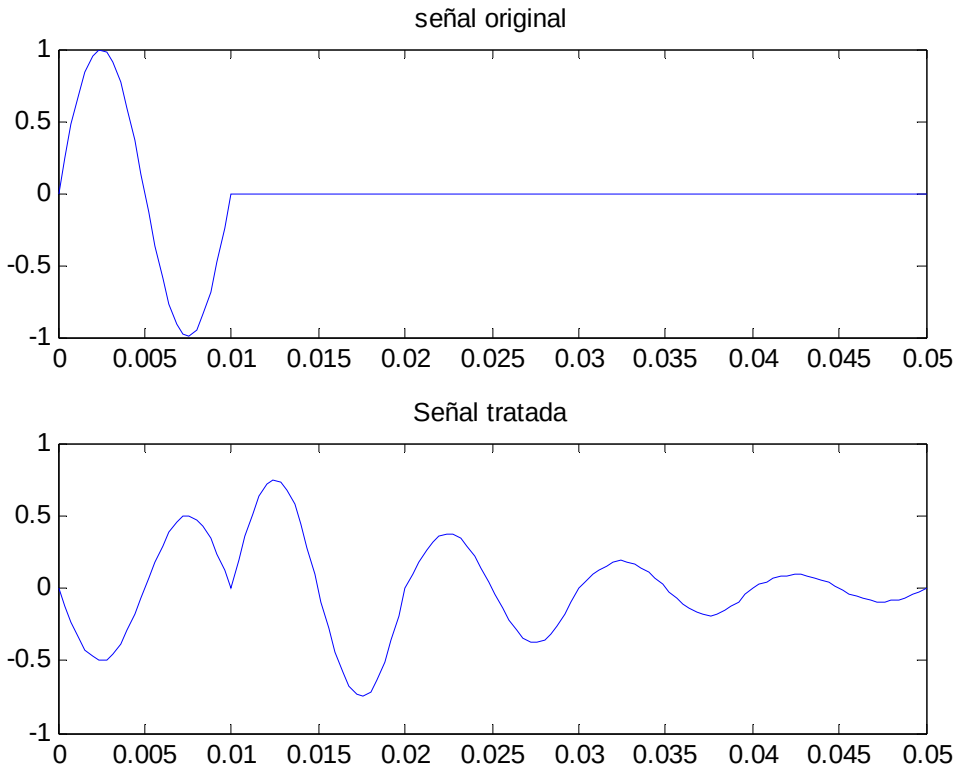


Figura 76. Bloque allpass para g=0.5 y retraso=0.01 segundos

Se observa que el resultado obtenido está en consonancia con la respuesta al impulso del bloque allpass: una inversión de fase con ganancia g y una atenuación progresiva. Se comprobó que el cambio de parámetros se corresponde con la relación que se mostraba en dicha respuesta al impulso unitario.

El algoritmo responde satisfactoriamente al cambio de los tiempos de retraso, así como al cambio de las constantes de escala.

3.8.2. SIMULACIÓN Y PROGRAMACIÓN EN C PARA MICROPROCESADOR.

El siguiente paso en el proceso de desarrollo software es el de eliminar Matlab todo lo posible para garantizar que el sofware diseñado funcionaría correctamente en un DSP. Para ello, Matlab tan solo se ha usado para generar el archivo de datos que hace las veces de entrada al sistema de procesado de audio, y para representar gráficamente el resultado de dicho procesado previa escritura del resultado del programa en C en un archivo de datos que Matlab pueda leer.

Así, el programa diseñado en C está prácticamente preparado para ser utilizado en un procesador DSP, pues ya trabaja muestra a muestra, usando los algoritmos en la forma adecuada para el trabajo sobre microprocesador. De esta forma, para adaptarlo al microprocesador será necesario eliminar la lectura y escritura sobre archivo de datos, corregir los índices de los bucles para que estos se encuentren en permanente ejecución y añadir las funciones de inicialización del microprocesador, así como las de configuración del conversor analógico-digital y digital-analógico y las de manejo por interrupciones o polling del canal de transmisión.

En cuanto a los resultados obtenidos, comentar que éstos son idénticos a los obtenidos en Matlab, por lo que carece de sentido volver a repetirlos uno a uno de nuevo. Se presenta un esquema ilustrativo del trabajo en esta fase de desarrollo:

Figura 77. Esquema de simulación en C.

El código generado únicamente en Matlab, y el generado en la segunda fase de uso intermedio de C, se incluyen en el anexo correspondiente.

3.8.3. IMPLEMENTACIÓN EN LENGUAJE C SOBRE PLATAFORMA DSP ADSP 21000.

En este objetivo se elabora un programa. La tarjeta DSP utilizada dispone de todo lo necesario para desarrollarlo: convertidor A/D (analógico/digital), ADSP 2100, convertidor D/A (digital/analógico),herramientas software. No cuenta con la interfaz con el usuario, que formaría parte de futuros desarrollos.

El programa inicializa todos los componentes de la placa necesarios, elige el efecto a realizar , que le será indicado por la variable efecto y comienza a ejecutar dicho efecto muestreando la señal de entrada procesándola numéricamente y enviando el resultado al conversor digital analógico para devolver la señal al campo analógico del audio vía amplificador. Debe señalarse que el código está estructurado y convenientemente comentado para ir siguiendo todos los pasos(ver un poco más adelante).

3.8.3.1. DSP SHARC 21061.

Aunque en un anexo se incluye el Data-sheet del DSP utilizado se hace necesario un resumen de sus características para continuar adecuadamente.

La SHARC EZ-KIT LITE ha sido la placa utilizada, se presenta como uno de los mejores valores de desarrollo hoy día. Permite, por un coste reducido, el acceso a uno de los mejores procesadores de señal en coma flotante. 32 bits de precisión simple(40 con precisión expandida) en coma flotante. Tres unidades computacionales independientes en paralelo: una ALU, un multiplicador y un registro de desplazamiento. Bancos configurables de memoria. Dos puertos serie de 40 Mbits por segundo. Dispone también de un puerto RS232 para comunicarse con PC donde se desarrollan los programas y de un conversor analógico/digital/analógico para los procesos de digitalización y reconstrucción de las señales, el AD1847 .

La placa puede funcionar independientemente de PC una vez tenga el programa en memoria.

3.8.3.2. ¿POR QUÉ UN DSP?

Aunque ya se ha hablado de ello con anterioridad, diremos que los procesadores digitales de señal son una clase especial de microprocesadores que están optimizados para realizar operaciones en tiempo real que es lo que se persigue. Es su arquitectura lo que los hace diferentes de los microprocesadores de propósito general siendo más baratos y más flexibles en estas aplicaciones. Así estos procesadores cuentan entre otras características con:

- Unidades computacionales rápidas y flexibles.
- Flujo libre de los datos hacia y desde las unidades computacionales.
- Precisión y rango dinámico expandidas en las unidades computacionales.
- Generadores direccionales duales.

Debe tenerse en cuenta que la precisión requerida actualmente es muy grande y supera muchas veces las capacidades de los microprocesadores, haciéndose casi imprescindible que un microprocesador pueda hacer operaciones en coma flotante para poder satisfacer la demanda del procesamiento digital de señales.

3.8.3.3. PROGRAMA DESARROLLADO.

Comentar aquí detalladamente cómo se ha realizado el programa, comentando las secciones y métodos más importantes. Para más detalle consultar la sección de código debidamente comentado. El programa se ha estructurado de la siguiente manera:

a) Declaraciones de variables y constantes de inicialización y propias de micro.

b) Declaraciones de variables y constantes de los efectos (variables globales).

c) Rutinas de los efectos una a una(con variables locales propias del efecto).

d) Rutinas de inicialización.

e) Programa principal.

a) Comenzaré explicando que al comienzo se encuentran todas definiciones de registros del DSP, los registros de la DMA. Este primer segmento de código se ha tomado tal cual de los manuales de arranque. Llega hasta el comentario /*variables procesado*/

b) Las variables de procesado son las variables globales que se utilizan de forma usual en todos los efectos. Es conveniente indicar que se han declarado como globales para mayor comodidad , para no redeclarar en cada efecto cada vez las mismas variables en todos los efectos como locales o tener que pasar gran cantidad de parámetros. Son:

- Búfferes de transmisión y recepción desde el conversor:

```
int rx_buf[3];
int tx_buf[3] = {0xcc40, 0, 0};
```

- Indicador de interrupción:

La llegada de un nuevo dato se maneja por interrupciones, de manera que flag se pone a uno cuando llega un nuevo dato, esto es chequeado por la función del efecto de ejecución, de manera que la pone a cero, efectúa el procesado numérico correspondiente y luego la envía al puerto serie para que en la siguiente interrupción sea mandada al conversor.

```
int flag=0;
```

- Llegada de datos salida de datos:

Al llegar el dato del conversor se guarda en lectura y cuando se quiere mandar en salida.

```
float lectura;
float salida;
```

- Variable que decide el efecto en ejecución, frecuencia de muestreo y escala:

La primera controla el efecto en ejecución, la segunda la frecuencia de muestreo del sistema y la tercera el valor máximo numérico del dato procedente del conversor. Es importante hacer notar aquí que la frecuencia de muestreo se puede elegir en la rutina de inicialización del conversor y el usuario deberá introducir el valor correcto tanto en la inicialización del conversor como en esta variable para que el sistema funcione correctamente. En cuanto al valor escala, este valor se usará para trabajar con todos los valores numéricos entre 0 y 1. Así, llega un dato, que al ser dividido por el valor máximo posible estará escalado entre 0 y 1, se realizarán todas las comparaciones y operaciones así, y después se devolverá el valor resultante al rango del conversor multiplicando por escala de nuevo. Esto se hace así para poder introducir los parámetros en tanto por uno lo que resulta sin duda más adecuado y "amigable" para el usuario.

```
int efecto=10;
float fmuest=18900; /*valor frecuencia de muestreo Hz*/
float escala=32767; /*valor escalado a 1*/
```

-Búfferes circulares:

El buffer trabajo almacena muestras de forma que cuando llena el buffer vuelve al comienzo a seguir guardando muestras a intervalos del período de muestreo. La longitud del buffer delimitará el máximo retraso con el que podemos extraer una muestra pasada. Así el número de muestras necesario para implementar un tiempo de retraso dado será **tiempo*fs**.

La tabla trabajo contiene dichas muestras, y longitud la longitud de la tabla. Debe ajustarse la longitud del buffer junto con la frecuencia de muestreo para la obtención de los tiempos de retardo deseados.

```
float trabajo[7600]={0};
float tabla[1]={0};
int longitud=7600;
```

- Variables para controlar la frecuencia de muestreo:

Según el valor de esta variable cambia la frecuencia de muestreo :

Samplerates (in kHz):

(0)	8	(1)	5.5125 (default)
(2)	16	(2)	11.025
(4)	27.42857	(5)	18.9
(6)	32	(7)	12.05
(8)	N/A	(9)	37.8
(10)	N/A	(11)	44.1
(12)	48	(13)	33.075
(14)	9.6	(15)	6.615

```
volatile static int ordered_gain;
volatile static int ordered_rate=5; /* ver tabla de frcuencia*/
```

- Variables globales para control de los osciladores de baja frecuencia:

Hacer notar, pues no se comentó específicamente nada a este respecto que los osciladores de baja frecuencia senoidales empleados en los efectos de chorus y flanger se han implementado con un filtro digital según la ecuación:

$$y(n)=-a\,y(n-1)-y(n-2)$$

donde:

$a = -2\cos w$	A=amplitud
$y(-1) = -A\sin w$	w=frecuencia(rad/muestra)
$y(-2) = -A\sin w$	

```
/*osciladores */
/*variables parámetros->las tiene la función de arranque*/
float pi=3.1415926;
float flfo=0.5;
float tdelay=0.001;
```

```
float barrido=0.009;
float amplitud;
float tlfo;
float ptosdec;
int ptos;
int periodo;
float m1;
float m2;
int p=0;
/*variables del filtro ->globales*/
float oscilador;
float frecuencia;
float z;
float v;
float w;
```

c) En cuanto a las rutinas de los efectos, a continuación las comentaremos una a una:

Antes de explicar una a una las funciones de los efectos, debe hacerse notar que el dato procedente del conversor en la variable lectura se escalará para dejarlo en tanto por uno dividiendo por el valor a fondo de escala del conversor, guardado en la variable escala (32767). Así mismo todos las funciones tienen un esquema semejante: cargan parámetros de control, esperan a que haya llegado un dato (indicador flag de interrupción a 1) y luego lo procesa (flan se pone a 0 para sucesivos datos), lo desescala multiplicando por escala y lo envía al conversor digital analógico, y queda en espera de un nuevo dato. Este esquema se repite **siempre** y cuando no se comente es por no reiterar explicaciones.

A continuación se detallan particularidades de las funciones una a una. Para una mayor comprensión consúltese la sección de código con los correspondientes comentarios.

- Distorsion1.

El único parámetro de control es el grado de saturación en tanto por uno al cual se limita la señal. El esquema de funcionamiento es sencillo y se muestra a continuación:

La función espera a que llegue un dato (flag=1), cuando se da este caso, se escala el dato como ya se ha comentado y se comprueba su valor, de forma que si es mayor que saturación la salida se limita al valor de +saturación. Igualmente si el dato fuera menor que –saturación , se limitaría a este valor. Si el dato está entre los valores positivo y negativo de saturación , el dato no se toca. Se desescala y se manda al conversor.

- Distorsion2:

El funcionamiento es idéntico , con la salvedad de que cuando se superen los valores de saturación o –saturación la salida es igual a salida=mix*lectura+nivel*saturación en el primer caso y salida=mix*lectura+nivel*-saturación en el otro caso.

-Noise Gate:

El único parámetro de control es otra vez saturación.

Una vez que el dato ha llegado , se escala como ya se ha comentado y sólo se hace salida igual a 0 cuando el dato está entre +saturación y –saturación, en otro caso salida es igual a lectura.

-Compresor:

Los parámetros serán umbral y ratio. El primero marca el nivel a partir del cual la señal se comprime. Y ratio la nueva pendiente de la función de transferencia lineal retocada.

Así pues una vez llegado el dato y escalado se cheque si el dato está por encima de + umbral o por debajo de –umbral, si no lo estuviera, no se hace nada y salida=lectura. Si alguno de los dos chequeos fueran positivos, se retoca la función de transferencia según salida=umbral+ratio*(lectura-umbral) o bien salida=-umbral +ratio*(lectura+umbral) respectivamente.

-Expansor:

Los parámetros son los mismos que en algoritmo anterior y la forma de operación muy similar: ahora cuando el dato esté comprendido entre +umbral y –umbral es cuando se retoca la función de transferencia según: salida=ratio*lectura. Si no está comprendido entre los valores positivo y negativo de umbral no se hace nada y lo que sale es igual a lo que entra.

- Ring Modulator:

Esta función es más compleja porque precisa de la generación de una señal senoidal propia de frecuencia selecionable que se vaya generando muestra a muestra. Para realizarla se usa también la función **arrancaring**, que fija los parámetros iniciales del oscilador interno, pues éste precisa de unas condiciones iniciales de las que partir. Así, este oscilador se ha realizado según:

$$y(n)=-a\,y(n-1)-y(n-2)$$

donde:

$a = -2\cos w$	A=amplitud
$y(-1) = -A\sin w$	w=frecuencia(rad/muestra)
$y(-2) = -A\sin w$	

Los parámetros de arranque necesarios son la frecuencia que se le quiere dar al oscilador, la frecuencia de muestreo del sistema y el número pi. Así el parámetro w en radianes/muestra debe calcularse a partir de los anteriores datos según frecuencia=(flfo/fmuest)*2*pi, siendo frecuencia la w teórica que se presentó arriba. La amplitud A queda fijada en la variable amplitud=1.

Una vez que se realiza el arranque del oscilador, la función ringmod continúa su operación. Los parámetros propiamente dichos del efecto serán ringain y mix. El primero controla la cantidad de efecto que se produce y el segundo la cantidad de señal original que se mezcla con el efecto. Así cuando llega el dato se genera un punto del oscilador y se multiplica éste por la lectura procedente del instrumento, siendo la salida finalmente salida=lectura*mix+ringain*(lectura*oscilador).

-Delay Base:

A partir de este efecto comienzan los algoritmos que usan la tabla trabajo. Su forma de ser usada ya se describió antes y no se repetirá. Los parámetros serán el tiempo de retardo del eco en segundos y la cantidad de eco que se oirá en tanto por uno referido al sonido original. La función calcula una vez cargados los parámetros el número de muestras que es necesario ir hacia atrás para recoger el eco y lo guarda en muestras según muestras= tiemporet*fmuest

Así, una vez que ha llegado el dato y se ha escalado, lo primero que se hace es guardarlo en la posición actual de la tabla(se lleva un contador i que va desde 0 a la longitud de trabajo, incrementándose cada vez que se guarda), comprobando que no estamos fuera en cuyo caso escribiríamos de nuevo al principio. A continuación se le resta a la posición donde nos encontramos de la tabla el número de muestras necesario para recoger el eco comprobando que no nos salimos por abajo en cuyo caso habría que continuar la resta desde el final de la tabla.

Si al recoger el eco estamos fuera de la tabla por abajo, se recalcula el índice de forma sencilla en j y finalmente salida= lectura+mix*trabajo[j]. Si no se está fuera de la tabla al calcular la resta , se hace directamente salida= lectura+mix*trabajo[i-rest].

Se debe hacer notar que el número de muestras a restar al índice i para recoger el eco puede ser un número no entero de forma que al efectuarse la resta el índice resultante no fuese adecuado para su uso como índice de un vector. Es por ello, que muestras se pasa al entero rest según rest=muestras.

-Delay Realimentado:

La forma de operación de esta función es muy semejante a la anterior, de modo que todo lo dicho es válido, salvo que ahora, llega el dato , se hacen todas las comprobaciones oportunas para guardar pero no se guarda el dato procedente del instrumento, sino lo que antes constituía la salida, esto es trabajo[i]=lectura+mix*trabajo[j] y salida=trabajo[i] en el caso de que al buscar el eco saliéramos de tabla por abajo(índice recalculado en variable j). O bien trabajo[i]=lectura+mix*trabajo[i-rest] y salida=trabajo[i] en caso de que la resta de la que hablábamos fuera factible para manejar vectores.

-Chorus:

Esta función es exactamente igual que delay base salvo porque el tiempo de retardo va variando según un oscilador senoidal de baja frecuencia implementado de la misma forma que en la función Ringmod. Así, se incluye la función arrancaseno que opera igual que arrancaring, pero con la forma que tienen los parámetros de control del oscilador para el efecto de chorus(ver sección de explicación de efectos). De esta forma la función de arranque necesitará el valor de barrido, la frecuencia de oscilación, el tiempo de delay base de la oscilación, pi y fmuest.

La función chorus por su parte, el único parámetro que contempla será la intensidad del coro en tanto por uno con respecto al sonido original. Debe comentarse que se incluyen una serie de variables auxiliares para realizar una interpolación lineal en el caso de que el oscilador proporcione tiempos de retardo no enteros(casi siempre) en cuyo caso se redondeará el no entero por encima y por debajo, se obtendrán los ecos correspondientes a ambos índices y se estimará el valor del eco si el tiempo hubiese sido no entero según la ecuación de una recta.

Así lo primero que se hace es arrancar el oscilador como ya se explicó, y cuando se tiene un dato procedente del conversor, se escala y se genera un punto del oscilador, que ahora es un tiempo de retraso. Este tiempo de retraso viene en segundos y se guarda en la variable oscilador, multiplicando dicha variable por la frecuencia de muestreo se

tiene por fin el número de muestras necesario para restar al contador de tabla actual. Lo que sucede es que este valor será normalmente no entero de ahí que se pase a entero en la variable rest y a la variable rest2 se le pasará el mismo numero +1, de forma que se apuntará a la muestra siguiente, quedando la teóricamente deseada entre ambas.

A continuación, teniendo en cuenta el manejo de tabla circular ya descrito, se guarda lo leído en la tala trabajo, se calculan los índices de las dos muestras a extraer en j y en g, con ellas dos se construye la ecuación de una recta tal y como se puede ver en el código, y con la parte decimal del tiempo del oscilador se hace una estimación del valor que debería obtenerse, que se guarda en estimación, por último salida=lectura+estimacion*mix.

Obsérvese que es idéntica a la función de delay base salvo porque el tiempo del eco va variando según oscilador y por el sencillo mecanismo de interpolación lineal, que requerirá ahora dos ecos en vez de uno.

-Flanger:

Esta función es idéntica a la anterior salvo en que lo que se guarda en la tabla es ahora igual a la salida, en vez del dato del conversor. Y en lugar de usar un oscilador senoidal para variar el tiempo de retraso, se usa un oscilador triangular, de forma que se generan dos rectas, una de pendiente positiva, si se esta en el primer semiperíodo y otra de pendiente negativa si se está en el segundo semiperiodo. Los parámetros del oscilador son la frecuencia y el barrido. En la función arrancatriangular se inicializan estos parámetros, se calcula el número de puntos que para ese período de oscilación y esa frecuencia de muestreo son necesarios como entero(ptos). Se calcula igualmente el semiperíodo en puntos que separa los dos tramos de recta(periodo) y las pendientes de ambos tramos. Se lleva un contador p para recorrer los tramos.

Al contrario de lo que ocurría antes, el mecanismo de interpolación se ha eliminado por presentar malos resultados. El único parámetro es también la fuerza de los ecos en relación al sonido original en tanto por uno.

Así la función lo primero que hace es inicializar el oscilador triangular, cuando llega un dato lo escala, genera un punto del oscilador en función de la situación del contador p con respecto al número de puntos teórico de cada tramo del oscilador. Se obtiene el número de muestras necesario para recoger el eco de la tabla y se convierte a entero. Se calcula el índice en j necesario para recoger el eco y se hace trabajo[i]=lectura+trabajo[j]*mix y salida=trabajo[i], lo que provoca la realimentación de ecos sucesivos.

-Reverberación realista:

Esta función es como cuatro delay básicos. Además se implementa un filtro paso bajo para lo cual hacen falta un par de tablas circulares más para guardar valores de algunos coeficientes del mismo, son buffilt y bufftap de cuatro elementos cada uno. Su manejo es como el de la tabla trabajo y en longbuff la longitud que es 4.

Los parámetros son las fuerzas del sonido original y de los cuatro ecos referidos en tanto por uno al sonido original. También son parámetros los tiempos de retraso en segundos.

Lo primero que hace la función es calcular el número de muestras correspondiente a cada uno de los tiempos de retraso introducidos. Estos tiempos se convierten en enteros en rest1,rest2,rest3,rest4.

Cuando llega un dato, se guarda en la tabla trabajo con el manejo ya conocido, se extraen los cuatro ecos y se guardan en muestras, muestras2,muestras3,muestras4. Se hace:

bufftap[ind]=muestras*mix1+muestras2*mix2+muestras3*mix3+muestras4*mix4

Esta es la entrada del filtro paso bajo. Se extrae de los buffers del filtro los coeficientes (ver teoría para ver la ecuación) y se hace

buffilt[ind]=0.4*s-0.2499*d+0.0441*t+0.5814*h+0.2142*k

donde s, d, t, h y k salen de los buffers. Por último salida=lectura*mix+buffilt[ind]. Nótese que ind es el contador de los buffers secundarios.

-**Phaser:**

Esta función es intermedia entre el chorus y el flanger. Se usa un oscilador senoidal como en chorus, pero implementando las ecuaciones de este efecto. La diferencia más sustancial es que esta función necesita dos tablas circulares de la misma longitud y grandes. Para ello, cuando sea necesario ejecutar este algoritmo deberá retocarse la longitud de la tabla trabajo y de la tabla de acuerdo con la memoria disponible y el tiempo de retardo deseado.

De esta forma, los parámetros del efecto son ganancia y mix. El primero es propio de la cadencia del filtro allpass y el segundo la clásica fuerza de los ecos sucesivos relativos al sonido original en tanto por uno. De esta manera, se arranca el oscilador con arrancaseno de la misma manera que en chorus, y cuando llega un dato se guarda en la tabla trabajo. A continuación se extrae un tiempo de retraso del oscilador y se calcula el número de muestras necesario correspondiente al tiempo de retraso para extraer el eco correspondiente y se pasa a entero en rest.

Si al extraer el eco nos salimos por abajo, recalculamos el índice en j y se hace tabla[i]=-ganancia*lectura+trabajo[j]+ganancia*tabla[j] y salida=mix*tabla[i]+lectura.

Si al extraer el eco estamos dentro de la tabla, se hace tabla[i]=-ganancia*lectura+trabajo[i-rest]+ganancia*tabla[i-rest]
salida=mix*tabla[i]+lectura.

Por último se desescala la salida y se envía al conversor.

d) Rutinas de inicialización.

Las rutinas o funciones de inicialización son todas aquellas funciones necesarias para inicializar los periféricos y el conversor para su funcionamiento. No se va a entrar en profundidad en la explicación de las mismas puesto que han sido extraídas de los ejemplos y documentación del microprocesador, por tratarse de funciones estándar de uso frecuente y no formar parte de los objetivos de desarrollo de este proyecto. Por tanto, se comentarán brevemente.

Void_set_input_gain(void)->fija la ganancia inicial adicional que da la placa.
Void_set_samplerate(void)->fija la frecuencia de muestreo del sistema.
Void_timer_lo_prior(void)->configura el timer para su correcto funcionamiento.

Configuración de DMA en transmisión y recepción:

Void_spt0_asserted(int sig_num)
Void_spr0_asserted(int sig_num)

Void setup_sports (void) –> Configuración del Puerto serie.
Void send_1847_config_cmds(void)->Configuración conversor.
Void init_21k(void)->Inicialización de timer, conversor e interrupciones.

e) Programa principal:
El programa principal responde a un esquema sencillo:

- Inicializa los registros del microprocesador.
- Resetea el conversor.
- Configura el puerto serie.
- Configura el conversor.
- Enciende el timer.

Todo ello usando las funciones del apartado anterior. Posteriormente, se encuentra el selector de efectos, que no es más que chequear el valor de la variable

efecto y según su valor ejecutar el efecto correspondiente indefinidamente y según la tabla:

efecto=0 -> Transparente (la guitarra suena sin efecto).
efecto=1 -> Distorsión1.
efecto=2 -> Distorsion2.
efecto=3 -> NoiseGate.
efecto=4 -> Compresor.
efecto=5 -> Expansor.
efecto=6 -> Ringmod
efecto=7 -> Delay Base
efecto=8 -> Delay Realimentado
efecto=9 -> Chorus
efecto=10 -> Flanger
efecto=11 -> Reverberación Realista
efecto=12 -> Phaser.

3.8.3.4. CÓDIGO DEL PROGRAMA.

Se presenta aquí el código del programa completo, para elegir el efecto deseado tan sólo hay que cambiar la variable efecto al valor del efecto deseado. Cada efecto está inicializado con unos valores que permiten la clara distinción del mismo. Tan sólo hay que ajustar los parámetros de los osciladores cuando se ejecute un efecto que lo necesite, y en caso de ejecutar un flanger deberán ajustarse las tablas circulares a 3900 elementos de longitud , así como la variable longitud.

```
/*******************************************************/
/*
Receives input from the AD1847 via the serial port and then transmits
the data back out the serial port for output on the AD1847.  Flag 2
(and the associated LED) blinks periodically. The input gain and
samplerate of the AD1847 can be altered while the program is running
by changing the ordered_rate and ordered_gain variables.
The following attributes of the AD1847 operation can be altered while
the program is running:  input gain (16 levels), and samplerate (14
frequencies).

Input Gain formula:  level * 1.5dB  (default level = 0 --> 0dB)

Samplerates (in kHz):
      ( 0)    8                     ( 1)    5.5125 (default)
      ( 2)    16                    ( 2)    11.025
      ( 4)    27.42857              ( 5)    18.9
      ( 6)    32                    ( 7)    12.05
      ( 8)    N/A                   ( 9)    37.8
      (10)    N/A                   (11)    44.1
      (12)    48                    (13)    33.075
      (14)    9.6                   (15)    6.615

Conversor configurado en modo 16 bits-> mayor valor de escala
                                        32767 por llevar signo.*/
```

```
/*********************************************************/
/* ADSP-2106x System Register bit definitions */

#include <def21060.h>
#include <21060.h>
#include <signal.h>
#include <sport.h>
#include <macros.h>
#include <math.h>

/*********************************************************/
                              /* DMA Chain pointer bit definitions */
#define CP_PCI 0x20000       /* Program-Controlled Interrupts bit*/
#define CP_MAF 0x1ffff       /* Valid memory address field bits   */

#define SetIOP(addr, val)  (* (int *) addr) = (val)
#define GetIOP(addr)       (* (int *) addr)

/*********************************************************/
#define SZ_regs_1847 16
int regs_1847[SZ_regs_1847] = {
      /* Note that the MCE bit is maintained throughout initial
         programming to hold off premature autocalibration. */
      0xc000,                 /* index 0 - left input control */
      0xc100,                 /* index 1 - right input control */
      0xc280,                 /* index 2 - left aux 1 input control */
      0xc380,                /* index 3 - right aux 1 input control */
      0xc480,                /* index 4 - left aux 2 input control */
      0xc580,                /* index 5 - right aux 2 input control */
      0xc600,                 /* index 6 - left dac control */
      0xc700,                 /* index 7 - right dac control */
      0xc85c,                 /* index 8 - data format */
      0xc909,                 /* index 9 - interface configuration */
      0xca00,                 /* index 10 - pin control */
      0xcb00,                 /* index 11 - no register */
      0xcc40,               /* index 12 - miscellaneous information */
      0xcd00,                 /* index 13 - digital mix control */
      0xce00,                 /* index 14 - no register */
      0x8f00};                /* index 15 - no register */
```

```
/* DMA chaining Transfer Control Blocks */
typedef struct {
   unsigned   lpath3;     /* for mesh mulitprocessing  */
   unsigned   lpath2;     /* for mesh multiprocessing  */
   unsigned   lpath1;     /* for mesh multiprocessing  */
   unsigned   db;         /* General purpose register  */
   unsigned   gp;         /* General purpose register  */
   unsigned** cp;         /* Chain Pointer to next TCB */
   unsigned   c;          /* Count register            */
   int        im;         /* Index modifier register   */
   unsigned * ii;         /* Index register            */
} _tcb;

_tcb rx_tcb = {0, 0, 0, 0, 0, 0, 3, 1, 0};      /* receive tcb */
_tcb tx_tcb = {0, 0, 0, 0, 0, 0, 3, 1, 0};      /* transmit tcb */

int cmd_blk[8];                                 /* command block */

static int xmit_count;
static int * xmit_ptr;

/* Variables procesado*/

int rx_buf[3];                            /* receive buffer */
int tx_buf[3] = {0xcc40, 0, 0};         /* transmit buffer-> igual */
int flag=0;
int efecto=10;

float fmuest=18900;              /*valor frecuencia de muestreo Hz*/
float escala=32767;              /*valor escalado a 1*/

float trabajo[7600]={0};           /*tabla circular de trabajo. */
float tabla[1]={0};    /*Tabla SOLO PARA PHASER. Igual a trabajo*/
int longitud=7600;           /*longitud tabla circular de trabajo*/

float lectura;
float salida;
volatile static int ordered_gain;
volatile static int ordered_rate=5;      /* ver tabla de frcuencia*/
```

```
/*osciladores */
/*variables parametros->las tieme la funcion de arranque*/

float pi=3.1415926;
float flfo=0.5;
float tdelay=0.001;
float barrido=0.009;
float amplitud;
float tlfo;
float ptosdec;
int ptos;
int periodo;
float m1;
float m2;
int p=0;

/*variables del filtro ->globales*/
float oscilador;
float frecuencia;
float z;
float v;
float w;

/**********RUTINAS EFECTOS*********************************/

/*O TALKTHROUGHT***************/

void transparente (void){
while(1){
if (flag==1){
flag=0;
salida=lectura;}
idle();
}}
/***********Distorsion1********/

void distorsion1(){

/*Parámetro*/
float saturacion=0.1;
```

```
while(1){

if (flag==1){
flag=0;
lectura=lectura/escala;

if(lectura>=saturacion){
salida=saturacion;}
else if (lectura<=-saturacion){
     salida=-saturacion;}
      else{salida=lectura;}

salida=salida*escala;}}

/*---------------Distorsion2--------------------------------------*/
void distorsion2(){

/*Parámetros*/
float saturacion=0.1;
float mix=0;
float nivel=1;

while(1){

if (flag==1){
flag=0;
lectura=lectura/escala;

if(lectura>=saturacion){
salida=mix*lectura+nivel*saturacion;}

else if (lectura<=-saturacion){
salida=mix*lectura+nivel*(-saturacion);}
    else{salida=lectura;}
salida=salida*escala;}
}}
```

```
/*--------------------------------NoiseGate-------------------------*/

void noisegate (){

/*Parámetro*/
float umbral=1.001;

while(1){

if (flag==1){
flag=0;
lectura=lectura/escala;

if((lectura<=umbral)&&(lectura>=-umbral)){
salida=0;}
else
{salida=lectura;}
salida=salida*escala;}
}}

/*--------------------------------Compresor-------------------------*/
void compresor (){

/*Parámetros*/
float umbral=0.1;
float ratio=0.1;

while(1){

if (flag==1){
flag=0;
lectura=lectura/escala;

if (lectura>umbral){
             salida=umbral+ratio*(lectura-umbral);}
             else if (lectura<-umbral){
              salida=-umbral+ratio*(lectura+umbral);}
              else{salida=lectura;}
salida=salida*escala;}
}}
```

```
/*--------------------------------Expansor--------------------------*/
void expansor (){

/*Parámetros*/
float umbral=0.1;
float ratio=0.9;

while(1){

if (flag==1){
flag=0;
lectura=lectura/escala;

if (lectura>-umbral && lectura<umbral){
             if (lectura>0){
                 salida=ratio*lectura;}
else{salida=lectura;}
salida=salida*escala;}}}



/*--------------------------------Ringmod---------------------------*/
void arrancaring(){

/*parametro->flfo. Necesarias pi,fmuest*/
amplitud=1;
frecuencia=(flfo/fmuest)*2*pi;

z=-2*cos(frecuencia);
v=0;
w=-amplitud*sin(frecuencia);
}

void ringmod (){

/*Parámetros*/
float ringain=1;
float mix=0;

/*arranca generador*/
arrancaring();
```

```
while(1){

if (flag==1){
flag=0;
lectura=lectura/escala;

/*obtener pto del oscilador*/
oscilador=-z*v-w;
w=v;
v=oscilador;

/*genera salida*/
salida=lectura*mix+ringain*(lectura*oscilador);
salida=salida*escala;}
}}

/*--------------------------------delay base.----------------------*/
void delaybase (){

int rest;
int i=0;
int j=0;
int a;
float muestras;

/*parámetros*/
float mix=1;
float tiemporet=0.3;

/*sin interpolar, se redondea el tiempo de retraso al exacto más
próximo*/

muestras=tiemporet*fmuest; /*numero de muestras atras*/
rest=muestras;            /* pasalo a entero*/

while(1){

if (flag==1){
flag=0;
lectura=lectura/escala;
```

```
if (i==longitud){
i=0;}
trabajo[i]=lectura;

/*manejo tabla circular.Obtención de índice del eco*/
if((i-rest)<0){
a=(i-rest);
j=(longitud+a);

/*generación salida si resta se sale por abajo*/
salida=lectura+trabajo[j]*mix;
salida=salida*escala;
i++;}

/*generación salida si resta normal*/
else{
salida=lectura+trabajo[i-rest]*mix;
salida=salida*escala;
i++;}}}}
/*----------------------delay realimentado.-------------------------*/
void delayrealm (){

/*parámetros*/
float mix=0.6;
float tiemporet=0.4;

float muestras;
int i=0;
int a,j;
int rest;

/*sin interpolar, se redondea al tiempo de retraso más próximo*/

muestras=tiemporet*fmuest; /*numero de muestras atras*/
rest=muestras;            /* pasalo a entero*/

while(1){

if (flag==1){
flag=0;
```

```
lectura=lectura/escala;

if (i==longitud){
i=0;}

 /*manejo tabla circular.Obtención de índice para eco.*/
if((i-rest)<0){
a=(i-rest);
j=(longitud+a);

/*Guarda en tabla y genera salida si nos salimos por abajo*/
trabajo[i]=lectura+trabajo[j]*mix;
salida=trabajo[i];
salida=salida*escala;
i++;}

/*Guarda en tabla y genera salida si no nos salimos por abajo*/
else{
trabajo[i]=lectura+trabajo[i-rest]*mix;
salida=trabajo[i];
salida=salida*escala;
i++;}}
}}

/*--------------------------------chorus.--------------------------*/
void arrancaseno(){

/* parámetros->barrido,flfo,tdelay*/
amplitud=barrido/2;
frecuencia=(flfo/fmuest)*2*pi;

z=-2*cos(frecuencia);
v=0;
w=-amplitud*sin(frecuencia);
}

void chorus (){

/*variables para interpolacion*/
float muestras;
float muestrasup;
```

```
float muestrainf;
float m;
float b;
float estimacion;
int rest;
int rest2;
/*resto*/
int i=0;
int j=0;
int g=0;
int a;

/*parámetros efecto->Parámetros oscilador en globales*/
float mix=0.4;

arrancaseno();

while(1){

if (flag==1){
flag=0;
lectura=lectura/escala;

/*Obten tiempo de retraso*/
oscilador=-z*v-w;
w=v;
v=oscilador;
oscilador=oscilador+tdelay+amplitud;

/* indices de muestras a interpolar*/
muestras=oscilador*fmuest;
rest=muestras;    /* este indice nos da la muestra más cerca*/
muestras=muestras+1;
rest2=muestras;   /*este indice nos da la muestra más lejos*/

/*Guarda en tabla*/
if (i==longitud){
i=0;}
trabajo[i]=lectura;
```

```
/*manejo tabla circular con dos indices para los ecos*/
/*j y g contienen los indices*/

if((i-rest)<0){
a=(i-rest);
j=(longitud+a);}
else{j=(i-rest);}

if((i-rest2)<0){
a=(i-rest2);
g=(longitud+a);}
else{g=(i-rest2);}

/*realize interpolación*/
     muestrasup=trabajo[j];
     muestrainf=trabajo[g];
     m=muestrasup-muestrainf;
     b=(rest2-rest);
     estimacion=muestrainf+m*b;

/*Genera salida*/
     salida=lectura+estimacion*mix;
     salida=salida*escala;
i++;}}}

/*--------------------------------flanger.--------------------------*/
/* Se implementa con oscilador triangular por mejor efecto*/
/*Se eliminó el interpolador porque causaba ruido*/

void arrancatriangular(){

tlfo=1/flfo;
ptosdec=tlfo*fmuest;
ptos=ptosdec;
periodo=ptos/2;
m1=(barrido/periodo);
m2=(-barrido/periodo);
}
```

```
void flanger (){

/*variables índices*/
float muestras;
int rest;

/*resto*/
int i=0;
int j=0;
int g=0;
int a;

/*parámetros efecto->Parámetros oscilador en globales*/
float mix=0.5;

/*oscilador triangular*/
arrancatriangular();

while(1){

if (flag==1){
flag=0;
lectura=lectura/escala;

/*Genera tiempo de retraso oscilador triangular*/
if(p<periodo){
oscilador=m1*p;}
else{
oscilador=barrido+m2*(p-periodo);}
oscilador=oscilador+tdelay;
p++;
if(p>ptos){p=0;}

/* indices de muestras a interpolar*/
muestras=oscilador*fmuest;
rest=muestras;    /* este indice nos da la muestra más cerca*/

/*genera índices de eco en j*/
if (i==longitud){
i=0;}
```

```
if((i-rest)<0){
a=(i-rest);
j=(longitud+a);}
else{j=(i-rest);}

/*Guarda en tabla y genera salida*/
     trabajo[i]=lectura+trabajo[j]*mix;
     salida=trabajo[i];
     salida=salida*escala;
i++;}}}

/*----------------------------Reverberación realista----------------*/

void reverbreal(){

/*variables auxiliaries*/
float buffilt[4]={0};
float bufftap[4]={0};
int   a,j,g,e,f;
float s,d,t,h,k;

float muestras;
float muestras2;
float muestras3;
float muestras4;

int  longbuff=4;
int i=0;
int ind=0;                /*Indice tablas buffer*/
int rest1;
int rest2;
int rest3;
int rest4;

/*Parámetros*/

float mix=1;
float mix1=0.7;
float mix2=0.5;
float mix3=0.4;
float mix4=0.3;
```

```
float tiemporet1=0.1;
float tiemporet2=0.2;
float tiemporet3=0.3;
float tiemporet4=0.4;

/*cálculo de muestras hacia atrás*/

muestras=tiemporet1*fmuest;
rest1=muestras;

muestras2=tiemporet2*fmuest;
rest2=muestras2;

muestras3=tiemporet3*fmuest;
rest3=muestras3;

muestras4=tiemporet4*fmuest;
rest4=muestras4;

while(1){

if (flag==1){
flag=0;
lectura=lectura/escala;

/*Guarda en tabla */
if(i==longitud){
i=0;}
trabajo[i]=lectura;

if(ind==longbuff){
ind=0;}

/*generación de índices de los ecos y obtención de los mismos */
if(i-rest1<0){
a=i-rest1;
j=longitud+a;
muestras=trabajo[j];}
else{muestras=trabajo[i-rest1];}
```

```
if(i-rest2<0){
a=i-rest2;
g=longitud+a;
muestras2=trabajo[g];}
else{muestras2=trabajo[i-rest2];}

if(i-rest3<0){
a=i-rest3;
e=longitud+a;
muestras3=trabajo[e];}
else{muestras3=trabajo[i-rest3];}

if(i-rest4<0){
a=i-rest4;
f=longitud+a;
muestras4=trabajo[f];}
else{muestras4=trabajo[i-rest4];}

/*Guarda en buffer de ecos*/
bufftap[ind]=muestras*mix1+muestras2*mix2+muestras3*mix3+muestras4
*mix4;

/*Obtención de datos en buffers secundarios */
if((ind-1)<0){
s=buffilt[3];
h=bufftap[3];}
else{
s=buffilt[ind-1];
h=bufftap[ind-1];}

if((ind-2)<0){
a=ind-2;
a=longbuff+a;
d=buffilt[a];
k=bufftap[a];}
else{
d=buffilt[ind-2];
k=bufftap[ind-2];}

if((ind-3)<0){
```

```
a=ind-3;
a=longbuff+a;
t=buffilt[a];}
else{t=buffilt[ind-3];}

/*Guarda en buffer del filtro y genera salida*/
buffilt[ind]=0.4*s-0.2499*d+0.0441*t+0.5814*h+0.2142*k;
salida=lectura*mix+buffilt[ind];
salida=salida*escala;
i++;
ind++;}}
}

/*--------------------------------Phaser--------------------------*/
/*OJO QUE REQUIERE DOS TABLAS CIRCULARES DE IGUAL DIMENSIÓN->RETOCAR
TRABAJO CUANDO ESTE EFECTO SE EJECUTE. SI NO, NO CABE EN MEMORIA*/
/* Con OSCILADOR*/

void phaser (){

/*resto*/
int rest;
float muestras;
int i=0;
int j=0;
int g=0;
int interpola=0;
int a;

/*Parámetros*/
float ganancia=0.5;
float mix=0.5;

/*Arranca el oscilador*/
arrancaseno();

while(1){

if (flag==1){
flag=0;
lectura=lectura/escala;
```

```
/*Guarda el dato en la tabla*/
if (i==longitud){
i=0;}
trabajo[i]=lectura;

/* Genera un retraso desde el oscilador*/
oscilador=-z*v-w;
w=v;
v=oscilador;
oscilador=oscilador+tdelay+amplitud;

/* indices de muestras a interpolar*/
muestras=oscilador*fmuest;
rest=muestras;

/*Índice del eco */
if((i-rest)<0){
a=(i-rest);
j=(longitud+a);

/*Guarda en tabla y genera salida si nos salimos por abajo*/
tabla[i]=-ganancia*lectura+trabajo[j]+ganancia*tabla[j];
salida=mix*tabla[i]+lectura;
salida=salida*escala;
i++;}

/*Guarda en tabla y genera salida si no nos salimos por abajo*/
else{tabla[i]=-ganancia*lectura+trabajo[i-rest]+ganancia*tabla[i-
rest];
salida=mix*tabla[i]+lectura;
salida=salida*escala;
i++;}}}}
```

```
/**********RUTINAS INICIALIZACIÓN******************/
// Variables which the user may set to control the Codec.

void set_input_gain( void )
{
    // If still transmitting commands, wait until done.
    if( xmit_count != 0 )
        return;

    // Limit input gain to valid values.
    ordered_gain &= 0x0f;

    // Put new input gain into 1847 register commands.
    regs_1847[0] = (regs_1847[0] & ~0x0f) | ordered_gain;
    regs_1847[1] = (regs_1847[1] & ~0x0f) | ordered_gain;

    // Get 1847 commands to set input gain and terminating
    // command into command block for transmission.
    cmd_blk[0] = regs_1847[0];
    cmd_blk[1] = regs_1847[1];
    cmd_blk[2] = regs_1847[15];

    // Set up pointer and counter to transmit commands.
    xmit_ptr = cmd_blk;
    xmit_count = 3;

    // Update current value to commanded value.
    current_gain = ordered_gain;

    return;
}
```

```
/*********************************************************/
/* Fija el samplerate*/
void set_samplerate( void )
{
    // If still transmitting commands, wait until done.
    if( xmit_count != 0 )
        return;

    // Limit sample rate to valid values.
    ordered_rate &= 0x0f;
    if( ordered_rate == 8 )
        ordered_rate = 9;
    if( ordered_rate == 10 )
        ordered_rate = 11;

    // Put new sample rate into 1847 register commands.
    regs_1847[8] = (regs_1847[8] & ~0x0f) | ordered_rate;

    // Get 1847 command to set sample rate and terminating
    // command into command block for transmission.
    cmd_blk[0] = regs_1847[8];
    cmd_blk[1] = regs_1847[15];

    // Set up pointer and counter to transmit commands.
    xmit_ptr = cmd_blk;

    xmit_count = 2;

    // Update current value to commanded value.
    current_rate = ordered_rate;

    return;
}

/*********************************************************/
/* Periodic timer interrupt                              */
/*********************************************************/
void timer_lo_prior( int sig_num )
{
    sig_num=sig_num;
```

```
    // Toggle flag 2 LED.
    set_flag(SET_FLAG2, TGL_FLAG);
}

/*********************************************************/
/* Serial port transmit DMA complete                     */
/*********************************************************/
void spt0_asserted( int sig_num )
{
    // Check if there are more commands left to transmit.
    if( xmit_count )
    {
        // If so, put the comand into the transmit buffer and update
count.
        tx_buf[0] = *xmit_ptr++;
        xmit_count--;
    }
}

/*********************************************************/
/* Serial port receive DMA complete                      */
/*********************************************************/
void spr0_asserted( int sig_num )
{
    // Copy received data buffers to transmit data buffers.
    lectura = rx_buf[1];            /*lee solo un canal*/
    tx_buf[1]= salida;
    tx_buf[2]= salida;
    flag=1;
}
/*********************************************************/
/*Configuración serial port*/
void setup_sports ( void )
{
    /* Configure SHARC serial port SPORT0 */

    /* Multichannel communications setup */
    sport0_iop.mtcs  = 0x00070007;/*transmit on words0,1,2,16,17,18*/
    sport0_iop.mrcs  = 0x00070007;/*receive on words 0,1,2,16,17,18*/
    sport0_iop.mtccs = 0x00000000;/*no companding on transmit*/
```

```
    sport0_iop.mrccs = 0x00000000; /* no companding on receive */

/* TRANSMIT CONTROL REGISTER */
/* STCTL0 <= 0x001c00f2      */
/*An alternate(and more efficient) way of doing this would be to*/
/*write the 32-bit register all at once with a statement like this: */
/*        SetIOP(STCTL0, 0x001c00f2);                              */
/* But the following is more descriptive...                       */

sport0_iop.txc.mdf   = 1;  /* multichannel frame delay (MFD)       */
sport0_iop.txc.schen = 1;  /* Tx DMA chaining enable               */
sport0_iop.txc.sden  = 1;  /* Tx DMA enable                        */
sport0_iop.txc.lafs  = 0;  /* Late TFS (alternate)                 */
sport0_iop.txc.ltfs  = 0;  /* Active low TFS                       */
sport0_iop.txc.ditfs = 0;  /* Data independent TFS                 */
sport0_iop.txc.itfs  = 0;  /* Internally generated TFS             */
sport0_iop.txc.tfsr  = 0;  /* TFS Required                         */

sport0_iop.txc.ckre  = 0;  /* Data and FS on clock rising edge     */
sport0_iop.txc.gclk  = 0;  /* Enable clock only during transmission*/
sport0_iop.txc.iclk  = 0;  /* Internally generated Tx clock        */
sport0_iop.txc.pack  = 0;  /* Unpack 32b words into two 16b tx's   */

sport0_iop.txc.slen  = 15;  /* Data word length minus one          */
sport0_iop.txc.sendn = 0;   /* Data word endian 1 = LSB first      */
sport0_iop.txc.dtype = SPORT_DTYPE_RIGHT_JUSTIFY_SIGN_EXTEND;
                       /* Data type specifier                  */
sport0_iop.txc.spen  = 0;   /* Enable (clear for MC operation)     */

    /* RECEIVE CONTROL REGISTER */
    /* SRCTL0 <= 0x1f8c20f2     */
sport0_iop.rxc.nch   = 31;  /* multichannel number of channels - 1  */
sport0_iop.rxc.mce   = 1;   /* multichannel enable                  */
sport0_iop.rxc.spl   = 0;   /* Loop back configure (test)           */
sport0_iop.rxc.d2dma = 0;   /* Enable 2-dimensional DMA array       */
sport0_iop.rxc.schen = 1;   /* Rx DMA chaining enable               */
sport0_iop.rxc.sden  = 1;   /* Rx DMA enable                        */
sport0_iop.rxc.lafs  = 0;   /* Late RFS (alternate)                 */
sport0_iop.rxc.ltfs  = 0;   /* Active low RFS                       */
sport0_iop.rxc.irfs  = 0;   /* Internally generated RFS             */
sport0_iop.rxc.rfsr  = 1;   /* RFS Required                         */
```

```
sport0_iop.rxc.ckre  = 0;   /* Data and FS on clock rising edge     */
sport0_iop.rxc.gclk  = 0;   /* Enable clock only during transmission*/
sport0_iop.rxc.iclk  = 0;   /* Internally generated Rx clock        */
sport0_iop.rxc.pack  = 0;   /* Pack two 16b rx's into 32b word      */

sport0_iop.rxc.slen  = 15;  /* Data word length minus one           */
sport0_iop.rxc.sendn = 0;   /* Data word endian 1 = LSB first       */
sport0_iop.rxc.dtype = SPORT_DTYPE_RIGHT_JUSTIFY_SIGN_EXTEND;
                       /* Data type specifier                  */
sport0_iop.rxc.spen = 0;    /* Enable (clear for MC operation)      */

    /* Enable sport0 xmit & rcv irqs (DMA enabled) */
    interrupt(SIG_SPR0I, spr0_asserted);
    interrupt(SIG_SPT0I, spt0_asserted);

    /* Set up Transmit Transfer Control Block for chained DMA */
tx_tcb.ii = tx_buf;     /* DMA source buffer address                */
tx_tcb.cp = &tx_tcb.ii; /* define ptr to next TCB (point to self)   */
SetIOP(CP2, (((int)&tx_tcb.ii) & CP_MAF) | CP_PCI);
                       /* define ptr to current TCB (kick off DMA) */
                       /* (SPORT0 transmit uses DMA ch 2)          */

    /* Set up Receive Transfer Control Block for chained DMA */
rx_tcb.ii = rx_buf;    /* DMA destination buffer address           */
rx_tcb.cp = &rx_tcb.ii; /* define ptr to next TCB (point to self)   */
SetIOP(CP0, (((int)&rx_tcb.ii) & CP_MAF) | CP_PCI);
                       /* define ptr to current TCB (kick off DMA) */
                       /* (SPORT0 receive uses DMA ch 0)           */

}
/*********************************************************/
/*Configuracion codec*/
void send_1847_config_cmds( void )
{
    // Set up pointer and counter to transmit commands.
    xmit_ptr   = regs_1847;
    xmit_count = SZ_regs_1847;

    // Wait for all commands to be transmitted.
    while( xmit_count )
        idle();
```

```
    // Wait for AD1847 autocal to start.
    while( !(rx_buf[0] & 0x0002) )
        idle();

    // Wait for AD1847 autocal to finish.
    while( rx_buf[0] & 0x0002 )
        idle();

    return;
}

/*********************************************************/
/*Inicializaiones*/
void init_21k( void )
{
    // Disable timer and set rate to 4 Hz.
    timer_off();
    timer_set( 10000000, 10000000 );

    // Initialize pointer and counter to transmit commands.
    xmit_count = 0;
    xmit_ptr   = regs_1847;

    // Enable interrupt nesting.
    asm( "#include <def21060.h>" );
    asm( "bit set mode1 NESTM;"  );

    // Enable timer (low priority) interrupt.
    interrupt( SIG_TMZ, timer_lo_prior );

    // Turn flag LEDs off.
    set_flag( SET_FLAG2, SET_FLAG );

    return;
}
```

```
/*********************************************************/
/* PROGRAMA PRINCIPAL                                    */
/*********************************************************/
void main ( void )
{
/*Inicialización general*/
    int x;

    // Initialize some SHARC registers.
    init_21k();

    // Reset the Codec.
    set_flag( SET_FLAG0, CLR_FLAG );    /* Put CODEC into RESET  */
    for( x=0 ; x<0xffff ; x++ )         /* Hold CODEC in RESET */
        ;
    set_flag( SET_FLAG0, SET_FLAG );    /* Release CODEC from RESET */

    // Configure SHARC serial port.
    setup_sports();

    // Send setup commands to CODEC.
    send_1847_config_cmds();

    // Turn on all LEDs.
    set_flag(SET_FLAG2, CLR_FLAG);

    // Turn on the timer.
    timer_on();

/* SELECTOR DE EFECTOS-> ejecución continua*/

if (efecto==0){
transparente();}

if (efecto==1){
distorsion1();}

if (efecto==2){
distorsion2();}
```

```
if (efecto==3){
noisegate();}

if (efecto==4){
compresor();}

if (efecto==5){
expansor();}

if (efecto==6){
ringmod();}

if (efecto==7){
delaybase();}

if (efecto==8){
delayrealm();}

if (efecto==9){
chorus();}

if (efecto==10){
flanger();}

if (efecto==11){
reverbreal();}

if (efecto==12){
phaser();}}

/*************************************************************/
// FIN PROGRAMA.
/*************************************************************/
```

3.8.4 .MONTAJE. CONCLUSIONES SOBRE LAS PRUEBAS Y OTRAS NOTAS.

El Montaje realizado para llevar a cabo las pruebas de simulación finales está comprendido de los siguientes elementos:

a) Guitarra acústica washburn EA-10.

b) Altavoces Tec sound modelo 707.

c) Placa de prueba SHARC EZ-KIT LITE

d) Fuente de alimentación no regulada 10 V, 1 A.

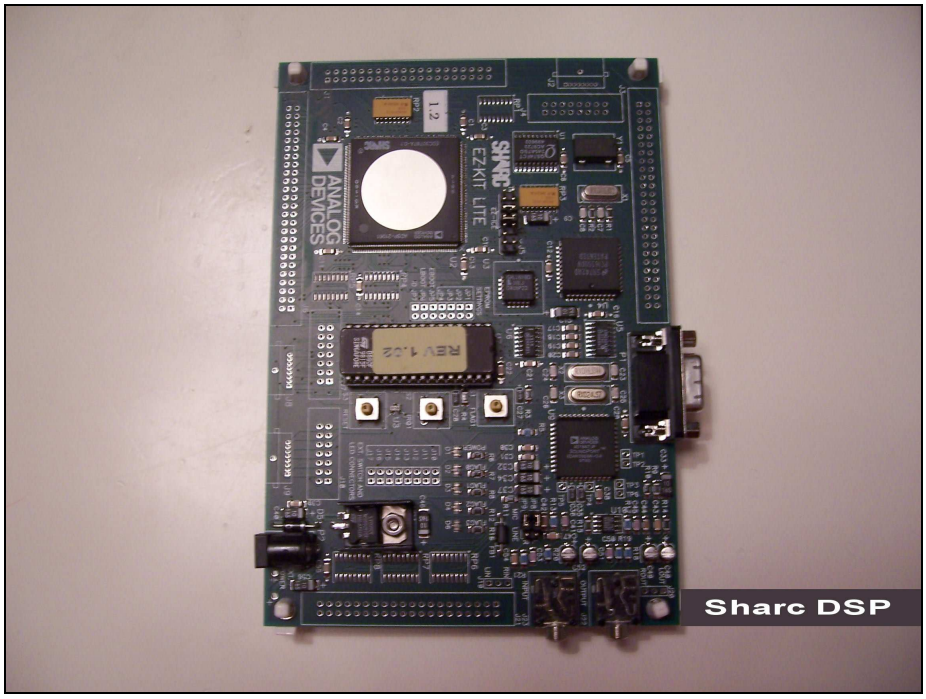
e) PC y software adecuado para el manejo de la tarjeta vía RS232.

A continuación se muestran algunas fotografías de los elementos y el montaje final completo.

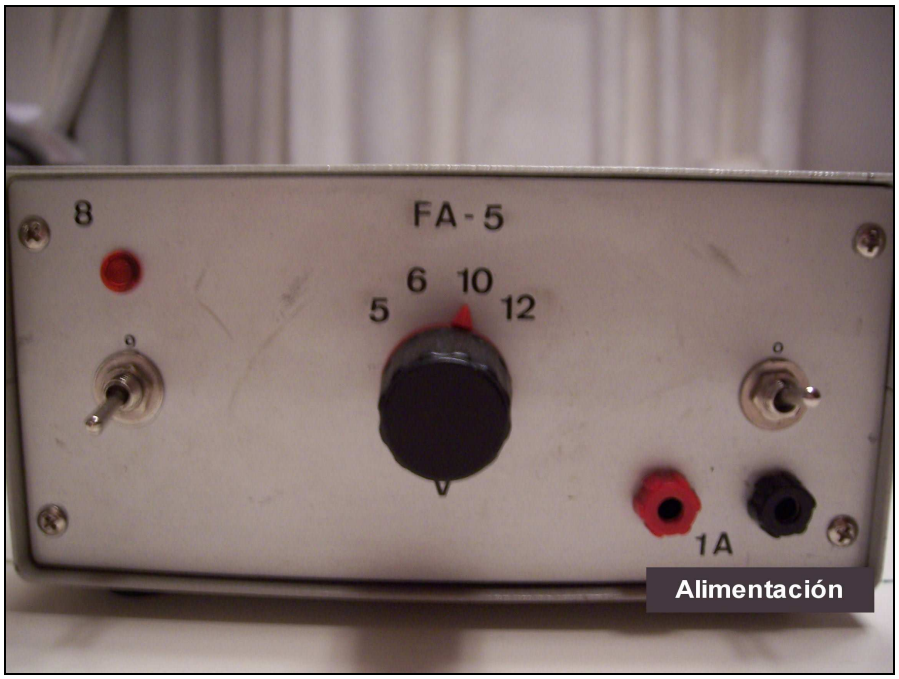
a)

b)

c)



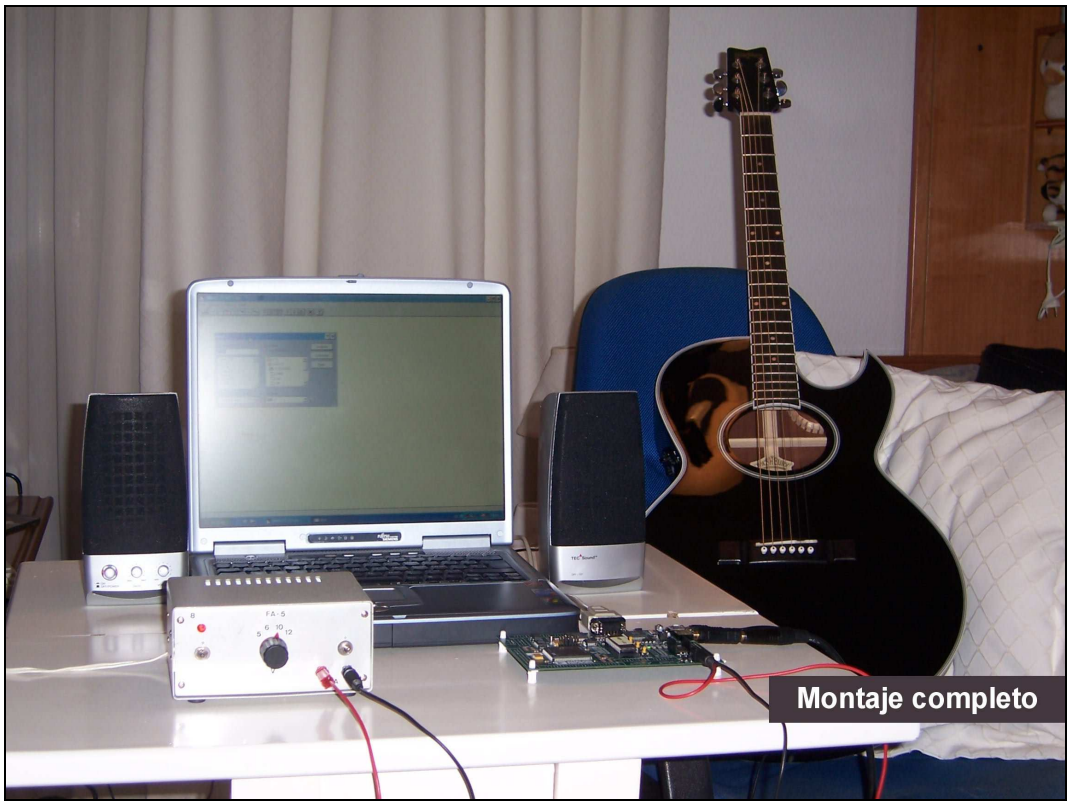
d)


8
FA-5
5
6
10
12
1A
Alimentación

e)


Windows
Haga clic en su nombre para comenzar
Aurora Perez Rodrig
Apagar equipo
PC y software

Por último , una foto del montaje completo:



Las pruebas realizadas sobre la tarjeta han sido muy satisfactorias obteniéndose los resultados esperados en todos los algoritmos, destacamos aquí no obstante algunas particularidades de interés. Se irán describiendo efecto por efecto:

Distorsion1-> Efecto esperado. Distorsión para solos de eléctrica.

Distorsión2-> Buen efecto cuando se utiliza igual que el anterior. El uso de los dos parámetros de control disponibles de forma simultánea no da buen resultado.

NoiseGate-> El algoritmo funciona correctamente, si bien no tiene mucho interés como efecto, sino más bien como herramienta de edición.

Compresor-> Buen efecto, parecido a distorsión1 pero más compacto, que es precisamente lo que se espera de la compresión.

Expansor-> El algoritmo funciona correctamente, si bien no tiene mucho interés como efecto, sino más bien como herramienta de edición tal y como sucedía con Noise Gate.

RingModulator-> Efecto particularmente llamativo por las asonancias metálicas que genera. Funcionamiento correcto.

Delay-> Efecto muy llamativo y muy usado. Funcionamiento correcto.

Delay realimentado-> Modificación del anterior efecto, con respuesta al impulso atenuada de forma progresiva, consiguiéndose un realzado del sonido aún más llamativo. Funcionamiento correcto.

Chorus->Efecto de coro con modulación del instrumento en coro ciertamente conseguido . Funciona correctamente.

Flanger-> Efecto muy espectacular exacto al buscado en teoría. Importante señalar aquí que en el código definitivo se eliminó el oscilador senoidal y se dejó únicamente el triangular, pues es el que realmente consigue el efecto deseado. Así mismo, se eliminó el interpolador lineal, pues éste estaba generando ruido y en vez de mejorar el resultado, lo empeoraba, de forma que los tiempos de retraso no múltiplos del período de muestreo simplemente se redondean al tiempo de muestreo múltiplo del período de muestreo más cercano.

Reberb-> Efecto de eco parecido al de delay realimentado. No obstante una escucha detenida del mismo nos muestra sus particularidades sobre aquél. De este modo, el efecto de reverberación realista implementado presenta un sonido más natural de lo que sería el eco dentro de una habitación. Así mismo se hace notar la atenuación de las altas frecuencias que reverberan menos que las notas graves, todo lo cual dota al efecto de mayor realismo, que es justo lo que se buscaba.

Phaser-> Este efecto está a medio camino entre el chorus y el flanger, de modo que se observan características de ambos. Se puede distinguir la oscilación propia del flanger y el sonido brillante del chorus. Si se quita el oscilador LFO el efecto es muy

leve y sin oscilación, siendo no obstante agradable y adecuado para solos de precisión en pasajes tranquilos. Funciona correctamente.

En general todos los algoritmos han funcionado según lo esperado, por lo que la implementación puede considerarse un éxito.

3.8.5. FUTUROS DESARROLLOS.

Una vez detallados los contenidos esenciales y fundamentales del proyecto, en este apartado se comentan posibles desarrollos futuros o ampliaciones posibles del mismo.

Una primera ampliación evidente sería el desarrollo de nuevos efectos o la modificación de algunos ya existentes que podrían admitir una mayor complejidad con el objetivo de lograr sonidos aún más conseguidos, como por ejemplo añadir un oscilador al efecto de phaser o bien implementar condiciones acústicas concretas de una sala en el efecto de reverberación, con el fin de reproducir el sonido en el interior de la misma, incluyendo las características especiales de ella, tales como las atenuaciones o amplificaciones de ciertos materiales, dimensiones y geometrías específicas, etc.

Un segundo desarrollo futuro importante sería el diseño de un sistema autónomo completo, esto es añadir una interfaz con el usuario de manera que éste pueda introducir el efecto que desea y los parámetros correspondientes al mismo en tiempo real. Así mismo se podría incluir aquí el diseño hardware de una nueva placa específica para la aplicación de este proyecto, optimizando los recursos de la misma, eliminando todos los componentes que no se usaran e incluyendo una memoria interna de mayor capacidad para poder contar con retardos más largos. Así mismo se podría incluir cualquier otro complemento pensando en futuras y sucesivas ampliaciones.

ANEXO A. CÓDIGO MATLAB.

Se incluye a continuación el código de las simulaciones preliminares para la obtención de resultados gráficos en Matlab. Se tiene un programa principal en el archivo principal.m y cada función.m es un efecto.

```
%-------------------------------------------------------------------------------------------
%PROGRAMA PRINCIPAL. Simula efectos para guitarra usando un tono senoidal
%perfecto. Los efectos se seleccionaran con un número de orden.
%Cada efecto se encuentra en una funcion.m. Ver descripcion en ayuda
%-------------------------------------------------------------------------------------------
% Datos de simulacion:
% PARAMETRO frecuencia señal original Hz.
% PARAMETRO timepo simulacion en seg.
% PARAMETRO numero de ptos por periodo
% PARAMETRO amplitud de la senoidal. escalada arango entre 0-1
% PARAMETRO amplitud de la senoidal. escalada a rango entre 0-1
clear
disp('Parametros simulacion:')
f=input('Teclee valor en Hz señal instrumento:');
A=input ('Amplitud de la señal:');
tsim=input('Teclee el tiempo total de simulacion(seg):');
n=input('Teclee el numero de ptos  por periodo:');
disp('seleccione efecto  1->distorsion1,2->distorsion2,3->noisegate,4->compresor,5-
>expansor')
efecto=input('6->ringmod,7->delay basico,8->delay realimentado,9->Chorus,10-
>Flanger:,11->Reverb:,12->Phaser:');

fs=n*f;                % frecuencia de muestreo
Ts=1/fs;
t=(0:Ts:tsim);         % vector tiempo.

% senoidal del instrumento musical muestreada a fs.
% Vector x señal original,
% Vector y,trabajo.
```

```
x=A*sin(2*pi*f*t);
D=size(x);

%selector efecto----------------------------------------------------------------------------:

if efecto==1;
distorsion1_func(x,t,D);

else if efecto==2;
distorsion2_func(x,t,D);

else if efecto==3;
noisegate_func(x,t,D);

else if efecto==4;
compresor_func(x,t,D);

else if efecto==5;
expansor_func(x,t,D);

else if efecto==6;
ringmod_func(x,t,D);

else if efecto==7;
%nueva generacion onda:
taux=zeros(1,D(2));
taux(1,1:n)=t(1,1:n);
x=sin(2*pi*f*taux);

delaybasico_func(x,t,D,Ts)

else if efecto==8;
%nueva generacion onda:
```

```
taux=zeros(1,D(2));
taux(1,1:n)=t(1,1:n);
x=sin(2*pi*f*taux);
delayreal_func(x,t,D,Ts);

else if efecto==9;
%nueva generacion onda:
taux=zeros(1,D(2));
taux(1,1:n)=t(1,1:n);
x=sin(2*pi*f*taux);
chorus_func(x,t,D,Ts);

else if efecto==10;
%nueva generacion onda:
taux=zeros(1,D(2));
taux(1,1:n)=t(1,1:n);
x=sin(2*pi*f*taux);
flanger_func(x,t,D,Ts);

else if efecto==11;
%nueva generacion onda:
taux=zeros(1,D(2));
taux(1,1:n)=t(1,1:n);
x=sin(2*pi*f*taux);
reverb_func(x,t,D,Ts);

else if efecto==12;
%nueva generacion onda:
taux=zeros(1,D(2));
taux(1,1:n)=t(1,1:n);
x=sin(2*pi*f*taux);
phaser_func(x,t,D,Ts);

else disp('Efecto no existente')
```

```
end
end
end
end
end
end
end
end
end
end
end
end
%Fin del programa--------------------------------------------------------------------------
function distorsion1_func(x,t,D)
%-------------------------------------------------------------------------------------------------
% DISTORSION BASICO: señal>sat salida=sat; señal<-sat salida=-sat ;recortador
% Argumentos funcion: vector x, vector y,dimension. Salidas:none
%-------------------------------------------------------------------------------------------------
% PARAMETRO->nivel de saturacion sat., suponiendo la señal escalada a 1
y=x;
disp=('algoritmo de DISTORSION seleccionado')
sat=input('Teclee nivel de limitacion:');

for I=1:D(2);
        if x(I)>sat;
           y(I)=sat;
        end;
         if x(I)<-sat;
            y(I)=-sat;
         end;

    end;
```

```
%Presentacion de resultados:
subplot(2,1,1);
plot(t,x);
title('Señal del instrumento');
subplot(2,1,2);
plot(t,y);
title('Señal tratada');
%-------------------------------------------------------------------------------------------
function distorsion2_func(x,t,D)
%-------------------------------------------------------------------------------------------
% DISTORSION2: señal>sat salida=a*entrada+b*sat; señal<-sat igual
% Argumentos funcion: vector x, vector y,dimension. Salidas:none
%-------------------------------------------------------------------------------------------
% PARAMETRO->nivel de saturacion sat., suponiendo la señal escalada a 1
% PARAMETRO-> cantidad de señal original a mezcla
% PARAMETRO->cantidad de saturacion a mezcla

y=x;
disp=('algoritmo de DISTORSION2 seleccionado')
a=input('cantidad señal original:');
b=input('cantidad saturacion:');
sat=input('Teclee nivel de limitacion:');

for I=1:D(2);
        if x(I)>sat;
           y(I)= a*x(I)+b*sat;
        end;
         if x(I)<-sat;
            y(I)=a*x(I)+b*(-sat);
         end;
end;

%Presentacion de resultados:
subplot(2,1,1);
```

```
plot(t,x);
title('Señal del instrumento');
subplot(2,1,2);
plot(t,y);
title('Señal tratada');

%-------------------------------------------------------------------------------------------------------
function noisegate_func (x,t,D)
%-------------------------------------------------------------------------------------------------------
% NOISE GATE: umbral<señal<umbral salida=0
% Argumentos funcion: vector x, vector y,dimension. Salidas:none
%-------------------------------------------------------------------------------------------------------
% PARAMETRO->Umbral de la puerta
y=x;
disp=('algoritmo de NOISE GATE seleccionado')
umbral=input('Teclee umbral:');

for I=1:D(2);
        if x(I)<umbral;
          if x(I)>-umbral
             y(I)=0;
          end
        end
    end
%Presentacion de resultados:
subplot(2,1,1);
plot(t,x);
title('Señal del instrumento');
subplot(2,1,2);
plot(t,y);
title('Señal tratada');
```

```
%-------------------------------------------------------------------------------------------------------
function compresor_func(x,t,D)
% COMPRESOR: señal>umbral; y(n)=umbral+m*(x(n)-umbral) idem señal<-umbral
% Argumentos funcion: vector x, vector y,dimension. Salidas:none
%-------------------------------------------------------------------------------------------------------
% PARAMETRO->nivel de umbral.
% PARAMETRO->ratio de la nueva recta. Debe ser menor que1
% Si fuera mayor que uno el efecto es una pseudo expansion.
y=x;
disp=('algoritmo de COMPRESION seleccionado')
umbral=input('Teclee umbral:');
ratio=input('ratio de compresion:');

for I=1:D(2);
        if x(I)>umbral;
           y(I)=umbral+ratio*(x(I)-umbral);
       else if x(I)<-umbral
            y(I)=-umbral+ratio*(x(I)+umbral);
           end
        end
    end

%Presentacion de resultados:
subplot(2,1,1);
plot(t,x);
title('Señal del instrumento');
subplot(2,1,2);
plot(t,y);
title('Señal tratada');
%funcion de transferencia:
figure
plot(x,y);
title('funcion de transferencia del compresor');
```

```
%-------------------------------------------------------------------------------------------------------
function expansor_func(x,t,D)
% EXPANSOR: if (x(I)>-umbral & x(I)<umbral);cambia pendiente
% Argumentos funcion: vector x, vector y,dimension. Salidas:none
%-------------------------------------------------------------------------------------------------------
% PARAMETRO->nivel de umbral.
% PARAMETRO->ratio de la nueva recta. Debe ser menor que 1
% no puede ser mayor que 1
y=x;
disp=('algoritmo de EXPANSION seleccionado')
umbral=input('Teclee umbral:');
ratio=input('ratio de expansion:');

%correccion ratio erroneo
if ratio>1|ratio<0
   ratio=1
   disp=('correccion automatica del ratio')
end

%algoritmo:
for I=1:D(2);
        if (x(I)>-umbral & x(I)<umbral);
          if x(I)>0
             y(I)=ratio*x(I);
          else
             y(I)=ratio*x(I);

          end
        end
     end

%Presentacion de resultados:
%formas de onda:
subplot(2,1,1);
```

```
plot(t,x);
title('Señal del instrumento');
subplot(2,1,2);
plot(t,y);
title('Señal tratada');
%funcion de transferencia:
figure
plot(x,y);
title('funcion de transferencia del expansor');
%-------------------------------------------------------------------------------------------------
function ringmod_func (x,t,D)
%RINGMOD: y(n)=x(n)*u(n) u(n)-> Modulador
%argumentos func: x,y,dimension,t.
%-------------------------------------------------------------------------------------------------
% señal del modulador.
% PARAMETRO: frecuencia del ring.
% PARAMETRO: ganancia del ring.
% PARAMETRO: ganancia señal.

y=x;
disp('RINGMODULATOR seleccionado')
fmod=input('Teclea la frecuencia del modulador:');
Ringain=input('ganancia del modulador:');
Mix=input('Teclea la ganancia de la entrada:');

z=sin(2*pi*fmod*t);

   for I=1:D(2);
       y(I)=Mix*x(I)+Ringain*(x(I)*z(I));
    end

%Presentacion de resultados:
subplot(3,1,1);
plot(t,x);
```

```
title('Señal del instrumento');
subplot(3,1,2);
plot(t,z);
title('Señal ring');
subplot(3,1,3);
plot(t,y);
title('Señal final');

%-------------------------------------------------------------------------------------------
function delaybasico_func (x,t,D,Ts)
%-------------------------------------------------------------------------------------------
% Algoritmo de retraso DELAY. Se prueba unicamente con un pulso para ver
% mejor sus efectos.Conviene que el tiempo de sim sea bastante mayor que un
% periodo para ver que pasa con las repeticiones
% y(n)= x(n)+Mix*x(n-m)
% PARAMETRO-> tiempo de retraso. Debe ser multiplo de Ts
% PARAMETRO-> Mix. fuerza de la repeticion.
% PARAMETRO->interpola=1, permite introducir tiempos no multiplo del de
% muestreo.
%-------------------------------------------------------------------------------------------

disp('DELAY BASICO seleccionado')
tdelay=input('introduce el tiempo de retraso en segundos:');
Mix=input ('introduce ganancia de retraso:');
interpola=input('¿usar interpolacion? 1->Si,No->otro:');
retraso=(tdelay/Ts);


%calcula interpolacion:
if interpola==1,

   retrasosup=floor(retraso); %redondea al entero mas proximo por abajo(min retraso)
    retrasoans=ceil(retraso);  %%redondea al entero mas proximo por arriba(max
retraso;
```

```
for I=1:D(2)
   if (I-retrasoans)<=0,
      y(I)=x(I);
   else
    muestrans= x(I-retrasoans);   %muestra mas antigua
    muestrasup= x(I-retrasosup);   %muestra mas nueva.
    m=muestrasup-muestrans;
    b=(retraso-retrasosup);
    estimacion=muestrans+m*b;
    y(I)=x(I)+Mix*estimacion;
   end
   end
 else

for I=1:D(2),
retras=floor(retraso); %redondea el retraso por abajo.No interpola.
   if (I-retras)<=0,
      y(I)=x(I);
   else
      y(I)=x(I)+Mix*x((I-retras));
   end
   end
end
%Presentacion de resultados:

subplot(2,1,1)
plot(t,x);
title('señal original')
subplot(2,1,2)
plot(t,y)
title('Señal tratada')
```

```
%-------------------------------------------------------------------------------------------------
function delayreal_func(x,t,D,Ts)
%-------------------------------------------------------------------------------------------------
% Algoritmo de retraso DELAYREALIMENTADO. Se prueba unicamente con un
pulso para ver
% mejor sus efectos.Conviene que el tiempo de sim sea bastante mayor que un
% periodo para ver que pasa con las repeticiones
% PARAMETRO-> tiempo de retraso. No tiene por que ser multiplo de Ts
% PARAMETRO-> Mix. fuerza de las repeticiones sucesivas.
% PARAMETRO->interpola=1, permite introducir tiempos no multiplo del de
% muestreo.
%-------------------------------------------------------------------------------------------------

disp('DELAY REALIMENTADO seleccionado')
y=x;
tdelay=input('introduce el tiempo de retraso en segundos:');
Mix=input ('introduce ganancia de retraso:');
interpola=input('¿usar interpolacion? 1->Si,No->otro:');
retraso=(tdelay/Ts);

%calcula interpolacion:
if interpola==1,

   retrasosup=floor(retraso); %redondea al entero mas proximo por abajo(min retraso)
    retrasoans=ceil(retraso);  %%redondea al entero mas proximo por arriba(max
retraso;

for I=1:D(2)
   if (I-retrasoans)<=0,
      y(I)=x(I);
   else
    muestrans= y(I-retrasoans);   %muestra mas antigua
    muestrasup=y(I-retrasosup);   %muestra mas nueva.
    m=muestrasup-muestrans;
```

```
    b=(retraso-retrasosup);
    estimacion=muestrans+m*b;
    y(I)=y(I)+Mix*estimacion;

  end
  end
else

retras=floor(retraso); %redondea el retraso por abajo.No interpola.
for I=1:D(2),

  if (I-retras)<=0,
     y(I)=x(I);
  else
     y(I)=y(I)+Mix*y((I-retras));
  end
  end
end

%Presentacion de resultados:

subplot(2,1,1)
plot(t,x);
title('señal original')
subplot(2,1,2)
plot(t,y)
title('Señal tratada')

%-----------------------------------------------------------------------------------------------
function chorus_func(x,t,D,Ts)
%-----------------------------------------------------------------------------------------------
% Algoritmo de retraso CHORUS. Se prueba unicamente con un pulso para ver
% mejor sus efectos.Conviene que el tiempo de sim sea bastante mayor que un
% periodo para ver que pasa con las repeticiones
```

```
% PARAMETRO-> tiempo de retraso base.No tiene por que ser multiplo de Ts
% PARAMETRO-> Profundidad de barrido
% PARAMETRO->frecuencia oscilador
% PARAMETRO-> Mix. fuerza de las repeticiones sucesivas.
% PARAMETRO->interpola=1, permite introducir tiempos no multiplo del de
% muestreo.
%-------------------------------------------------------------------------------------------------

disp('CHORUS seleccionado')
y=x;
tdelay=input('introduce el tiempo de retraso base en segundos:');
barrido=input('Profuncidad barrido LFO:');
flfo=input('frecuencia LFO en Hz:');
Mix=input ('introduce ganancia de retraso:');
z=barrido/2+tdelay+(barrido/2*sin(2*pi*flfo*t));                    %genera LFO
interpola=input('¿usar interpolacion? 1->Si,No->otro:');

%IMP-> tdelay+barrido=maximo delay.
%calcula interpolacion:
if interpola==1,

for I=1:D(2)

    retraso=(z(I)/Ts);
    retrasosup=floor(retraso);    %redondea al entero mas proximo por abajo(min
retraso)
    retrasoans=ceil(retraso);     %redondea al entero mas proximo por arriba(max retraso;

   if (I-retrasoans)<=0,
      y(I)=x(I);
   else
    muestrans= x(I-retrasoans);   %muestra mas antigua
    muestrasup= x(I-retrasosup);   %muestra mas nueva.
    m=muestrasup-muestrans;
```

```
    b=(retraso-retrasosup);
    estimacion=muestrans+m*b;
    y(I)=x(I)+Mix*estimacion;

  end
  end
else

for I=1:D(2),
retraso=floor(z(I)/Ts); %redondea el retraso por abajo.No interpola.
  if (I-retraso)<=0,
     y(I)=x(I);
  else
     y(I)=x(I)+Mix*x((I-retraso));
  end
  end
end

%Presentacion de resultados:

subplot(3,1,1)
plot(t,x);
title('señal original')
subplot(3,1,2)
plot(t,y)
title('Señal tratada')
subplot(3,1,3)
plot(t,z)
title('LFO')
```

```
%-------------------------------------------------------------------------------------------------
function flanger_func(x,t,D,Ts)
%-------------------------------------------------------------------------------------------------
% Algoritmo de retraso CHORUS. Se prueba unicamente con un pulso para ver
% mejor sus efectos.Conviene que el tiempo de sim sea bastante mayor que un
% periodo para ver que pasa con las repeticiones
% PARAMETRO->tipo oscilador.1->senoidal,2->triangular.
% PARAMETRO-> tiempo de retraso base.No tiene por que ser multiplo de Ts
% PARAMETRO-> Profundidad de barrido
% PARAMETRO->frecuencia oscilador
% PARAMETRO-> Mix. fuerza de las repeticiones sucesivas.
% PARAMETRO->interpola=1, permite introducir tiempos no multiplo del de
% muestreo.
%-------------------------------------------------------------------------------------------------

disp('FLANGER seleccionado')
y=x;
tdelay=input('introduce el tiempo de retraso base en segundos:');
barrido=input('Profuncidad barrido LFO:');
flfo=input('frecuencia LFO en Hz:');
Tlfo=1/flfo;
Mix=input ('introduce ganancia de retraso:');
generador=input('selecciona generador: 1->senoidal,2->triangular:');
interpola=input('¿usar interpolacion? 1->Si,No->otro:');

%generacion y operacion con onda senoidal-----------------------------------------
if generador==1
z=barrido/2+tdelay+(barrido/2*sin(2*pi*flfo*t));
%generacion ----------------------------------------------------------------:
else
periodo=0;
for I=1:D(2),

%periodos posteriores:
```

```
if t(I)-(periodo*Tlfo)>=0&t(I)<=(periodo*Tlfo+1/2*Tlfo)
m=barrido/(Tlfo/2);
z(I)=tdelay+(m*(t(I)-(periodo*Tlfo)));
end
if t(I)-(periodo*Tlfo+1/2*Tlfo)>=0&t(I)<=(periodo+1)*Tlfo
m=-barrido/(Tlfo/2);
z(I)=tdelay+(barrido+m*(t(I)-(periodo*Tlfo+1/2*Tlfo)));
end
if t(I)>=Tlfo*(periodo+1),
periodo=periodo+1
end
end
end
%operaciones flanger----------------------------------------------:
%IMP-> tdelay+barrido=maximo delay.
%calcula interpolacion:
if interpola==1,
for I=1:D(2)

    retraso=(z(I)/Ts);
    retrasosup=floor(retraso);    %redondea al entero mas proximo por abajo(min
retraso)
    retrasoans=ceil(retraso);     %redondea al entero mas proximo por arriba(max retraso;

   if (I-retrasoans)<=0,
      y(I)=x(I);
   else
    muestrans= y(I-retrasoans);   %muestra mas antigua
    muestrasup= y(I-retrasosup);   %muestra mas nueva.
    m=muestrasup-muestrans;
    b=(retraso-retrasosup);
    estimacion=muestrans+m*b;
    y(I)=y(I)+Mix*estimacion;
```

```
   end
   end
else

for I=1:D(2),
retraso=floor(z(I)/Ts); %redondea el retraso por abajo.No interpola.
   if (I-retraso)<=0,
      y(I)=x(I);
   else
      y(I)=y(I)+Mix*y((I-retraso));
   end
   end
end

%Presentacion de resultados:

subplot(3,1,1)
plot(t,x);
title('señal original')
subplot(3,1,2)
plot(t,y)
title('Señal tratada')
subplot(3,1,3)
plot(t,z)
title('LFO')
%-------------------------------------------------------------------------------------------------
function phaser_func (x,t,D,Ts)
%-------------------------------------------------------------------------------------------------
% Algoritmo de PHASER. Se prueba unicamente con un pulso para ver
% mejor sus efectos.Conviene que el tiempo de sim sea bastante mayor que un
% periodo para ver que pasa con las repeticiones
% PARAMETRO-> ganancia allpass.
% PARAMETRO-> Mix. fuerza de la repeticion.
% PARAMETRO-> retraso puede no ser multiplo de Ts
```

```
% PARAMETRO-> interpola=1, permite introducir tiempos no multiplo del de
% muestreo.
%------------------------------------------------------------------------

disp('PHASER seleccionado')
tdelay=input('introduce el tiempo de retraso en segundos:');
g=input ('introduce ganancia allpass:');
Mix=input ('introduce Mix:');
interpola=input('¿usar interpolacion? 1->Si,No->otro:');
retraso=(tdelay/Ts);

%calcula interpolacion:
if interpola==1,

   retrasosup=floor(retraso); %redondea al entero mas proximo por abajo(min retraso)
    retrasoans=ceil(retraso);  %%redondea al entero mas proximo por arriba(max
retraso;

for I=1:D(2)
   if (I-retrasoans)<=0,
       y(I)=-g*x(I);
   else
    muestrans= x(I-retrasoans);   %muestra mas antigua
    muestrasup= x(I-retrasosup);   %muestra mas nueva.
    m=muestrasup-muestrans;
    b=(retraso-retrasosup);
    estimacionx=muestrans+m*b;

    muestrans2= y(I-retrasoans);   %muestra mas antigua
    muestrasup2= y(I-retrasosup);   %muestra mas nueva.
    m2=muestrasup2-muestrans2;
    b2=(retraso-retrasosup);
    estimaciony=muestrans2+m2*b2;
```

```
    y(I)=-g*x(I)+estimacionx+g*estimaciony;

   end
   end
 else

for I=1:D(2),
retras=floor(retraso); %redondea el retraso por abajo.No interpola.
   if (I-retras)<=0,
      y(I)=-g*x(I);
   else
   y(I)=-g*x(I)+x(I-retras)+g*y(I-retras);
   end
   end
end

%Presentacion de resultados:

subplot(2,1,1)
plot(t,x);
title('señal original')
subplot(2,1,2)
plot(t,y)
title('Señal tratada')

%-------------------------------------------------------------------------------------------------
function reverb_func (x,t,D,Ts)
%-------------------------------------------------------------------------------------------------
% Algoritmo de REVERB. Se prueba unicamente con un pulso para ver
% mejor sus efectos.Conviene que el tiempo de sim sea bastante mayor que un
% periodo para ver que pasa con las repeticiones
% PARAMETRO-> tiempo de retraso1.
% PARAMETRO-> tiempo de retraso2.
% PARAMETRO-> tiempo de retraso3.
```

```
% PARAMETRO-> tiempo de retraso4.
% PARAMETRO->interpola=1, permite introducir tiempos no multiplo del de
% muestreo.
%------------------------------------------------------------------------

disp('REVERB seleccionado')

Mix =input ('introduce ganancia de general:');

tdelay1=input('introduce el tiempo de retraso1 en segundos:');
Mix1=input ('introduce ganancia de retraso1:');
tdelay2=input('introduce el tiempo de retraso2 en segundos:');
Mix2=input ('introduce ganancia de retraso2:');
tdelay3=input('introduce el tiempo de retraso3 en segundos:');
Mix3=input ('introduce ganancia de retraso3:');
tdelay4=input('introduce el tiempo de retraso4 en segundos:');
Mix4=input ('introduce ganancia de retraso4:');
interpola=input('¿usar interpolacion? 1->Si,No->otro:');

retraso1=(tdelay1/Ts);
retraso2=(tdelay2/Ts);
retraso3=(tdelay3/Ts);
retraso4=(tdelay4/Ts);
w=zeros(1,D(2));

%calcula interpolacion:
if interpola==1,

retrasosup1=floor(retraso1); %redondea al entero mas proximo por abajo(min retraso)
retrasoans1=ceil(retraso1);  %%redondea al entero mas proximo por arriba(max retraso;

retrasosup2=floor(retraso2); %redondea al entero mas proximo por abajo(min retraso)
retrasoans2=ceil(retraso2);  %%redondea al entero mas proximo por arriba(max retraso;
```

```
retrasosup3=floor(retraso3); %redondea al entero mas proximo por abajo(min retraso)
retrasoans3=ceil(retraso3);  %%redondea al entero mas proximo por arriba(max retraso;

retrasosup4=floor(retraso4); %redondea al entero mas proximo por abajo(min retraso)
retrasoans4=ceil(retraso4);  %%redondea al entero mas proximo por arriba(max retraso;

for I=1:D(2)

   if (I-retrasoans1)>0
   muestrans1= x(I-retrasoans1);   %muestra mas antigua
   muestrasup1= x(I-retrasosup1);   %muestra mas nueva.
   m=muestrasup1-muestrans1;
   e=(retraso1-retrasosup1);
   estimacion1=muestrans1+m*e;
   else
   estimacion1=0;
   end
   if (I-retrasoans2)>0
   muestrans2= x(I-retrasoans2);   %muestra mas antigua
   muestrasup2= x(I-retrasosup2);   %muestra mas nueva.
   m=muestrasup2-muestrans2;
   e=(retraso2-retrasosup2);
   estimacion2=muestrans2+m*e;
   else
   estimacion2=0;
   end
   if (I-retrasoans3)>0
   muestrans3= x(I-retrasoans3);   %muestra mas antigua
   muestrasup3= x(I-retrasosup3);   %muestra mas nueva.
   m=muestrasup3-muestrans3;
   e=(retraso3-retrasosup3);
   estimacion3=muestrans3+m*e;
   else
   estimacion3=0;
```

```
   end
   if (I-retrasoans4)>0
   muestrans4= x(I-retrasoans4);   %muestra mas antigua
   muestrasup4= x(I-retrasosup4);   %muestra mas nueva.
   m=muestrasup4-muestrans4;
   e=(retraso4-retrasosup4);
   estimacion4=muestrans4+m*e;
   else
   estimacion4=0;
   end

   if (I-3)<1

y(I)=Mix*x(I)+Mix1*estimacion1+Mix2*estimacion2+Mix3*estimacion3+Mix4*estim
acion4;
   else
   z(I)=Mix1*estimacion1+Mix2*estimacion2+Mix3*estimacion3+Mix4*estimacion4;
   w(I)=0.4*w(I-1)-0.2499*w(I-2)+0.0441*w(I-3)+0.5814*z(I-1)+0.2142*z(I-2);
   y(I)=Mix*x(I)+w(I);
   end

end

 else

for I=1:D(2),

   retras1=floor(retraso1); %redondea al entero mas proximo por abajo(min retraso)
   retras2=floor(retraso2); %redondea al entero mas proximo por abajo(min retraso)
   retras3=floor(retraso3); %redondea al entero mas proximo por abajo(min retraso)
   retras4=floor(retraso4); %redondea al entero mas proximo por abajo(min retraso)

   if (I-retras1)<=0
   muestra1=0;
```

```
    else
    muestra1=x(I-retras1);
    end
    if (I-retras2)<=0
    muestra2=0;
    else
    muestra2=x(I-retras2);
    end
    if (I-retras3)<=0
    muestra3=0;
    else
    muestra3=x(I-retras3);
    end
    if (I-retras4)<=0
    muestra4=0;
    else
    muestra4=x(I-retras4);
    end

    if (I-3)<1
    y(I)=Mix*x(I)+Mix1*muestra1+Mix2*muestra2+Mix3*muestra3+Mix4*muestra4;
    else

    z(I)=Mix1*muestra1+Mix2*muestra2+Mix3*muestra3+Mix4*muestra4;
    w(I)=0.4*w(I-1)-0.2499*w(I-2)+0.0441*w(I-3)+0.5814*z(I-1)+0.2142*z(I-2);
    y(I)=Mix*x(I)+w(I);

    end
    end

end
```

```
%Presentacion de resultados:

subplot(2,1,1)
plot(t,x);
title('señal original')
subplot(2,1,2)
plot(t,y)
title('Señal tratada')
```

ANEXO B. DOCUMENTACIÓN ADICIONAL.

B.1. C PROGRAMS ON THE ADSP-2106x .

Se incluye en este apartado una nota de aplicación sobre cómo programar en C para la plataforma DSP elegida. Así mismo se detallan algunas particularidades interesantes como documentación adicional.

B.2. SHARC DSP21061 DATA SHEET.

En el datasheet del microprocesador se recogen todos las características adicionales a tener en cuenta del chip.

4. BIBLIOGRAFÍA:

[1] PROAKIS,J.G,MANOLAKIS,D.G **Tratamiento digital de señales.**.3ª ED Prentice Hall 1998.

[2] JULIUS O.SMITH III **(**jos@ccrma.stanford.edu**)** . **Digital Waveguide**. Center for Computer Research in Music and Acoustics (CCRMA) **.** Department of Music**,** Stanford University Stanford, California 94305.

[3] JULIUS O.SMITH III **(**jos@ccrma.stanford.edu**)** .**Digital Filters.** Center for Computer Research in Music and Acoustics (CCRMA)**.** Department of Music**,** Stanford University **.**
Stanford, California 94305.

[4] STEVEN W. SMITH,)**The Scientist and engineer´s Guide to Digital Signal Processing.** Ph.D. California Technical Publishing. ISBN 0-9660176-3-3 (1997

[5] MOORER,JAMES. **About this reverberation business.** Computer Music Journal, Volume 3, number(1979);13-28.

[6] Harmony- Central Effects Explained. Información sobre efectos digitales.

http://www.harmony-central.com/Effects/effects-explained.html.

[7] Proyectos DSP. Universidad de república. Informe sobre tecología DSP aplicada a efectos de sonido.

http://iie.fing.edu.uy/ense/asign/sisdsp/proyectos/1999/flanging/informe.htm

[8] Digital Waveguide Stanford University .
website

http://ccrma-www.stanford.edu/~jos/waveguide/

[9] Digital Filters. Stanford University website.
http://www-ccrma.stanford.edu/~jos/filters/filters.html

5.PLIEGO DE CONDICIONES GENERALES Y ECONÓMICAS.

Las condiciones que se exponen a continuación son de obligado cumplimiento por las partes contratantes:

1. Todas las condiciones y cláusulas a las que hace referencia el presente documento de condiciones, tratan sobre la contratación efectiva por parte de personas físicas o jurídicas de un sistema de control activo del ruido implementable en distintos sistemas y modelos.

2. Todos los circuitos y montajes se ajustarán a los valores y recomendaciones contenidas en la documentación del proyecto. El comprador no tendrá facultad para efectuar cambios por sí mismo. En cualquier caso será obligación del comprador consultar cualquier duda que pudiera surgir en la interpretación del proyecto.

3. La contratación del presente proyecto se regirá por la norma del código de comercio vigente. En cualquier caso, se entenderán aceptadas por el comprador las condiciones recogidas en dichas normas, aún cuando no aparezcan en el presente pliego de condiciones.

4. Todos los precios que figuran en el presupuesto del presente proyecto son los vigentes en el momento de su elaboración. El hecho de una subida de precios en alguna de las partidas no dará derecho al comprador a ninguna indemnización económica, aunque dicha subida se produjera dentro del plazo de entrega. En todo caso se respetarán a todos los efectos los presupuestos contenidos en este proyecto que se entenderán aceptados por el comprador en su totalidad.

5. No se aceptarán otras condiciones que las que hayan sido convenidas de mutuo acuerdo y por escrito. En cualquier caso, si alguna de dichas condiciones modificara el presente proyecto en alguna de sus partes, habrá de hacerse constar en el documento de aceptación por ambas partes. En caso contrario, se entenderá la modificación como no válida. Cualquier acuerdo verbal con un representante no será válido hasta que sea ratificado por la parte interesada.

6. Serán causa de anulación del contrato alteraciones en medidas y características que por su magnitud impidan el funcionamiento correcto de alguna parte componente del dispositivo. A tal fin, el comprador está obligado a conocer el uso que se pretende de cada parte del proyecto.

7. Las dos partes se comprometen desde la fecha de la firma del contrato a llevar a cabo todo lo que en el mismo se estipula.

8. Ante una reclamación de alguna de las partes o discrepancia en lo que concierne al cumplimiento de lo firmado, será el Tribunal de Madrid (o instancias superiores en su defecto) el único ante el que se puede recurrir, y por tanto el dictamen o sentencia que se emita obligará a ambas partes.

9. Al firmarse el contrato, el vendedor queda comprometido a facilitar a la otra parte toda la información necesaria para la instalación y buen funcionamiento de los equipos.

10. Existirá un plazo de garantía de un año a partir de la entrega del equipo frente a cualquier defecto técnico que pueda presentarse. Dicha garantía quedará sin efecto si se demuestra que el objeto ha sido objeto de una manipulación o utilización indebida. Cumplido dicho plazo de garantía, el vendedor queda obligado a encargarse de la reparación del equipo durante un plazo de 3 años. Fuera de este período quedará a su criterio el atender o no los requerimientos que el comprador le formule. Durante el plazo de garantía, la totalidad de los gastos originados por las reparaciones será atendida por el vendedor.

6.PRESUPUESTO

6.1.- PRESUPUESTOS PARCIALES

6.1.1.- COSTES DE INGENIERÍA.

En este grupo se incluyen las horas dedicadas al estudio del estado del arte, estudio la teoría y diseño de algoritmo s de tratamiento de señal, así como de todas las matemáticas asociadas, aprendizaje del lenguaje de programación Matlab y C aplicado al DSP. Lectura y estudio de la extensa documentación de la DSP SHARC 21061. Así pues los costes de ingeniería obtenidos son:

Estudio del estado del arte	50h
Estudio de la teoría de procesamiento de señal	75h
Estudio de la documentación	200h
Desarrollo de los programas	300h
Pruebas	1750h
Total de horas de ingeniería	800h
Coste de una hora de ingeniería	36euros
Total del coste de ingeniería	**28.800euros**

6.1.2.- COSTES DE ADQUISICIÓN

Suponiendo que los productos utilizados se amortizan en 5 años de uso, vamos a calcular el coste real de los productos que no se han adquirido exclusivamente para la realización del proyecto, ya que su coste debe repartirse entre las horas efectivas de utilización, aproximadamente 1 año.

El coste de los productos utilizados en la realización de este proyecto, seguido de las horas de uso y del coste amortizado se detalla a continuación. Aquel/os recursos utilizados que no aparecen ponderados por el coste de amortización, se suponen de uso exclusivo para este proyecto:

Pc FUJITSU SIEMENS AMILO.	2500*1/5	Euros
Placa EZ-KIT LITE: (SHARC DSP2106).	102*1/5	Euros
Total de coste de los recursos empleados.	**520**	**Euros**

6.1.3.- COSTES FINANCIEROS.

Los costes financieros se calculan sobre los costes de ingeniería y de adquisición, suponiendo un coste del dinero anual del 10% Y teniendo en cuenta que la duración oficial del proyecto ha sido de 9 meses.

Así pues, los costes financieros del proyecto ascienden a:

$$(520 + 28800) \cdot 0.1 \cdot \frac{9}{12} = 2199 \text{ Euros}$$

6.2.- PRESUPUESTO GENERAL.

Costes de ingeniería	28800	Euros
Costes de adquisición	520	Euros
Costes financieros	2199	Euros
Presupuesto General	**31519**	**Euros**