ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO DE TELECOMUNICACIÓN ESPECIALIDAD SONIDO E IMAGEN

Título del proyecto:

"Diseño y test de una mesa de mezclas analógica con efectos sonoros usando Arduino."

Tutor: Carlos A. de la Cruz Blas

Alumno: Gonzalo Thomas Erviti

Fecha: 21 Enero 2013

AGRADECIMIENTOS

- Al tutor del proyecto Carlos A. de la Cruz Blas por su ayuda y apoyo.
- A mis compañeros de laboratorio por la ayuda prestada.
- A mi familia y amigos.

ÍNDICE

CAPÍTULO 1

1. INTRODUCCIÓN Y OBJETIVOS.

Este PFC presenta una mesa de mezcla electrónica utilizando un diseño de modo mixto, utilizando amplificadores operacionales para el diseño analógico y un microcontrolador basado en la plataforma Arduino para realizar efectos de audio. La mesa de mezclas de audio o mezcladora de sonidos, es un dispositivo electrónico al cual se conectan diversos elementos emisores de audio, tales como micrófonos, entradas de línea, samplers, sintetizadores, gira discos de vinilos, reproductores de cd, reproductores de cintas, etc. Una vez que las señales sonoras entran en la mesa estas pueden ser procesadas y tratadas de diversos modos para dar como resultado de salida una mezcla de audio, mono, multicanal o estéreo. El procesado habitual de las mesas de mezclas incluye la variación del nivel sonoro de cada entrada, ecualización, efectos de envío y de inserción, panorámica y balance.

OBJETIVO DEL PFC

El objetivo de este proyecto es la simulación, caracterización y construcción PCB de una mesa de mezclas, en su mayoría analógica, pero con elementos digitales que permitirá realizar efectos sonoros a través de un convertidor analógico digital y su procesamiento con un microcontrolador (Arduino).

Para llevarlo a cabo, la mesa será divida en subsistemas tales como: adaptador de impedancias de entrada, filtrado, suma de señales y efectos digitales con la ayuda de un microcontrolador y su conversión analógica digital. Para su realización y puesta en marcha se hará uso de herramientas de simulación de verificación eléctrica para la parte analógica y uso del lenguaje C para programar al microcontrolador.

1.1DESCRIPCIÓN DEL PFC

El proyecto de fin de carrera consiste en el diseño de una mesa de mezclas con los elementos más importantes. En la Fig. 1.1 se muestra un diagrama general donde se muestran las partes más importante de la mezcladora en este PFC, el adaptador de impedancias, preamplificador, ecualizador, sumador y master se harán de forma analógica mientras que la parte de efectos se hará con la ayuda de un microcontrolador basado en Arduino.

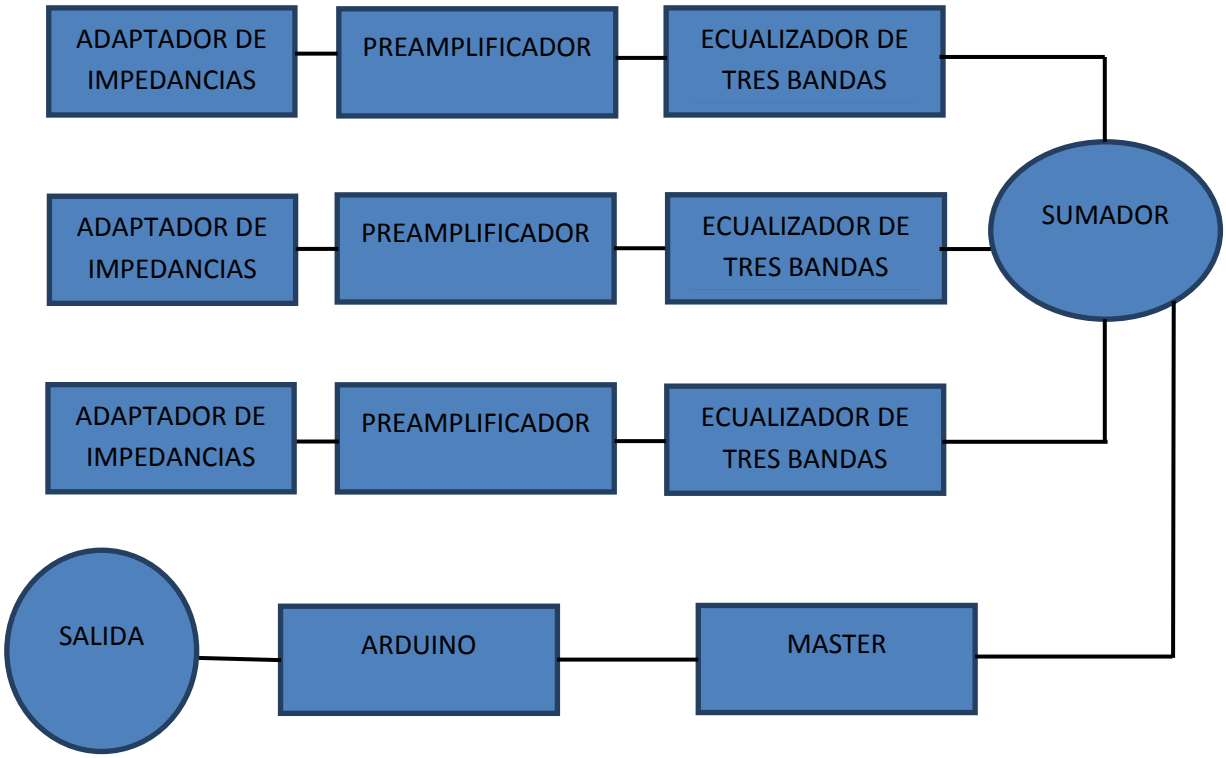


Figura 1.1.1: Diagrama de bloques formado por las partes de la mesa de mezclas.

En los párrafos siguientes se describen en más detalle las etapas de la Fig. 1.1.1, así como la forma de su diseño empezando por la parte analógica y posteriormente la digital.

Parte analógica

Como se muestra en la Fig. 1.1.1, la primera parte de la mesa de mezclas es el adaptador de impedancias, para entradas de micrófono y de nivel de línea. Los valores típicos de impedancia para una entrada de micrófono oscilan alrededor de unos 600 Ω, mientras que los valores típicos de impedancia para nivel de línea oscilan cerca de 10 kΩ, por lo que se hace imprescindible esta etapa de la mesa, ya que si no las señales de entrada se verían afectadas de manera considerable.

Una vez que la señal atraviesa la parte del adaptador de impedancias, la siguiente etapa es el preamplificador, su función es la de aumentar la señal hasta el nivel deseado por el usuario, para las posteriores etapas. Una que la señal tiene el nivel deseado, esta será procesada mediante un ecualizador. Constará de un filtro de graves, medios y agudos. Para la elección del filtro se tendrán en cuenta características tales como numero de amplificadores operacionales necesitados, así como número de componentes electrónicos, respuesta en frecuencia y la sensibilidad del filtro a variaciones de los componentes.

La variación del nivel sonoro de cada entrada, es la función principal de la mesa de mezclas. Su realización será mediante un circuito sumador analógico. Al igual que en la etapa de filtrado serán estudiados varios circuitos sumadores, y se escogerá el más adecuado dependiendo de la relación entre cantidad de componentes y calidad de la suma de señales.

Por último se llevará a cabo un último circuito (master), que servirá al usuario para ajustar el nivel sonoro total a la salida de la mesa.

Parte digital.

La parte digital de esta mesa de mezclas consta de un microcontrolador programable, Arduino, que será colocado a la salida del master. Con la ayuda de un conversor analógico digital las señales de audio serán procesadas por el microcontrolador para realizar diversos efectos en el mundo digital; posteriormente las señales digitales serán reconvertidas a analógico para interaccionar con la mesa de mezclas.

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software - hardware flexibles, fáciles de usar y con un costo más que accesible. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos. Los programas hechos con Arduino se dividen en tres partes principales: *estructura*, *valores* (variables y constantes), y *funciones*. El Lenguaje de programación Arduino se basa en C/C++.

Para construir el sistema completo analógico y digital se construirá un PCB con la parte analógica de la mesa de mezclas y posteriormente se acoplara el microcontrolador con la plataforma Arduino a esta PCB y programar varios efectos de audio, tales como reverb, delay...

La salida del master de la mesa de mezclas será llevada hasta un pin de entrada analógica del microcontrolador Arduino. La unión se realizara mediante un conector. Arduino convertirá la señal de entrada analógica en digital, y procesará esta señal a tiempo real.

La parte analógica de la mesa de mezclas, será testeada primeramente montando el circuito en una protoboard, y por medio del equipo apropiado se comprobará que el circuito se comporta igual que en la simulación realizada en el OrCad PSpice. Una vez comprobado se procederá a la impresión del circuito en una placa PCB y se soldarán los componentes y se volverá a comprobar su correcto funcionamiento. En lo referido a la parte digital, se comprobará que el microcontrolador Arduino este configurado correctamente, tanto para la conversión A/D, como para el procesado de la señal.

En la siguiente sección se dan más detalles de la metodología a utilizar junto con algunas consideraciones que serán tomadas en el diseño final

- **Metodología de desarrollo:**

El objetivo es realizar el diseño y la construcción de una mesa de mezclas analógica, pero con una parte digital y con las mejores características posibles. El primer paso será coger destreza en la utilización del simulador eléctrico ORCAD Pspice ya que se pretende realizar numerosas simulaciones de circuitos para estudiar sus comportamientos. Además, una vez finalizado el estudio de los diferentes sistemas de adaptación, se utilizará OrCad Editor para desarrollar el PCB necesario para fabricar el circuito impreso final.

Por otra parte y de forma paralela, se necesita conocimientos del lenguaje C y experiencia en programación de microcontroladores, para realizar la parte de efectos sonoros. En la actualidad los microcontroladores tienen un gran número de elementos que hacen sus aplicaciones más fáciles, así aprovechando esta posibilidad se hará uso de su conversión analógico digital para procesar el sonido en formato digital y hacer diversos efectos en tiempo real dentro de las posibilidades del microcontrolador. En esta parte la selección del microcontrolador estará basada en Arduino.

Algunas consideraciones

En los últimos años se está empezando a diseñar circuitos electrónicos que operan a baja tensión, con niveles tales como 5 V, 3.3 V o incluso 1,8 V, el propósito es que la mesa de mezclas opere a 5 V. Serán necesarios los amplificadores operacionales tipo *rail to rail* los cuales pueden alcanzar tensiones de salida muy cercanas a las tensión de alimentación del amplificador. De lo contrario al operar la mesa de mezclas a una tensión de 5V y no utilizar este tipo de amplificadores, la tensión máxima de salida seria alrededor de 2V aproximadamente, por lo que afectaría de manera sustanciosa a la señal que pasa por el amplificador operacional.

Es una característica importante ya que aumentan parámetros tan importantes como el SNR y el rango dinámico. Otro factor a tener en cuenta es el ancho de banda del amplificador operacional, ya que hará falta un amplificador operacional *rail to rail* que tenga un ancho de banda de 20Hz a 20KHz como mínimo.

Como se ha explicado anteriormente, se realizaran simulaciones con el programa OrCad PSpice, de todos los bloques de la mesa de mezclas. La elección final del circuito más apropiado estará condicionada por varios factores. Se trata de llegar a un compromiso entre la cantidad de componentes necesarios, y la calidad de la respuesta de este, por eso se llevarán a cabo varias simulaciones, y se estudiaran todas y cada una de ellas, teniendo en cuenta sus ventajas y desventajas para seleccionar el diseño y componentes mejores para la mesa.

Respecto a la parte digital, el primer paso con el microcontrolador Arduino, será instalar los drivers necesarios para su uso y conexión con el ordenador. La conexión se realizará mediante USB. El siguiente paso es la descarga del sofware gratuito, ofrecido por Arduino, para la programación de este. Se hará uso de su convertidor A/D, se programarán efectos sonoros digitales, y Arduino procesara a tiempo real la señal de entrada, que será la señal de salida de la mesa de mezclas.

El microcontrolador Arduino trae incorporado varios pines, tanto de entrada como de salida y de alimentación para hacer la conexión con el PCB analógico. La unión entre la mesa de mezclas y Arduino se realizara mediante un conector, de la salida de la última etapa a un pin de entrada del microcontrolador.

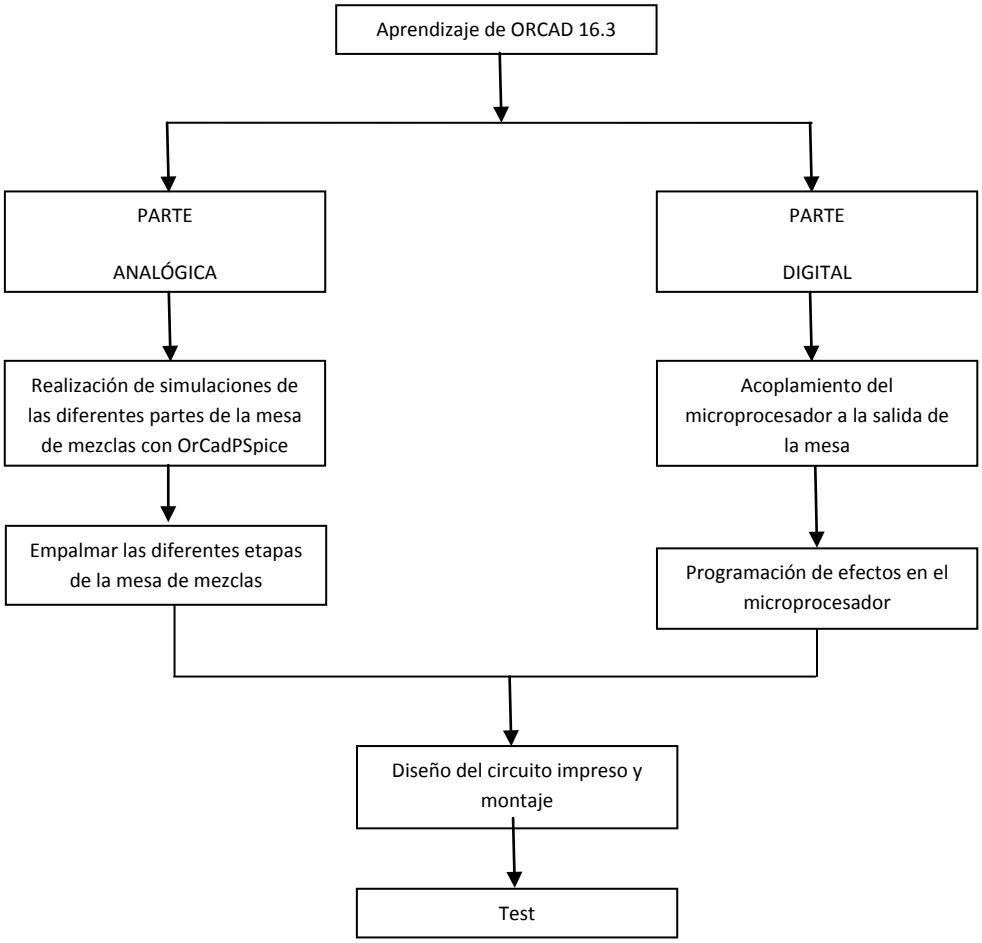


Figura 1.1.2. Procesos para realizar el proyecto

El procedimiento descrito anteriormente puede resumirse en la Fig. 1.1.2. El diagrama mostrado anteriormente representa los diferentes pasos a seguir para la realización del proyecto. Como se puede observar se divide en dos ramas, parte analógica y parte digital. La memoria está organizada de igual manera que este esquema.

Primeramente en el capítulo 2 se empezará por explicar la parte analógica realizada para la mesa de mezclas, se mostraran los circuitos diseñados así como sus respectivas simulaciones.

La rama de la parte digital seguirá se encuentra en el capítulo 3, en este capítulo se explicaran las características más relevantes tanto de Arduino como de los efectos digitales. Más tarde se procederá a explicar un funcionamiento más detallado de Arduino.

Las dos últimas partes de este proyecto, donde se juntan estas dos ramas mencionadas anteriormente, son el diseño de la placa PCB (capitulo 4), y el testeo de todo el sistema (capitulo 5). La memoria seguirá este orden descrito, de tal manera que en capítulo de diseño del circuito impreso se explicaran los pasos más relevantes y la manera en la que se ha diseñado la PCB, asi como el proceso de montaje de esta. Por último se explicará cómo se ha llevado a cabo el testeo de la mesa de mezclas, y se expondrán varios resultados de esté testeo. Finalmente se darán las conclusiones y líneas futuras del PFC en el capítulo 6.

CAPÍTULO 2

DISEÑO ANALÓGICO

En este capítulo se va a tratar las partes del proyecto que conciernen a la electrónica analógica. En primer lugar se va a proceder a mostrar algunos circuitos electrónicos básicos para el diseño de posteriores circuitos, entre ellos se encuentran filtros, amplificadores…La siguiente parte que seguirá será los circuitos diseñados para la elaboración de la mesa de mezclas, dentro de este apartado se mostrarán las simulaciones llevadas a cabo. Por último se mostrarán algunos circuitos descartados para el diseño de la mesa de mezclas.

2.1 INTRODUCCIÓN A CIRCUITOS BÁSICOS

En esta parte se van a introducir algunos circuitos electrónicos básicos, explicando brevemente sus propiedades. Estos circuitos son los que servirán para realizar circuitos más complejos posteriormente.

Se empezará por varios circuitos pasivos, principalmente filtros, y se continuará con circuitos activos basados en opamps.

Filtro paso bajo

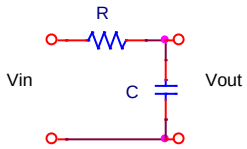


Figura 2.1.1. Filtro pasivo paso bajo.

En la figura anterior se muestra el esquemático de un filtro pasivo RC. Su funcionamiento se basa en que a medida que va aumentando la frecuencia de la señal la reactancia capacitiva del condensador va disminuyendo, y se alcanza un valor de frecuencia donde la reactancia del condensador y de la resistencia se igualan, a esto se le llama frecuencia de corte, y viene dada por la fórmula:

$$f_c = \frac{1}{2\pi RC}$$

A esta frecuencia de corte la señal se atenúa 3 dB, y al ser un filtro de primer orden su función de transferencia cae 10dB/década a partir de ese punto.

Filtro paso alto

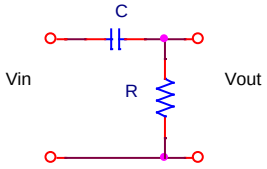


Figura 2.1.2. Filtro paso alto pasivo.

En la figura 2.1.2 se muestra un filtro paso alto pasivo, su funcionamiento es prácticamente al del paso bajo, el condensador no deja pasar las frecuencias bajas hasta que que la reactancia capacitiva iguale al valor de la resistencia, y en ese punto se encuentra la frecuencia de corte del filtro. Este punto tiene las mismas propiedades que el paso bajo y se calcula de la misma manera.

Filtro paso banda

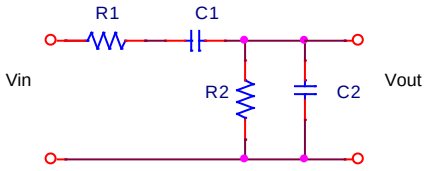


Figura 2.1.3. Filtro paso banda.

En la figura 2.1.3 se muestra un filtro paso banda de primer orden pasivo. Básicamente consiste en un filtro paso alto y un paso bajo colocados en cascada, de esta manera se consigue que únicamente se deje pasar un rango de frecuencias. El funcionamiento es el mismo que en los anteriores filtros. Las frecuencias de corte son calculadas con la misma fórmula, pero con valores distintos para cada filtro.

Circuitos activos con OPAMP

A continuación se presentan varios circuitos activos, cuentan todos ellos con un amplificador operacional.

Circuito seguidor.

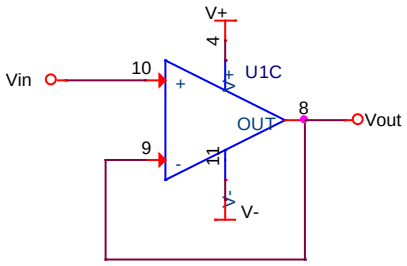


Figura 2.1.4. OPAMP como seguidor.

En la figura anterior se muestra el esquemático de un amplificador operacional operando como seguidor. Debido a las propiedades de los opamps este circuito es comúnmente usado para eliminar efectos de carga, o como adaptador de impedancias. La tensión de entrada es la misma que la tensión de salida, y la impedancia de entra se considera infinito

Amplificador no inversor

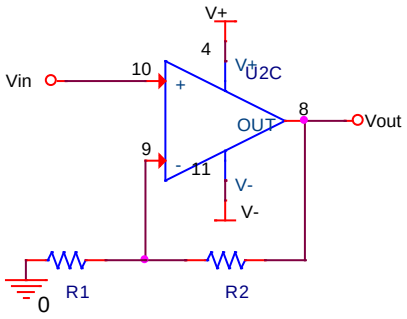


Figura 2.1.5. Amplificador no inversor.

En la figura anterior se representa un opamp como amplificador no inversor. La tensión de entrada se aplica al pin positivo, y en el pin negativo se replica esta tensión, y la ganancia del circuito se puede calcular medienteuun divisor de tensión:

$$V_o = V_i \left(1 + \frac{R2}{R1}\right)$$

Tiene la ventaja que la impedancia de entrada es infinita.

Integrador

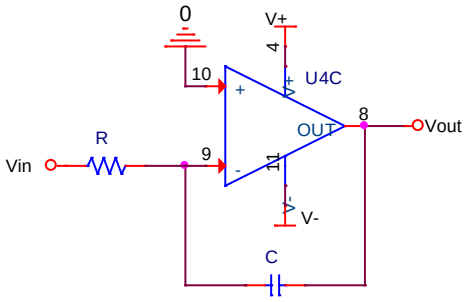


Figura 2.1.6. OPAMP como integrador.

EN la figura 2.1.6 se muestra la configuración de un opamp como integrador. Este sistema integra e invierte la señal de entra (V_{in}y V_{out} son funciones dependientes del tiempo).

Su función viene dada por:

$$Vout = \int_0^t -\frac{Vin}{RC}\,dt$$

Derivador

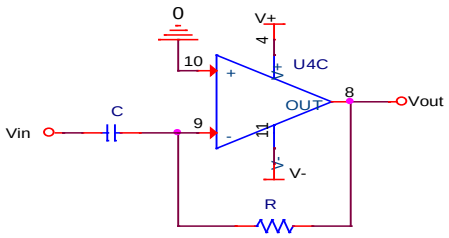


Figura 2.1.7. OPAMP como derivador.

En la figura anterior se muestra el esquemático de un opamp como derivador. Esta configuración del opamp deriva la señal de entrada respecto al tiempo, su función viene dada por:

$$Vout = -RC\frac{dVin}{dt}$$

Filtro paso bajo activo.

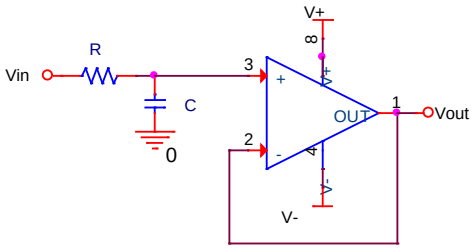


Figura 2.1.8. Filtro bajo activo.

La figura 2.1.8 muestra un filtro paso bajo activo de primer orden. Realmente se basa en un filtro paso bajo RC seguido de un operacional como seguidor, de esta manera se consigue una impedancia de salida elevada. El cálculo de la frecuencia de corte es idéntico al del filtro paso bajo pasivo.

Cabe destacar que de esta misma manera se pueden conseguir filtro paso alto, paso banda y rechazo banda.

Filtro paso bajo activo de segundo orden.

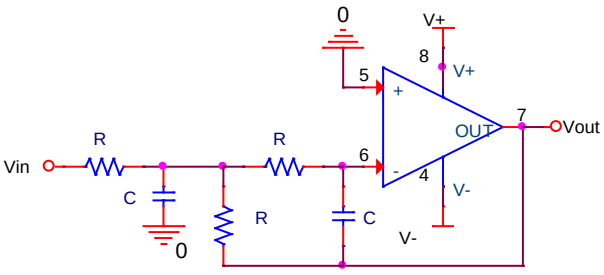


Figura 2.1.9. Filtro paso bajo activo de segundo orden.

En la figura anterior se muestra el esquemático de un filtro paso bajo activo de segundo orden. Básicamente son dos filtros paso bajo RC de primer orden, el primer filtro lo forman R y

el primer condensador conectado a tierra, y el segundo filtro lo forman el condensador que está conectado a la patilla negativa y la resistencia R que está conectada a la realimentación del operacional. La otra resistencia existente se usa únicamente para que la ganancia sea 0dB. El cálculo de la frecuencia de corte del filtro viene dado por la siguiente formula:

$$fc = \frac{1}{2\pi R\sqrt{C_1 C_2}}$$

De esta misma manera que ha sido diseñado un filtro paso bajo de segundo orden, se pueden conseguir filtros paso alto, paso banda y rechazo banda de segundo orden.

2.2 CIRCUITOS INCLUIDOS EN EL DISEÑO DE LA MESA DE MEZCLAS.

En este apartado se expondrán las características principales del diseño en primer lugar, y posteriormente los circuitos diseñados junto con sus respectivas simulaciones. A continuación se muestras un diagrama de bloques con las diferentes etapa existentes en la mesa de mezclas.

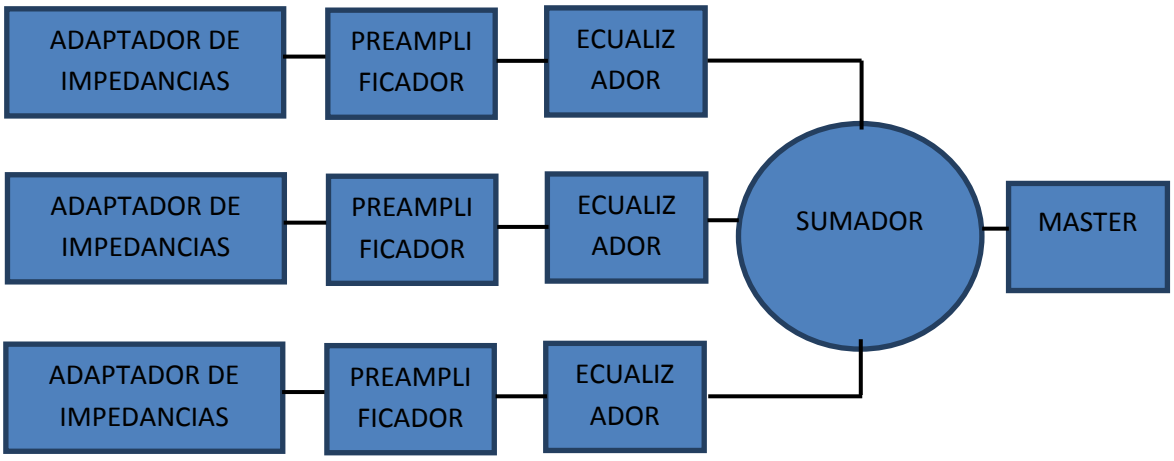


Figura 2.2.1. Diagrama de bloques de la mesa de mezclas.

En el diagrama de bloques como los tres canales existentes tienen en común las diferentes etapas, y posteriormente hay dos etapas comunes.

El orden de las etapas es importante, ya que de no ser el orden adecuado, esto afectaría de manera no deseada a las señales de entrada. El adaptador de impedancias se encargará de evitar que se pierda señal debido a la conexión entre el dispositivo de entrada y la mesa. La siguiente etapa en la cascada será el preamplificador, su función es la de aumentar la amplitud de la señal para operar a los niveles deseados, y más tarde poder manipularla. El ecualizar seguirá a esta última etapa, acentuando o atenuando según el gusto del usuario las tres bandas frecuenciales con las que este cuenta.

Estas tres etapas se encuentran en los tres canales independientes, y a los siguientes elementos en la cascada son el sumador y el master, que estas dos son comunes a los tres canales. El sumador es el encargado de mezclar las tres señales de entrada en una única, y poder

ajustar la cantidad de señal que se quiera que se sume de cada canal. Y por último la etapa de master, su función es la de ajustar la amplitud de la señal final.

Sección Alimentación del circuito

En primer lugar, la alimentación de la mesa de mezclas será de 0-5V en DC, con esto se intenta ser compatible con el actual mercado dentro de la electrónica, pudiendo ser así esta mesa portátil. Sin embargo esto implica tener en cuenta varias características, al no tener tensión negativa y positiva, se hace necesaria la creación de un nivel intermedio entre los dos voltajes, tierra analógica .La señal de audio está definida entre -1V y 1V, por lo que al ser introducida en el circuito quedaría eliminada la parte negativa de la señal. Para solventar este problema, se aumenta el nivel de continua de la señal de audio hasta 2.5 V al entrar a la mesa de mezclas, es decir en lugar de usar el nivel de tierra de la fuente de alimentación de 0 V, usará la llamada tierra analógica, que equivale a 2.5 V, así la señal de audio queda dentro del rango de alimentación. No sólo hace falta para la señal de audio, sino que se hace necesaria para la alimentación de los amplificadores operacionales. Usualmente los amplificadores se alimentan con tensiones negativas y positivas que varían desde +- 20V hasta +-3, sin embargo en este caso serán alimentados con 5V y 0V. Estos amplificadores pueden soportar esta alimentación (single supply). Los amplificadores operacionales, irán alimentados de la siguiente manera, por su entrada de tensión positiva se introducirán los 5V provenientes de la fuente de alimentación, la patilla de voltaje negativo se llevara a la tierra de la fuente de alimentación. De las entradas del amplificador operacional, se elegirá la entrada positiva para introducirla la tierra analógica, y la entrada inversora para la señal de audio (excepto en el generador de 2.5V).

Para la alimentación de los amplificadores ya sea de 5V o de 2.5 V se utilizan condensadores de desacoplo, situados justo a las entradas, de esta manera la tensión a la entrada estará estabilizada.

Como muestra el diagrama de bloques anterior (figura 2.2.1) , hay tres etapas conectadas en cascada por cada canal, más dos etapas finales conectadas en cascada igualmente y al tener varias etapas constituidas mayormente por amplificadores operacionales puede producirse un offset no deseado a lo largo de todas estas. Para su medición y control se colocaran conectores en la realimentación de los amplificadores operacionales en las etapas consideradas más críticas. Serán colocados en la etapa preamplificadora del primer canal, y en la etapa de master final, así se podrá ver como varía este offset, y se podrá regular.

Generador de tierra analógica (2.5 V)

Al alimentarse el circuito únicamente con una tensión continua de 5V, es necesario la creación de lo que se llama “tierra analógica “. De esta manera los amplificadores operacionales pueden operar en todo el rango de la señal.

El circuito descrito en la figura 2.2.2 es un simple divisor de tensión, en el cual a partir de 5V conseguimos una tensión de 2.5 V.

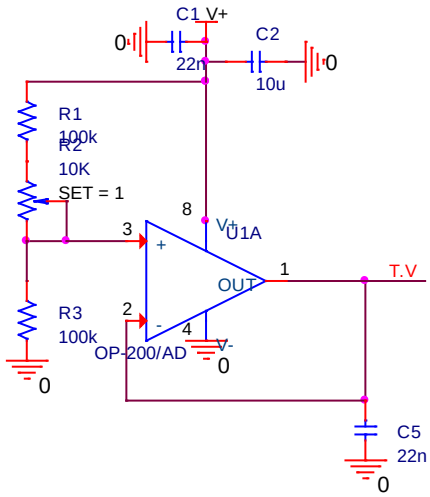


Figura 2.2.2. Esquemático del generador de 2.5 V.

El potenciómetro servirá para regular el offset e intentar que la tensión se mantenga a un nivel de 2.5 V a lo largo del circuito.

Este diseño con alimentación single supply requiere opamps especiales, que soporten este tipo de alimentación. En la actualidad hay multitud de estos integrados que son compatibles tanto con alimentaciones positivas y negativas como con una única alimentación. Encontrar un amplificador operacional que opere con alimentación de 0-5V no es difícil, sin embargo el objetivo es que la señal de audio tenga el mayor rango posible, y para aprovechar este rango, limitado normalmente por los opamps usados, se pasara a usar opamps llamados rail-to-rail. Estos amplificadores tienen la característica de que sus tensiones de salida son casi idénticas a las tensiones de alimentación, mientras que en los amplificadores más tradicionales con tensiones de alimentación de ±15V, las tensiones de salida suelen alcanzar alcanzan máximos de ±12V usualmente. El amplificador rail-to-rail elegido es el modelo OPA340, es un amplificador que puede operan con tensión de alimentación de +5V, sus características principales se exponen a continuación.

-Rail-to-Rail output: 1mV

-Ancho de banda: 5.5 MHz

-THD + NOISE: 0.0007 % (1KHz)

- Voltajes de operación: +2.7-5.5V

Como se puede observar, cumple con las necesidades perfectamente. El ancho de banda es muy superior al necesitado, tiene baja distorsión armónica, y puede alimentarse con únicamente 5V. El fabricante especifica que la salida puede alcanzar un máximo de un mV menos que la tensión de alimentación. Además de todo esto el fabricante recomienda este opamp para aplicaciones de audio entre otras.

Una vez que se ha aclarado todo lo relacionado a la alimentación del circuito así como otras características generales más relevantes, se pasa a explicar en orden cada una de las etapas que conforman la mesa de mezclas. Volviendo al diagrama de bloques mostrado al principio de

este apartado la primera etapa será el adaptador de impedancias, que se explicara junto al preamplificador, seguidas del ecualizador, y por último la etapa sumadora y el master.

Adaptador de impedancias y preamplificador.

La primera etapa es el adaptador de impedancias, es lógico que vaya en primer lugar, ya que de esta manera nos asegura la máxima transferencia de voltaje entre la señal de la fuente de audio y la señal entrante a la mesa. Es necesaria ya que se tiene diferentes tipos de impedancias de entrada, micrófonos dinámicos, micrófonos de condensador, entradas de línea…todas ellas tienen su impedancia característica, pueden variar desde los 600Ω (de un micrófono) y los 10KΩ aproximadamente (nivel de línea).

Es el circuito más simple de todo el diseño (mostrado en la figura 2.2.3), se basa en que la entrada de un amplificador operacional ideal tiene una impedancia infinita, y en que la salida tiene una impedancia prácticamente nula. Es el clásico circuito del amplificador operacional usado como buffer.

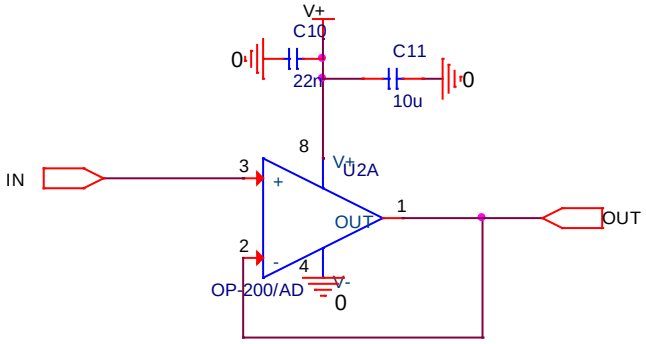


Figura 2.2.3. Esquemático del adaptador de impedancias.

En la figura 2.2.4 se muestra la respuesta en frecuencia del adaptador de impedancias. Se observa como prácticamente su respuesta es plana, hay un aumento hacia los 5KHz pero es despreciable ya que es una subida de unos 6µV.

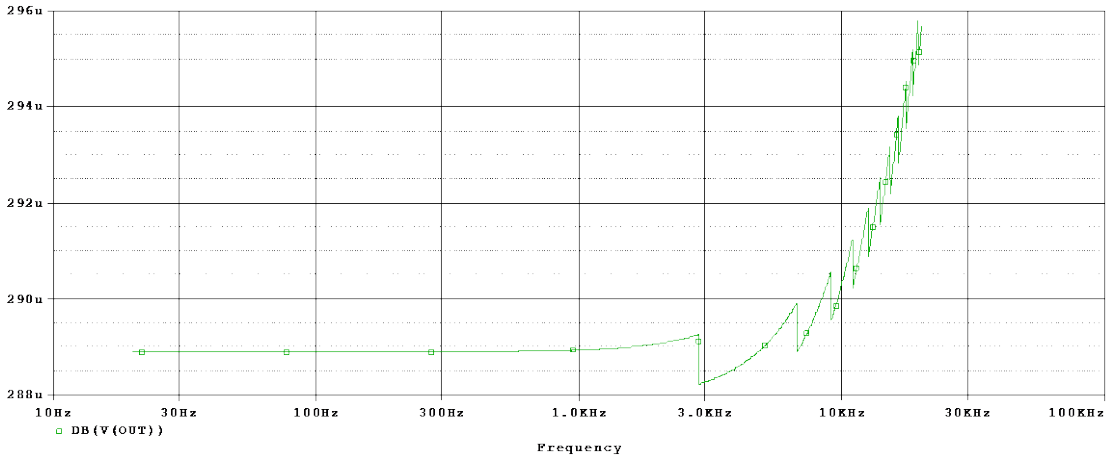


Figura 2.2.4. Respuesta en frecuencia del adaptador de impedancias

Conectada directamente con la salida del adaptador de impedancias se encuentra la etapa preamplificadora.

El preamplificador es la segunda etapa, tiene como función aumentar la amplitud de la señal con la mínima distorsión posible. La señal que procede de un micrófono es muy débil, menor que 1mV, es necesario aumentar la amplitud de esta señal para el posterior procesado, se aumentara la relación señal ruido, y la señal llegará a un nivel audible. En el caso de entradas con nivel de línea la etapa preamplificadora se utiliza igualmente, aunque no haría falta tanta ganancia como para la entrada de micrófono, las señales con nivel de línea, suelen estar en un rango audible.

Existe gran diversidad de preamplificadores, pasivos, activos, con válvulas de vacío, de bajo ruido. Los preamplificadores utilizan por lo general electrónica discreta, es decir se basan en transistores, acompañados por otros elementos pasivos. Los preamplificadores con válvulas de vacío son más caros y difíciles de diseñar, la respuesta de una válvula de vacío es muy diferente a la de cualquier tipo de transistor o de opamp, y es esto lo que caracteriza cualquier circuito de audio diseñado a partir de válvulas de vacío. Otro tipo de preamplicadores activos usualmente están basados en diseños con amplificadores operacionales, estos circuitos son más amigables en cuanto a diseño y son baratos, y es por estas dos razones por lo que el preamplicador de la mesa de mezclas será de tipo activo.

Se trata de un circuito muy básico. La ganancia de salida viene dada por la división entre la suma de la resistencia y el potenciómetro en realimentación y la resistencia a la entrada de este. Destacar que no existe ganancia negativa, su función no es la de bajar el nivel de la señal, sino la de aumentarlo. Tiene una ganancia máxima de 60dB. El usuario es capaz de regularla mediante el potenciómetro.

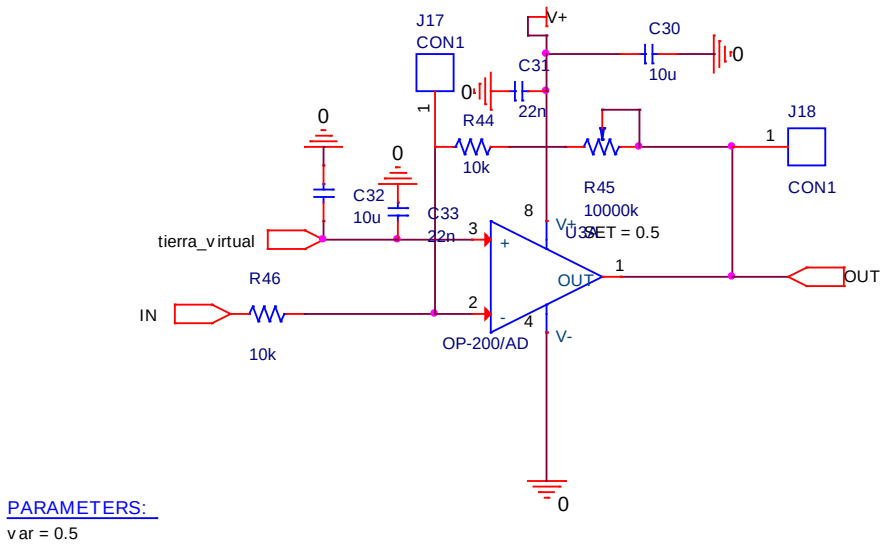


Figura 2.2.5. Esquemático del preamplificador.

Como se puede observar en la figura 2.2.5 en esta etapa se encuentran los conectores en la etapa de realimentación, que servirán para medir el offset. Se han colocado en esta etapa ya que será una de las que más afecte debido a su elevada ganancia.

En la figura 2.2.6 se observa la respuesta en frecuencia de la etapa preamplificadora. Se representan diferentes valores del potenciómetro, desde ganancia 0dB hasta 60dB. La respuesta se mantiene plana hasta los 50 dB de ganancia, al seguir aumentando la respuesta decae a partir de los 5kHz. Debido a la elevada ganancia y la limitación de los OPAMS.

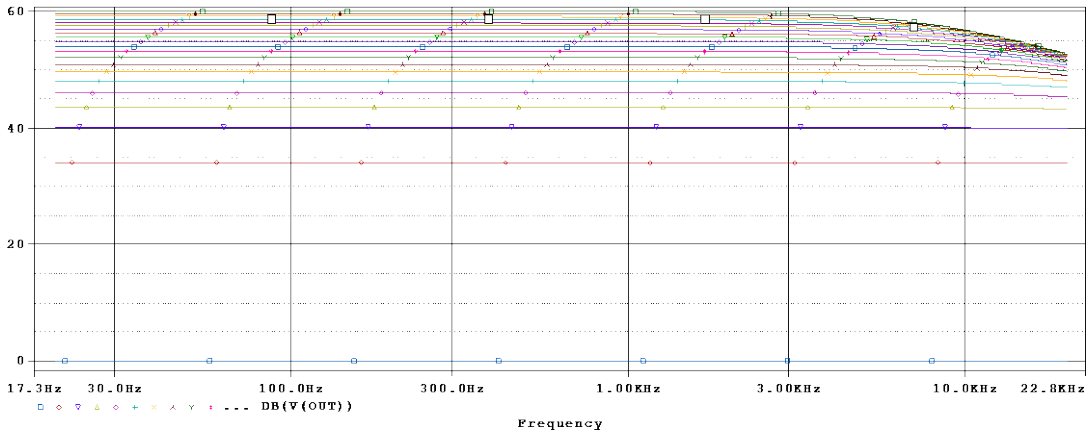


Figura 2.2.6. Respuesta en frecuencia de la etapa preamplificadora.

Ecualizador

Una vez que la señal de entrada está al nivel deseado, el siguiente paso es la ecualización, es una etapa importante en una mesa de mezclas, su función es la de resaltar matices, o simplemente abrir un hueco en el espectro para la posterior mezcla. Cuantas más opciones de ajuste tenga el ecualizador más versatilidad ofrece al usuario. Existen varios tipos de ecualizadores, tipo shelving, paramétricos, gráficos, activos, pasivos…La diferencia entre un ecualizador activo y pasivo reside principalmente en que el ecualizador pasivo es incapaz de aumentar la amplitud en un rango de frecuencias, solo puede atenuarlas, ya que no tiene alimentación, mientras que el activo es capaz de aumentar o disminuir el rango frecuencial deseado. Los ecualizadores paramétricos, reciben este nombre ya que permiten ajustar varios parámetros en cada banda del ecualizadores, tales como factor de calidad (factor Q), ancho de banda…Los ecualizadores gráficos constan de múltiples bandas, el más común es de bandas de octava que recorre 10 octavas, que son 30 Hz, 60 Hz, 125 Hz, 250 Hz, 500 Hz, 1 KHz, 2 KHz, 4 KHz, 8 KHz, 16 KHz. Los ecualizadores tipo shelving, son iguales que los gráficos, pero suelen constar de dos o tres bandas de ecualización.

En este caso se eligió un ecualizador shelving, que consta de tres bandas, graves, medios y agudos. Este fue elegido gracias a que cuenta con tres banda ajustables, es activo, y únicamente cuenta con un opamp. El filtro se muestra en la Cada uno de los tres potenciómetros regula una de estas tres bandas. El primer potenciómetro de 100k controla la banda de graves, el segundo potenciómetro de 100k controla la banda de medios, y el potenciómetro de 500k controla la banda de agudos.

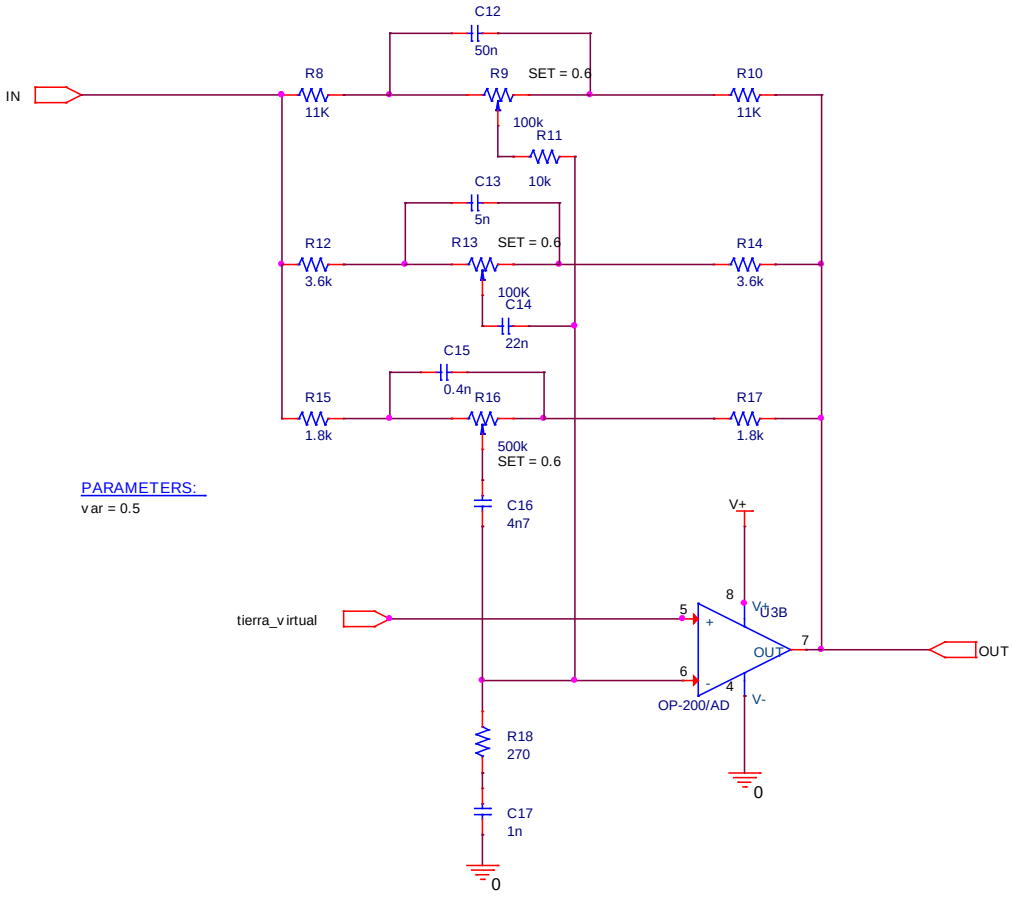


Figura 2.2.7. Esquemático del ecualizador.

En la figura 2.2.8 se observa la respuesta en frecuencia del control de tonos, para el barrido completo del valor de los potenciómetros. Se observan tres máximos cada uno en una banda de frecuencias, para la banda de graves el máximo se alcanza en los 20Hz, alcanzo ganancias de ±19dB, en la banda de medios el máximo se alcanza en los 1500 Hz siendo este ±16dB y para la banda de agudos el máximo de ±9dB se encuentra en 10KHz. La señal de entrada a de estar limitada a una frecuencia de 20 KHz ya que la función de transferencia del filtro a frecuencias más altas no responde de manera deseada.

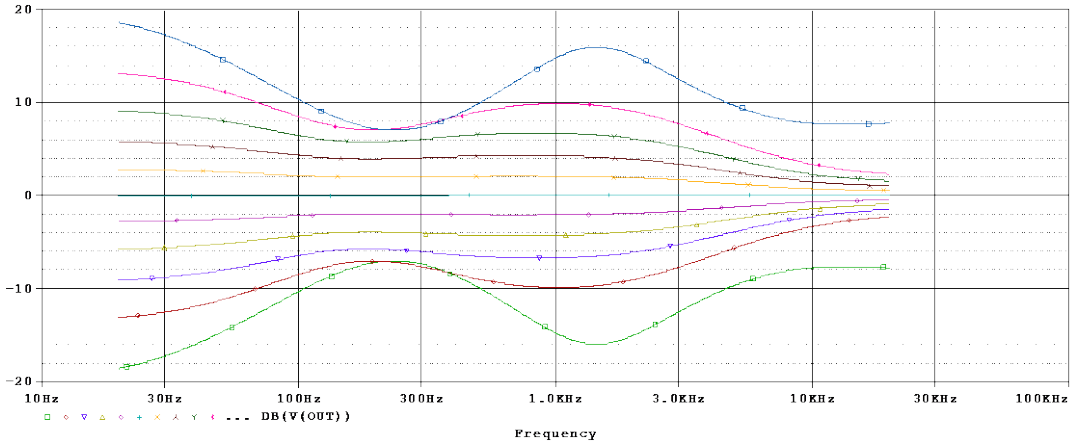


Figura 2.2.8. Respuesta en frecuencia del ecualizador.

Sumador

Es la siguiente etapa donde se juntan y se suman las señales procedentes de los tres canales independientes. La etapa sumadora recibe las tres señales provenientes de los tres diferentes canales, y el usuario ajusta a su gusto la cantidad de señal que quiere que se mezcle.

Al igual que en el caso del preamplificador, existen varios tipos de estos circuitos, pasivos, activos, con válvulas de vacío etc. Se volvió a escoger la opción de circuitos basados en opamps, por las razones expuestas con anterioridad.

Una etapa fundamental en la mesa mezclas, el circuito sumador. Es un circuito muy básico al cual le entran las señales de los diferentes canales, y son sumadas cada una de ellas con la ganancia que el usuario desee. El circuito está diseñado, para obtener una ganancia máxima de 6dB y una mínima de -20dB en cada canal de entrada. El esquemático se muestra en la figura 2.2.9.

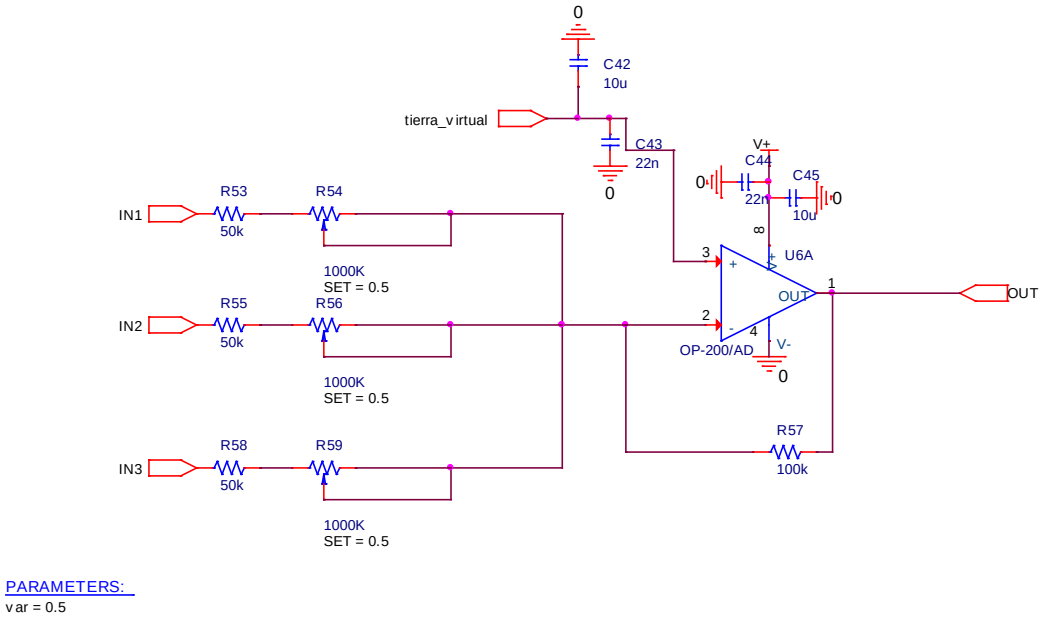


Figura 2.2.9. Esquemático del circuito sumador.

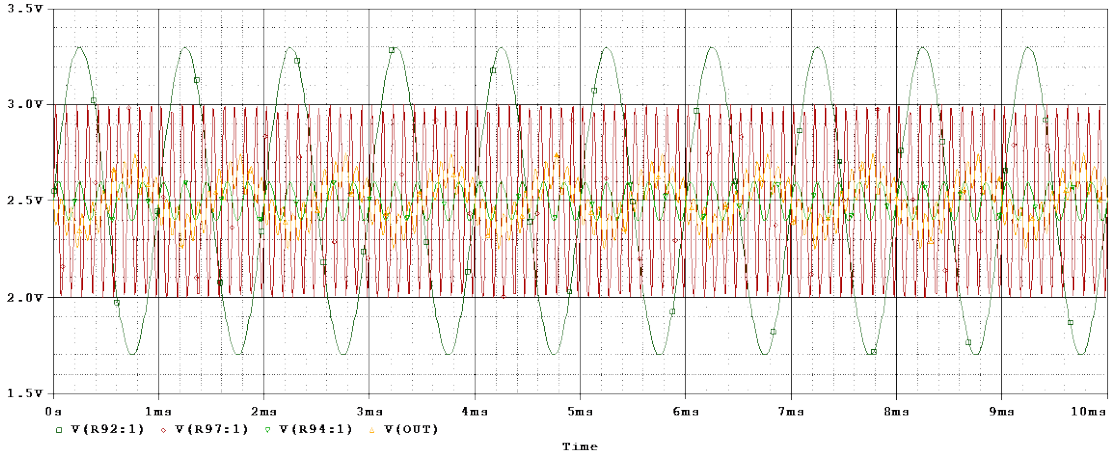


Figura 2.2.10. Suma de distintas señales, representada en el eje temporal.

En la figura 2.2.10 se muestra la simulación del circuito sumador, en el que entran tres señales sinusoidales, de 1KHz con 0.8 V de amplitud (color verde oscuro), 5KHz con 0.1V de amplitud (color verde claro) y 10KHz con 0.5V de amplitud (color rojo). La salida se muestra en color naranja.

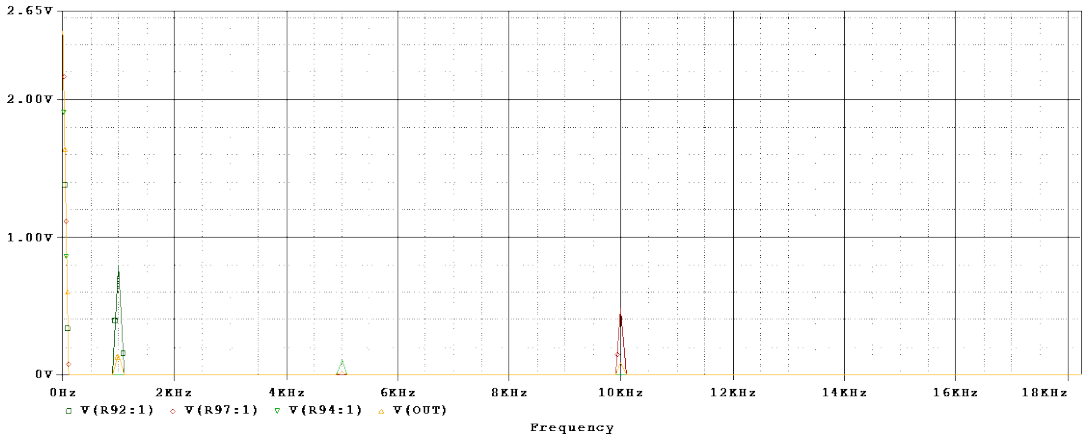


Figura 2.2.11. Suma de distintas señales, representada en el eje frecuencial.

En la figura 2.2.11 se muestra el espectro de las señales expuestas anteriormente, se observa cómo cada componente espectral tiene componente para la salida, con amplitud correspondiente a la ganancia con la que se simulo.

A continuación se muestra la salida del circuito sumador, la suma de las tres señales de entrada, tanto en el eje temporal como en el frecuencial.

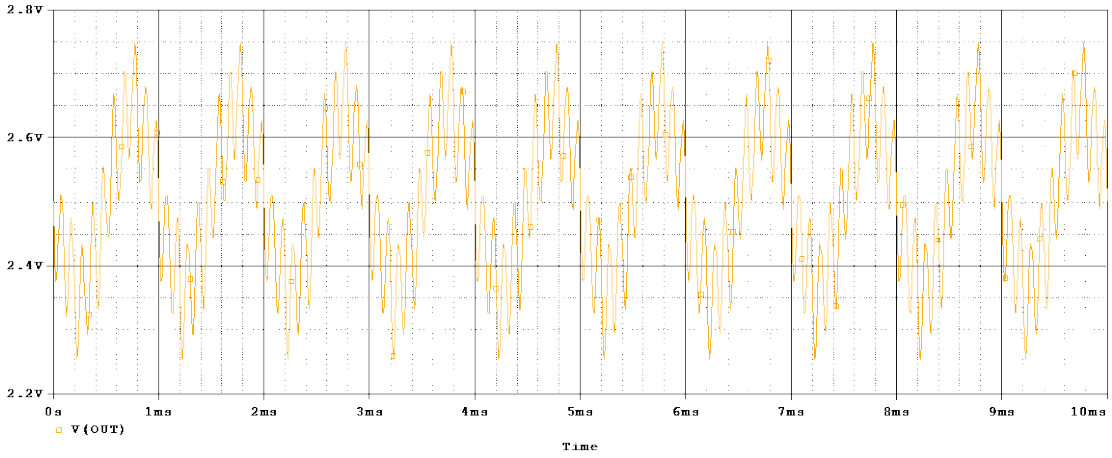


Figura 2.2.12. Suma de distintas señales, representada en el eje temporal.

En esta figura se aprecia mejor como la señal resultante adopta la forma de la señal de frecuencia más baja de entrada, un seno de 1KHz y como las demás señales al ser de frecuencia más altas quedan sumadas a esta.

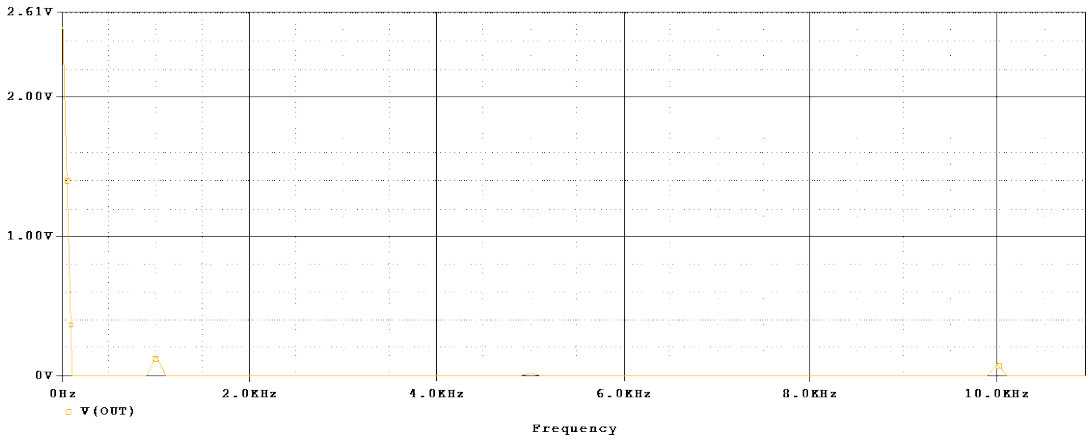


Figura 2.2.13. Suma de distintas señales, representada en el eje frecuencial.

Se aprecia en la figura 2.2.13, las componentes frecuenciales de la señal resultante de la suma de las tres señales de entrada. Coinciden con las componentes frecuenciales de las señales de entrada. Resaltar que se encuentra también una componente en DC de 2.5 V, la tierra analógica.

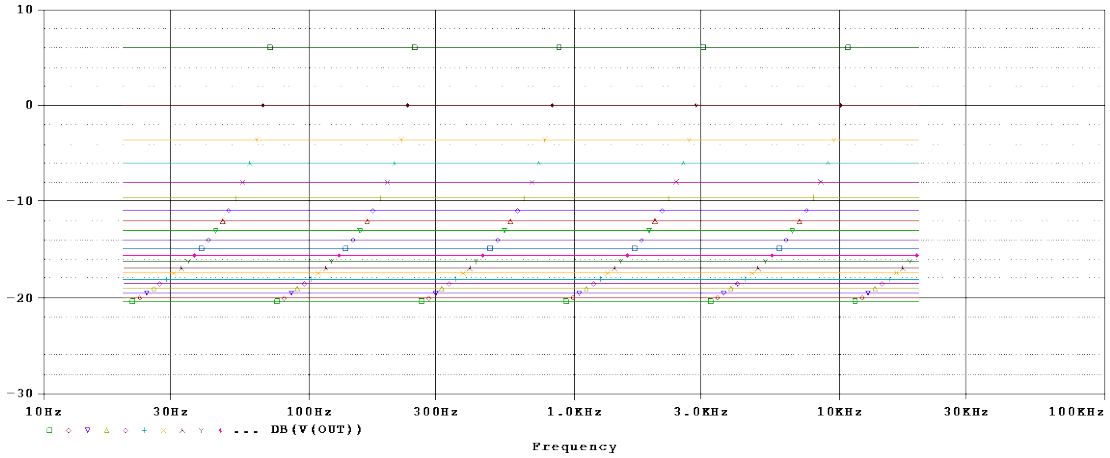


Figura 2.2.14. Respuesta en frecuencia de la etapa sumadora.

En la figura 2.2.14 se muestra la respuesta en frecuencia del circuito sumador, como se ha comentado antes la máxima ganancia que tiene es de 6dB, y con una ganancia mínima de -20 dB. Su respuesta es plana en todo el rango frecuencia audible.

Master

La etapa de master, constituye la última parte de la mesa. El master únicamente regula el nivel final de la señal resultante de la suma de los tres canales.

Su función es la de ajustar la amplitud de salida de la señal mezclada. Al igual que en los anteriores diseños su ganancia varía en un rango de -20 dB a +6 dB. De esta última etapa saldrá la señal hacia Arduino.

Al igual que en la etapa preamplificadora, se han colocado conectores para el ajuste del offset, debido a que es la etapa final.

Como se observa en la figura 2.2.15 se trata de un circuito muy básico, donde la ganancia de este viene ajustada por el potenciómetro.

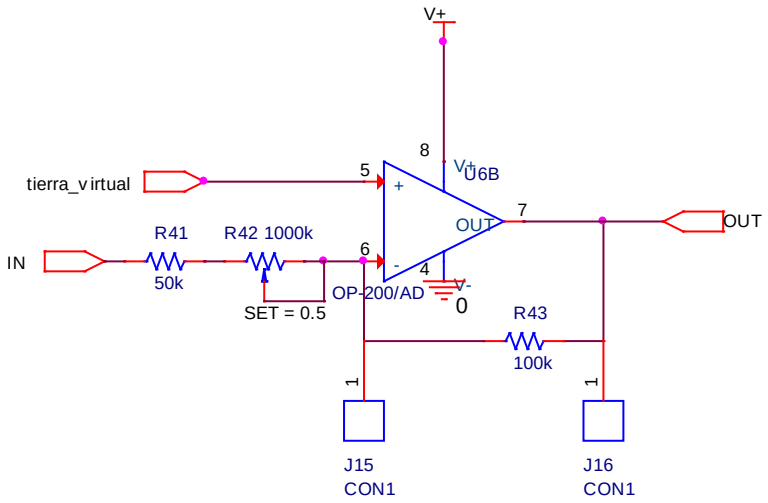


Figura 2.2.15. Esquemático del circuito master.

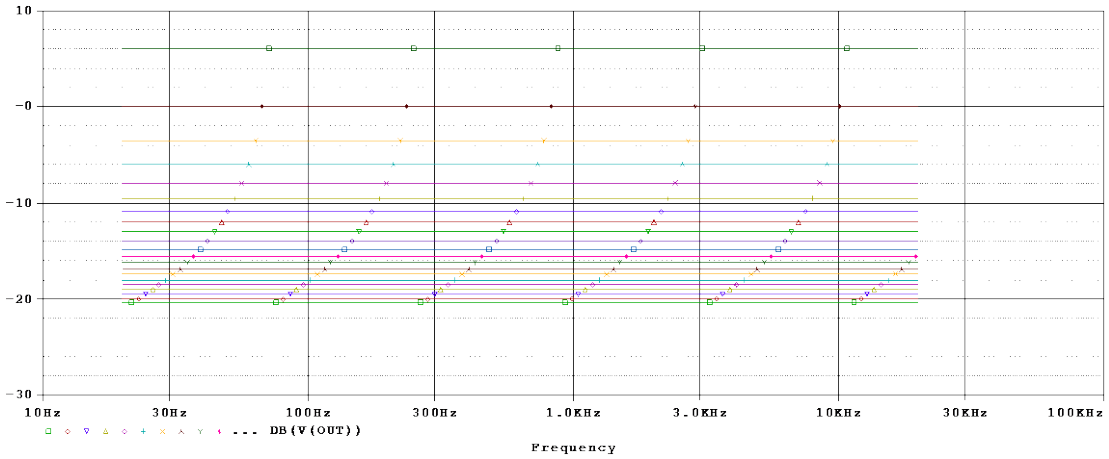


Figura 2.2.16. Respuesta en frecuencia del circuito master.

En la figura 2.2.16 se representa la respuesta en frecuencia del circuito master, observándose una mínima ganancia de -20 dB y una máxima de +6dB.

2.3 OTROS CIRCUITOS DESCARTADOS.

En esta parte se van a mostrar algunos de los circuitos descartados para la elaboración de la mesa de mezclas.

A continuación se representan tres de ellos, un ecualizador, un preamplificador y un circuito sumador.

Ecualizador tipo Baxandall

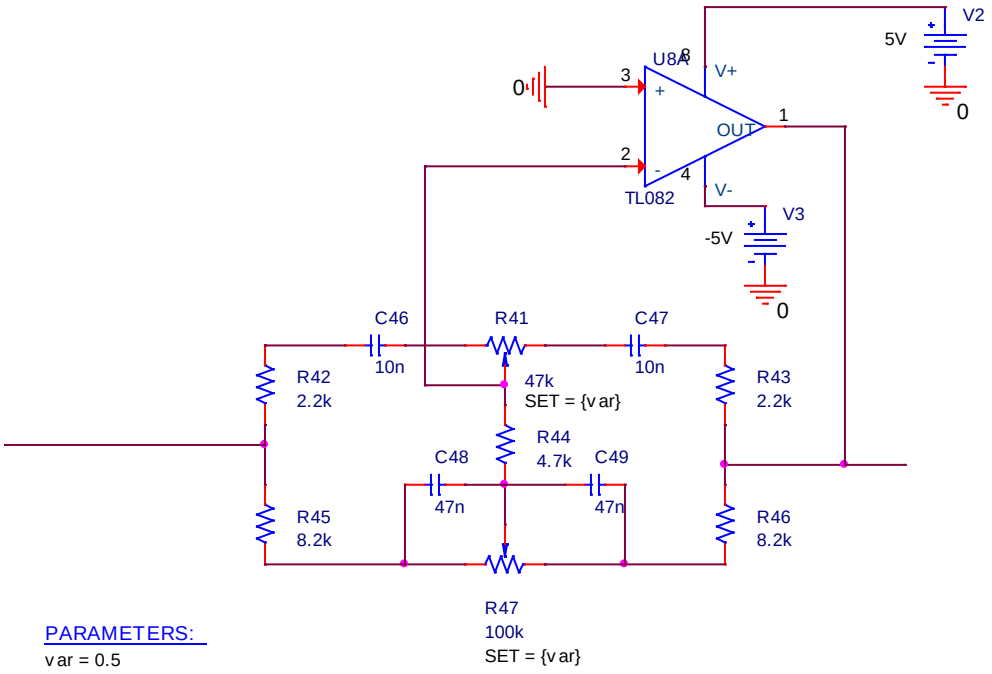


Figura 2.3.1. Ecualizador de dos bandas.

Este primer circuito fue descartado ya que únicamente cuenta con dos bandas de ecualización, mientras que con unos pocos componentes más se puede conseguir un ecualizador similar pero de tres bandas.

Circuito sumador

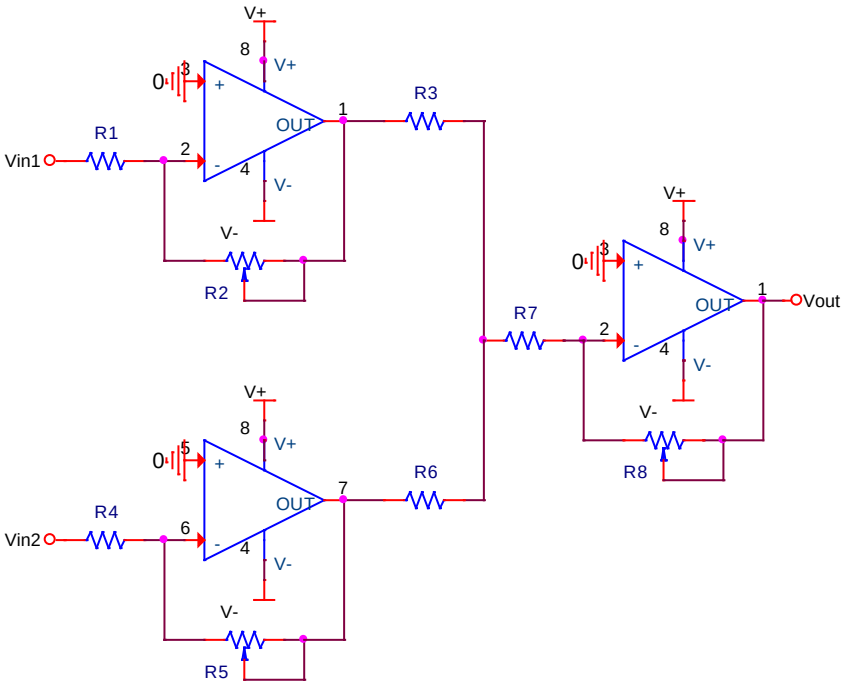


Figura 2.3.2. Esquemático de un circuito sumador.

Se trata del clásico circuito sumador básico que se puede encontrar en una mesa de mezclas de prestaciones básicas, sin embargo fue descartado debido al gran número utilizado de amplificadores operacionales, uno por cada canal, más uno a la salida. Se optó por concentrar el circuito sumador con un solo amplificador operacional.

Circuito preamplificador.

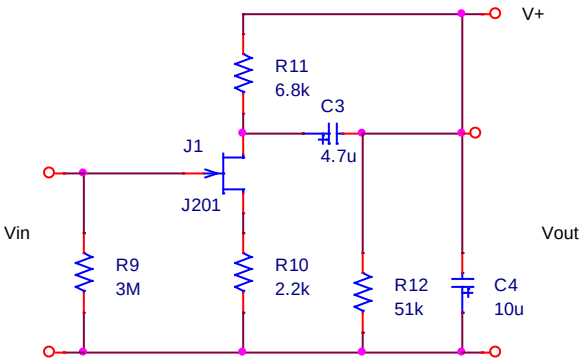


Figura 2.3.3. Esquemático de un circuito preamplificador.

Este circuito está basado en un transistor JFET especial para para aplicaciones de audio. Todo este tipo de circuitos preamplificadores basados en transistores fueron descartados, ya que actualmente se diseñan con integrados, lo cual facilita el diseño, además en términos de distorsión los basados en OPAM son mucho mejores que los de tipo discreto.

CAPÍTULO 3

DISEÑO DIGITAL

El diseño digital de la meza de mezclas está formada por un microcontrolador basado en Arduino. Una vez que las señales han sido procesadas y sumadas en la mesa de mezclas, la señal resultante entra en la parte digital para ser procesada y agregar varios efectos para ser retornada al master. El objetivo es usarel microcontrolador para realizar una conversión analógica digital para realizar un procesado digital, y programar efectos sonoros digitales, posteriormente se realizara la conversión digital analógica de esta señal procesada.

En este capítulo se introducirán las principales características del microcontrolador utilizado para este objeto (Arduino UNO), las principales características de los efectos digitales más comunes, y por último se pasara a explicar el funcionamiento del microcontrolador, así como se programaron los efectos.

3.1. INTRODUCCIÓN A ARDUINO Y SUS CARACTERÍSTICAS GENERALES

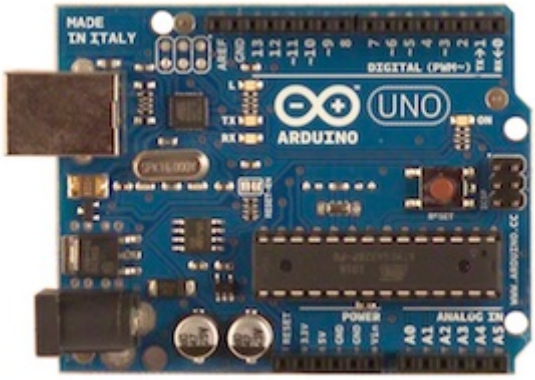
Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El hardware consiste en una placa con un microcontroladorAtmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (bootloader) que corre en la placa, pero básicamente es un conjunto de librería programadas en c y c++.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente. Al ser open-hardware, tanto su diseño como su distribución son libres. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.

A continuación se muestran alguna de las variantes que ofrece Arduino en términos de hardware.

Arduino Uno

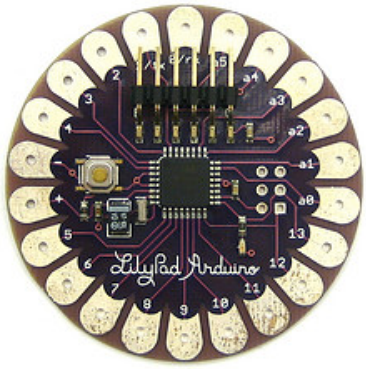


El Arduino Uno es una buena opción para comenzar a usar Arduino, proporciona una base sólida para los principiantes y tiene muchas de las opciones que se desea a medida que exploras la plataforma.

Arduino Nano

El Arduino Nano es rasgo por rasgo casi lo mismo que el Arduino Uno, pero es aproximadamente 1/3 del tamaño.

ArduinoLilypad

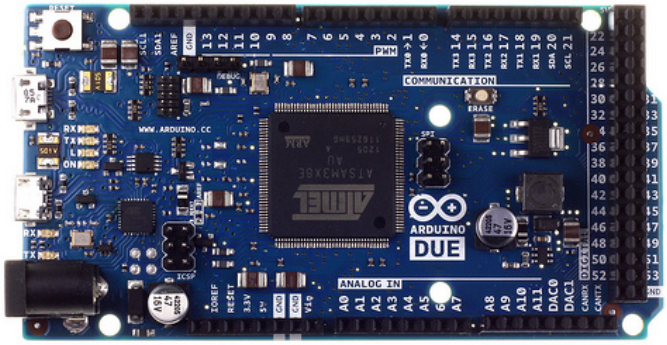


El Lilypad tiene un diseño único que se puede coser en la tela para proyectos que se puedan vestir o para arte.

Arduino Mega 2560

El Mega 2560 era hasta hace poco el Arduino más completo de todos, cuenta con un gran número de pines de E/S y con un controlador más potente que los anteriores Arduinos.

Arduino DUE



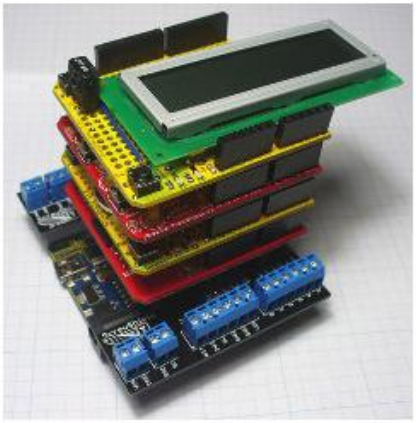
El Arduino DUEtiene más prestaciones que cualquier otro de la familia Arduino. Este es el Arduino más grande y mejor que puedes conseguir, pero puede que no se necesite tanto poder si los proyectos a realizar no lo requieren. También es un poco más caro.

Arduino Fio

El Arduino FIO está diseñado para aplicaciones inalámbricas. El usuario puede subir sus sketches con un cable FTDI o una placa adicional adaptadora Sparkfun. Además, si utiliza un adaptador de USB a XBee modificado, se pueden subir tus sketches de forma inalámbrica. La tarjeta viene sin conectores pre-montados, permitiendo el uso de diversos tipos de conectores o la soldadura directa de los cables.

Tiene conexiones para una batería de polímero de Litio e incluye un circuito de carga a través de USB. En el reverso de la placa tiene disponible un zócalo para módulos XBee.

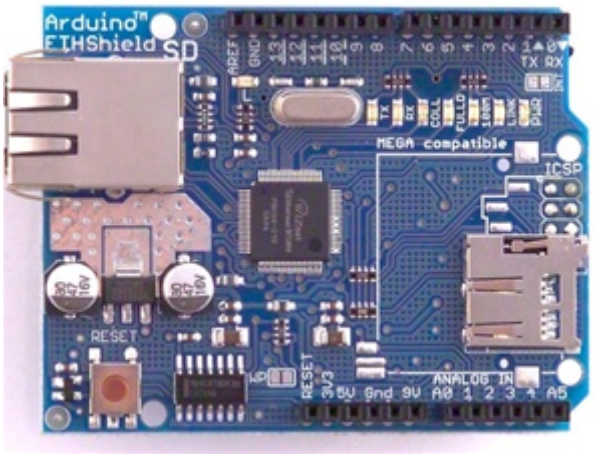
Accesorios de Arduino (Shields)



Un shield es una placa impresa que se pueden conectar en la parte superior de la placa Arduino para ampliar sus capacidades, pudiendo ser apilada una encima de la otra.

Los shields permiten añadir funcionalidad a tu Arduino al instante. Tienen pines que se colocan justo encima de tu Arduino y puedes aprovechar inmediatamente lo que sea que el shield pueda hacer. Puedes agregar múltiples shields al mismo tiempo.

Shield de Ethernet



Este es uno de los shields más populares, ya que permite a Arduino poder usar internet para fines de comunicación y control. El shield de Ethernet es uno de los más versátiles del mercado.

Shield de XBee

El shieldXBee hace que la comunicación inalámbrica de punto a punto sea fácil. Se puede utilizar este shield para conectar en red dos Arduinos o para crear una malla de red completa de Arduinos etc.

Shield de Motor

Arduino puede controlar motores y servos sin necesidad de un shield, pero el shield de motor aumenta esta capacidad. Puedes utilizar este shield para diseñar tu propio robot y controlarlo a distancia.

Entradas y salidas

Consta de 14 entradas digitales configurables entrada i/o salidas que operan a 5 voltios. Cada pin puede proporcionar o recibir como máximo 40 mA. Los pines 3, 5, 6, 8, 10 y 11 pueden proporcionar una salida PWM (Pulse WidthModulation). Si se conecta cualquier cosa a los pines 0 y 1, eso interferirá con la comunicación USB. Diecimila también tiene 6 entradas analógicas que proporcionan una resolución de 10 bits. Por defecto miden de 0 voltios (masa) hasta 5 voltios, aunque es posible cambiar el nivel más alto, utilizando el pin Aref y algún código de bajo nivel.

Especificaciones básicas de los microcontroladores ATMEL.

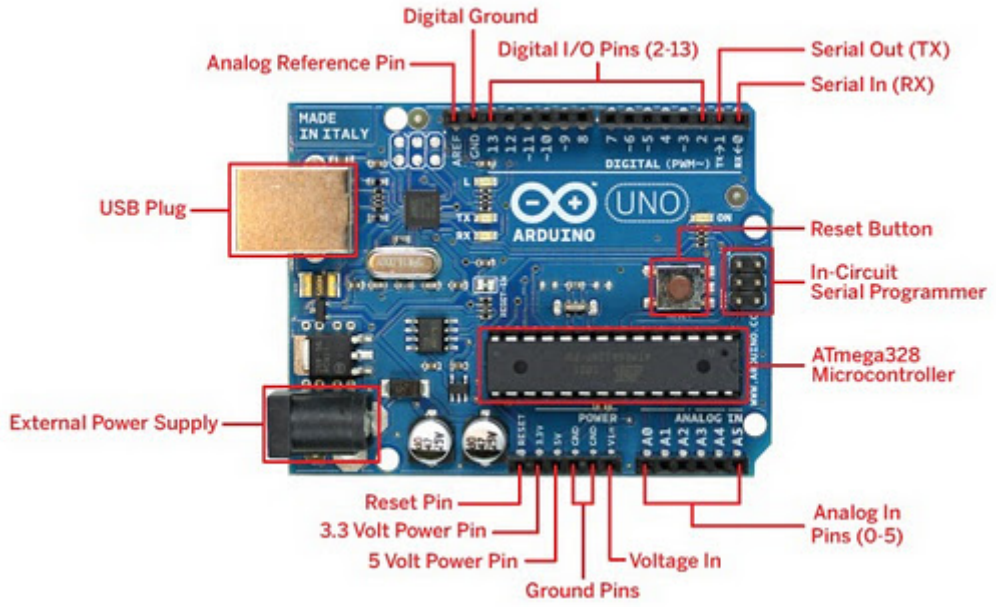
	Atmega168	Atmega328	Atmega1280
Voltaje operativo	5 V	5 V	5 V
Voltaje de entrada recomendado	7-12 V	7-12 V	7-12 V
Voltaje de entrada límite	6-20 V	6-20 V	6-20 V
Pines de entrada y salida digital	14 (6 proporcionan PWM)	14 (6 proporcionan PWM)	54 (14 proporcionan PWM)
Pines de entrada analógica	6	6	16
Intensidad de corriente	40 mA	40 mA	40 mA
Memoria Flash	16KB (2KB reservados para el bootloader)	32KB (2KB reservados para el bootloader)	128KB (4KB reservados para el bootloader)
SRAM	1 KB	2 KB	8 KB
EEPROM	512 bytes	1 KB	4 KB
Frecuencia de reloj	16 MHz	16 MHz	16 MHz

Eleccion de Arduino

Entre toda la gama de Arduino´s existentes se escogió el Arduino UNO. Fue elegido por sus prestaciones en comparación con los demás Arduinos. De toda la gran variedad son interesantes para este objetivo, el Arduino Nano, Arduino UNO y Arduino Mega, de estos se eligió a Arduino UNO ya que tiene un precio bajo y más prestaciones que Arduino Nano, aunque menos que Arduino Mega.

Arduino UNO (pines de entrada y salida)

A continuación se muestra los diferentes pines de entrada y salida, así como su distribución en Arduino UNO.



La programación en Arduino es básicamente en C, más básico que C++ de Flash. El programa de Arduino siempre tiene una estructura básica que se muestra a continuación:

Figura 3.1.1. Estructura de un programa en Arduino.

En la figura anterior se muestra la estructura básica de un programa en Arduino. El primer bloque (bloque 0) son los comentarios del programa, son opcionales. En bloque 1 se efectúan las declaraciones de variables que se utilizarán en el programa con posterioridad. El bloque 2 (voidsetup) hace referencia a la configuración inicial del programa, es donde se

efectuaría la manipulación de registros por ejemplo. Por último el bloque 3 (voidloop) es un blucle infinito que se repite constantemente.

El lenguaje cuenta con varias funciones ya predefinidas que facilitan la programación, tales como digitalRead,digitalWrite, analogRead…Al ser un software libre, existe una gran variedad de librerías para distintas aplicaciones.

En cuanto al aprendizaje de la programación en Arduino, existen libros que explican diferentes aplicaciones de este microcontrolador, sin embargo al ser un software libre la mayor parte de información se encuentra en internety puede accederse de forma gratuita.

3.2. EFECTOS DIGITALES

En general, se puede decir que un efecto de audio digital es simplemente un sistema que procesa una señal de entrada y da como resultado otra de salida. Pueden ser LTI o no. El objetivo normalmente es simular un efecto físico o producir alteraciones en el sonido que sean acústicamente interesantes.

A continuación se describirán varios efectos digitales, tanto su definición como su implementación.

ECO

El eco es un efecto que simula el rebote del sonido en una pared. Se supone que el oyente está en el mismo punto físico que la fuente de sonido. El sonido viaja hasta la pared, en donde sufre una atenuación, y vuelve hasta el oyente.

En otras palabras: en un instante de tiempo, lo que escucha el oyente, es decir, la salida y[n] del sistema será la suma de: 1) La muestra de entrada en ese instante x[n], ya suponemos que el oyente está en el mismo sitio que la fuente. 2)Más una muestra pasada de la señal, con un cierto retardo debido al tiempo de propagación, y una atenuación causada tanto por la distancia como por la absorción de la pared.

La ecuación que describe al eco queda entonces:

$$\mathbf{y[n]= x[n]+ ax[n - D]} \qquad \mathbf{(3.2.1)}$$

La ecuación 3.2.1, describe el efecto del eco simple, donde x[n] sería la señal emitida por la fuente, y ax[n − D] seria la señal x[n] atenuada por un factor “a” y retrasada D muestras.

ECO MÚLTIPLE

El eco múltiple es una versión más realista de la reverberación existente en un recinto cerrado, aunque no llega a ser exactamente eso:

$$\mathbf{y[n]=x[n]+}\sum_{m=0}^{M-1} \boldsymbol{am\ x[n - Dm]} \qquad \mathbf{(3.2.2)}$$

En la ecuación 3.2.2 se describe el eco múltiple el cual consiste en superponer M ecos diferentes provenientes de M paredes, cada una con una distancia y atenuación propias. Es más parecido al sonido real de un recinto, aunque se eliminan todas las reflexiones posteriores, propias de una reverberación real.

VIBRATO

El vibrato es un efecto muy común. Se da de forma natural entre los instrumentos musicales acústicos, así como en la voz cantada. Consiste en la variación sinodal del pitch generado. Es un efecto que tiende a enriquecer el espectro del timbre producido, siempre que esté dentro de unos límites razonables.

Desde el punto de vista espectral se puede decir que el espectro se estira y se encoge de manera periódica. La clave de la implementación es recordar que comprimir una señal en el tiempo es lo mismo que expandirla en frecuencia y expandir una señal en el tiempo es lo mismo que comprimirla en frecuencia.

Y la implementación intuitiva es más o menos directa, se trata de coger la señal retardada, siendo dicho retardo variable y sinodal.

En los instantes temporales en los que el retardo va aumentando, se estará expandiendo la señal y, en consecuencia, el pitch disminuye. En los instantes temporales en los que el retardo va disminuyendo, se estará comprimiendo la señal y, en consecuencia, el pitch aumenta.

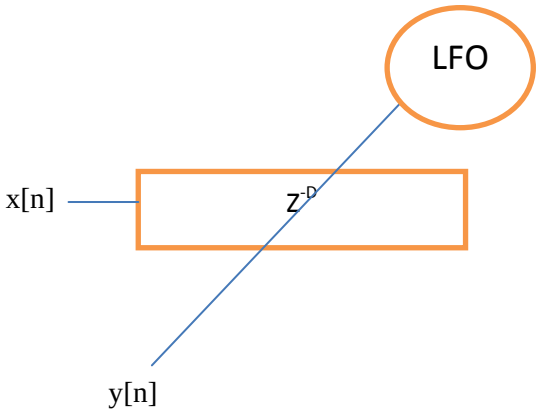


Figura 3.2.1. Diagrama de bloques de un vibrato.

En la figura 3.2.1 se muestra el diagrama de bloques del el efecto vibrato, donde x[n] representa la señal de entrada, y y[n] es la señal a la que se le aplicado el vibrato. La señal de entrada entraría en el sistema donde un LFO (LowFrecuencyOsclilator) controla el retraso aplicado a la señal. El bloque Z^{-D} hace de buffer donde se encuentran las muestras anteriores, y el LFO es el que decide que muestra enviar a la salida. La salida es una de las muestras que se encuentran en el bloque Z^{-D}.

FLANGER

El flanger es un efecto de audio muy utilizado en la música comercial desde varios años. Su propósito, como el de tantos otros efectos, es tan sólo distorsionar la señal original para que sea más agradable o atractiva. Consiste en la suma de la señal directa, con una copia suya retrasada y con vibrato.

La implementación es directa a partir del vibrato, con las siguientes anotaciones. Al contrario que en el vibrato, el offset del retardo no es importante en este caso sí que hay que definir el retardo entre la señal original y su copia retardada. El retardo suele ser del orden de 0 ms para el mínimo y 1 ó 2 ms para el máximo. La frecuencia del LFO también suele ser lenta, entre 0.1 y 1 Hz, típicamente. No obstante, existen flangers más rápidos. A continuación se muestra el digrama de bloques del flanger:

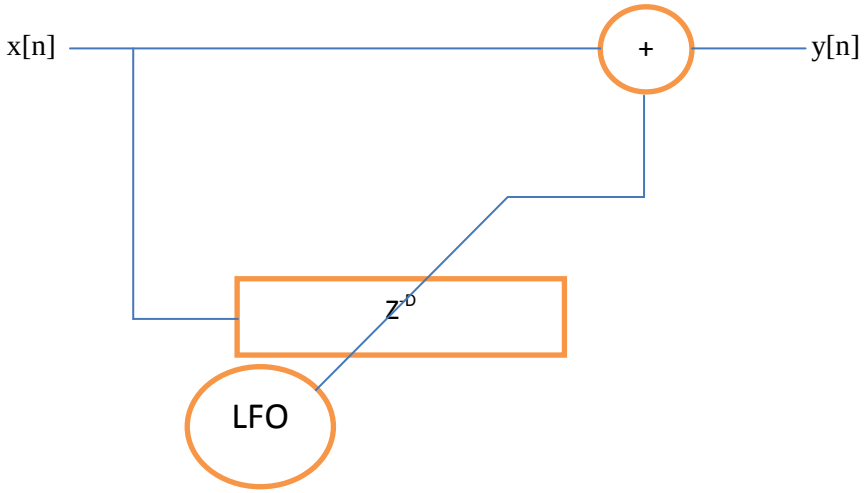


Figura 3.2.3. Diagrama de bloques del flanger.

En la anterior figura se observa como la implementación del flanger es directa a partir del vibrato, donde lo único que se debe hacer es sumar la señal original a esta señal con vibrato. Cabe destacar que cambian parámetros respecto al vibrato, como pueden ser la frecuencia del LFO , o el retraso.

PHASER

El phaser es otro efecto similar al flanging en cuanto a sus propiedades acústicas. En este caso se trata de coger una copia de la señal de entrada y filtrarla a través de un filtro notch. Posteriormente la señal filtrada se suma con la original. El filtro notch, es un filtro variable que se va moviendo lentamente arriba y abajo en frecuencia.

El resultado es que, por la zona en donde se mueve el notch, se producen realces y atenuaciones de ciertas frecuencias debido a las diferencias de fase entre las dos señales combinadas.

El movimiento del notch puede ser automático (mediante el uso de un LFO) o controlado externamente.

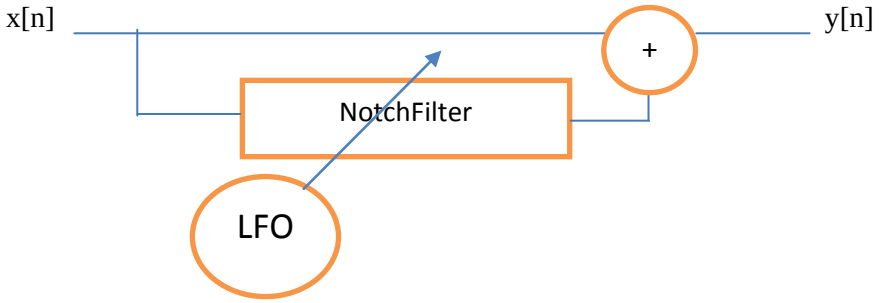


Figura 3.2.3. Diagrama de bloques del phaser.

En la figura 3.2.3 se muestra el diagrama de bloques de un phaser, en el cual la variación del filtro notch está controlada por un LFO. La señal de entrada x[n], pasa por este filtro y más tarde se suma esta señal filtrada con la original. La variación del filtro puede ser controlada por el usuario usando por ejemplo pedal de expresión.

WAH-WAH

El wah-wah es en cierta manera la versión opuesta al phaser. En lugar de filtrar la copia de la señal con un notch, se utiliza un filtro paso banda para realzar una cierta banda del espectro. Dicho filtro se mueve por el espectro igual que el phaser, pudiendo estar controlado por un LFO o por el propio usuario.

En el caso del wah-wah es más común que el usuario controle el propio filtro.

MODULACIÓN

La modulación ring es sencillamente la versión más simple de una modulación AM. A continuación se muestra la ecuación que describe este efecto.

$$y[n]=x[n]\cos(2\pi f_o n) \qquad (3.2.3)$$

En la ecuación 3.2.3 se observa como el efecto de modulación únicamente consiste en multiplicar la señal original x[n] por un coseno de la frecuencia f0 que elija el usuario.

Algunas consideraciones a tener en cuenta, desplaza el espectro, pero no de manera armónica, con lo que la distorsión musical es notable se suele utilizar para distorsionar la voz(en películas) u otros sonidos para crear efectos sonoros y también se puede emplear en música con ciertos instrumentos de síntesis, pero es necesario probar a ver cómo queda.

A continuación se muestra el diagrama de bloques que describe este efecto:

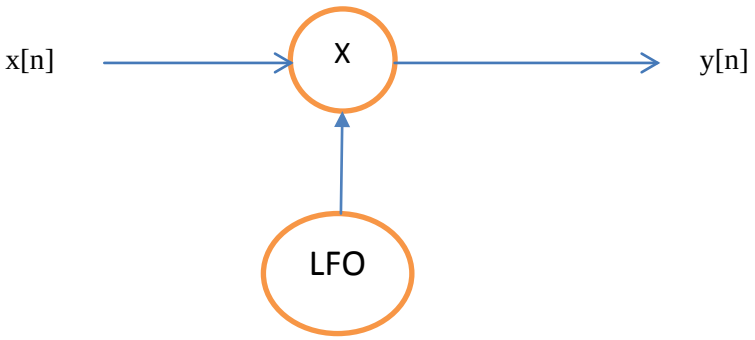


Figura 3.2.4. Diagrama de bloques de la modulación.

En la figura 3.2.4 se observa como la implementación de la modulación es muy sencilla, únicamente se trata de multiplicar a la señal de entrada x[n] por un LFO de la frecuencia que elija el usuario.

TRÉMOLO

El trémolo es otro efecto de modulación de amplitud, con la siguiente expresión:

$$y[n]=x[n](1+\mu\cos(2\pi f_o n)) \qquad (3.2.4)$$

En la ecuación anterior se define el tremolo como la multiplicación de la señal de entrada por un coseno alzado y atenuado, centrado en la unidad. Este provoca un rizado periódico en la envolvente de la señal. La frecuencia de oscilación es lenta, con lo que el oído más que

percibirlo como una modulación lo percibe como una variación rápida en la amplitud de la señal.

3.3. CRITERIO UTILIZADO PARA LA ELECCIÓN DE LOS EFECTOS DIGITALES A REALIZAR

Hay varios factores que influyen a la hora de seleccionar los efectos digitales a realizar, pero hay dos que cabe resaltar.

El primero es tener en cuenta las capacidades de Arduino, (únicamente cuenta con 2 kbyte de memoria RAM, y una velocidad de 16MHz) y el segundo se basa en el estudio de los efectos usados más comúnmente en el mundo musical, y observar las opciones que ofrece el mercado actual en este tipo de mesas de mezclas con efectos digitales.

Efectos digitales con filtrado en frecuencia, quedan totalmente descartados por la capacidad de Arduino para realizar este filtrado. EL chip Atmega 328 es incapaz de calcular los coeficientes de la ecuación en diferencias que describe a un filtro variante en el tiempo, y llevar a cabo el filtrado de la señal.

La multiplicación es un operador a evitar también. Para realizar esta operación el microcontrolador necesita de muchos ciclos de clock, lo que encarece el resultado final, al no poder mantener una frecuencia de muestreo lo suficientemente alta.

Es recomendable también evitar osciladores locales generados por el propio microcontrolador, ya que al igual que la multiplicación consume muchos recursos de este. La mejor opción para esto es generar una tabla con un oscilador y almacenarla en la memoria Flash, formando parte del sketch, ya que esta tabla no va a ser modificada.

Los efectos más usados en el mundo de la música y para estas aplicaciones son el reverb (con varios tipos de reverb), eco-delay, flanger, chours, phaser, pitch-shifter.Las mesas de mezclas que cuentan con efectos digitales que se pueden encontrar en la actualidad en el mercado, cuentan con efectos de Reverb(varios tipos como, Hall, Room ,Reverse..), efectos tipo Delay(EarlyReflection, Echo, Delay…), Chorus, FlangerPhaser, y efectos que afectan al pitch.Varios de estos efectos citados requieren de una modulación o de un filtrado, por lo que quedan descartados debido a su coste computacional. Efectos como el chours requieren de un gran coste computacional también, ya que requiere generar en cada iteración varios valores aleatorios. Nos quedaremos con efectos tipo reverb,eco-delay, flanger…

Por lo general los efectos viables para su realización con Arduino son efectos en el dominio temporal.

Siguiendo con el estudio previo a la programación de los efectos digitales, hay que destacar la capacidad de almacenamiento del chip Atmega328, consta con una memoria RAM de 2Kbyte. El conversor analógico digital puede alcanzar frecuencias de muestro suficientemente altas como para no perder información de la señal, sin embargo tiene una resolución de 8-bit. En el mejor de los casos se podría realizar una conversión analógica digital con una frecuencia de muestreo mayor de 40000Hz, y una resolución de 8-bit. Sin embargo con esta configuración se obtendrían 40000byte por segundo, y la memoria RAM del microcontrolador es de 2Kbyte. Para efectos en el dominio temporal es una memoria realmente escasa, y esto da lugar a dos posibles opciones, la primera de ellas sería aumentar la memoria

RAM del microcontrolador, para poder satisfacer las necesidades a frecuencias de muestreo suficientemente altas, y la segunda opción, que es la llevada a cabo en este caso, es la de bajar la frecuencia de muestro para compensar esa falta de memoria para el almacenamiento de las muestras. En el apartado 3.5 se detalla el problema y se muestran algunas soluciones llevadas a cabo, sin embargo el problema siguiente fue que los tiempo de volcado y recuperación de datos era muy lento para poder ser utilizado en tiempo real, por lo que esta posible solución fue descartada

3.4. CÓDIGO CON LOS EFECTOS DIGITALES

Como ya se ha comentado con anterioridad, Arduino se basa en un microcontrolador de la casa Atmel, en el caso de Arduino UNO se trata del ATMEGA328P. Arduino se ha hecho tan popular porque permite programar de manera muy sencilla, pudiendo hacer uso de las capacidades del microcontrolador muy fácilmente. Sin embargo se trata de un microcontrolador, y Arduino también permite al usuario un control completo sobre este microcontrolador, acceso a los registros, temporizadores, velocidades de clock etc.

Lo primero que se realizó fue una vista general acerca del microcontrolador, su estructura, especificaciones básicas etc.

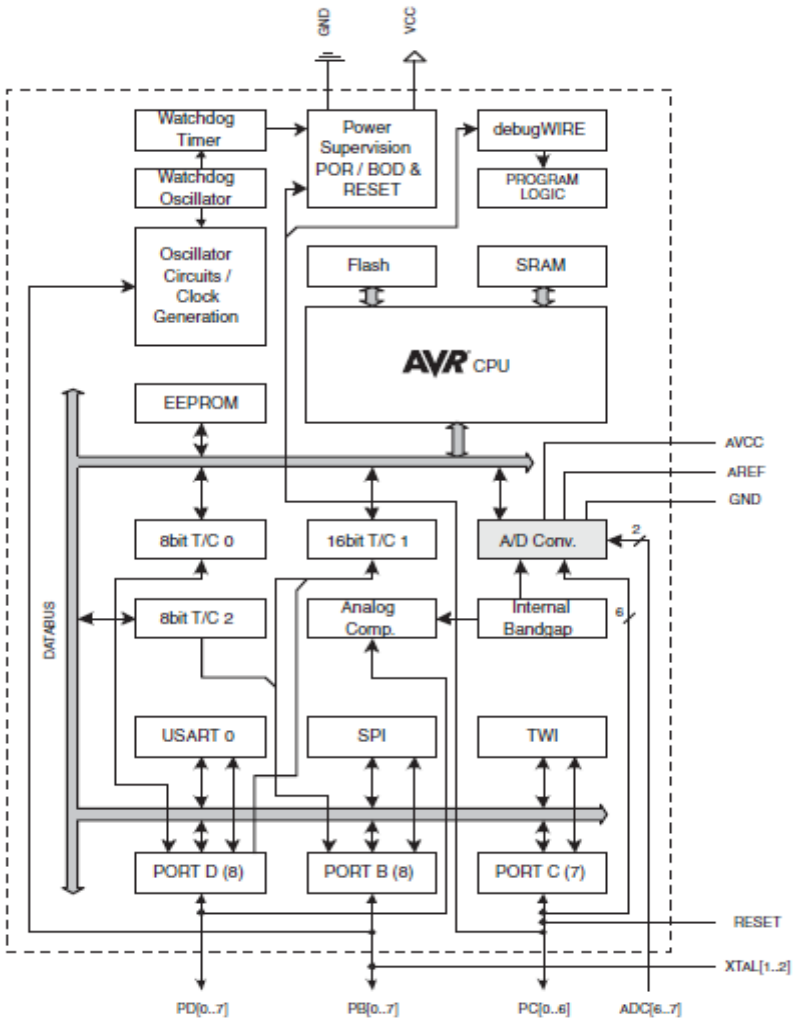


Figura 3.4.1. Estructura interna del microcontrolador Atmega328P

En la figura 3.4.1 se muestra la estructura interna del microcontrolador, donde cabe resaltar la existencia de tres temporizadores, un conversor A/D, un comparador analógico, osciladores…

El primer paso a la hora de programar los efectos digitales, fue separar este proceso en varias partes. La primera de ellas fue la conversión analógica digital, y la posterior conversión digital analógica. Lo siguiente fue lograr el almacenamiento en la memoria del microcontrolador de los datos obtenidos. Por último se programó los efectos digitales.

En primer lugar se va a proceder a explicar la manera en la que se realizó la conversión A/D y D/A.

La señal de entrada a Arduino es una señal analógica por lo que deberemos de definir un pin analógico como entrada. Para el proceso de conversión A/D, el chip Atmega cuenta con un conversor. Existen dos maneras de hacer uso de este conversor, la primera y más sencilla, es mediante la función "*analogRead*" que ofrece Arduino, sin embargo esta función está muy limitada y no se sabe que frecuencia de muestreo utiliza a primera vista, ni tampoco es posible definirla como el usuario crea conveniente, está orientada para manipular entradas de sensores de una manera directa y sencilla. La otra manera es más compleja pero a su vez permite que el usuario obtenga mejores resultados haciendo que el conversor se ajuste a sus necesidades, se trata de hacer uso de los registros del microcontrolador, de tal manera que se ajuste esta conversión a los requisitos.

A continuación se muestra la estructura de bloques del conversor.

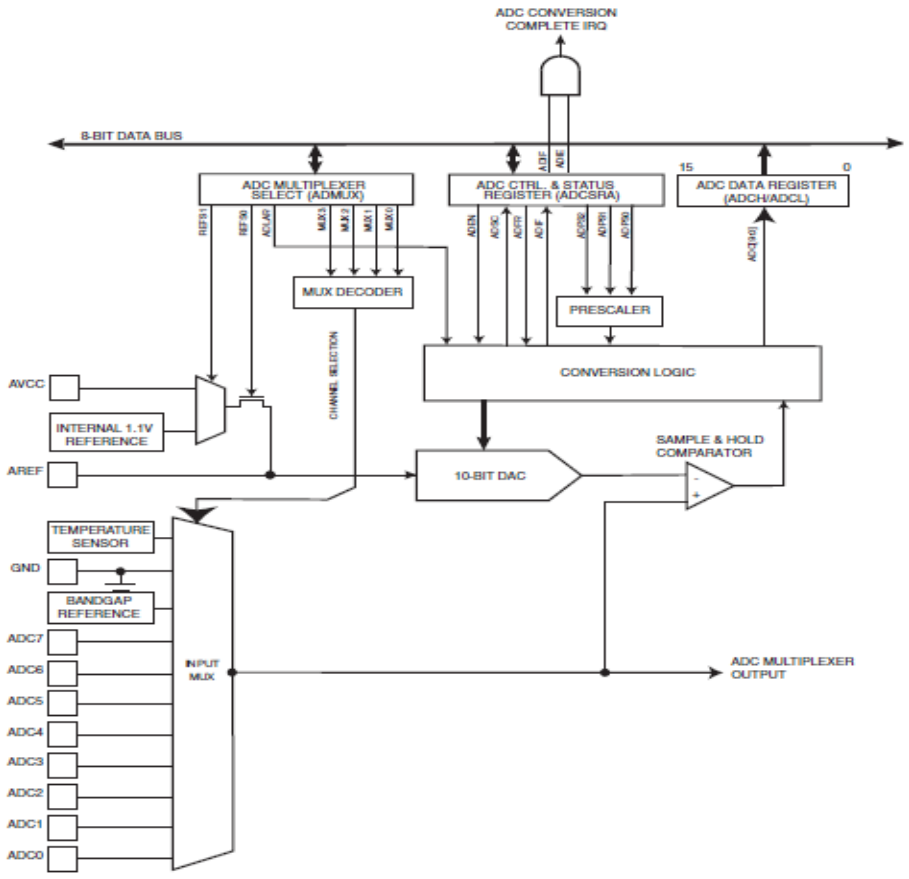


Figura 3.4.2. Diagrama de bloques del conversor A/D.

En la figura 3.4.2 se muestran los principales bloques del conversor. Cuanta con un multiplexor a la entrada de este, en el cual el usuario puede elegir en cualquier momento el canal de entrada al conversor. Cuanta con varias entradas (AREF, AVCC) que sirven al usuario para ajustar el rango en el que desea efectuar la conversión. Del BUS de datos de 8 bits en la parte superior se observa como el conversor recibe datos que le sirven para elegir la entrada del multiplexor, así como la frecuencia de muestreo deseada, y por supuesto enviar los datos de salida a los registros correspondientes.

El conversor analógico digital del chip cuenta con una resolución de 10 bits, un tiempo de conversión de 13-260µs, seis canales de entrada multiplexados etc. Aunque el microcontrolador cuente con una resolución de 10-bits el fabricante recomienda no usar esta resolución sino de 8-bit para frecuencias de muestro relativamente altas.

Existen dos modos de realizar la conversión analógica-digital, el Single Modeconversion y Auto-TriggeredMode. Este último realiza las conversiones seguidas una detrás de otra una vez que el usuario ha especificado la primera conversión. El modo Single ModeConversion efectúa una única vez la conversión. Se decidió usar el Single ModeConversion ya que de esta manera está más controlado el flujo de datos y el usuario puede ajustar la frecuencia de muestro más todavía de lo que el microcontrolador permite, además es necesario utilizar un temporizador para la conversión digital analógica como se explicara más adelante.

Para ajustar la frecuencia de muestreo el conversor cuenta con un *presclaer*. A continuación se muestra la estructura de este *prescaler*.

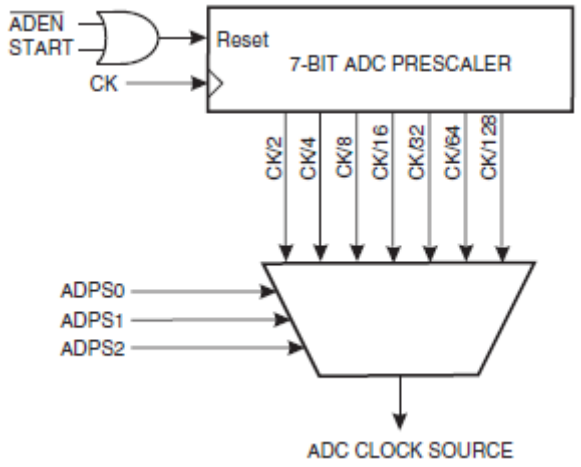


Figura 3.4.3. Estructura del prescaler.

En la figura 3.4.3 se observa la estructura del prescaler con el que cuenta el conversor, su función es la de dividir la señal de clock, proveniente del oscilador interno, por un factor que permite ajustar la frecuencia de muestro a la deseada. El factor de prescaler se escribe en los registros ADPS0-2.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Figura 3.4.4. Valores posibles del prescaler.

En la anterior figura se muestra los diferentes valores que puede tomar este *prescaler*.

Al haber elegido el Single Mode Conversión, se programa la conversión mediante interrupciones producidas por un temporizador, aunque el microcontroladorAtmega permite generar una interrupción cada vez que se efectúa una conversión, pero de esta manera solo se podría almacenar los datos y no sería posible realizar la conversión digital analógica. El paso siguiente es estudiar los tres temporizadores que ofrece el chip Atmega, y elegir el más adecuado.

Es importante la elección del temporizador ya que Arduino no cuenta con un conversor D/A, y la conversión se realiza mediante modulación PWM, y esta conversión corre a cargo del temporizador.

De los tres temporizadores/contadores que cuenta Arduino, es este caso se utiliza el Timer/Counter2. Es necesario que cuente con un modo de operación llamado Fast PWM, que es el que más adelante hará de conversor digital-analógico, sin embargo este modo está presente en los tres temporizadores, el temporizador uno se descartó ya que es de 16 bits, y no es necesario para este caso. Se decidió utilizar el Timer/Counter2 por comodidad de los pines correspondientes.

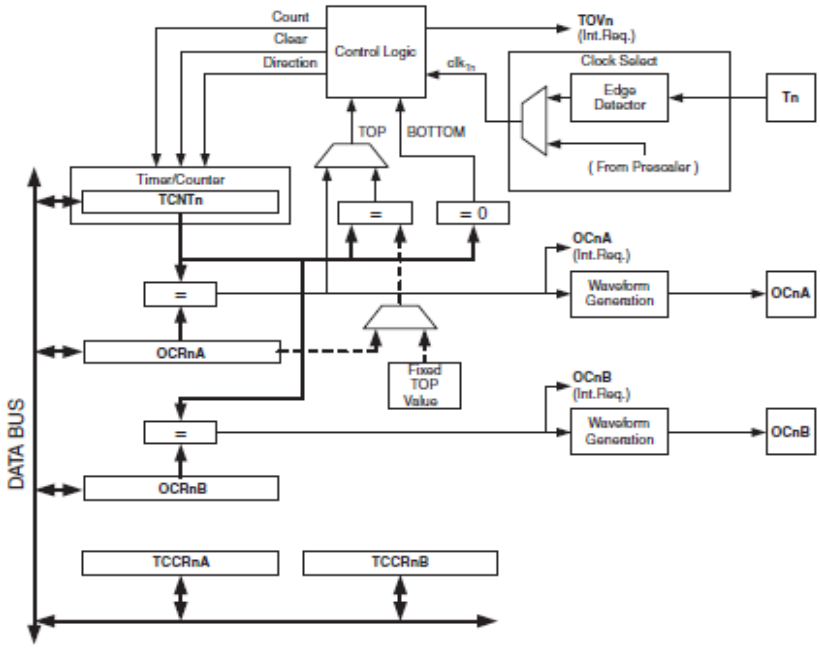


Figura 3.4.5. Esquema del Timer/Counter2.

En la figura 3.4.5 se muestra el diagrama de bloques del Timer2, a continuación se va a explicar algunos de estos bloques usados.

Todos los temporizadores tienen varios modos de operación, en general los modos de este temporizador son comunes a los demás.

En primer lugar el Normal Mode, este modo sirve únicamente para generar interrupciones, toma valores desde 0x00 hasta 0xFF donde es generada una interrupción. El siguiente modo es el Clear Timeron Compare Match, este modo se utiliza para manipular la resolución del contador. Otro modo es el PhaseCorrect PWM Mode, en este modo el contador incrementa hasta 0xFF, y en vez de volver a incrementar hasta 0xFF decrementa a partir de este valor hasta 0x00, Atmega recomienda este modo para aplicación de control de motores.

El ultimo modo ofrecido por Atmega es el Fast PWM mode, que es el que se utilizará en este caso. Este modo ofrece una generación de onda con PWM a alta frecuencia, el contador incremeta hasta 0xFF y vuelve a 0XFF. La salida es generada comparando el valor en el registro OCR2A o OCR2B con el valor del TCNTn, que es el valor que toma el contador de 0x00 a 0xFF, y según sea mayor o menor que este valor se escribe un 1 o un 0 en el OCnA o OCnB, existen dos modos, el invertido y el no invertido, la única diferencia es que uno escribe un cero si es mayor el valor del registro y el otro escribe un uno.

A continuación se muestra como configurar los parámetros descritos anteriormente, así como la velocidad del contador mediante los prescaler.

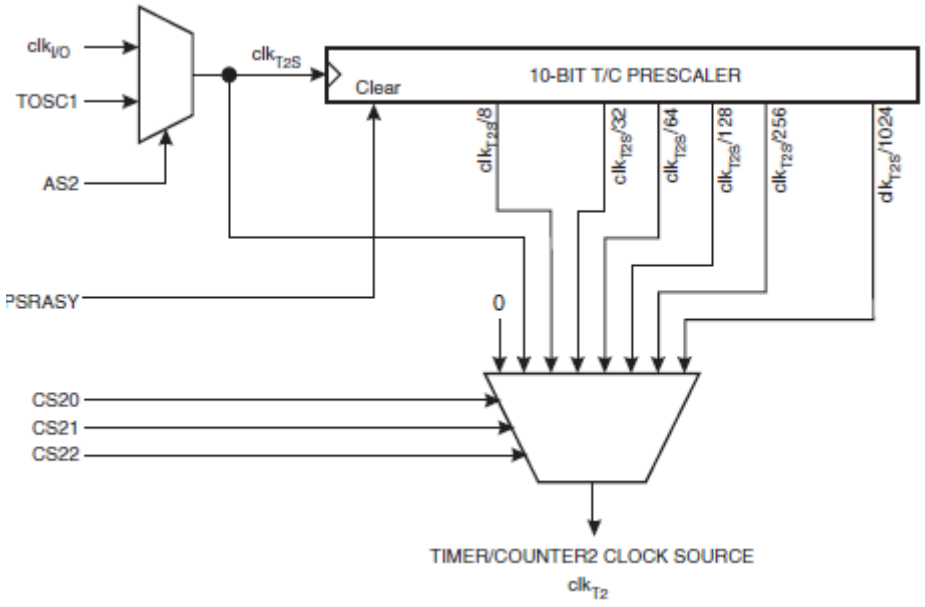


Figura 3.4.6. Diagrama de bloques del prescaler para el Timer2.

En la figura anterior se muestra el esquema de cómo funciona el prescaler para el timer2, a simple vista se observa como los bits CS22-0 son los que seleccionan el prescaler deseado. En este caso conviene fijar un prescaler no muy alto ya que la frecuencia de muestreo vendrá marcada por el conversor, y si se ajusta el timer a un valor muy alto es posible que se produzca antes la interrupción que la conversión analógica digital. Se optó por no utilizar ningún prescaler y de esta manera la frecuencia con la que se produce cada interrupción viene

dada por la división entre la frecuencia de clock del microcontrolador entre las 256 veces que cuenta el contador, quedando así 62745 Hz.

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk_{T2S}/(No prescaling)
0	1	0	clk_{T2S}/8 (From prescaler)
0	1	1	clk_{T2S}/32 (From prescaler)
1	0	0	clk_{T2S}/64 (From prescaler)
1	0	1	clk_{T2S}/128 (From prescaler)
1	1	0	clk_{T2S}/256 (From prescaler)
1	1	1	clk_{T2S}/1024 (From prescaler)

Figura 3.4.7. Valores del prescaler del timer2.

En la figura anterior se muestra los diferentes valores que puede tomar el prescaler del timer2 según lo que el usuario escriba en los bits CS22-0.

Estos bits se encuentran en el registro TCCR2B.

Bit	7	6	5	4	3	2	1	0	
(0xB1)	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 3.4.8. Registro TCCR2B.

En la figura anterior se muestra los bits en el registro TCCR2B. Los dos primeros bits, los FOC2A-B sirven para el modo normal, sin generación PWM. El bit WGM22 se explica más adelante, y por último los bits CS22-0, que sirven para fijar el prescaler del timer2.

A continuación se explica el registro TCCR2A, donde es posible seleccionar el modo deseado de operación del timer2, asi como sus características.

Bit	7	6	5	4	3	2	1	0	
(0xB0)	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 3.4.9. Registro TCCR2A.

En la figura 3.4.9 se muestra los bits que contiene el registro TCCR2A. En primer lugar están los bits COM2A1-0, que según el modo elegido afectan de manera diferente, únicamente se explicara los valores para el modo Fast PWM.

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	WGM22 = 0: Normal Port Operation, OC0A Disconnected. WGM22 = 1: Toggle OC2A on Compare Match.
1	0	Clear OC2A on Compare Match, set OC2A at BOTTOM, (non-inverting mode).
1	1	Set OC2A on Compare Match, clear OC2A at BOTTOM, (inverting mode).

Figura 3.4.10. Valores para COM2A1-0.

Estos valores controlan el comportamiento del pin correspondiente al OC2A, aunque sirven de igual manera para el OC2B, en ese caso deberían de ser configurados en los bits, COM2B1-0. En este caso se eligió el non-inverting mode.

Los dos últimos bits, WGM21-0, así como el bit WGM22 del registro TCCR2B servirán para elegir el modo de funcionamiento de timer2.

Mode	WGM22	WGM21	WGM20	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1)(2)]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Figura 3.4.11. Modos de funcionamiento del Timer2.

En la figura 3.4.11 se muestran todos los posibles modos de funcionamiento del timer2, en este caso se eligió el modo 3.

Todos estos parámetros funcionan de igual manera para los tres timers, aunque cabe resaltar que el timer1 es un contador con 16 bits, y que permite más modos de funcionamiento.

Una vez elegido el temporizador, el prescaler, y el modo de funcionamiento, el siguiente paso es estudiar los vectores de interrupción generados por este temporizador, para poder crear la interrupción y programar las órdenes deseadas.

Todavía queda por configurar varios parámetros del conversor A/D. El primero de ellos es el del multiplexor, el microcontrolador necesita saber qué entrada es la que el usuario desea que entre al conversor A/D. Esto se consigue con la escritura de bits en el registro ADMUX.

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 3.4.12. Bits en el registro ADMUX

En la figura 3.4.12 se muestra el registro ADMUX donde los bits MUX3-0 sirven para elegir el canal de entrada para el conversor analógico digital.

MUX3...0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8[1]
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

Figura 3.4.13. Selección de canales de entrada.

En la figura 3.4.13 se muestran los diferentes valores que se le pueden dar a estos cuatro bits.

El registro ADMUX sirve al usuario para controlar el voltaje de referencia del conversor, así como dos variantes para la escritura de la salida de los datos del conversor.

En cuanto al voltaje de referencia para el conversor, este será modificado con los bits RESF01-0. Básicamente sirve para ajustar el rango de valores de la conversión.

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

Figura 3.4.14. Valores para los bits RESF1-0.

En la figura anterior se muestran los valores que pueden tomar los bits del registro ADMUX, RESF1-0. Arduino cuenta con un pin llamado AREF, sirve al usuario para ajustar el rango de la conversión, este pin marcará el máximo voltaje para la conversión.

Para almacenar los bits de salida del conversor Arduino ofrece dos opciones de guardado. Cuenta con dos registros específicos para almacenar estos datos, pero el usuario es capaz de almacenarlos según él quiera de dos maneras diferentes.

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Figura 3.4.15. Registros ADCL y ADCH.

Según se escriba en el bit ADLAR, en el registro ADMUX los bits de salida del conversor se almacenan de diferente manera, tal y como muestra la figura XX.

Dentro de cada interrupción se le especificara que empieza una nueva conversión digital, escribiendo en el registro ADCSRA.

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 3.4.16. Bits en el registro ADCSRA.

En la figura anterior se muestra el registro ADCSRA, el primer bit ADEN sirve para activar o desactivar el conversor, escribiendo un “1” o un “0”. El bit ADSC sirve para realizar una nueva conversión poniendo este bit a uno. El bit ADATE sirve para activar el modo Auto Trigger. ADIE es el bit que nos permite controlar las interrupción es generadas al acabar una conversión. Los bits ADPS0-2 nos sirven para ajustar el *prescaler* del conversor.

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Figura 3.4.17. Valores para el prescaler del divisor

La forma de mandar la salida después del conversor analógico-digital, y la posterior conversión digital analógica mediante PWM, es escribiendo el bit de salida que deseamos en el registro OCRnA, o OCR2B en este caso.

Una vez que se ha adecuado el temporizador y en conversor analógico digital, el siguiente paso es programar los efectos digitales, en el Anexo I se muestra el código.

Se han programado tres efectos digitales, de los cuales es controlado un parámetro de estos por medio de un potenciómetro, a este potenciómetro irán conectados 5V, y tierra, mientras que la patilla variable de este irá conectada a un pin de Arduino, que efectuara la conversión analógico digital, y mediante este parámetro se controlaran los efectos.

Se han elegido los siguientes efectos digitales:

-Delay: Se controla la cantidad de retraso.

-Vibrato: Se controla la mezcla con la señal original.

-Flanger: Se controla la mezcla con la señal.

Resaltar que en este código se ha hecho uso de una librería existente para escribir en la memoria Flash de Arduino, y más tarde acceder a ella.

El programa consta de cuatro partes principales

La primera de ellas es la definición de todas las variables utilizadas, así como la inclusión de las librerías necesarias.

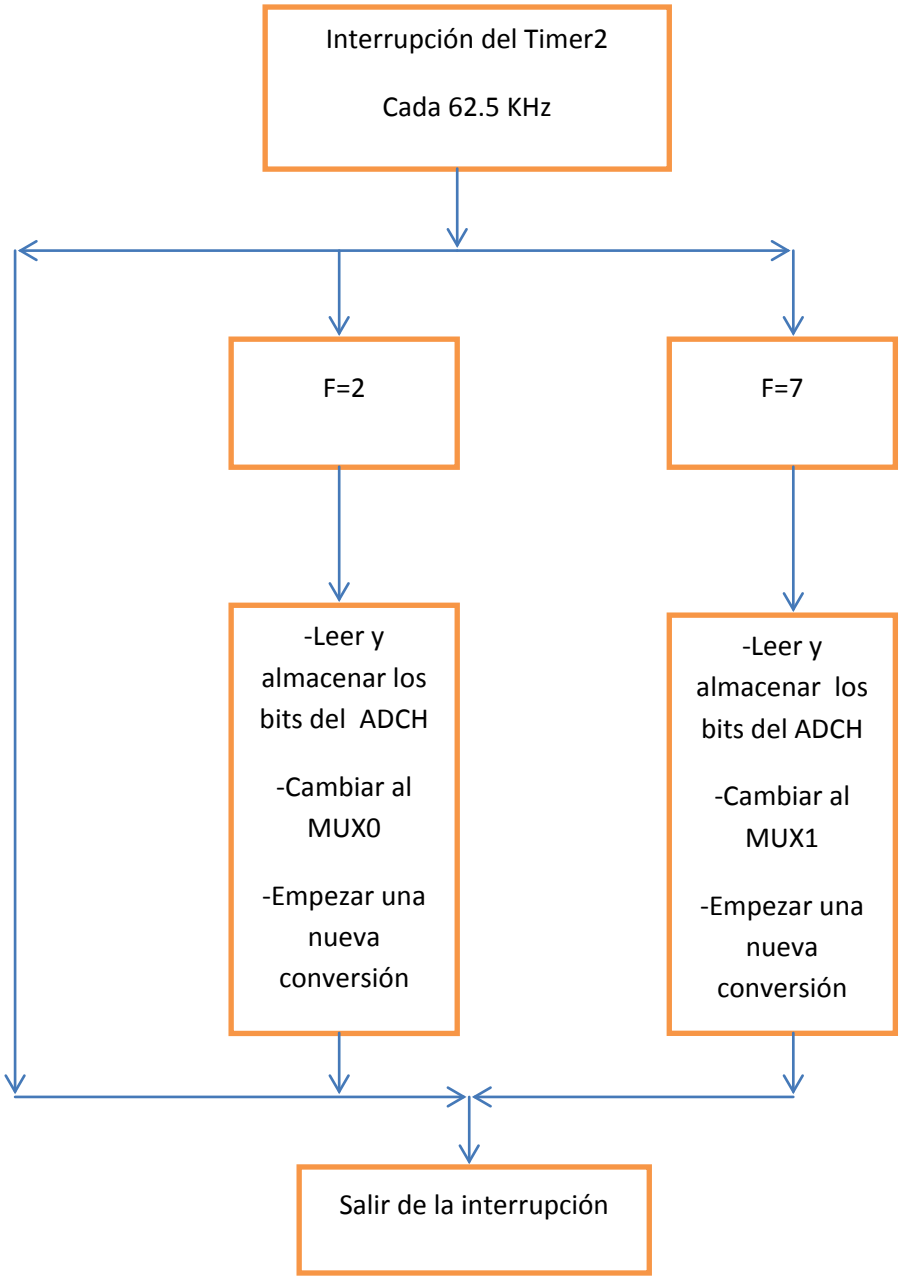
La segunda parte llamada por el compilador, “voidsetup”, es en la cual se definen todos los parámetros de los temporizadores, conversor...modificando los registros, esta parte será ejecutada una única vez.

La tercera parte llamada “voidloop”, se trata de un loop que se repetirá infinitamente, es aquí donde se encuentra el código para los efectos digitales.

La ultima y cuarta parte es la interrupción creada por el timer2, en esta interrupción se cogerán los datos procedentes del conversor, y se le indicara que efectué la siguiente conversión.

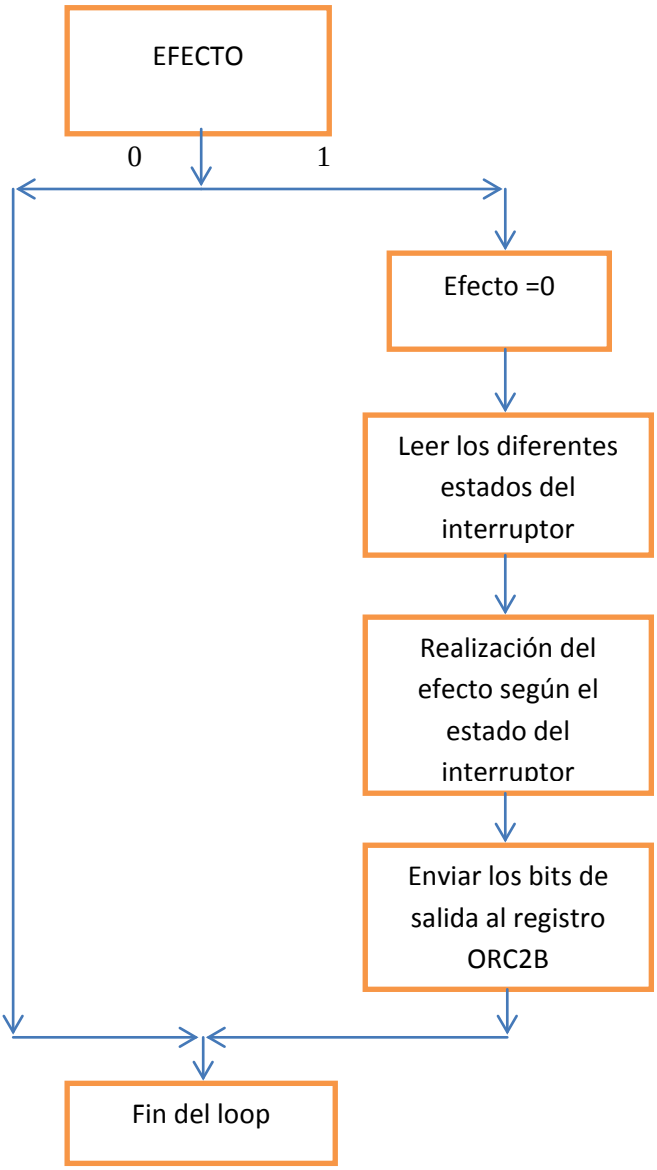
A continuación se muestran los diagramas de flujo para la interrupción y para el programa principal

-Diagrama de flujo de la interrupción



El microcontrolador está programado para que se produzca una interrupción cada 62.5KHz, y dentro de esta interrupción se creó una variable "f" que va incrementando cada vez que se produce una interrupción. De esta manera cuando la variable alcanza el valor de 2 el conversor leerá la entrada A1, que es la correspondiente al potenciómetro que controla los parámetros de los efectos, y cuando su valor alcance 7 leerá la señal de audio de entrada.

-Diagrama de flujo del programa principal.



Este diagrama muestra cómo funciona el loop del programa. Dentro de la interrupción existe una variable que toma el valor de uno cuando la variable f tiene valor 7, y es en este momento, cuando efecto=1, donde el procesado se lleva a cabo, de esta manera se evita que entre cada vez que salga de la interrupción. Una vez que ha entrado en el programa principal, lo primero que realiza es una lectura de los estados de los pines asociados a los interruptores, para así saber que efecto realizar. Por último, después del procesado envía los bits al registro asociado al timer2, ORC2B.

3.5. OPCIONES PARA EL AMPLIAMIENTO DE MEMORIA

Como se ha explicado con anterioridad, el proyecto alcanzo un punto de compromiso. La memoria del microcontrolador era muy escasa, y para arreglarlo se intentó buscar varias opciones para aumentar la memoria de este.

La primera manera que se barajo fue acoplar una tarjeta SD, a Arduino. Una tarjeta de memoria SD utiliza la comunicación SPI (figura 3.5.1). El Bus SPI es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj.

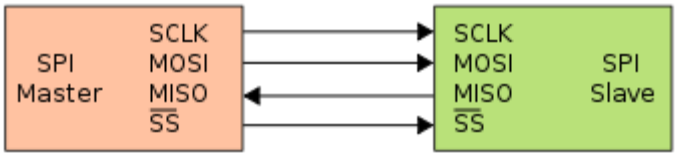


Figura 3.5.1. Comunicación SPI.

En la figura anterior se ilustra la comunicación SPI, en este caso Arduino sería el maestro, y la tarjeta SD el esclavo.

Hay que tener en cuenta que Arduino funciona a 5V, y una tarjeta SD funciona a 3.3V, por lo que los dados deberán tener el estado lógico a 3.3V. A continuación se muestra el esquema a seguir.

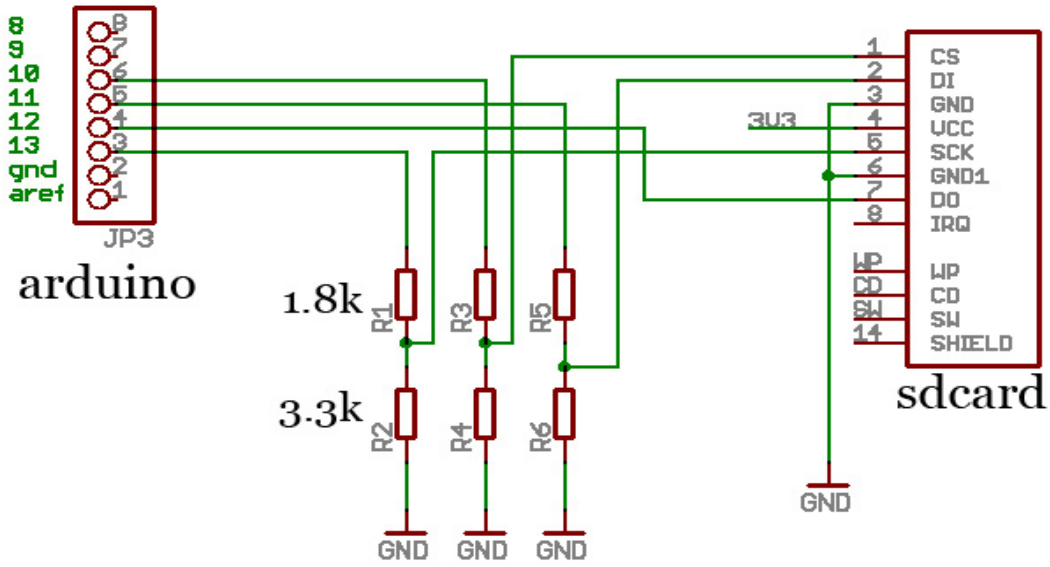


Figura 3.5.2. Esquema de conexión entre Arduino y una tarjeta SD.

En la figura 3.5.2 se muestra el esquema de conexión entre Arduino y una tarjeta SD, simplemente consta de tres divisores de tensión que convierten la tensión de salida de Arduino de 5V a 3.3V para la tarjeta SD.

Debido a que Arduino es una plataforma libre, existen gran variedad de librerías, en particular hay una librería específica para la comunicación con una tarjeta SD, la librería SD. Prácticamente la comunicación esta lista entre Arduino y la tarjeta SD.

Este estándar de comunicación presenta un problema para este propósito, la comunicación es en serie, y cada segundo debería enviar y recibir 40000 muestras de 8 bit. A esto hay que añadir que la librería existente en Arduino necesita abrir un fichero para almacenar los datos, y cerrar este fichero para guardarlo, al igual que para la lectura de este fichero. Es decir tendría que abrir y cerrar 80000 veces el fichero, para almacenar los datos en el y para leer los datos, y enviar y recibir 640000 bits por segundo.

Todos estos condicionantes hacen inviable esta opción para el ampliamiento de memoria, y poder almacenar y acceder a las muestras del conversor a tiempo real.

El siguiente paso en el intento de ampliar la memoria de forma externa, fue probar la comunicación en paralelo. Se contaba con dos memorias FIFO para este propósito, exactamente el modelo IDT 7206.

En la siguiente figura se observa un esquema de conexión de estas memorias.

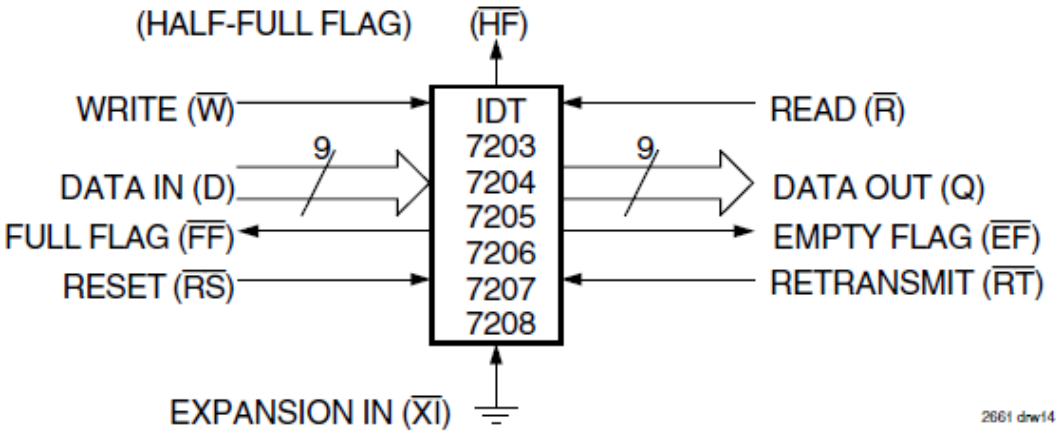


Figura 3.5.3. Esquema conexión memoria IDT 7206.

Arduino cuenta con un número limitado de puertos y son insuficientes para este tipo de comunicación. Arduino Mega si cuenta con mayor número de puertos, y sería posible interconectar Arduino Mega y la memoria 7206.

Al ver que Arduino UNO no contaba con los puertos suficientes para la comunicación en paralelo de 8 bits, se barajó la opción de la comunicación en paralelo con una memoria que permitiese la comunicación en paralelo de 4 bits, pero el número de operaciones por segundo era inviable.

La transferencia de datos entre Arduino y una memoria externa es posible realizarla, pero con velocidades de transmisión inferiores a la requerida para esta aplicación. El caso de transferencia en serie es inviable ya que requiere un gran número de operaciones por segundo las cuales Arduino es incapaz de realizar. La comunicación con una memoria externa tipo FIFO queda descartada debido a la falta de puertos para establecer esta comunicación.

CAPÍTULO 4

REALIZACIÓN DEL CIRCUITO IMPRESO

En este capítulo se procederá a explicar el proceso seguido a la hora de la realización de la PCB. Se empezará con una breve introducción acerca de las principales características de un circuito impreso. Seguidamente se procederá al diseño en el PCB Editor, y por último se hablará del montaje.

4.1 INTRODUCCIÓN A LAS CARACTERÍSTICAS BÁSICAS DE UNA PCB.

En electrónica, un circuito impreso, tarjeta de circuito impreso o PCB, es una superficie constituida por caminos o pistas de material conductor laminadas sobre un sustrato no conductor. El circuito impreso se utiliza para conectar eléctricamente - a través de los caminos conductores, y sostener mecánicamente - por medio del sustrato, un conjunto de componentes electrónicos. Los caminos son generalmente de cobre mientras que el sustrato se fabrica de resinas de fibra de vidrio reforzada (la más conocida es la FR4), cerámica, plástico, teflón o polímeros como la baquelita.

La placa de circuito impreso a de diseñarse según el tipo de componentes que se vayan a utilizar, existen dos tipos los llamados THD (throughtholedevide), y SMD (surfacemountdevice). Los componentes THD son introducidos por una cara de la PCB y soldados por la otra, sin embargo los componentes SMD solo hacen uso de una cara y se sueldan en esa misma cara. Otra diferencia fundamental es el tamaño de estos componentes, por lo general los SMD son más pequeños que los THD.

Una placa PCB puede componerse desde una hasta numerosas capas, hay varios conceptos que afectan al diseñar una PCB multicapa, ya que tiene que existir una conexión física entre las diferentes capas. Una PCB de una única cara consta de una cara donde están las pistas y se sueldan los componentes y otra cara donde son colocados los componentes. Las PCB de más de una cara necesitan interconectar estas caras entre ellas, y esto se hace por medio de las llamadas vías. Una vía es un canal que conecta una cara de la PCB con otra.

A la hora del diseño de una PCB hay que tener en cuenta también las condiciones de trabajo de la tarjeta que el diseño sea apto para automatizar el proceso de montaje, la anchura de las pistas conductoras y separación entre ellas, la disipación de calor, la emisión e inmunidad frente a interferencias electromagnéticas.

Para la fabricación de PCBs existen dos métodos principales, la fabricación por ataque químico y o la fabricación haciendo uso de un fresadora. Dentro del ataque existen dos grandes familias, la impresión serigráfica y el fotograbado, estas dos técnicas consisten en grabar la el dibujo de la PCB sobre una placa de cobre para más tarde eliminar el cobre no deseado. La fabricación por medio de una fresadora se basa en que la fresa va recibiendo comandos desde un programa que controla el cabezal de la fresa, los ejes *x*, y yz. Los datos para controlar la máquina son generados por el programa de diseño, y son almacenados en un archivo en formato HPGL o Gerber.

4.2 DISEÑO EN PCB EDITOR.

En este caso caben dos opciones para el diseño de la PCB, hacerla de una capa o de dos capas. A ser posible se indicó que se hiciese de una única capa.

Durante todo el diseño se hará uso de componentes THD.

A continuación se presentan los principales pasos seguidos para el diseño del circuito impreso:

1. ***Diseño del esquemático con OrCAD Capture.***

Creación del proyecto, colocación de componentes, creación de bloques jerárquicos, creación de símbolos y simulaciones.

2. ***Creación y asignación de footprints***

Los footprints son las huellas de los componentes tal y como irán soldados en el PCB. Hay que asignar su footprint a cada componente del esquemático que queramos incluir. Existe una librería de footprints ya diseñados que ofrece OrCAD. Si no existe el footprint deseado OrCAD facilita los medios necesarios para el diseño de estos.

3. ***Acondicionamiento final de los esquemáticos y extracción del 'netlist' desde Capture.***

Una vez organizado el esquemático, se procede a la extracción del netlist, que es un fichero que sirve para exportar toda la información de los circuitos al editor de layout, que en este caso será PCB Editor.

4. ***Diseño del layout con PCB Editor.***

En PCB Editor se procederá al diseño de la PCB a partir del netlist previamente importado. Colocaremos los componentes, las pistas, los planos de masa y el resto de elementos del PCB.

5. ***Extracción de los ficheros Gerber a partir del layout.***

Estos archivos contienen la información necesaria para la fabricación del PCB. En este caso deben ser compatibles con la máquina fresadora que se encarga de fabricar las placas.

1. ***Diseño del esquemático con OrCAD Capture.***

Una vez dibujado el esquemático y haber simulado con éxito las diferentes etapas, es conveniente organizar el esquemático.

En este caso el circuito diseñado requiere un gran número de componentes, y en OCAD Capture cada uno tiene un número asociado, mediante una de las múltiples opciones ofrecidas por el programa se numeraran en orden todos los componentes, esto facilitara su localización posteriormente.

Como ya se ha comentado la mesa de mezclas consta de tres canales y varias etapas en cada uno de ellos, el siguiente paso será crear una estructura de bloques que conforme todas las etapas y canales de la mesa de mezclas.

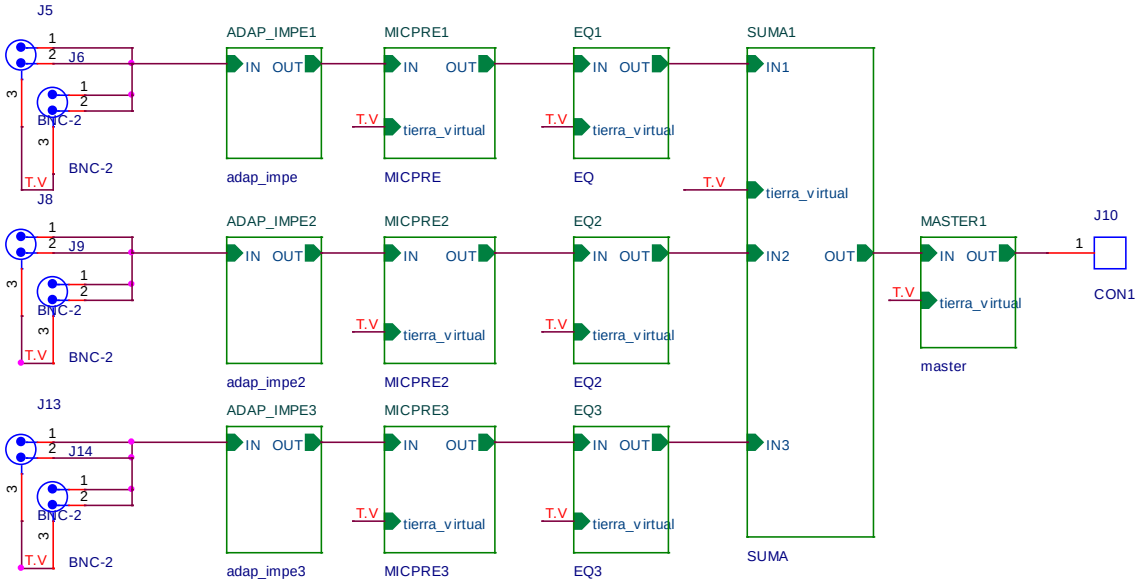


Figura 4.2.1. Estructura de bloques jerárquicos

En la Figura 4.2.1 se muestran todas las partes principales de la mesa de mezclas. Si se recorre el diagrama de izquierda a derecha se observa que existen tres canales, donde lo primero son los conectores, que dan paso al adaptadores de impedancias. Este da paso al preamplificador siendo este el segundo bloque, al cual le sigue el ecualizador. Notar que en estas dos etapas existen dos entradas, una es la de la señal de audio y otra hace referencia a la tierra analógica. El bloque que une a los tres canales es la etapa sumadora, que consta de tres entradas, una para cada canal, y una entrada adicional para la tierra analógica. Por último se encuentra el master al cual le entrará la salida del circuito sumador y la tierra analógica. Para realizar la conexión con Arduino se pondrá un conector.

El último paso será añadir las partes restantes al esquemático, el circuito correspondiente a la conexión con Arduino, la alimentación de la placa, y el divisor de tensión, quedando como resultado final:

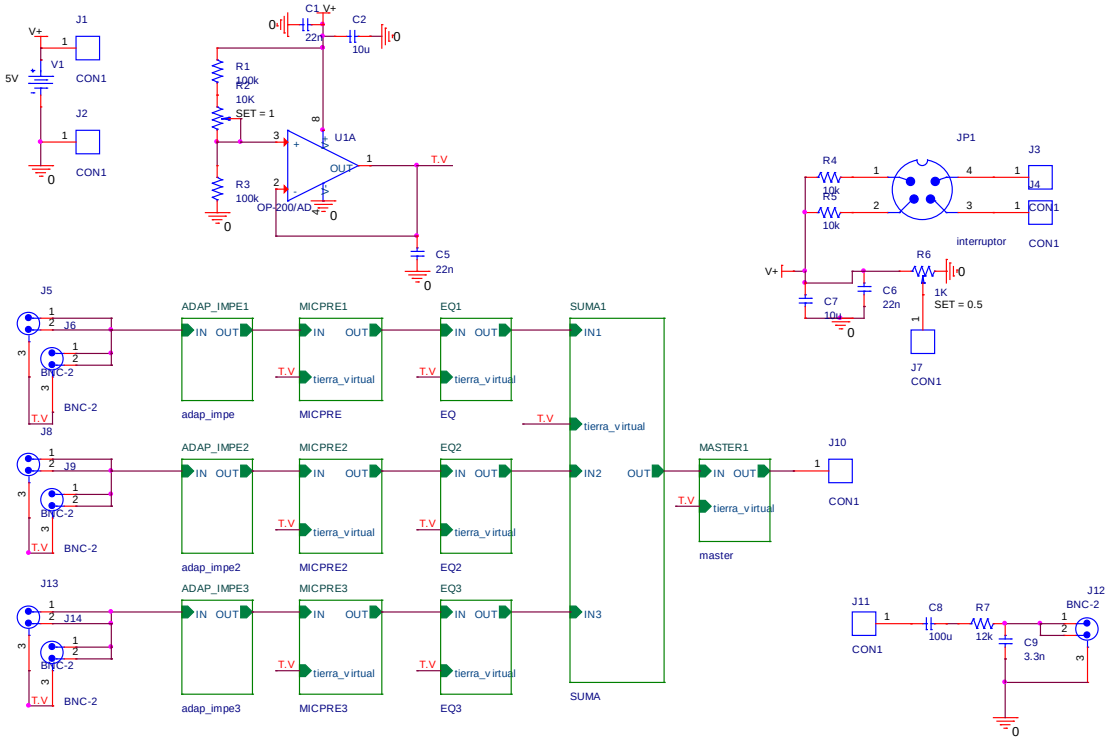


Figura 4.2.2. Esquemático completo.

En esta figura se añaden nuevos elementos respecto a la figura 4.2.1, como son los conectores para la alimentación, el generador de 2.5V, el filtro final RC y un pequeño circuito que consta de un potenciómetro y un interruptor, el cual hará uso Arduino.

2. Asignación de footprints

Una placa PCB puede componerse desde una hasta numerosas capas, en este caso en particular se intentó la realización de este circuito impreso por medio de una única capa, aunque más tarde fue necesario incluir otra capa más. Hay varios conceptos que afectan al diseñar una PCB multi-capa, ya que tiene que existir una conexión física entre las diferentes capas

Para el diseño del circuito impreso, lo segundo que ha de realizarse es la asignación de los footprints a cada componente del esquématico en OrCAD Capture.

Un footprint es la “huella” de un componente en su formato físico, tal y como irá colocado en el PCB. En la figura 4.2.3 se muestra el footprint típico de una resistencia en PCB Editor.

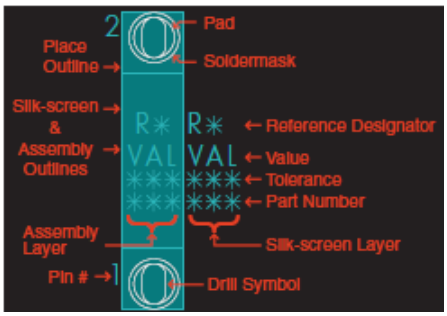


Figura 4.2.3. Footprint de una resistencia.

Los elementos principales de un footprint son los ‘pines’, que se componen del ‘drill’, que es el agujero para colocar la pata del componente, y el ‘pad’, que es la parte metálica soldable que rodea al agujero. En el caso de componentes superficiales los pines no tendrán agujero y consistirán únicamente en un pad. Cada pin tiene asociado lo que se conoce como ‘padstack’. El padstack es un elemento que define las características del pad en cada uno de los niveles de la placa. Aunque en nuestro caso nos limitamos a capas de sólo dos caras, los PCB comerciales puede tener un número mucho mayor de niveles.

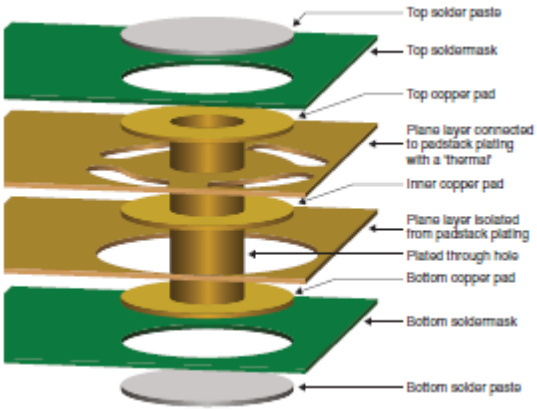


Figura 4.2.4. Ejemplo de los diferentes niveles que puede tener una PCB.

En la figura 4.2.4 se muestra un ejemplo de una PCB de dos capas, donde las capas superior e inferior esta representadas en verde, y las dos capas intermedias en amarillo, se puede observar como existe una unión entre las diferentes capas, y como esta unión se ensancha creando el pad del componente, que puede o no estar unido a esta capa.

OrCAD cuenta con una librería extensa de componentes junto con sus correspondientes footprints, pero componentes de tipo conector son necesarios crearlos manualmente.

Footprints usados:

Res400: para las resistencias.

Dip8_3: para los opamps.

Cap300: para los dos tipos de condensadores

Pot: para los potenciómetros

Dip4_3: para el interruptor correspondiente al circuito de Arduino

Jumper1: para diferentes conexiones.

Pese a la amplia librería de OrCAD de footprints, en este caso, es necesaria la creación de dos de ellos, los conectores de alimentación, y los conectores de Jack. OrCAD incluye un programa llamado Pad Designer, en el cual diseñaremos el pad de nuestros conectores.

En el caso de los conectores de alimentación, el Pad a diseñar debe de tener 6mm de diámetro, y dos milímetros más de radio superior (figura 4.2.5).

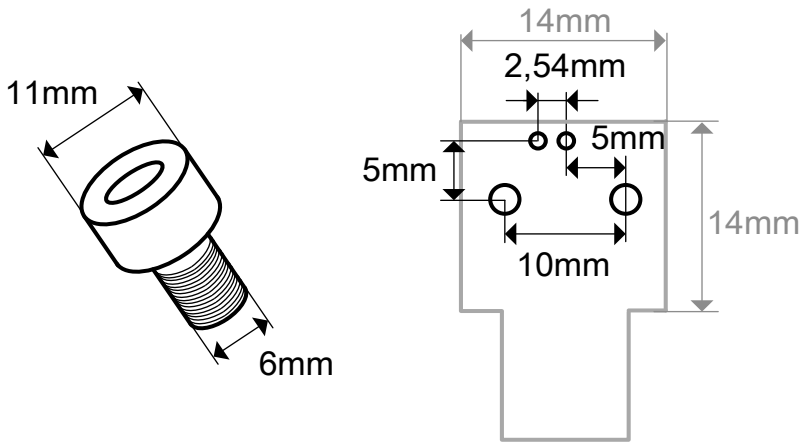


Figura 4.2.5. Conector de alimentación.

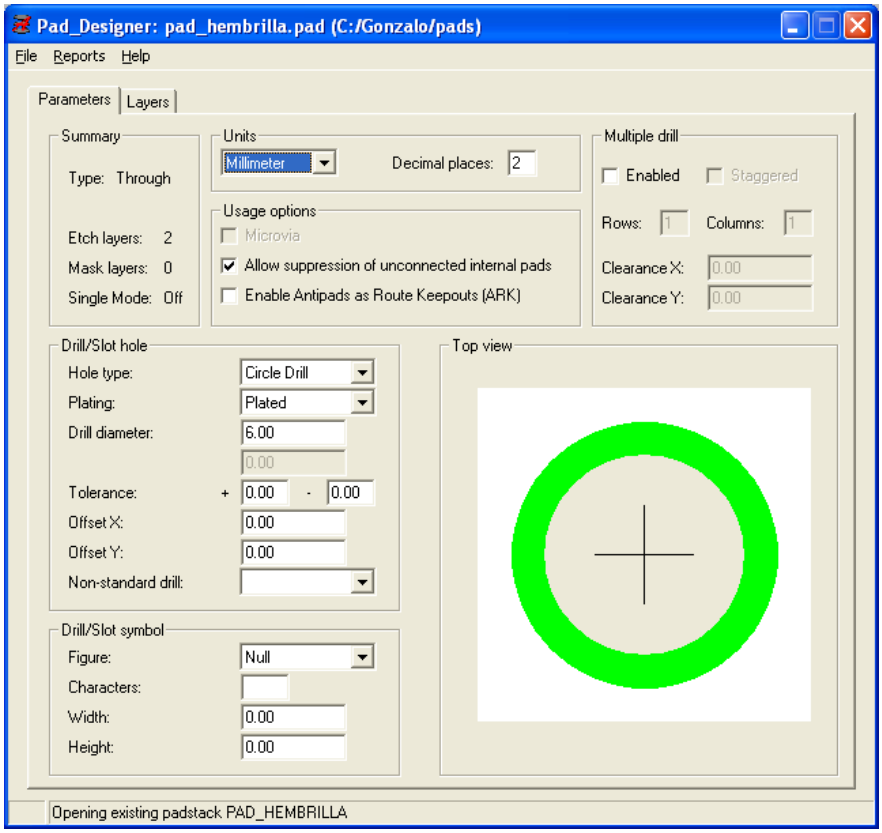


Figura 4.2.6. Ejemplo del diseño de un pad.

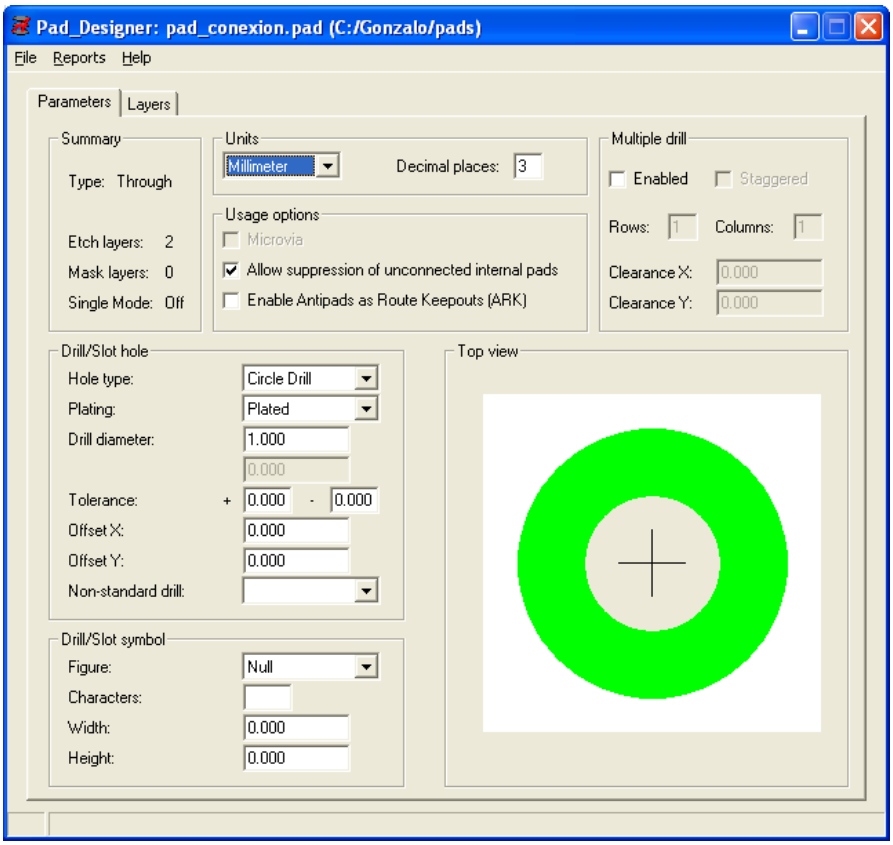


Figura 4.2.7. Ejemplo del diseño de un pad.

En la figuras 4.2.6 y 4.2.7 se muestra el diseño de dos pads diferentes con el programa ofrecido por OrCad, Pad Designer. El usuario elige los diámetros interiores y exteriores dando asi la forma básica al pad.

Una vez creados los Pads correspondientes a los conectores, se procede a la creación del footprint correspondiente a cada conector, se procederá con PCB Editor.

Se trata de posicionar según las medidas del conector los Pads diseñados, y especificar la Package Geometry del conector, este último paso es importante ya que nos marca el espacio que ocupa el conector.

A continuación se muestran los dos conectores diseñados. En la figura 4.2.8 el conector de alimentación y en la figura 4.2.9 el conector Jack.

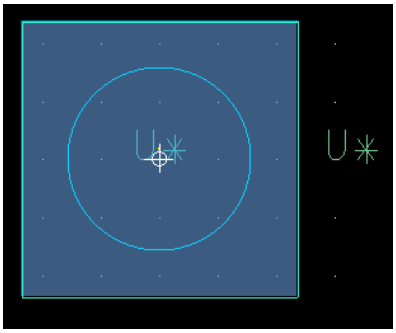


Figure 4.2.8. Conector de alimentación

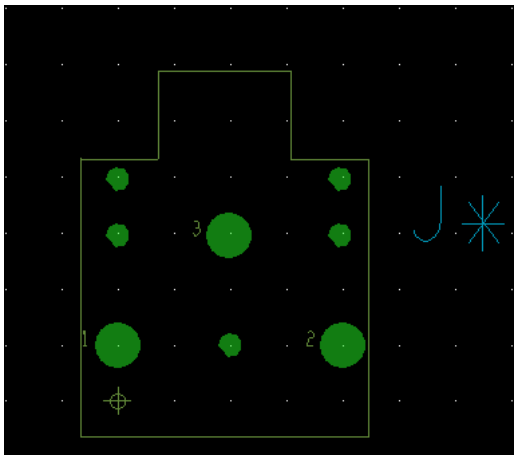


Figura 4.2.9. Conector Jack.

3. ***Acondicionamiento final de los esquemáticos y extracción del 'netlist' desde Capture.***

El 'netlist' es un fichero que contiene la información de los esquemáticos del proyecto, será necesario para el diseño posterior en PCB Editor. Antes de proceder a la extracción de este fichero es recomendable realizar comprobaciones del esquemático, para asegurar que todo está bien conectado, ya que no hay vuelta atrás.

4. ***Diseño del layout con PCB Editor.***

El primer paso es la colocación de todos los componentes, de una forma adecuada para que posteriormente beneficie a la hora de enrrutar.

Antes de efectuar el enrrutado entre componentes ha de considerarse la posterior fabricación del circuito impreso, en este caso se parte de una placa cubierta completamente por las dos caras de una capa de cobre, y el dibujo y agujeros realizados se harán mediante una fresadora, hay que tener en cuenta las limitaciones de esta fresadora.

La anchura de cada pista del circuito impreso la definimos a 0.7 mm de grosor, y la separación entre pistas a 1mm, ya que la fresadora es incapaz de tolerar una separación menor. Las dos placas recubiertas totalmente de cobre se convierten en una ventaja para el diseño, ya que se pueden aprovechar los planos restantes de cobre como conexión de tierra, por lo que las conexión a tierra serán despreciadas primeramente.

Al constar de dos capas es posible conectar una capa con la otra en el momento deseado, es decir a la hora de diseñar una pista, es posible "saltar" de una capa a otra y continuar dibujando la pista en la otra capa, a este proceso PCB Editor le llama VIA, sin embargo es recomendable evitarlo, ya que el proceso de fabricación del circuito impreso no cuenta con los medios necesarios para realizar esta interconexión, se tendría que realizar a mano con un pedazo de alambre, lo que afectaría a la señal que circule por esta pista. Es evidente que ha de existir un salto de pista en algún momento, y al no ser recomendable por medio de una VIA, se optó por hacerlo por medio de los componentes, es recomendable evitar los componentes difíciles de soldar por arriba, tales como opamps, potenciómetros etc.

La última consideración es que se ha de evitar los cambios de dirección en las pistas de 90º, PCB contiene una opción que suaviza las trazas de las pistas automáticamente, aunque se

prefirió no usarla y hacerlo de manera manual ya que esta opción puede desajustar lo hecho con anterioridad.

El proceso de enrrutado se efectuó en varias fases. La primera fue enrrutar las partes del circuito en la que se acumulaban más componentes o pines, tales como opamps. Se muestra en la figura 4.2.10.

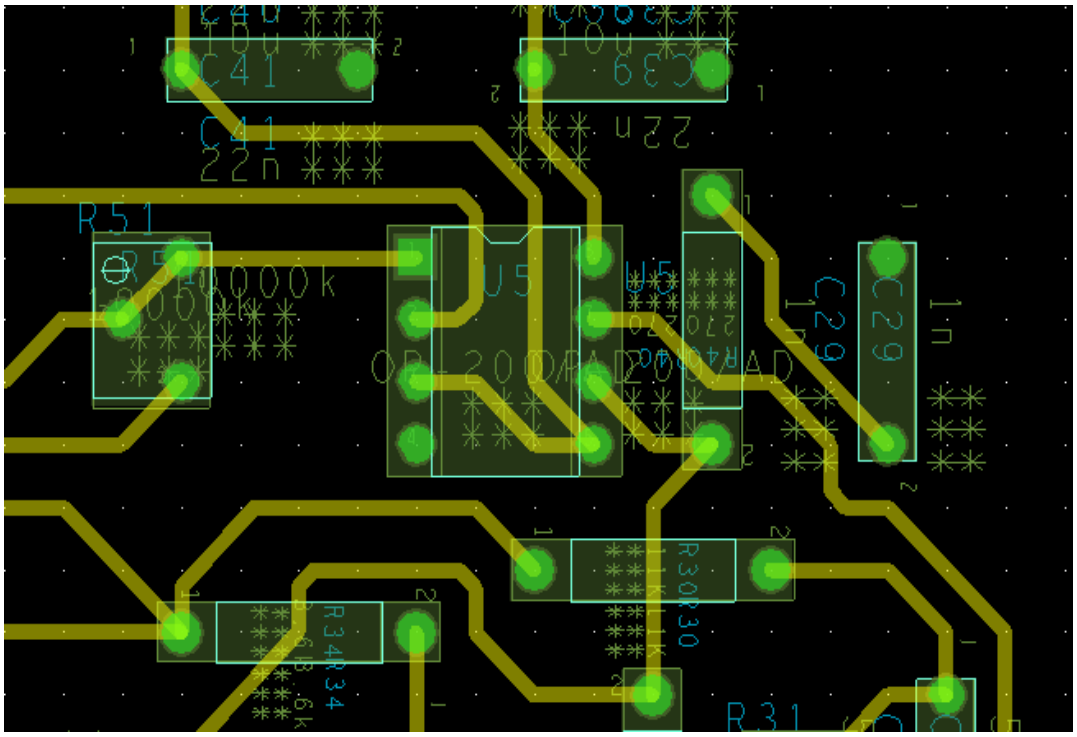


Figura 4.2.10. Ejemplo de enrrutado en las cercanías a un opamp.

Una vez enrutados los seis opamps del diseño, se pasó a enrutar las partes cercanas a estos. Ejemplo en la figura 4.2.11.

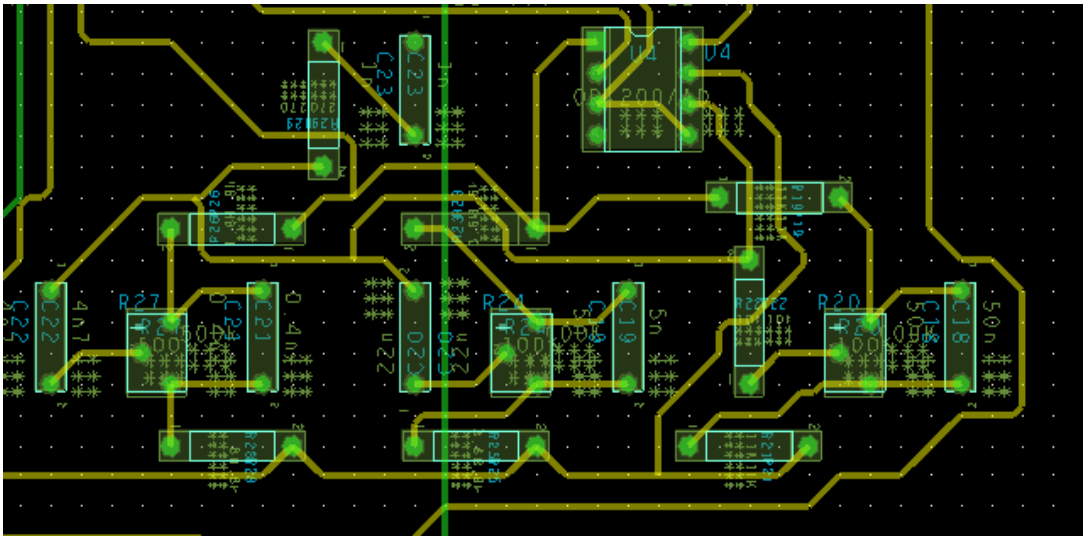


Figura 4.2.11. Ejemplo de enrutado de los componentes cercanos a un opamp.

Una vez enrutado las partes cercanas a los opamps, y los componentes de alrededor son pocas las pistas restantes, aunque son las más complicadas de diseñar, ya que ya existe un gran

número de ellas, y estas nuevas pistas tendrán que abrirse espacio entre las ya existentes. Es en este punto donde se hace imprescindible el uso de VIAS, aparecen en color verde. Se intentó usar VIAS únicamente para el voltaje ya sea de 5 V como de 2.5V y evitar usar VIAS con la señal de audio.

En la figura 4.2.12 se muestra el resultado final del enrutado del diseño del circuito impreso.

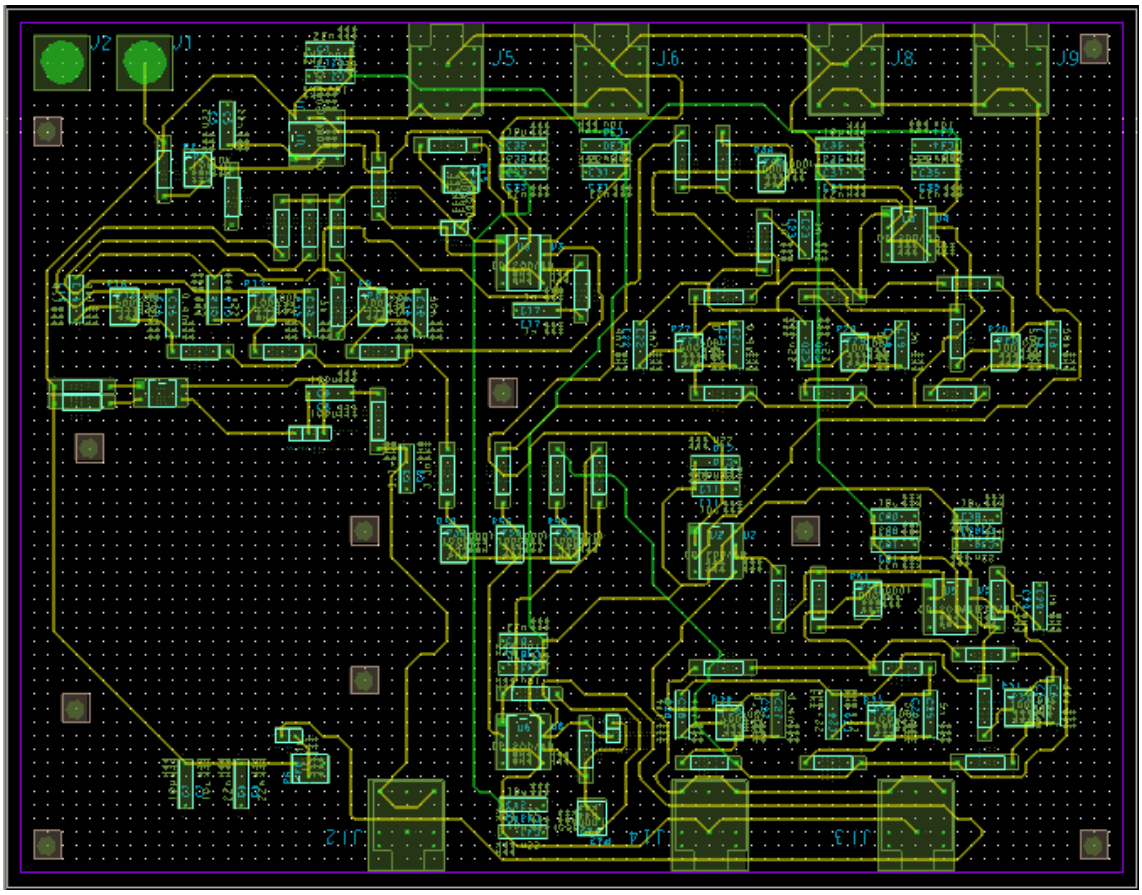


Figura 4.2.12. Resultado final del enrutado.

Una vez que se ha finalizado prácticamente el enrutado, se añadirán agujeros o también llamados MountingHoles para las patas de la PCB, asi como para sujetar a Arduino dentro de esta. Dentro de la variedad que ofrece OrCAD elegiremos el MTG125, ya que se adecua a los soportes que se encuentran en el laboratorio, y a la medida de los agujeros de Arduino. En la figura 4.2.13 se muestra un mountinghole, y en la figura 4.2.14 se observan los mountingholes para Arduino.

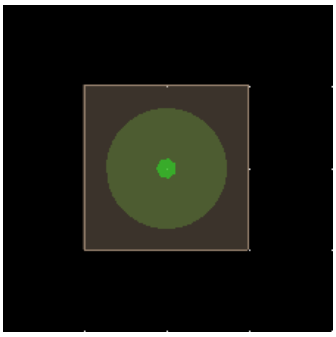


Figura 4.2.13. Ejemplo de un MountingHole MTG125

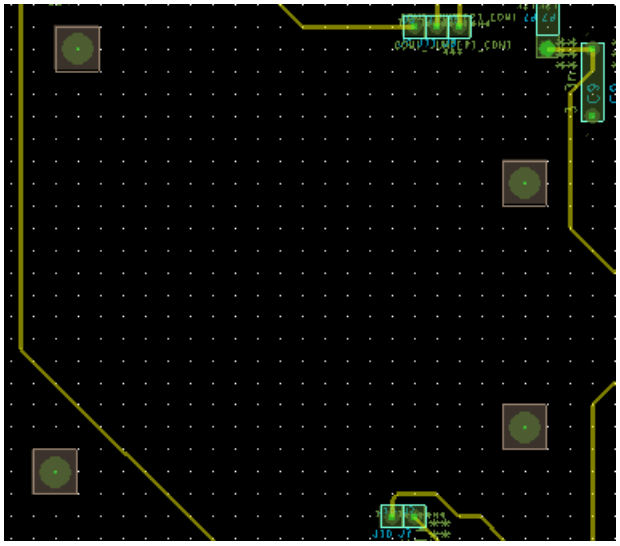


Figura 4.2.14 Varios MountingHoles que conforman la estructura de Arduino.

Para acabar el diseño se procederá a la introducción de planos de tierra. En ningún momento se han interconectado los nodos correspondientes a tierra, y aprovechando las propiedades físicas de la placa que será usada para el circuito impreso, los nodos pertenecientes a la tierra se incluirán todos en un plano, tanto en la capa superior como en la inferior.

El circuito impreso será fabricado mediante una fresadora que eliminara las partes de cobre que se desee. La placa inicialmente está recubierta totalmente por cobre, y este cobre se puede aprovechar usándolo como plano de tierra. En el diseño no se ha colocado ninguna pista que uno los diferentes nodos que pertenecen a tierra, sino que se ha colocado dos planos que los contienen a todos. Esta técnica además de facilitar la fabricación es recomendable que haya metal conectado al plano de tierra en la mayor cantidad posible.

A continuación se muestra en la figura 4.2.15 y 4.2.16 , los planos de tierra de inferior e inferior y superior, respectivamente,

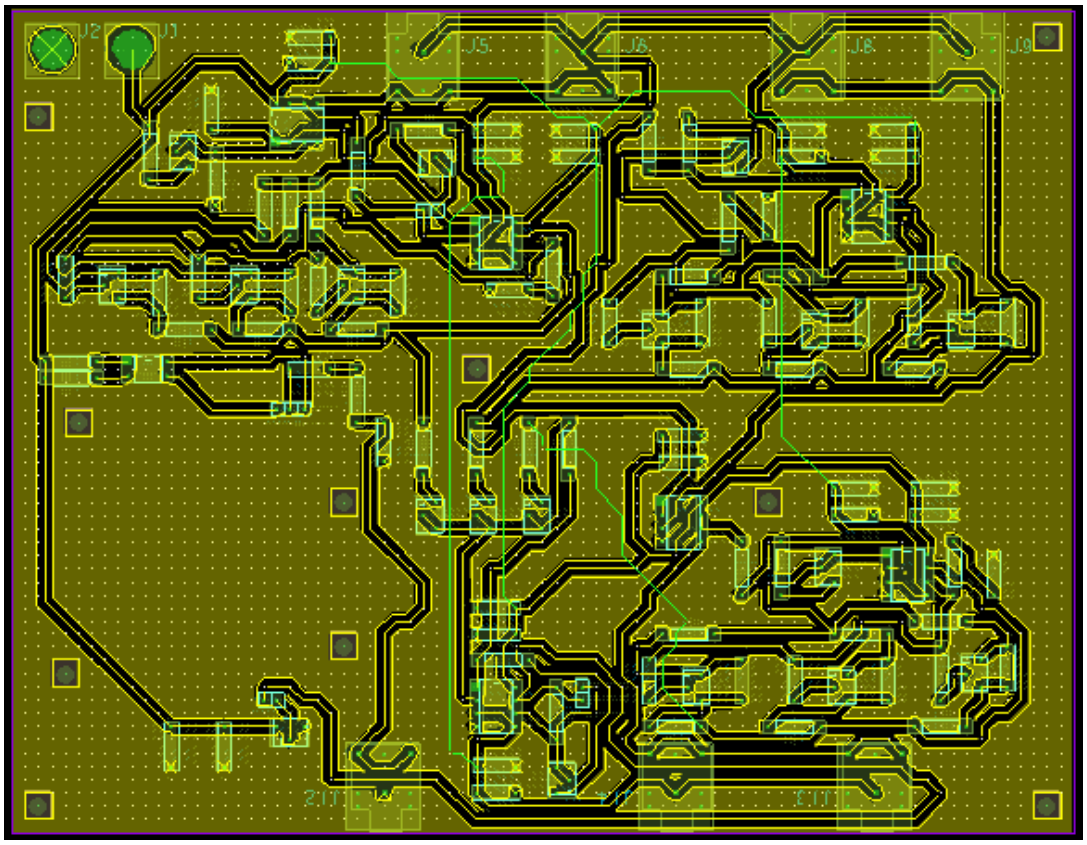


Figura 4.2.15. Plano de tierra de la capa inferior.

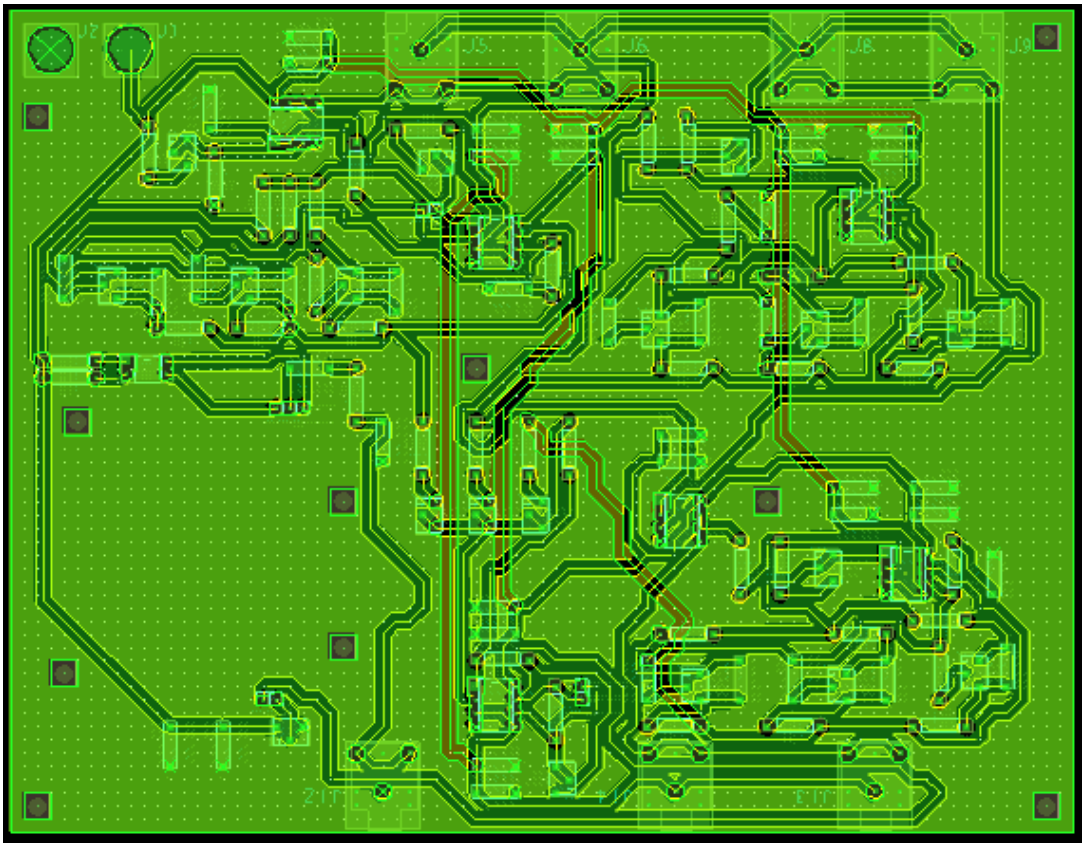


Figura 4.2.16. Planos de tierra inferior y superior.

5. Extracción de los ficheros Gerber a partir del layout.

Una vez que se ha finalizado el diseño del circuito impreso únicamente que da un paso, y es el de la extracción de los ficheros necesarios para la implementación física de la PCB.

Antes de proceder a la extracción de estos ficheros conviene comprobar por última vez el diseño, y ver que no hay errores.

Los ficheros Gerber son los archivos que se extraen del layout una vez terminado y que contienen toda la información del mismo que se necesita para fabricarlo. Estos ficheros son los que se le proporciona a la máquina para que fabrique la placa.

Existen varios tipos de ficheros Gerber pero en este caso se extraerán cuatro que serán los siguientes: TOP.art y BOTTOM.art (información de la distribución del cobre en las capas TOP y BOTTOM respectivamente), otro fichero .art con el contorno de la placa (BoardOutline) y finalmente un fichero de extensión .drl con la información de los taladros.

Cabe destacar que durante prácticamente todo el proceso de diseño del circuito impreso se intentó hacer uso únicamente de una capa, aunque se hizo imposible y se tuvo que proceder a añadir otra capa en el diseño.

4.3 MONTAJE

Una vez que la placa de circuito impresa esta lista, es siguiente paso es comprobar que está en buen estado, que no tiene ninguna pista rota, cortocircuitos etc.

En un principio la placa fue diseña con dos caras, pero únicamente se fabricó de una cara debido a que el número de pistas en la cara superior era pequeño. Sin embargo esto se tradujo en un problema que fueron los planos de tierra, ya que en la capa superior existía un gran plano de tierra, y en la capa inferior este plano era más pequeño y muy poco comunicado entre sí, debido al gran número de pistas existentes en esta cara.

Se debatió entre fabricar otra vez la placa por dos caras o si aprovechar la placa ya fabricada. Finalmente se optó por aprovechar la placa ya fabricada.

Existían dos problemas, el primero de ellos y más importante era la tierra, ya que había gran cantidad de nodos conectados a tierra que no estaban comunicados entre sí. Este problema fue solucionado mediante una cinta de cobre. Se trata de una cinta adhesiva, por una cara es una lámina de cobre, y por otra es pegamento (figura 4.3.1).

Se cortó la cinta en tiras finas, y se rodeó la cara superior de la placa con esta cinta. Lo siguiente fue cortar más tiras de la cinta a medida y llevar cinta hasta los nodos que deberían estar conectados a tierra. Se hizo un agujero en la cinta para que el componente la traspasase.

Al no tener cobre por las dos caras todas las tiras estaban aisladas entre ellas, se optó por soldar las uniones entre tiras, de esta manera todas las tiras estaban en cortocircuito.

Por último ya que en la cara superior no existe plano de tierra, solamente existe en la capa inferior, se hizo necesario conectar estos dos planos (el plano de la capa superior formado por las tiras de cinta de cobre, y el plano de tierra de la capa inferior). Para esta interconexión

fue usada de nuevo la cinta de cobre, fue cortada una pequeña tira y se bordeo la placa con esta tira. Finalmente se procedió a la soldadura de esta cinta de cobre en las dos caras.

De esta manera el problema de las tierras quedó solucionado. En la figura 4.3.1-3 se muestra este proceso.

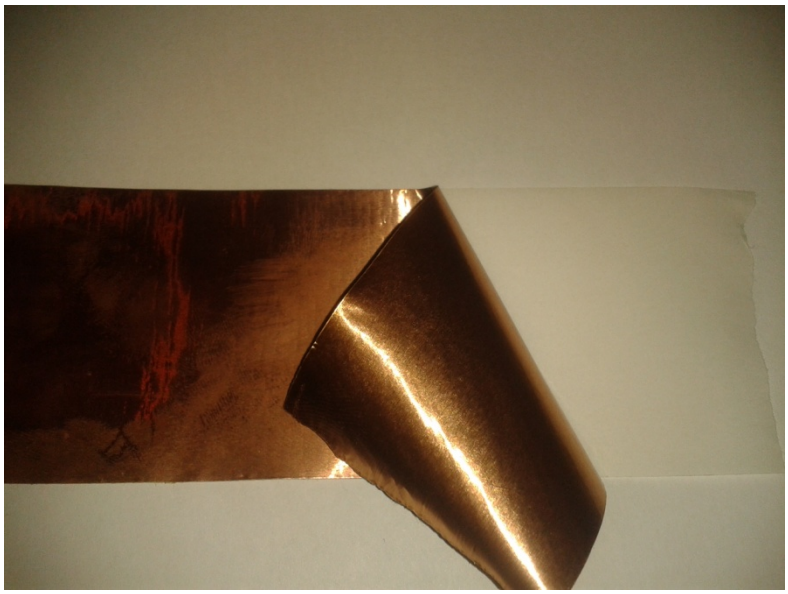


Figura 4.3.1. Cinta de cobre adhesiva.

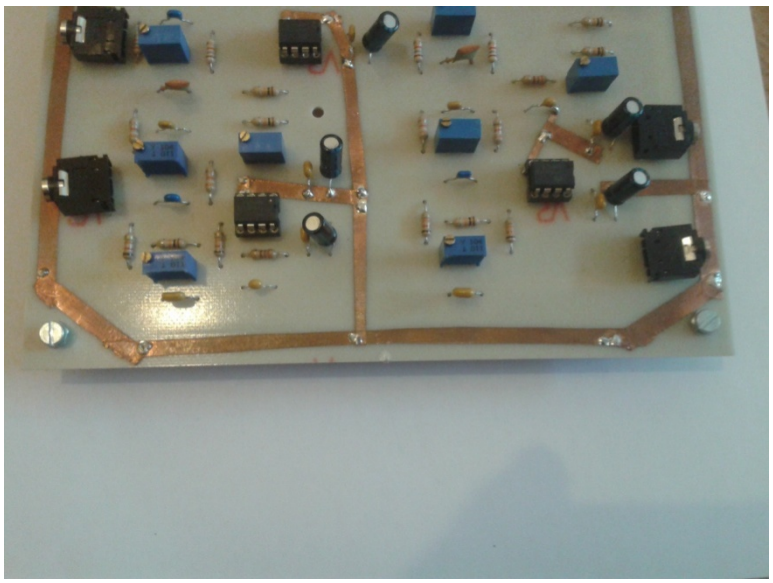


Figura 4.3.2. Pistas con la cinta de cobre.

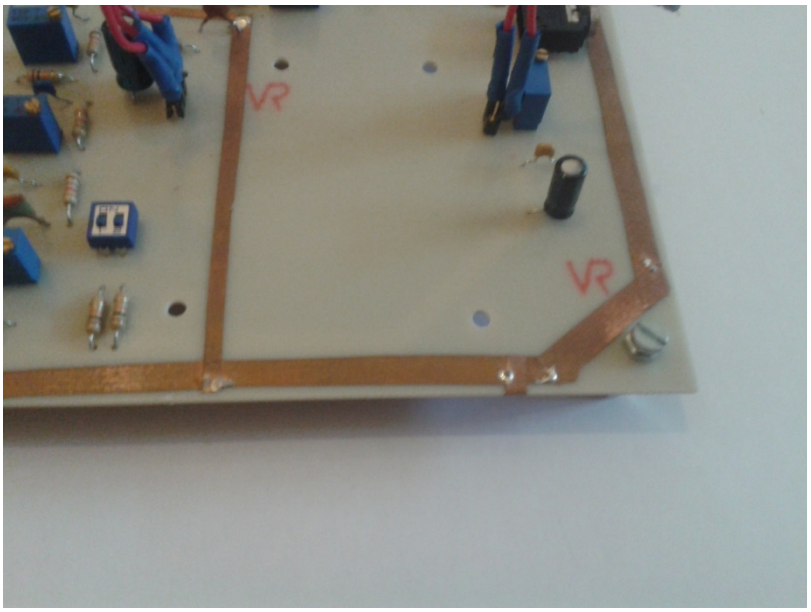


Figura 4.3.3. Unión entre planos de masa de las distintas capas.

El siguiente problema a afrontar la inexistencia de pistas en la cara superior. Había dos opciones, hacer las pistas con la cinta de cobre de la misma manera que se utilizó para los planos de tierra, o crearlas mediante cables entre componentes en la capa inferior. Se optó por la segunda solución, lo cual presentaba una nueva incógnita, el tipo de cable a usar, cable de alambre rígido, o cable compuesto de hilos de cobre, el cual es más manejable. Se optó por esta segunda opción, y se descartó el cable de alambre rígido, ya que esta rigidez podría causar algún problema mecánico.

Una vez solucionados los problemas se pasó al montaje de los componentes. La placa estaba diseñada para componentes THD (thoughtholedevice), por lo que el montaje de componentes será igual para todos. Cada componente atraviesa la capa superior y es soldado en la capa inferior.

Debido a que en la cara superior se había creado con anterioridad planos de tierra, se hacía necesaria la soldadura de algún componente por las dos caras, ya que en la cara inferior el nodo del componente estaba asilado y al soldar en la capa superior también este nodo dejaría de estar aislado.

El primer componente que se montó fueron las patas a través de los mountingholes que sujetarán a la PCB.

Lo ideal a la hora de colocar todos los componentes, es empezar por los componentes más bajos en cuanto a tamaño, como las resistencias, e ir aumentando su tamaño, hasta que los últimos componentes a soldar sean los más altos, como por ejemplo los condensadores o potenciómetros, esto facilita el montaje. Sin embargo en este caso, al no ser una PCB contrastada, el circuito únicamente se simulo y se probó por partes en una protoboard, se optó por montar por partes la PCB, es decir se montó los conectores de alimentación y de conexión, y después se montó una a una las diferentes partes de un canal, empezando por el adaptador de impedancias, hasta acabar en el sumador, y posteriormente en el circuito master. De esta manera al comprobar que un único canal funcionaba perfectamente se pasó al montaje de los siguientes

canales. Cabe mencionar que este procedimiento realizado es el típico en un diseño mixto analógico digital como fue descrito en el capítulo uno.

Las pistas creadas mediante cables de hilo de cobre fueron soldadas usando las patas de los componentes por los dos extremos del cable, y soldando el cable con las patas de los componentes.

Hubo problemas con algún cortocircuito, debido a la dificultad de soldar en la capa superior sobre la cinta de cobre, pero mediante un multímetro, estos problemas fueron detectados y solucionados.

A continuación se muestra una figura de la PCB completa, y posteriormente una figura donde se diferenciarán las diferentes etapas.

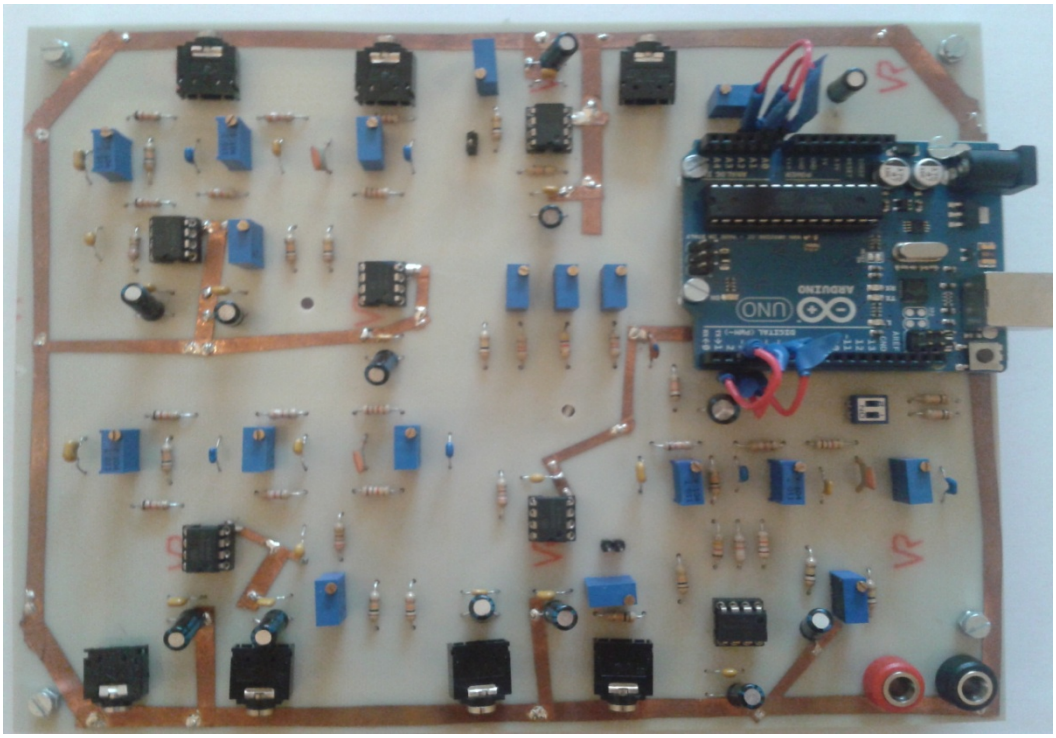


Figura 4.3.4. Circuito impreso completo con Arduino

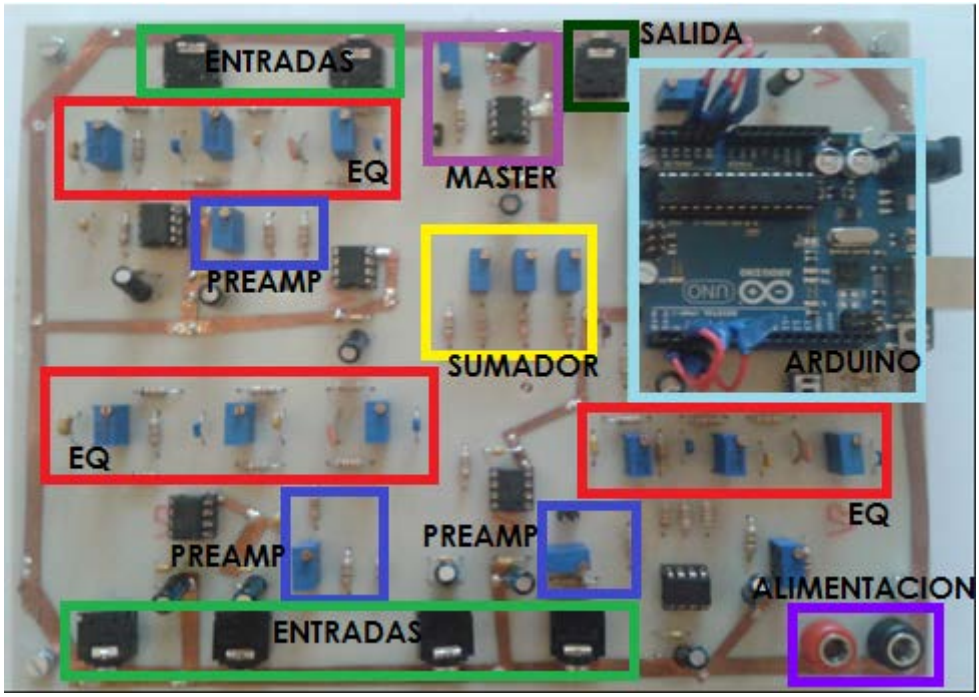


Figura 4.3.5. Diferentes etapas del circuito impreso.

En la figura 4.3.5 se muestran las diferentes etapas, en morado se muestran las entradas de alimentación de la placa. En verde se muestran las entradas de audio y en verde oscuro la única salida del sistema. En azul las etapas preamplificadoras, en rojo los ecualizadores, en amarillo el circuito sumador, en rosa la etapa de master, y en azul claro Arduino.

CAPÍTULO 5

PRUEBAS Y RESULTADOS EXPERIMENTALES DE LA PLACA

En este capítulo, se darán los resultados experimentales obtenidos de la mesa de mezcla. Primero se describe la parte analógica y luego la parte digital. En la parte digital con Arduino se ha utilizado una salida PWM y un filtro de primero orden para formar un conversor digital analógico para recobrar la señal y regresarla en el master de la mesa.

5.1 PARTE ANALÓGICA

Como ya se ha comentado anteriormente lo primero que se realizo fue la comprobación de los planos de tierra creados manualmente con la cinta e cobre, usando un multímetro.

La placa fue montada en un principio por etapas, aprovechando que se trata de una mesa de mezclas, y que tiene tres canales independientes, pero idénticos en cuanto a componentes, se empezó por montar un único canal y por etapas, es decir, lo primero que se realizo fue el circuito que se encarga de generar los 2.5 V, más tarde el adaptador de impedancias, el preamplificador y por último el ecualizador.

En cuanto al generador de 2.5 V, la comprobación de su funcionamiento se basaba en comprobar los voltajes en las entradas y salidas del amplificador operacional. Para el adaptador de impedancias se procedió de igual manera. Además, en la placa se pusieron condensadores de desacoplo para evitar posibles oscilaciones que se introdujeran por la fuente de alimentación.

La siguiente etapa fue el preamplificador, en este caso se usó un generador de funciones y un osciloscopio, así como el multímetro. La prueba residía en introducir una señal sinusoidal, y por medio del potenciómetro del preamplificador se fue variando la ganancia que esta etapa aporta, de esta manera se observó como el preamplificador se comportaba según lo esperado, la señal sinusoidal alcanza un rango de 0.1-4.9 V sin recortarse, una vez superado el límite la señal se recortaba. Este procedimiento se comprobó para varias frecuencias (1000 Hz, 5000 Hz y 10KHz).

Respecto al ecualizador al ser un circuito de tamaño grande en comparación con las demás etapas se estudió con más detalle. Se utilizó un analizar de espectros para comprobar que la respuesta en frecuencia de esta etapa era tal y como la simulación mostraba. Se extrajeron los datos del analizador de espectros y con Matlab se realizó una gráfica con estos datos. Se tomaron dos posiciones de este ecualizador, con todos los potenciómetros al máximo y con todos los potenciómetros al mínimo, de esta manera queda caracterizada la respuesta en frecuencia para los dos extremos. A continuación se muestra la gráfica obtenida.

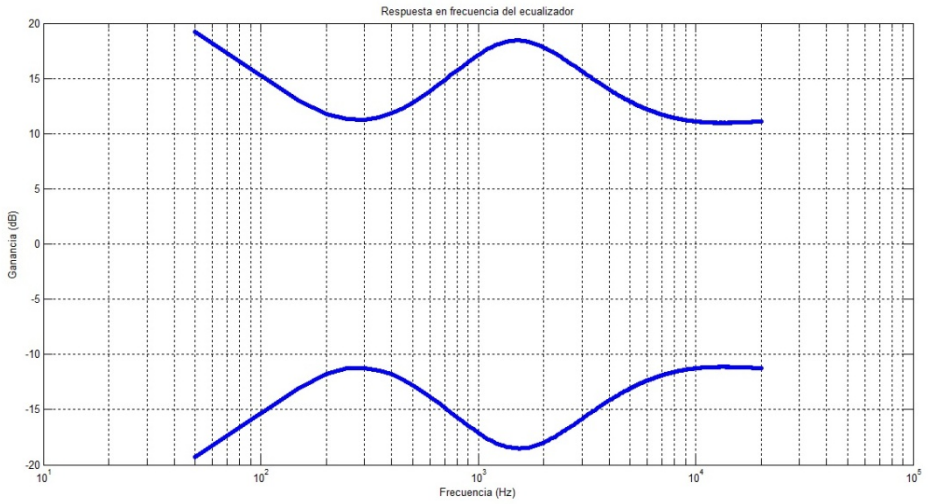


Figura 5.1.1.Respuesta en frecuencia de la etapa ecualizadora.

En la figura anterior se muestra la respuesta en frecuencia del ecualizador, que es acorde con la respuesta obtenida mediante simulación. El cambio más sustancial que se aprecia es que en la banda de agudos, el máximo en la simulación se encontraba en 10KHz y con un valor de ±9dB, y en la respuesta obtenida con el analizador de espectros el valor obtenido a 10KHz es de ±11 dB. Esto es debido a las tolerancias de los componentes discretos, lo que hace que las respuestas no sean idénticas.

Una vez que se comprobó que un canal funcionaba perfectamente, se procedió al montaje de los dos canales restante, así como la etapa sumadora, y el master.

Al igual que las anteriores etapas, para comprobar y testear estas dos etapas se hizo uso de un generador de funciones, y de un osciloscopio. A continuación se muestra una captura del osciloscopio.

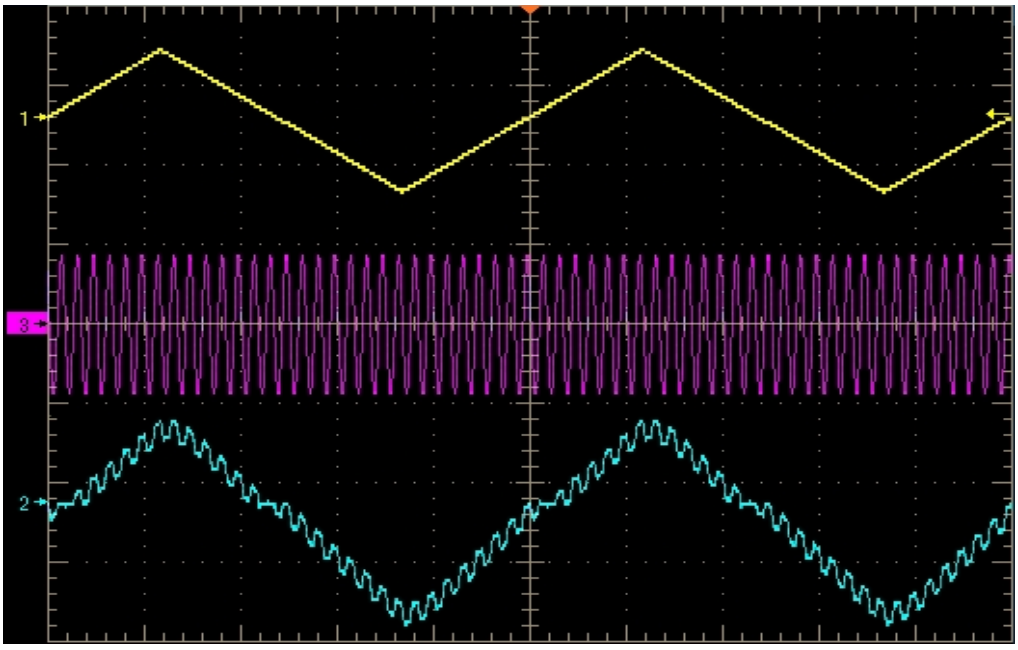


Figura 5.1.2. Captura de osciloscopio.

En la figura 5.1.2 se observan tres señales, la prueba consistía en generar una señal triangular de frecuencia baja (500 Hz) e introducirla por un canal de la mesa de mezclas, y generar una señal sinusoidal de frecuencia superior (15KHz) y amplitud menor, e introducirla por otro canal.

La suma de la señal triangular y de la señal sinusoidal está representada en azul, donde se distingue como la señal resultante es la esperada, es decir se mantiene la forma de la señal rectangular pero la señal sinusoidal de frecuencia mayor queda superpuesta a esta.

A la vez que se comprobó la suma de la señal, se verifico como la ganancia varía según el usuario desee ajustando los potenciómetros, uno por cada canal. De igual manera se verifico la etapa de master, variando la amplitud de la señal a la salida de la etapa sumadora.

5.2 PARTE DIGITAL

El testeo de la parte digital es el correspondiente a Arduino. Una vez que las señales de audio de entrada son amplificadas, ecualizadas y mezcladas, Arduino recibe una señal con una componente de 2.5V en continua, y que puede tener un rango de entre 0-5V. El conversor digital analógico opera a 5V, y es capaz de convertir dentro del rango de 0-5V. Se aprovecha de esta manera todo el rango y se mejora la relación señal ruido.

Para la realización del testeo de la parte relacionada con Arduino se diseñó un circuito, que pasaría a ser el mismo que en la PCB, y se construyó en una protoboard. Este circuito consta de un divisor de tensión para subir la señal de entrada a 2.5V, un potenciómetro para regular parámetros de los efectos, un interruptor para cambiar de efecto, y un filtro paso bajo RC a la salida de Arduino.

Las pruebas para esta parte fueron varias. La primera de ellas era usar solo el conversor con la componente continua y verificar que en realidad la salida del conversor era la mitad del rango del conversor. El conversor cuenta con 8 bits, por lo que al convertir la componente continua la salida debía ser 127.

Otro tipo de comprobación relacionadas con la programación fueron realizadas a través del “Serial Monitor” que ofrece el compilador de Arduino, de esta manera se aseguraba que el comportamiento del programa era el deseado.

Una vez que se comprobó que el código funcionaba, se introdujo varias señales de audio y se escuchaban a la salida de Arduino, para comprobar que los efectos digitales funcionaban.

A continuación se muestran capturas del osciloscopio (figuras 5.2.1-3), donde la entrada es una señal sinusoidal, y se muestra la salida de Arduino, después de haber convertido esta señal en digital, y posteriormente haberla generado por medio de modulación PWM. Para la señal de entrada se generó un barrido con una señal sinusoidal desde 20 Hz hasta 3906 Hz, ya que la frecuencia de muestreo de Arduino este ajustada a 7812 Hz. Se muestran los diferentes ciclos de trabajo de la señal PWM.

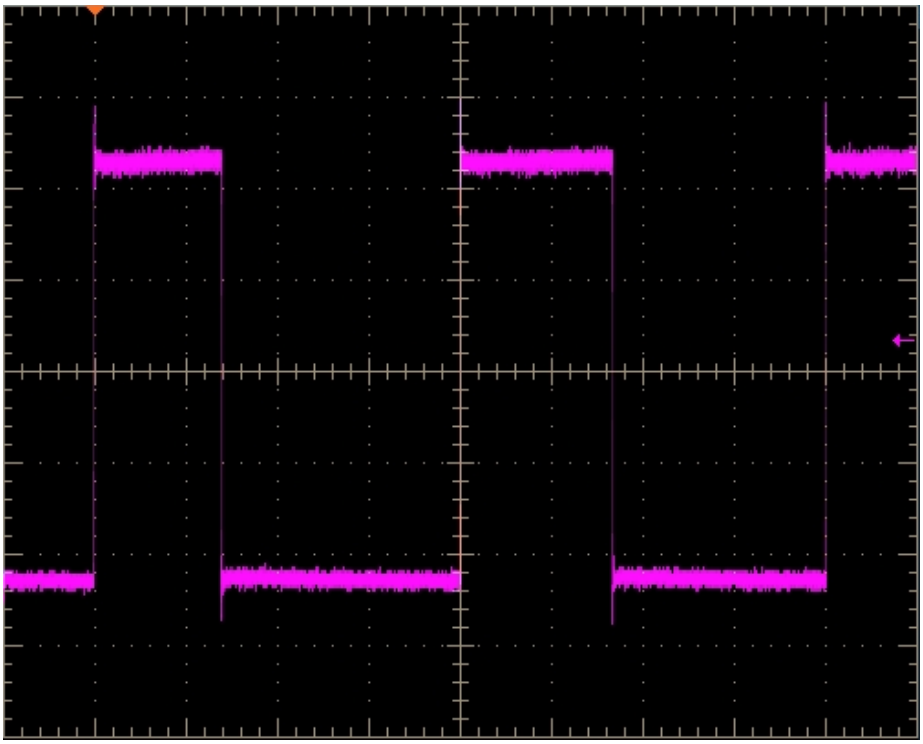


Figura 5.2.1. Captura de osciloscopio.

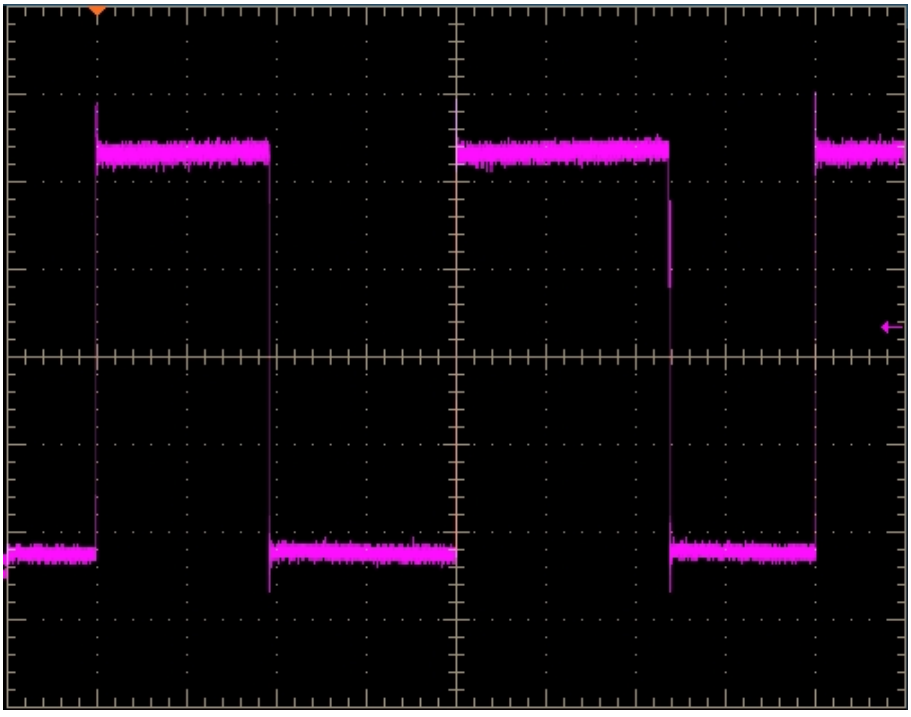


Figura 5.2.2. Captura de osciloscopio.

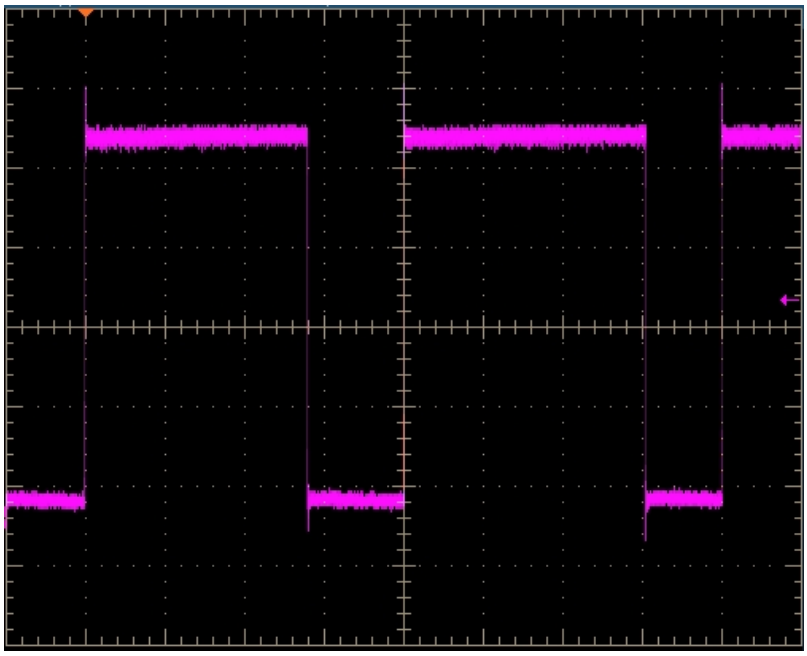


Figura 5.2.3. Captura del osciloscopio.

En las figuras anteriores, se muestra como Arduino al no tener conversor digital analógico genera la señal analógica de salida por medio de modulación PWM. Las tres figuras están ordenadas de menor a menor ciclo de trabajo de la señal. La primera figura correspondía con los instantes de tiempo donde el seno tiene menor amplitud, y la última figura correspondería con los instantes donde el seno alcanza sus máximos de amplitud, ya que el ciclo de trabajo es mayor. En los siguientes párrafos se describe como una modulación PWM se puede utilizar para generar una señal analógica.

Una señal PWM está compuesta por pulsos cuadrados, estos pulsos siempre tienen el mismo periodo, sin embargo su duración (duty cicle) depende de la amplitud de la señal original, tal y como se muestra en la figura 5.2.4.

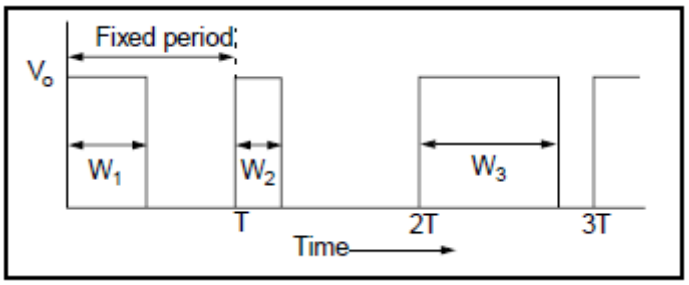


Figura 5.2.4. Ejemplo de una señal PWM.

Al hacer un análisis espectral aparecen varios armónicos no deseados en la señal, siendo el primero el que posee mayor amplitud de todos ellos, este armónico corresponde a la frecuencia de la señal PWM, en este caso a 62.5KHz.

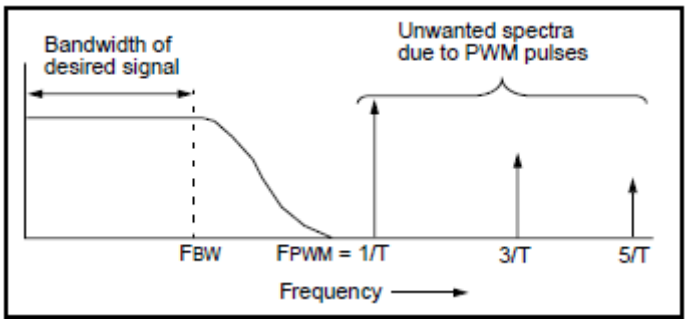


Figura 5.2.5. Espectro de una señal PWM

Teóricamente la señal original podría llegar en este caso hasta 62.5KHz que es donde se sitúa el primer armónico, y a partir de ahí realizar un filtrado paso bajo ideal, aunque usualmente el ancho de banda de la señal origina es siempre menor a la frecuencia de modulación (figura 5.2.5). En el caso que concierne se sabe que la señal analógica ha sido muestreada con una frecuencia de 7812 Hz, por lo que será necesario filtrar a partir de la mitad de la frecuencia de muestro según el teorema de Nyquist, para evitar el aliasing, y de esta manera se eliminará las componentes de orden superior producido por la modulación PWM de igual manera.

A continuación se presenta una captura del osciloscopio donde se representa la entrada sinusoidal de entrada, y la salida (figura 5.2.6). A la salida de Arduino se puso un filtro paso bajo RC con frecuencia de corte la mitad que la frecuencia de muestro, de esta manera tal y como se ha dicho con anterioridad se elimina el ruido producido por la conversión y por la modulación PWM.

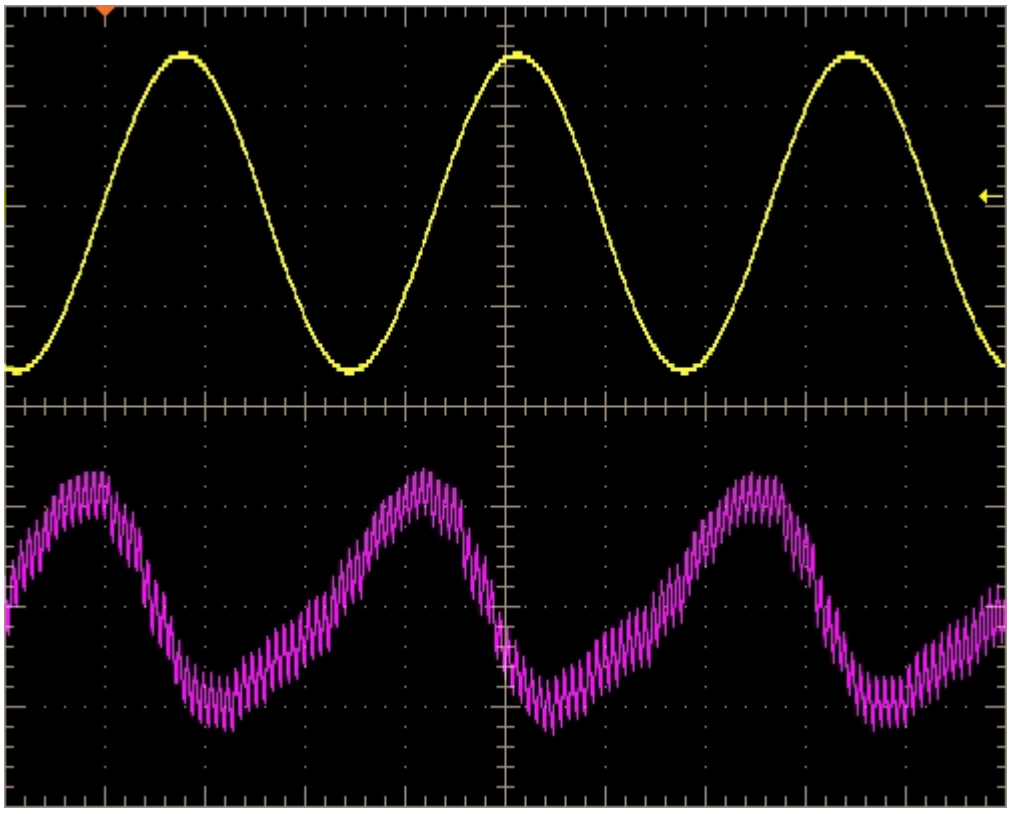


Figura 5.2.6. Captura del osciloscopio.

En la figura anterior se observa cómo el seno original es recuperado aunque con alguna distorsión debida al filtro, además de que no se ha quitado todo el ruido deseado, se observa un rizado en la envolvente del seno producido por frecuencias más altas que la de este. Esto es debido a que el filtro es de primer orden y es necesario diseñar un filtro de mayor orden para eliminar con mayor eficacia este ruido producido.

Sin embargo sabiendo que la frecuencia del seno era del 1500Hz se puso un condensador electrolítico en paralelo antes del filtro RC, y se consiguió eliminar la componente de alta frecuencia, ya que este condensador forma un filtro paso bajo, de frecuencia de corte 3183Hz con la resistencia de 50Ω de la salida de Arduino. A continuación se muestra una figura en la que se representa la señal original y la señal obtenida después del filtrado.

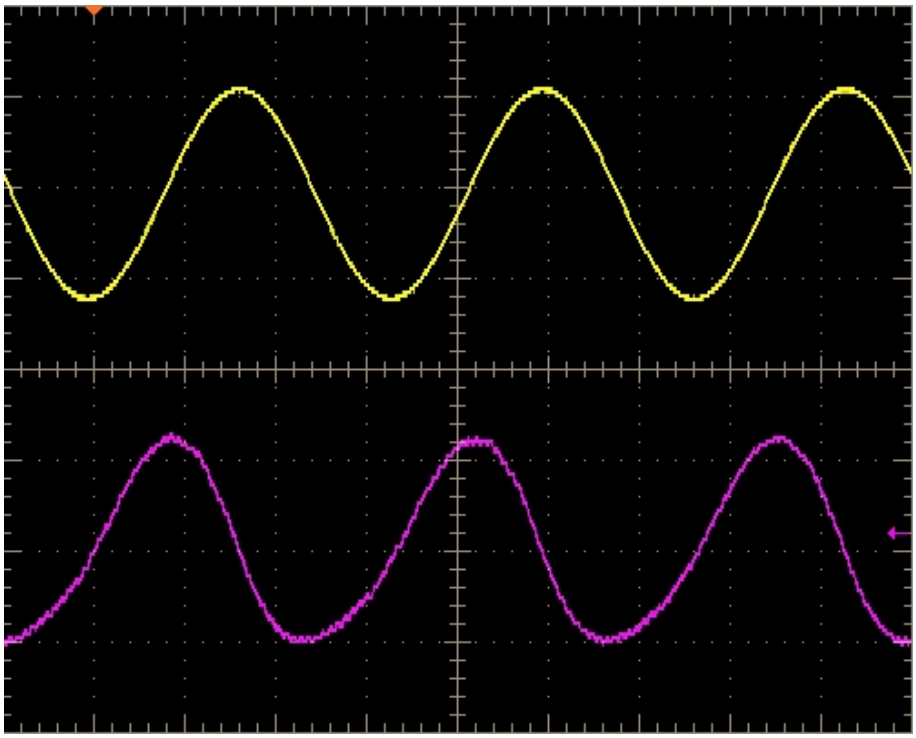


Figura 5.2.7. Captura del osciloscopio.

Como se puede observar la componente de alta frecuencia ha sido eliminada aunque la distorsión debida al filtrado es todavía notable.

En el CD de la memoria se pueden encontrar un fichero mp3 con los diferentes efectos probados donde se nota satisfactoriamente cada uno de ellos.

CAPÍTULO 6

CONCLUSIONES Y LINEAS FUTURAS

En este proyecto se ha diseñado una mesa mezclas analógica cumpliendo todas las características requeridas en los objetivos del diseño, y se ha conseguido usar Arduino como un procesador digital para llevar a cabo efectos digitales.

La mesa de mezclas está alimentada con 0 y cinco voltios, por lo que se tuvo que adaptar el diseño para que este pudiese funcionar con alimentación única. Uno de los objetivos principales del diseño era que la señal de audio tuviese más rango dinámico que la que tiene por normalidad, aprovechando que la mesa está alimentada hasta cinco voltios, se fue capaz de aprovechar este rango eficientemente, quedando la señal al final de la mesa dentro de un rango de 0-5V también. Para conseguir que la señal pudiese tener un rango tan amplio se hizo necesario el uso de amplificadores operacionales rail-to-rail.

Las etapas diseñadas dentro de cada canal cumplen las especificaciones más comunes dentro de una mesa de mezclas. El adaptador de impedancias, hace posible que cualquier tipo de entrada que se conecte a la mesa no quede atenuada, o modificada. El preamplificador cuenta con una ganancia máxima de 60dB. El ecualizador es de tres bandas, lo que permite al usuario mayor flexibilidad en cuanto a la ecualización. La etapa sumadora puede aportar a cada canal una ganancia de 6 dB o de -20 dB, y la etapa de master aporta el mismo rango de ganancias que la etapa sumadora.

Dentro de la parte que respecta a Arduino resaltar que al ser un hardware y software libre existe un gran número de aplicaciones creadas por usuarios, pero muy pocas de estas aplicaciones utilizan a Arduino como un procesador de la señal para llevar a cabo efectos de audio digitales. El objetivo del proyecto era la implementación por medio de este microcontrolador de efectos digitales, teniendo en cuenta las limitaciones inherentes propias. Como ya se ha explicado anteriormente el principal problema es su velocidad y su escasa memoria, por lo que hubo que adaptarse a sus prestaciones, y la solución adoptada fue reducir la frecuencia de muestreo considerablemente. Además Arduino UNO no cuenta con un conversor analógico digital, y la señal analógica de salida es generada mediante la modulación PWM, la cual puede ser usado como un conversor digital analógico de baja prestaciones que pudiera utilizarse para aplicaciones de voz de media calidad que podría utilizarse por ejemplo en juguetes.

Respecto a Arduino decir que es una plataforma muy versátil, su principal ventaja es su facilidad para programar y obtener resultados fácilmente, aunque en este caso se haya sacado prácticamente todo el rendimiento de este microcontrolador, manipulando sus registros etc… Arduino cuenta con instrucciones muy sencillas que facilitan la programación sin necesidad de manipular registros.

Líneas futuras

Estas limitaciones concernientes a Arduino pueden evitarse con una gran ventana de posibilidades. A continuación se van a exponer varias de ellas.

1) La primera posibilidad es la de cambiar Arduino UNO por Arduino DUE, ya que prácticamente el código seria idéntico. Este Arduino monta el chip AtmegaAT91SAM3X8E, que cuenta con más prestaciones que el Atmega328 que usa Arduino UNO. Sus principales ventajas son el aumento de memoria RAM (96 KB) y de velocidad (84MHz), además cuenta con dos salidas procedentes de DAC, lo que permitiría una mayor calidad. Con este Arduino sería posible la comunicación con un FIFO externo en paralelo, aunque no sería necesario ya que cuenta con 96KB de memoria RAM. El inconveniente de esta opción es el precio de Arduino DUE ya que duplica el precio del Arduino UNO.

2) La segunda opción es la de reemplazar Arduino por un DSP (digital signal procesor), debido a sus bajas prestaciones en comparación con un DSP. Los DSP son dispositivos muy utilizados en la actualidad debido a su bajo costo y sus prestaciones. Se puede adquirir un DSP con 256KB de memoria RAM, 16 bits de resolución, dos ALU´s… por la mitad de precio que Arduino UNO. Es verdad que el DSP tiene muchas menos opciones de uso que un microcontrolador, pero para este caso en particular sería una buena solución.

3) La siguiente opción es la de usar un conversor analógico digital y digital analógico externo, de esta manera Arduino haría las funciones de una ALU (arithmetic logic unit) únicamente operando con las muestras de entrada. De esta manera no se solucionaría el problema de la memoria pero se podrían permitir otro tipo de efectos que implican operaciones como la multiplicación.

4) A la salida de Arduino se ha diseñado un filtro paso bajo RC, para eliminar la frecuencia imagen creada después de la conversión. Sin embargo para eliminar estas frecuencias de una manera efectiva es necesario filtro de orden elevado. Muchos DSP´s incluyen este filtro, pero Arduino no lo incluye, y es por eso por lo que se barajó la posibilidad de introducir un filtro paso bajo de orden elevado a la salida de este. En el mercado existe gran variedad de circuitos intregados, que son filtros analógicos en los cuales el usuario puede diseñar varios parámetros, como frecuencia de corte, factor de calidad…Muchos de esos requieren señal de clock sin embargo se encontró un circuito integrado que cumplía con los requerimientos para este caso, el modelo MAX 274. Es un integrado que permite generar respuesta de filtros paso bajo, de tipo, Chebyshev, Butterworth, Bessel…Permite diseñar filtros de orden ocho como máximo. De esta manera la frecuencia imagen sería filtrada de una manera más efectiva.

En el PFC se han llevado todas las facetas de un desarrollo mixto analógico utilizando como herramienta Cadence Orcad, para realizar las capturas esquemáticas, simulaciones con Pspice y creación PCB con la herramienta PCB Editor. Por otra parte, para llevar a cabo la parte digital se ha utilizado editores de texto, compiladores y programadores USB de microcontroladores basado en Arduino.

ANEXO I

CÓDIGO DE LOS EFECTOS DIGITALES

```
// Incluir libreria para escribir en la Flash
#include <avr/pgmspace.h>
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))

// Definición de variables
longx=0 ;
inty=0;
intsalida;
intsalida1;
intbuffer1;
intosc;
bytesenal ;
intsenal1;
intfs=0;
PROGMEM bytecoseno[]={30,30,…,30};
bytebuffer_r[1700];
bytebuffer[120];
intpini =A0; / /declaracion pines
intpinpot=A1;
intpino=3;
intcantidad = 0;
constintbuttonpin= 2;
intbuttonState2 = 0;

voidsetup()
{
Serial.begin(9600); // Iniciar velocidad para comunicacion
Serie

pinMode(pino,OUTPUT);// Definicion de los pines
pinMode(buttonpin, INPUT);
pinMode(buttonpin2, INPUT);

cbi(ADCSRA,ADPS2); // prescale del ADC
sbi(ADCSRA,ADPS1);
sbi(ADCSRA,ADPS0);

cbi(ADMUX,REFS1); // voltaje referencia para ADC
sbi(ADMUX,REFS0);

cbi(ADMUX,MUX3); //seleccionar la entrrada A0 del
multipexor del ADC
cbi(ADMUX,MUX2);
cbi(ADMUX,MUX1);
cbi(ADMUX,MUX0);

sbi(TCCR2A,WGM21);
sbi(TCCR2A,WGM20);
sbi(TCCR2A,COM2B1); //Modo no inverting
cbi(TCCR2A,COM2B0);
```

```
cbi(TCCR2B,CS22); // Prescaler del timer 2 puesto a 1
cbi(TCCR2B,CS21);
sbi(TCCR2B,CS20);

sbi(DDRB,3); / / Activar el puerto de PWM del Timer2

cbi (TIMSK0,TOIE0); // Desactivar el timer0
sbi (TIMSK2,TOIE2); // Activar el timer2
}
voidloop()
{
// Condicion para entrar en el bucle
if(efecto==1){
efecto=0;
// Leer el estado de los interruptores
if(buttonState == HIGH && buttonState2== LOW) {
// Eliminar el nivel de DC
senal1=senal-127;
// Coger la muestra del buffer
buffer1=buffer_r[x]-127;
// Añadir a la senal original la retrasada
salida = (buffer1*cantidad)/255+senal1;
//Cctualizar buffer
buffer_r[x]=senal;
// Incrementar x sin que sobrepase el tamano del buffer
x++;
if(x>1800){x=1;}
// Limitar la señal
if(salida<-127){
salida=-127;}
if(salida>127){
salida=127;}
// Sumar el nivel de DC
salida1=salida+127;
// Enviar la salida al registro del timer2
OCR2B=salida1;
if(buttonState2 == HIGH &&buttonState == LOW) {
// Incremento de x sin sobrepasar el tamaño del buffer
x++;
if(x>120){x=1;}
// Incremento de y sin sobrepasar el tamaño de la tabla con
el coseno
y++;
if(y>7800){y=1;}
// Leer la tabla en la Flash con el coseno
osc=pgm_read_byte(&coseno[y]);
osc=osc/4;
// Eliminar el nivel de DC
senal1=senal-127;
//Sumar a la senal la muestra retrasada variable
if(x-osc>0){
salida=senal1+(buffer[x-osc]-127);
}
else{
salida=senal1+(buffer[x-osc+120]-127);
}
// Almacenar en el buffer la senal
buffer[x]=senal;
```

```
// Limitar la senal
salida=127;}
// Sumarle el nivel de DC
salida1=salida+127;
// Mandar al registro del timer2 la señal
OCR2B=salida1;
}
else{
if(buttonState2 == HIGH &&buttonState == HIGH) {
// Incremento de x sin sobrepasar el tamaño del buffer
x++;
if(x>120){x=1;}
// Incremento de y sin sobrepasar el tamaño de la tabla con
el coseno
y++;
if(y>7800){y=1;}
// Leer la tabla en la Flash con el coseno
osc=pgm_read_byte(&coseno[y]);
// Eliminar el nivel de DC
senal1=senal-127;
//Obtener la muestra retrasada variable
if(x-osc>0){
salida=(buffer[x-osc]-127);
}
// Almacenar la muestra
buffer[x]=senal;
// Limitar la senal de salida
if(salida<-127){
salida=-127;}
if(salida>127){
salida=127;}
// Agregarle el nivel de DC
salida1=salida+127;
// Enviar al registro del timer2 la senal
OCR2B=salida1;
}
else{ OCR2B=senal;// La senal sin ningun efecto
}
}
}
}
}
// Interrupcion creada por el timer2
if(fs==7){
fs=0;
senal=ADCH;// Coger la senaldespues de la conversion del
registro ADCH
efecto=1;// Efecto=1 para que entre en el bucle
sbi(ADMUX,MUX0); / / Cambiar el multiplexor canal 1
sbi(ADCSRA,ADSC); / / Empezar la siguiente conversion
}
if(fs==2){
cantidad=ADCH;//Coger la cantidad despues de la conversion
del registro ADCH
cbi(ADMUX,MUX0); / / Cambiar el multiplexor canal 0
sbi(ADCSRA,ADSC); / / Empezar la siguiente conversion
}
}
```

Para la realización de este programa se necesita una tabla la cual almacena todos los valores de en coseno, se realizó un programa en Matlab que crea esta tabla y el cual se adjunta en el Anexo II.

ANEXO II

PROGRAMA EN MATLAB PARA ALMACENAR VALORES DE UN COSENO EN UNA TABLA

```
vector='';
oscilador=0;
fori=1:7900
oscilador(i)=60+120*0.5*cos(2*pi*i/(7900/10));
    osc1=ceil(oscilador(i));
osc=num2str(osc1);
vector1=[','osc];
vector=[vector vector1];
end
```

ANEXO III

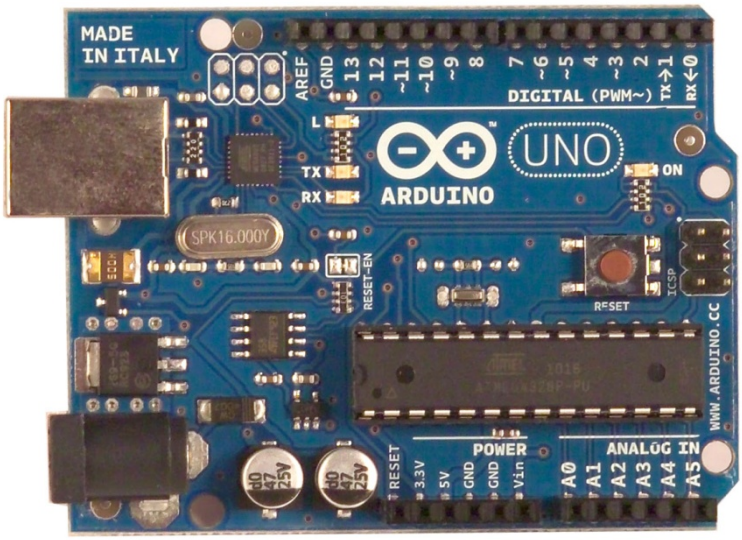
TUTORIAL DE ARDUINO

1- Primeros pasos

1.1- Consigue un Arduino y un cable USB

En este tutorial asumimos que estás usando un placa UNO. Si tienes cualquier otra placa necesitas leer la página correspondiente a la placa que uses en esta guía de iniciación.

También necesitarás un cable estandar USB (Type A aType B), como los que se usan para conectar, por ejemplo, una impresora USB. (En el caso de la placa Arduino Nano necesitarás un cable de conexión A a conexión mini-B).



1.2- Descarga el IDE de Arduino

Descarga la última versión de la página de descargas.

Cuando la descarga finalice, descomprime el fichero. Asegúrate de mantener la estructura de directorios. Haz doble click en la carpeta *arduino-00XX* para abrirla. Deberías ver una serie de ficheros y carpetas ahí dentro.

1.3- Conecta la placa

Conecta la placa Arduino a tu ordenador usando el cable USB. el LED verde indicador de la alimentación (nombrado como **PWR** en la placa) debería quedar encendido a partir de ese momento.

1.4- Instala los *drivers*

Cuando conectas la placa, Windows debería inicializar la instalación de los *drivers* (siempre y cuando no hayas utilizado ese ordenador con una placa Arduino anteriormente).

En Windows Vista y Windows 7, los drivers deberían descargarse e instalarse automáticamente.

En Windows XP, se abrirá el diálogo de instalación de Nuevo Harware:

- Cuando te pregunten: **¿Puede Windows conectarse a Windows Update para buscar el software?** selecciona **No, no esta vez**. Haz click en *Siguiente*.
- Selecciona **Instalar desde una lista o localización específica (Avanzado)** haz click en *Siguiente*.
- Asegurate que **Buscar los mejores drivers en estas localizaciones** esté seleccionado; deselecciona **Buscar en medios removibles**; selecciona **Incluye esta localización en la búsqueda** y navega al directorio **drivers/FTDI USB Drivers** dentro de la carpeta de Arduino que has descomprimido previamente. (La versión más reciente de los *drivers* se puede encontrar en la página web del fabricante delchip FTDI.) Haz click en *Siguiente*.
- El asistente de instalación buscará los *drivers* y te anunciará que encontró un "USB Serial Converter" (se traduce por *Conversor USB-Serie*). Haz click en *Finalizar*.
- El asistente de instalación de hardware volverá a iniciarse. Repite los mismos pasos que antes y selecciona la misma carpeta de instalación de los *drivers*. Esta vez el sistema encontrará un "USB Serial Port" (o *Puerto USB-Serie*).

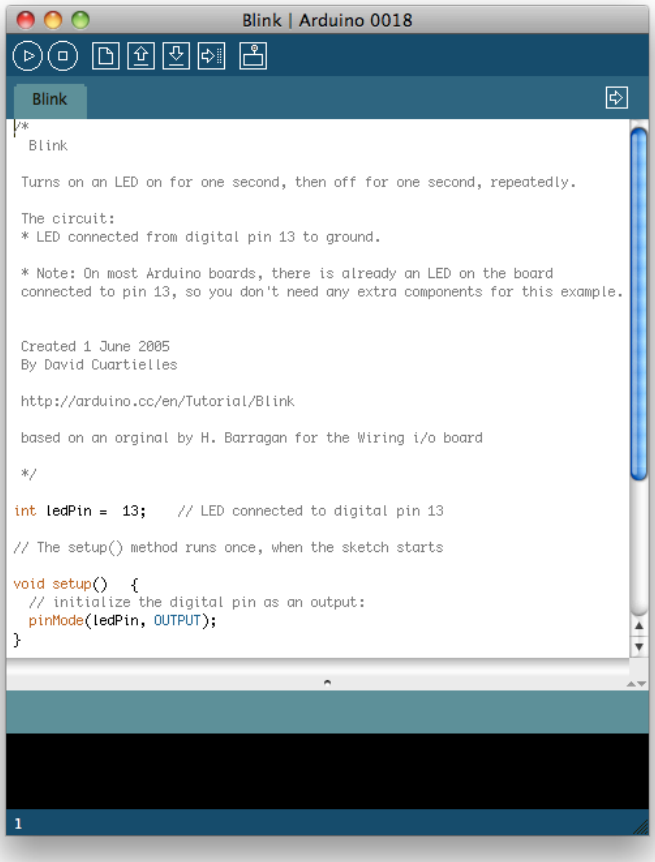
Puedes comprobar que los *drivers* se han instalado correctamente abriendo la carpeta del Administrador del Dispositivos, en el grupo *Dispositivos* del panel de control del sistema. Busca "USB Serial Port" (o *Puerto USB-Serie*)en la sección de puertos; esa es tu placa Arduino.

1.5- Ejecuta la Aplicación Arduino

Haz doble click en la aplicación Arduino.

1.6- Abre el ejemplo *Blink*

Abre el programa de ejemplo para hacer parpadear un LED ("LED blink"): **File >Examples> Digital >Blink**.



1.7- Selecciona tu placa

Necesitarás seleccionar el tipo de placa de tu Arduino en el menu**Tools >Board**. Para las nuevas placas Arduino con el chip ATmega 328 (comprueba el texto escrito en el chip de la placa).

1.8- Selecciona tu puerto serie

Selecciona el dispositivo serie de la placa Arduino en el menú Tools | Serial Port (*Herramientas* | *Puertos Serie*). Lo más probable es que sea **COM3** o mayor (**COM1** y **COM2** se reservan, por regla general para puertos serie de hardware). Para asegurarte de cual es, puedes desconectar la placa y volver a mirar el menú; el puerto de la placa habrá desaparecido de la lista. Reconecta la placa y selecciona el puerto apropiado.

1.9- Sube el *sketch* a la placa

Ahora simplemente pulsa sobre el botón "Upload" en el Entorno Arduino. Espera unos pocos segundos - deberías ver parpadear los led RX y TX de la placa. Si el volcado del código es exitoso verás aparecer el mensaje "Done uploading" en la barra de estado. (*Aviso:* Si tienes una placa Arduino Mini, NG, u otras placas, necesitarás presionar el boton de reseteo de la placa inmediatamente antes de presional el botón "Upload" el Entorno de programación Arduino.)

Unos pocos segundos despues de finalizar el volcado del programa deberías ver cómo el led de la placa conectado al pin 13 (L) comienza a parpadear (con un color naranja). Si ocurre esto ¡enhorabuena! Ya tienes tu Arduino listo y funcionando.

2- Estructura Arduino

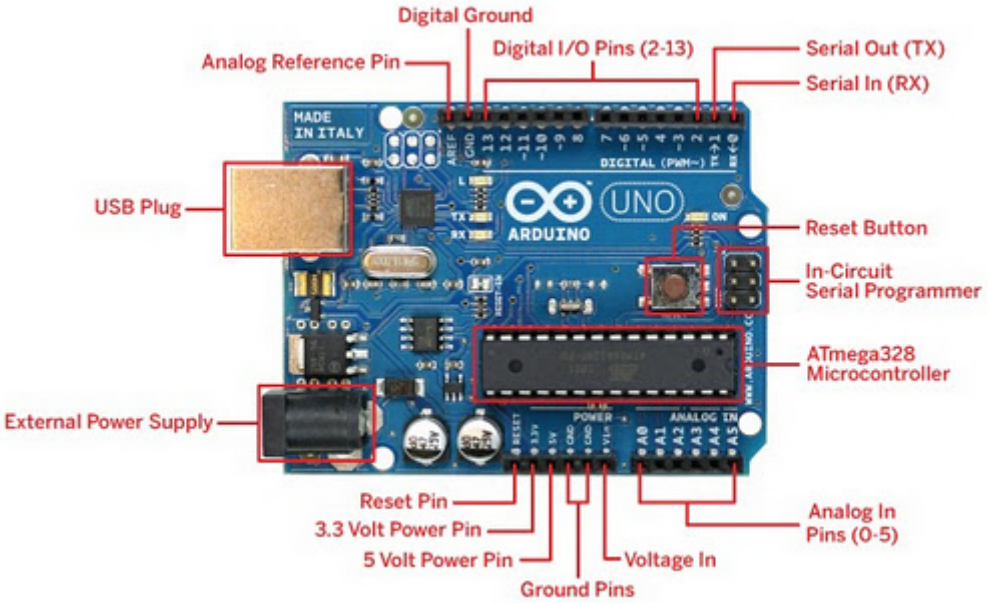
Bloque 0 – Comentarios (OPCIONAL)

Bloque 1 – Declaración de las variables que vamos a utilizar

Bloque 2 – Configuración inicial del programa

Bloque 3 – Bucle infinito que contiene el conjunto de instrucciones que se repiten constantemente

3- Pines de Arduino UNO



4- Lenguaje y comandos básicos

• Lenguaje básicamente en C (más simple que el C++ de flash, y menos variantes que processing).

• **pinMode ()** – ajusta el pin en salida (input) o en entrada (output).

• **digitalWrite ()** – ajusta el pin digital en encendido/ apagado (high/low).

• **digitalRead()** – lee el estado del pin digita.

• **analogRead()** – lee el estado del pin analógico.

• **analogWrite()** – ajusta el pin analógico en la variante de valores

• **delay()** – cantidad de tiempo en espera.

• **millis()** – actualiza el tiempo.

Salidas Digitales

Los pines digitales se pueden emplear como salidas o entradas digitales, es decir, se puede escribir niveles altos (5V) o bajos (0V) de tensión a cada uno de los pines y viceversa, excepto los pines 0 (TX) y 1 (RX) que se emplean para la comunicación serie o comunicación de Arduino con otros dispositivos.

Comandos básicos:

pinMode(pin, modo), sirve para declarar un pin digital como entrada (INPUT) o como salida (OUTPUT). Los pines analógicos son, por defecto,de entrada.

delay(tiempo), sirve para parar los procesos de la placa durante tiempo milisegundos y delayMicroseconds(tiempo) sirve para parar los procesos de la placa durante tiempo microsegundos.

digitalWrite(pin, valor),sirve para escribir un valor al pin digital, el valor podrá ser 1 lógico (HIGH=5v) o 0 lógico (LOW=0v)

setup() es la función de configuración de los pines de Arduino y sólo se ejecuta una vez, mientras que loop() se ejecuta una y otra vez hasta que apaguemos el sistema, o se gasten las baterías.

Comandos básicos:

digitalRead(pin),sirve para leer un valor del pin digital que señalemos, el valor podrá ser 1 lógico (HIGH=5v) o 0 lógico (LOW=0v)

Entradas digitales

– Divisor de tensión resistivo

Un divisor de tensión es una configuración de circuito eléctrico que reparte la tensión de una fuente entre una o más impedancias conectadas en serie.

Un divisor resistivo es un caso especial donde ambas impedancias, son puramente resistivas.

Se utiliza para leer valores de sensores.

• Arduino sólo puede leer voltajes, y muchos sensores son resistivos (varían la resistencia). Por eso hay que usar circuitos de este tipo para leer el valor de los sensores.

Comandos básicos:

digitalRead(pin),sirve para leer un valor del pin digital que señalemos, el valor podrá ser 1 lógico (HIGH=5v) o 0 lógico (LOW=0v)

Entradas analógicas

Los **pines analógicos** se emplean como entradas analógicas, es decir, se puede recibir tensiones entre 5V y 0 voltios. Los pines analógicos, al contrario que los pines digitales, no necesitan ser declarados como modo INPUT (entrada) o OUTPUT (salida).

Conversión analógico-->digital(ADC) en Arduino: consiste en transformar un valor de_

tensión en un número que pueda ser comprendido por un dispositivo de lógica digital. Arduino puede convertir tensiones de 0 a 5 voltios en números enteros que van del 0 al 1023. En otras palabras representa la información en números de 10 bits (resolución).

Comandos básicos:

analogRead(pin), Lee o captura el valor de entrada del especificado pin analógico, la tarjeta Arduino realiza una conversión analógica a digital de 10 bits. Esto quiere decir que mapeará losvalores de voltage de entrada, entre 0 y 5 voltios, a valores enteros comprendidos entre 0 y 1023

Salidas analógicas

Arduino dispone de varios pines para generar salidas PWM, a través algunos de los pines digitales. Dependiendo del modelo de la placa y sobre todo del chip Atmega de que disponga la placa tendremos 3 o 6 salidas PWM, que están marcadas en la placa:

A diferencia de las entradas analógicas, en las que el conversor analógico digital nos daba un valor entre 0 y 1023, para generar una salida digital el rango es de 0 a 255. Donde 0 equivale a 0V y 255 a 5V

Los pines analógicos, al contrario que los pines digitales, no necesitan ser declarados como modo INPUT(entrada) o OUTPUT (salida).

Comandos básicos:

analogWrite(pin, value), Escribe el valor especificado en el pin PWM correspondiente. Dicho valor, como se ha mencionado, tiene que estar entre 0 y 255.

5- Algunos ejemplos

BUTTON

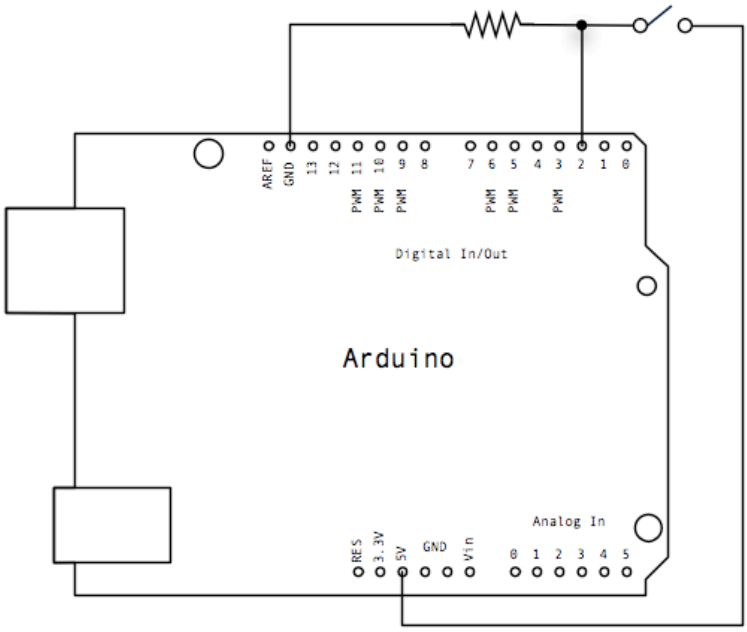
Los pulsadores o switches conectan dos puntos de un circuito al ser pulsados. Este ejemplo enciende el LED integrado en el pin 13 cuando pulsas el botón.

Conecta tres cables a la placa Arduino. Los dos primeros, rojo y negro, conectan a las dos hileras largas verticales de los laterales de la placa universal (*breadboard*) para proporcionar acceso a la fuente de 5 voltios y a masa (*ground*). El tercer cable va desde el pin digital 2 a una patilla del pulsador. Esa misma patilla del pulsador se conecta a través de una resistencia *pull-down* (en este caso 10 KOhms) a masa. El otro extremo del pulsador se conecta a la fuente de 5 voltios.

Cuando el pulsador está abierto (sin pulsar) no hay conexión entre las dos patas del pulsador, de forma que el pin está conectado a tierra (a través de la resistencia *pull-down*) y leemos un LOW (bajo ó 0). Cuando el botón se cierra (pulsado), se establece la unión entre sus dos extremos, conectando el pin a 5 voltios, por lo que leemos un HIGH (alto ó 1).

También puedes cablear el circuito en sentido contrario, con una resistencia "pull-up" que mantenga la entrada en HIGH, y que pase a LOW cuando se pulse el botón. Así, el comportamiento del programa (sketch) se invertirá, con el LED normalmente encendido y apagado cuando se pulsa el botón.

Si se desconectas el pin digital de E/S del todo, el LED puede parpadear de forma errática. Esto se debe a la entrada es "flotante", es decir, al azar se tornará en HIGH o LOW. Por eso se necesita la resistencia *pull-up* o *pull-down* en el circuito.



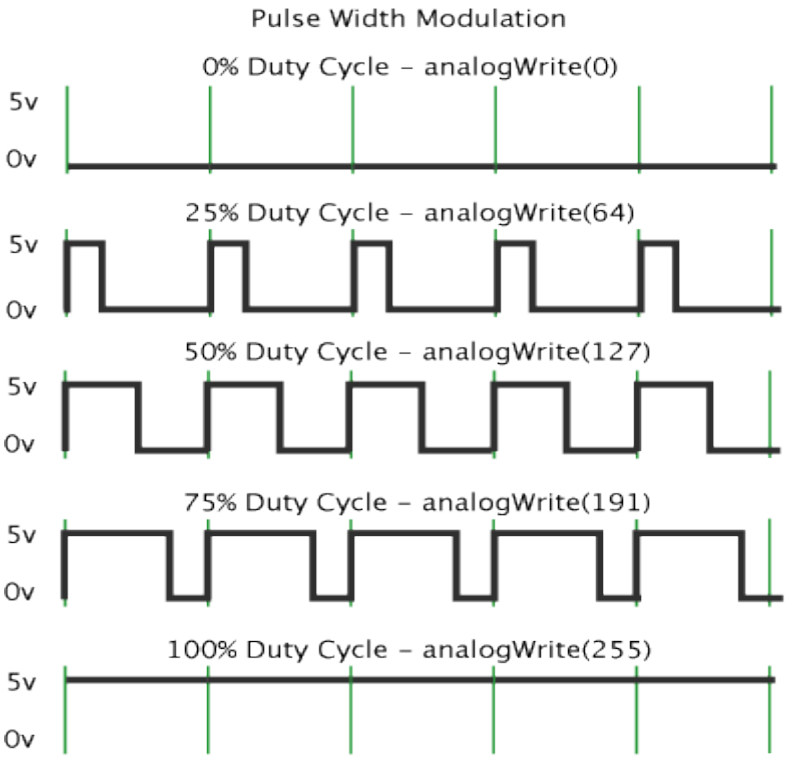
Sketch:

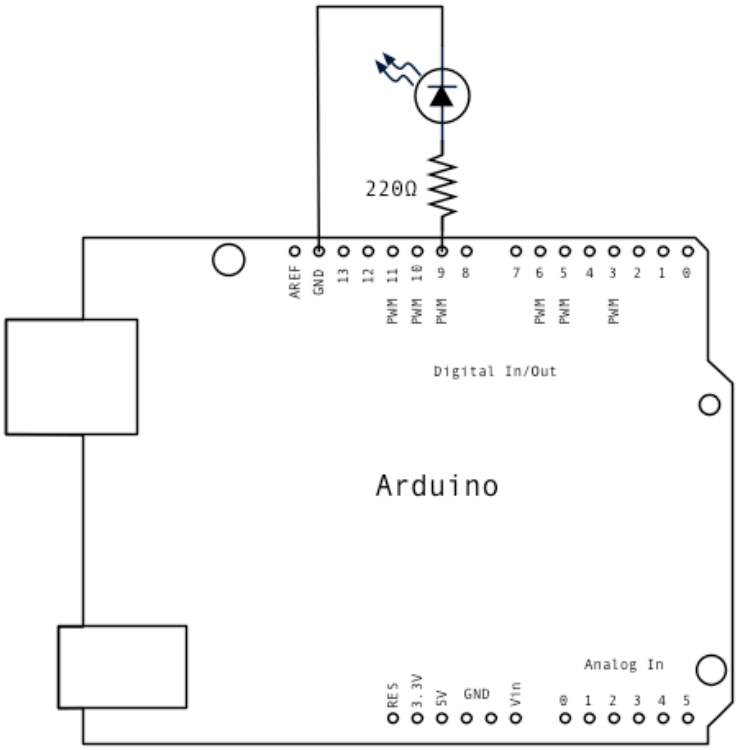
```
// set pin numbers:

constintbuttonPin = 2;     // the number of the pushbutton pin

constintledPin =  13;      // the number of the LED pin

// variables will change:

intbuttonState = 0;         // variable for reading the pushbutton status

voidsetup() {

  // initialize the LED pin as an output:

pinMode(ledPin, OUTPUT);

  // initialize the pushbutton pin as an input:

pinMode(buttonPin, INPUT);

}



voidloop(){
```

```
 // read the state of the pushbutton value:
buttonState = digitalRead(buttonPin);
 // check if the pushbutton is pressed.
 // if it is, the buttonState is HIGH:
if(buttonState == HIGH) {
   // turn LED on:
digitalWrite(ledPin, HIGH);
 }
else {
   // turn LED off:
digitalWrite(ledPin, LOW);
}
```

FADING

Demuestra el uso de la salida analógica (PWM) para apagar gradualmente un LED.





Sketch:

```
intledPin = 9;    // LED connected to digital pin 9

voidsetup()  {
  // nothing happens in setup
}

void loop()  {
  // fade in from min to max in increments of 5 points:
for(intfadeValue = 0 ; fadeValue<= 255; fadeValue +=5) {
    // sets the value (range from 0 to 255):
analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
delay(30);
  }

  // fade out from max to min in increments of 5 points:
for(intfadeValue = 255 ; fadeValue>= 0; fadeValue -=5) {
    // sets the value (range from 0 to 255):
analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
delay(30);
  }
}
```

BIBLIOGRAFÍA

[1]-CREACIÓN DE NUEVOS COMPONENTES PARA ORCAD 10.3. Miguel Pareja Aparicio. Ediciones técnicas Marcombo

[2]-THE SPICE HANDBOOK OF 50 BASIC CIRCUITS. Steven M. Sandler and Analytical Engineering, Inc.

[3]-ELECTRÓNICA. Allan R. Hambley. Pearson, Prentice Hall

[4]-EQUIPOS DE SONIDO. Francisco Ruiz Vassalloo

[5]-OPERATIONAL AMPLIFIERS SELECTION GUIDE. Analog Devices

[6]-ARDUINO COOKBOOK. Michael Margolis

[7]- ATMEL Atmega 328 Microcontroler Datasheet

[8]- SINGLE-SUPPLY CIRCUIT COLLECTION, Texas Instruments, Appendix A, Literature Number SLOA091

[8]-A SINGLE-SUPPLY OP-AMP CIRCUIT COLLECTION, Texas Instruments, Literature Number SLOA058

[9]- CHARACTERIZATION AND CALIBRATION OF THE ADC ON AN AVR, ATMEL.

[10]- http://www.arduino.cc/