UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

"INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES"

TÍTULO:

"DISEÑO E IMPLEMENTACIÓN DE UN CROSSOVER DE TRES VÍAS UTILIZANDO DSP CON PARÁMETROS CONTROLADOS POR UN DISPOSITIVO ANDROID"

AUTOR:

DARÍO JAVIER ASTUDILLO AMÁN

DIRECTOR:

ING. DEYSI INCA

RIOBAMBA – ECUADOR

2016

Los miembros del Tribunal de Graduación del proyecto de investigación del título: DISEÑO E IMPLEMENTACIÓN DE UN CROSSOVER DE TRES VÍAS UTILIZANDO DSP CON PARÁMETROS CONTROLADOS POR UN DISPOSITIVO ANDROID presentado por Darío Javier Astudillo Amán y dirigido por: Deysi Vilma Inca Balseca.

Una vez escuchada la defensa oral y revisado el informe final del proyecto de investigación con fines de graduación escrito en los cuales se ha constatado el cumplimiento de las observaciones realizadas, remite la presente para uso y custodia en la biblioteca de la Facultad de Ingeniería de la Universidad Nacional de Chimborazo.

Para constancia de lo expuesto firman:

Ing. Paulina Vélez ---------- **Firma**

Presidente del Tribunal

Ing. Deysi Inca ---------- **Firma**

Directora del Proyecto

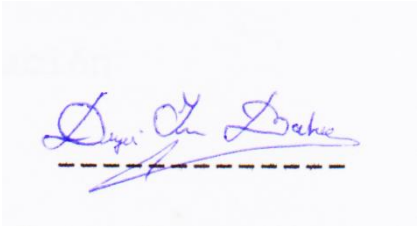
Ing. Geovanny Cuzco ---------- **Firma**

Miembro del Tribunal

CETIFICACIÓN DEL TUTOR

Certifico que el presente trabajo de investigación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones. Con el tema "*Diseño e implementación de un crossover de tres vías utilizando DSP con parámetros controlados por un dispositivo Android*" ha sido elaborado por Darío Javier Astudillo Amán, el mismo que ha sido revisado y analizado en un cien por ciento por el asesoramiento de mi persona en calidad de Directora, por lo que se encuentra apto para su presentación y defensa respectiva.

Es todo cuanto puedo informar en honor a la verdad.

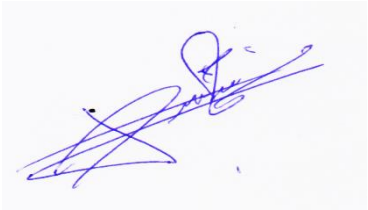


Ing. Deysi Inca

Tutora del Proyecto de Investigación

AUTORÍA DE LA INVESTIGACIÓN

"La responsabilidad del contenido de este Proyecto de Graduación, nos corresponde exclusivamente a: Darío Javier Astudillo Amán e Ing. Deysi Vilma Inca Balseca; y el patrimonio intelectual de la misma a la Universidad Nacional de Chimborazo".



Darío Javier Astudillo Amán
C.I. 160069199-0

AGRADECIMIENTO

A Dios, ya que gracias a su providencia, logré culminar este Trabajo de Titulación; a mi esposa Diana y mi hija Kristen que día a día me mostraron su comprensión y apoyo; a mis padres Elsa y Kléber, y mis abuelitos Abelardo y Gonzalina; quienes de muchas maneras han colaborado con la conclusión de este proyecto.

A mis profesores, especialmente a la Ing. Yesennia Cevallos, Ing. Fabián Gunsha e Ing. Deysi Inca, quienes compartieron su conocimiento conmigo en toda mi etapa estudiantil en la Universidad Nacional de Chimborazo.

Finalmente a todos mis amigos, de manera especial a Nathaly, Christian, Javier, Fray Mauricio Ruiz, Eddie García y Gilda Gutiérrez por todo el apoyo brindado; todos ellos tendrán mi eterna gratitud.

DEDICATORIA

Dedico este trabajo especialmente a Dios, Quien es lo más importante en mi mente y con quien cuento a cada minuto, también se lo dedico a Diana Barrionuevo y Kristen Astudillo, las mujercitas que cada vez que llego a casa me reciben con sus brazos abiertos; a mis hermanos Adrián y Verónica; de igual manera a mis padres Elsa y Kléber por su apoyo incondicional en toda mi etapa estudiantil.

ÍNDICE GENERAL

ÍNDICE DE CUADROS

ÍNDICE DE GRÁFICOS

RESUMEN

El propósito de este Trabajo de Investigación, es implementar un crossover de tres vías, que permita manipular sus frecuencias de corte de manera inalámbrica, mediante un dispositivo con una aplicación Android.

El crossover cuenta con tres filtros digitales (pasa bajos, pasa altos, pasa banda) diseñados a través de filtros FIR, implementados en microcontroladores con funciones de Procesamiento Digital de Señales (DSP) utilizando el método de ventaneo.

Con la interfaz Android es posible la manipulación de las frecuencias de corte del crossover digital, que permite mejorar el sistema y eliminar las frecuencias que no quieran ser percibidas.

Las señales de salida resultantes pueden ser escuchadas a través de audífonos independientes o a su vez en altavoces con un sistema de amplificación.

Las pruebas realizadas en este proyecto, como son la comunicación, la eficiencia de los filtros y la velocidad de procesamiento de los microcontroladores, han demostrado que este sistema electrónico cumple con los objetivos planteados.

UNIVERSIDAD NACIONAL DE CHIMBORAZO

CENTRO DE IDIOMAS INSTITUCIONAL

Ms. Edison Salazar 17 de Agosto de 2016

ABSTRACT

The purpose of this Investigation Project is to implement a three crossover channel that enables to manipulate its cutoff frequencies wirelessly, through a device with an Android application.

This crossover has three digital filters (low-pass, high-pass, band-pass) designed through FIR filters, implemented on microcontrollers with Digital Signal Processing functions (DSP) using windowing method.

With Android interface is possible to manipulate cutoff frequencies from digital crossover, which permits to improve the system and remove undesired frequencies.

Output signals can be listened through independent earphones or amplified speakers.

Test done in this Project, such as communication, filters efficiency and microcontrollers processing speed, had shown that this electronic system accomplishes the planed objectives.

INTRODUCCIÓN

El audio digital ha tenido avances vertiginosos, hasta el punto que la reproducción de audio digital es superior a la analógica.

Una reproducción acústica de óptima calidad requiere de filtros, amplificadores y un buen arreglo de altavoces.

En la actualidad existen crossover analógicos los cuales son utilizados para mejorar la calidad de sonido pero la limitante es que estos crossover cuentan con frecuencias de corte fijas; debido a este inconveniente se presenta la propuesta de diseñar un crossover de tres vías con frecuencias de corte variables.

La utilización de frecuencias de corte variables es viable ya que el audio digital presenta un espectro amplio de frecuencias siendo el alcance máximo del oído humano las frecuencias de 20 KHz.

Dentro del Procesamiento Digital de Señales la manera más adecuada para la manipulación de audio, es la utilización de filtros FIR debido a que su funcionamiento es estable. Estos filtros utilizan procesos matemáticos como la convolución y las sumatorias acarreadas para cumplir su propósito.

CAPÍTULO I

1.- FUNDAMENTACIÓN TEÓRICA

1.1.- CROSSOVER

Un crossover (filtro de cruce), está denominado como un circuito electrónico que divide frecuencias, es decir un dispositivo capaz de filtrar las frecuencias en la entrada, para que a la salida solo pase una determinada banda de frecuencias. Teniendo en cuenta que el margen que ocupa esta banda, depende de la configuración del sistema en sí mismo. (CUENCA, David, GOMEZ, Eduardo, 2001)

1.2.- MUESTREO

El muestreo está definido como la obtención de manera periódica de muestras de una señal analógica, a través de un conversor Análogo-Digital, con una frecuencia de muestreo: Fs[1]. El período de muestreo T, es el tiempo en segundos que existe entre las muestras. El muestreo permite convertir una señal continua en una señal discreta. (López, 2000) Para definir el periodo de muestreo T, se debe considerar que: la frecuencia de muestreo debe ser mayor que la máxima componente espectral contenida en la señal x (t), como mínimo el

[1] Frequency Sampling

doble. Este concepto es conocido como Teorema de muestreo de Nyquist. (López, 2000)

Sin embargo la frecuencia de muestreo puede ser mayor al doble del máximo en frecuencia de la señal x (t), cuanto mayor sea la frecuencia de muestreo, mayor será la fidelidad de la señal digitalizada, pero a su vez, mayores serán los recursos requeridos en velocidad y memoria de procesamiento por lo que deberían utilizarse dispositivos más robustos y con una mejor capacidad. (Clavijo Mendoza, 2011)

$$Fs \geq 2\, F_{max}$$

1.3.- FUNCIONES DE TRANSFERENCIA

Una función de transferencia es la relación existente entre la entrada de una señal y su salida después de pasar por un bloque de proceso. Las funciones de transferencia son estudiadas y manipuladas en términos de la frecuencia compleja y no en el dominio del tiempo continuo. (Clavijo Mendoza, 2011)

1.4.- AUDIO DIGITAL

El audio digital, está definido como la digitalización de sonido real, ya sea procedente de voces, instrumentos musicales acústicos o electrónicos, grabaciones, entre otras, para ser tratados en los microprocesadores. Es la representación de una señal audible mediante números, en general codificados en forma binaria. (López, 2000)

Para convertir una señal eléctrica a una señal digital, es necesaria la utilización de los ADC o conversores analógico-digital. En la **Figura 1**, puede observarse un ejemplo de este proceso.

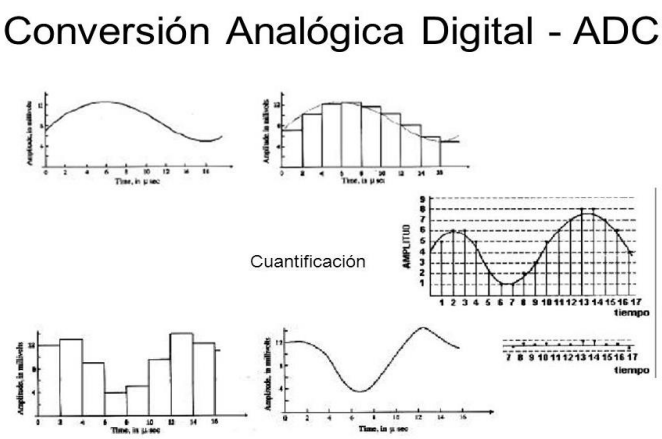


Figura 1.- Ejemplo de conversión Analógica-Digital
Fuente.-
http://images.slideplayer.es/1/33693/slides/slide_8.jpg

1.5.- CONVOLUCIÓN

Una convolución es un operador matemático que transforma dos funciones: h y k en una tercera función, que es la función resultante, o de salida, ésta representa la magnitud en la que son superpuestas h y una versión trasladada e invertida de k. Lo cual matemáticamente es expresado por:

$$y(n) = \sum_{k=-\infty}^{+\infty} [h(k)x(n-k)]$$

La ecuación anterior describe la forma general de la convolución, sin embargo, la longitud de la función h(n),

es finita y su máximo es M, por lo tanto la ecuación puede ser rescrita de la siguiente forma:

$$y(n) = \sum_{k=0}^{M} [h(k)x(n-k)]$$

Cada vez que una muestra de la salida y(n) es calculada por medio de la convolución, son requeridas M muestras de la señal x(n). (Clavijo Mendoza, 2011)

1.6.- FILTROS

Un filtro es un sistema que permite el paso de las componentes de la señal existentes en un determinado intervalo frecuencial (banda de paso) y no deja pasar al resto de señales (banda atenuada o de rechazo). (CONSTANTE CASTRO, 2003).

El ancho de banda de un filtro, comprende todas aquellas frecuencias capaces de atravesar el circuito, la amplitud de la señal de salida no debe ser menor al 70 % del valor de la señal aplicada; este punto es el de media potencia ó -3dB[2]. Las frecuencias que están dentro de los puntos de media potencia, son denominadas como frecuencias de corte. (CONSTANTE CASTRO, 2003)

1.6.1.- FILTROS DIGITALES

Un filtro digital, es un sistema que dependiendo de las variaciones de señales de entrada en el tiempo y amplitud, realiza un procesamiento matemático sobre dicha señal, mediante varios tipos de transformadas, o mediante el uso de operaciones matemáticas como la convolución,

[2] Decibelios

obteniéndose en la salida el resultado del procesamiento matemático o la señal de salida. (López, 2000)
Las operaciones básicas a realizarse son sumas acumulativas de productos en los que los factores varían en cada operación.

1.6.1.1.- FILTROS IIR

Los filtros IIR[3], son filtros digitales, en los que si la entrada es una señal impulso, la salida tendrá un número infinito de términos no nulos, es decir nunca vuelve al reposo. Estos filtros no presentan mucha carga computacional pero pueden ser inestables aun cuando fueren diseñados para ser estables. (López, 2000)
Los filtros IIR utilizan la siguiente ecuación:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] + \sum_{k=0}^{N} a_k y[n-k]$$

1.6.1.2.- FILTROS FIR

Los filtros FIR[4], son un tipo de filtros digitales cuya respuesta a una señal impulso como entrada tendrá un número finito de términos no nulos. Este tipo de filtros tiene especial interés en aplicaciones de audio. Además son siempre estables; por el contrario también tienen la desventaja de necesitar un orden mayor respecto a otros filtros para cumplir las mismas características. Esto quiere decir que requieren de un mayor gasto computacional. (López, 2000)
La ecuación que utilizan los filtros FIR es la siguiente:

[3] Infinite Impulse Response (Respuesta Infinita al Impulso)
[4] Finite Impulse Response (Respuesta Finita al Impulso)

$$y\,[n] = \sum_{k=0}^{N-1} a[k]\,x[n-k]$$

1.6.1.3.- VENTANAS EN LOS FILTROS DIGITALES

Las ventanas son aplicadas a las funciones de transferencia h(n), el objetivo de las ventanas es mejorar y suavizar la respuesta espectral de los filtros FIR. (Clavijo Mendoza, 2011)

Las ventanas de mayor uso son: Rectangular, Hamming, Hanning, Blackman. En la **Figura 2**, están detalladas las diferentes respuestas en frecuencia de las ventanas más utilizadas.

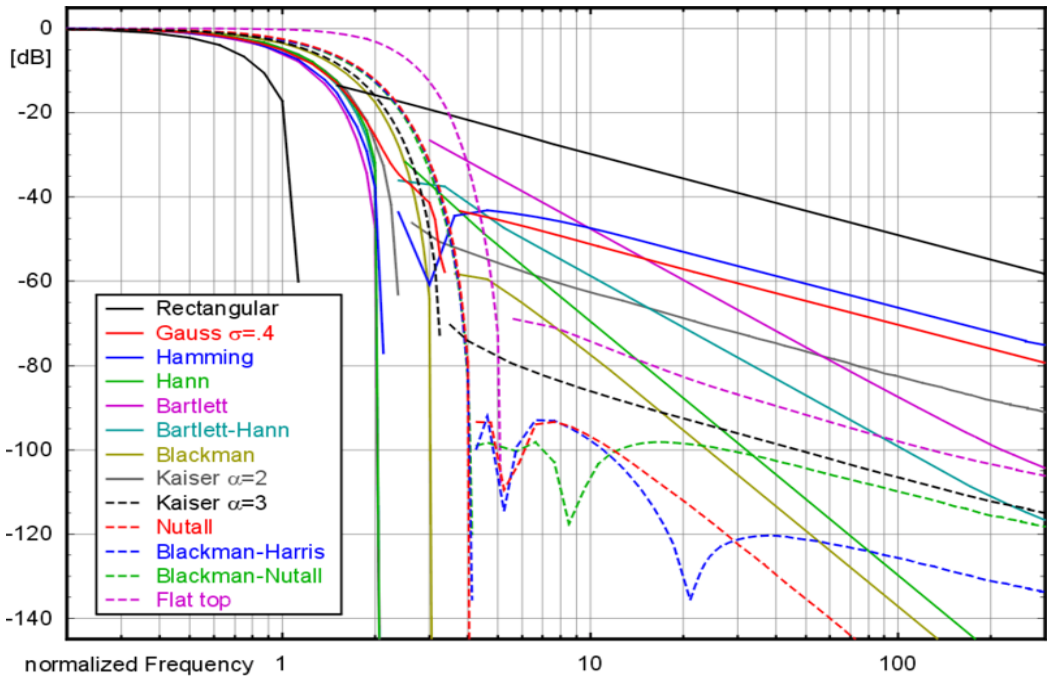


Figura 2.- Respuesta en frecuencias de las ventanas.

Fuente.- http://4.bp.blogspot.com/-1ymh3nBNfzA/UmCILZcPCII/AAAAAAAAAQQ/0ufuTxH2Dx4/s1600/FIR+1.jpg

1.6.1.4- FILTRO PASA BAJAS

Los filtros pasa bajas están caracterizados por tener una frecuencia de corte a partir de la cual las frecuencias inferiores son permitidas; y las frecuencias superiores son atenuadas. (Clavijo Mendoza, 2011) La respuesta en frecuencia de dicho filtro, está mostrada en la **Figura 3.**

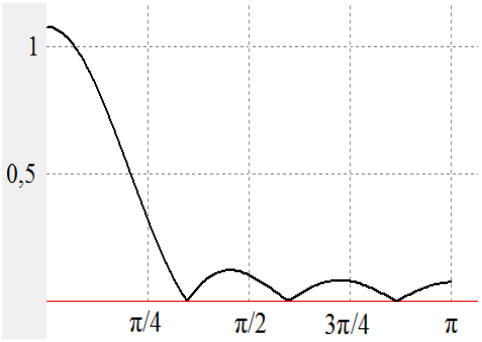


Figura 3.- Respuesta de un filtro pasa bajas de orden 17
Fuente: (Clavijo Mendoza, 2011)

La función de transferencia h(n) para el filtro digital es:

$$h(n) = \begin{cases} \dfrac{\sin(w_c n)}{\pi n}, n \neq 0 \\ \dfrac{w_c}{\pi}, n = 0 \end{cases}$$

1.6.1.5.- FILTRO PASA ALTAS

Los filtros pasa altas permiten el paso de las frecuencias superiores a la frecuencia de corte, y suprimen las frecuencias inferiores a la misma, como el ejemplo de la **Figura 4.**

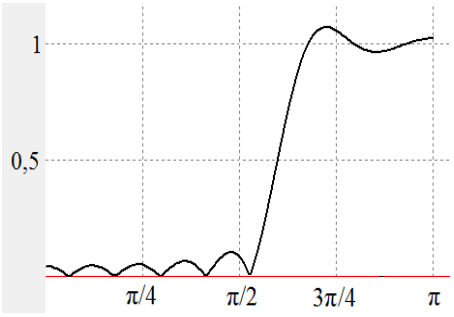


Figura 4.- Respuesta de un filtro pasa altas de orden 17
Fuente: (Clavijo Mendoza, 2011)

El cálculo de la función de transferencia h(n), para estos filtros, está definido por las siguientes ecuaciones:

$$h(n) = \begin{cases} \dfrac{-\sin(w_c n)}{\pi n}, n \neq 0 \\ 1 - \dfrac{w_c}{\pi}, n = 0 \end{cases}$$

1.6.1.6.- FILTRO PASA BANDA

Los filtros pasa banda permiten el paso de una porción de frecuencias entre Wc1 y Wc2, y el resto de frecuencias son eliminadas, (Clavijo Mendoza, 2011) el ejemplo es presentado en la **Figura 5.**

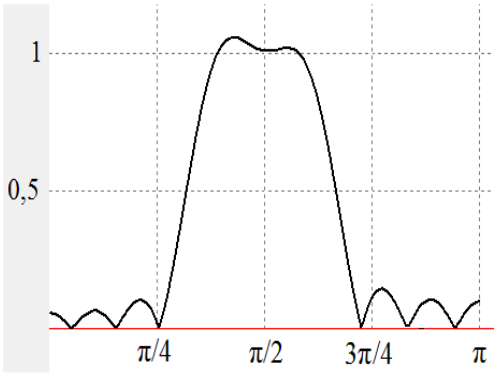


Figura 5.- Respuesta de un filtro pasa banda de orden 17
Fuente: (Clavijo Mendoza, 2011)

La función de transferencia para este tipo de filtros es calculada por medio de la siguiente ecuación:

$$h(n) = \begin{cases} \dfrac{\sin(w_{c2}n) - \sin(w_{c1}n)}{\pi n}, n \neq 0 \\ \dfrac{w_{c2} - w_{c1}}{\pi}, n = 0 \end{cases}$$

1.7.- PROCESAMIENTO DIGITAL DE SEÑALES

"Una señal está definida, como aquella cantidad física que varía con el tiempo, espacio o cualquier otra variable o variables independientes" (E. Soria Olivas, 2003).

El procesamiento digital, requiere transformar las señales de entrada a un formato digital; esto sucede en una etapa llamada conversión analógica-digital (A/D). (ALVARADO MOYA, 2006)

La señal digital después es tratada en el procesador digital de señales, que puede ser un microcontrolador, o inclusive circuitos digitales específicamente diseñados para realizar las tareas de procesamiento deseadas. El último paso del procesamiento digital de señales consiste en convertir la salida del bloque procesado a una señal analógica, lo que ocurre en el llamado conversor digital-analógico (D/A). En muchos casos de módulos de análisis de señal, la última etapa no es necesaria, pues la información a extraer puede obtenerse fácilmente de las representaciones digitales. (ALVARADO MOYA, 2006)

La **Figura 6** muestra el diagrama del procesamiento.

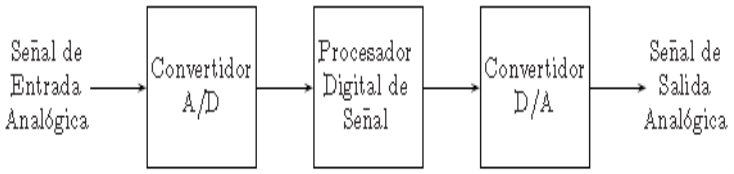


Figura 6.- Procesamiento Digital de una Señal Analógica
FUENTE.- (ALVARADO MOYA, 2006)

1.7.1.- DISEÑO DE FILTROS FIR POR EL MÉTODO DE VENTANAS

En el dominio de frecuencia $H(w)=H_d(w)*W(w)$. Puesto que W (w) tiene generalmente un lóbulo central y lóbulos laterales, el efecto en la convolución es suavizar la respuesta H_d (w). (ALVARADO MOYA, 2006)
Para reducir el efecto de estos lóbulos laterales son utilizadas ventanas, las cuales mejoran la señal de salida.

Algunas ventanas típicas y sus características están representadas en la **Tabla 1.**

Nombre función ventana	**Ancho de Transición (Hz) (normalizado)**	**Rizo (dB) Pasabanda**	**Relación (dB) lóbulo principal lóbulos laterales**	**Atenuación (dB) Máxima Parabanda**	**función ventana** $w(n),\ \|n\|\leq(N-1)/2$
Rectangular	0,9 / N	0.7416	13	21	1
Hanning	3,1 / N	0.0546	31	44	$0.5+0.5COS\left(\frac{2\pi n}{N}\right)$
Hamming	3,3 / N	0.0194	41	53	$0.54-0.46COS\left(\frac{2\pi n}{N}\right)$
Blackman	5,5 / N	0.0017	57	75	$0.42-0.50COS\left(\frac{2\pi n}{N-1}\right)+0.08COS\left(\frac{4\pi n}{N-1}\right)$
Káiser	5,71 / N	0.000275		90	$\frac{I_o\left[\beta\sqrt{\left(1-\left(\frac{2n}{(N-1)}\right)^2\right)}\right]}{I_o(\beta)}$

Tabla 1.- Funciones Utilizadas como ventanas.
Fuente.- (ALVARADO MOYA, 2006)

La variable h(n) es la respuesta al impulso del filtro diseñado, ésta es la resultante de multiplicar la respuesta al impulso deseada h_d (n) con la función de ventana W(n).

$$h(n) = h_d(n)\, w(n)$$

1.7.2.- ESPECIFICACIONES DE LOS FILTROS FIR

En el diseño de los filtros FIR, están presentes algunos elementos normalizados a tomar en cuenta.

La **Figura 7** muestra la gráfica del filtro pasa bajos con sus elementos normalizados en función de la frecuencia.

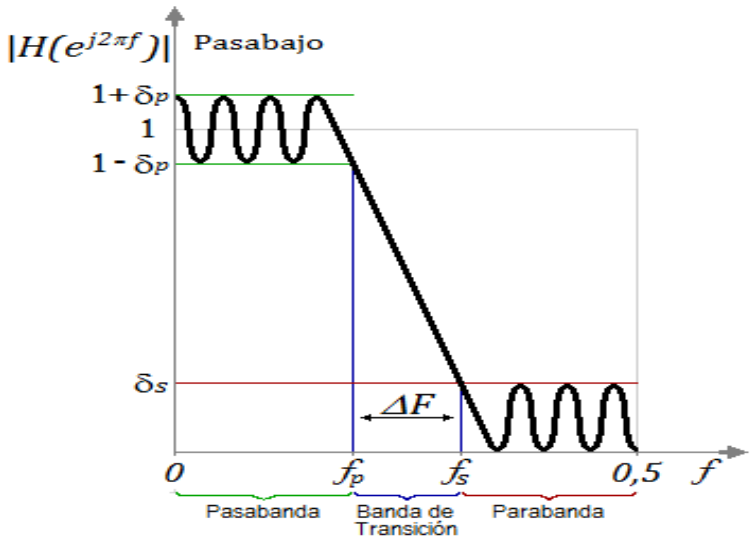


Figura 7.- Elementos de un filtro FIR pasa bajos.
Fuente.- https://es.filterdesign.org/filterdesign/Dise%C3%B1o_de_Filtros_de_Respuesta_Finita_al_Impulso#/media/File:FiltroFIRPasaBajo.png

La frecuencia de corte normalizada es hallada mediante la siguiente ecuación:

$$fc = fp + \frac{\Delta F}{2} = fp + \frac{fs - fp}{2} = \frac{fs + fp}{2}$$

La frecuencia de borde de pasa banda (fp) puede ser encontrada con la siguiente fórmula:

$$fp = \frac{fcorte}{fmuestreo}$$

Y la banda de transición en hercios es encontrada mediante la relación:

$$banda\ de\ transición = \Delta F * Fmuestreo$$

La **Figura 8** muestra los elementos normalizados del filtro FIR pasa banda.

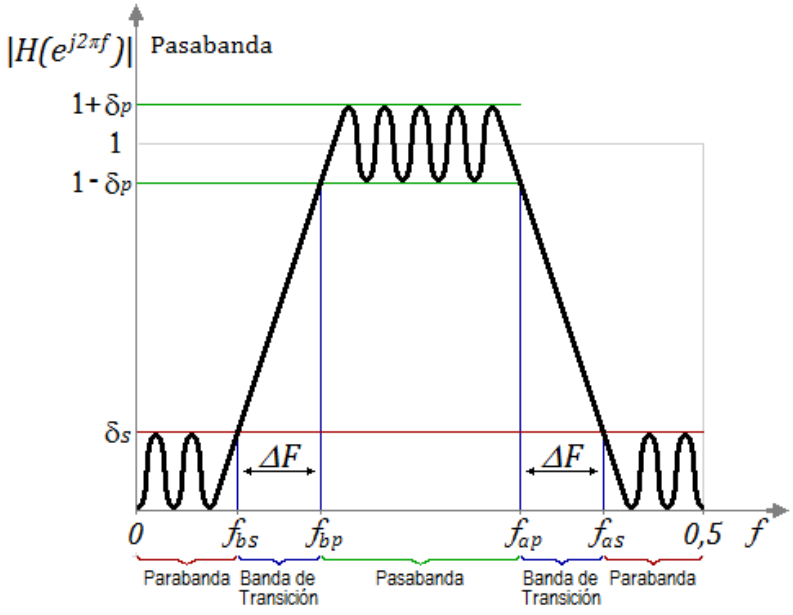


Figura 8.- Elementos de un filtro FIR pasa banda.
Fuente.- https://es.filterdesign.org/filterdesign/Dise%C3%B1o_de_Filtros_de_Respuesta_Finita_al_Impulso#/media/File:FiltroFIRPasaBanda.png

Las frecuencias de corte normalizadas son resueltas mediante las siguientes ecuaciones:

$$fb = fbp - \frac{\Delta F}{2} = fbp - \frac{fbp - fbs}{2} = \frac{fbp + fbs}{2}$$

$$fa = fap + \frac{\Delta F}{2} = fap + \frac{fas - fap}{2} = \frac{fap + fas}{2}$$

La **Figura 9** muestra los elementos normalizados del filtro FIR pasa altos.

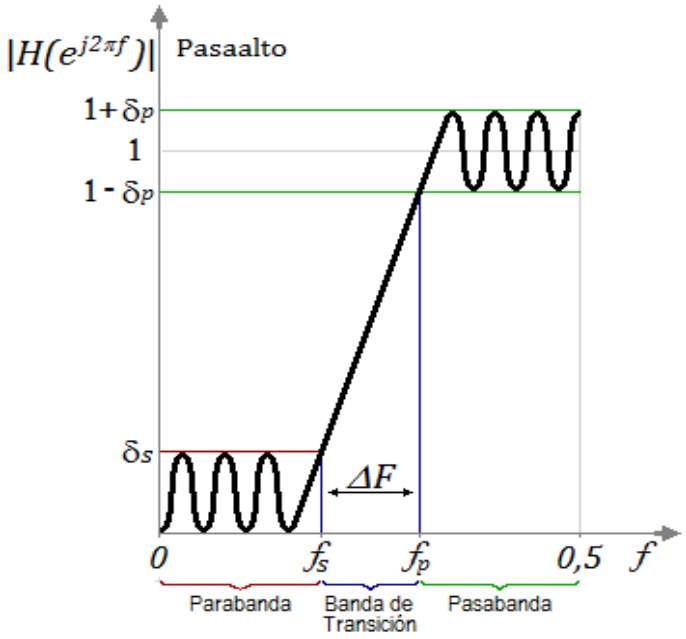


Figura 9.- Elementos de un filtro FIR pasa altos.
Fuente.- https://es.filterdesign.org/filterdesign/Dise%C3%B1o_de_Filtros_de_Respuesta_Finita_al_Impulso#/media/File:FiltroFIRPasaAltos.png

La frecuencia de corte normalizada es resuelta mediante la siguiente ecuación:

$$fc = fp - \frac{\Delta F2}{2} = fp - \frac{fp - fs}{2} = \frac{fp + fs}{2}$$

Las frecuencias de pasa banda, y la banda de transición; son resueltas por la misma ecuación, indistintamente del tipo de filtro.

1.7.3.- CONVERSIÓN DIGITAL - ANALÓGICA CON PWM

La conversión digital-analógica con PWM, consiste en tomar la señal modulada por ancho de pulso, y realizar la demodulación por medio de un filtro pasa bajas. Este filtro, debe tener como frecuencia de corte un valor mucho menor a la frecuencia de muestreo en una proporción cercana a 10 veces, dado que su objetivo es eliminar la frecuencia de muestreo. (Clavijo Mendoza, 2011)

El cálculo de la frecuencia de corte del filtro es obtenido por la siguiente fórmula:

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

1.8.-SISTEMA OPERATIVO ANDROID

Android es un sistema operativo móvil basado en Linux, que junto con aplicaciones middleware[5] está enfocado para ser utilizado en dispositivos móviles con pantalla táctil. Cuenta con una plataforma de descarga de aplicaciones y juegos llamada Google Play, la cual tiene más de 100 mil opciones para descargar, la mayoría gratuitas. (https://www.android.com/play/)

[5] Software que permite intercambio de información

1.8.1.- ANDROID STUDIO

Android Studio, es el IDE[6] oficial para el desarrollo de aplicaciones Android fundamentado en Intellij IDEA, otro entorno de desarrollo para la plataforma Java. (Developers, 2014)

1.8.2.- LENGUAJE DE PROGRAMACIÓN JAVA

Java es una plataforma informática que está orientada a los objetos, su principal meta es que los desarrolladores de aplicaciones creen sus compilaciones y puedan ejecutarlas en infinidad de dispositivos. La escritura de las aplicaciones y applets[7] de Java, requiere de herramientas de desarrollo como son: JDK[8], que incluye Java Runtime Environment, un compilador, las API[9] y JRE[10] permiten applets escritos en el lenguaje de programación de Java para ejecutarlos en varios exploradores. (Java, 2014)

1.8.3.- PAQUETE ANDROID DEVELOPER TOOLS

El ADT (Android Developer Tools) incluye los componentes esenciales de Android SDK[11] y una versión del IDE de Android Studio, este permite compilar las aplicaciones desarrolladas, en un emulador como interfaz de un dispositivo Android, con las características requeridas para la visualización de la aplicación diseñada. (Developers, 2014)

---

[6] Entorno de Desarrollo Integrado
[7] Elemento para desarrollar una página web como imagen o fragmento de texto
[8] Java Development Kit
[9] Interfaz de Programación de Aplicaciones
[10] Java Runtime Environment
[11] Kit de Desarrollo de Software

1.8.4.- KIT DE DESARROLLO DE SOFTWARE (DSK)

Los DSK son herramientas de desarrollo de software, éstos proporcionan las librerías API que le permite al programador: crear, compilar y corregir las aplicaciones de Android. (Developers, 2014)

1.9. - MÓDULO WI-FLY RN-XV

El módulo RN-XV es un módulo inalámbrico que utiliza el estándar 802.11 g/b[12], es una solución especialmente diseñada para comunicaciones a distancias cortas como son las redes domiciliarias, está en el protocolo TCP/IP[13]. (Networks, 2011). La **Figura 10** presenta el módulo mencionado.

Figura 10. - Módulo Wi-fly RN-XV
Fuente. - Datasheet Módulo Wi-fly RN-XV

Las características principales de este módulo son:

- Potencia de transmisión de 0 dBm[14] a 12 dBm.
- Consumo de energía ultra bajo.
- Interfaces de hardware TTL[15] UART[16]

[12] Indica las normas de funcionamiento en una red de área local inalámbrica
[13] Protocolo de Control de Transmisión / Protocolo de Internet
[14] Unidad de medida de potencia: Decibelio-milivatio
[15] Lógica Transistor a Transistor

- Tasa de datos de hasta 464 Kbps[17] en UART.
- 3 entradas de sensor analógicas.
- 8 pines digitales de entrada/salida para propósitos generales. (Networks, 2011)

El **Anexo 5** contiene la hoja de datos completa de este módulo Wireless.

1.10.- COMUNICACIÓN LOCAL INALÁMBRICA

La tecnología inalámbrica, usa ondas de radiofrecuencia de potencia baja, y una banda específica de uso libre o privado para transmitir información entre dispositivos. Las redes inalámbricas están clasificadas de diversas maneras, al tomarse en cuenta la distancia y el alcance para el desarrollo del control local, hay que tomar en cuenta la red WLAN[18] o estándar IEEE[19] 802.11, llamada comúnmente como wi-fi, ésta es una de las tecnologías de comunicación inalámbrica que utiliza ondas. (Mishra, 2004)

1.10.1.- RED AD-HOC

Una red AD-HOC es un tipo particular de red, sin control central y sin conexión con el "mundo externo"; permiten la comunicación entre dispositivos (punto a punto) sin utilizar un punto de acceso, utiliza las señales de radio frecuencia para lograr intercambiar

[16] Universal Synchonous Asynchronous Receiver Transmitter (Transmisor-Receptor Asíncrono Universal)
[17] Kilobits por segundo
[18] Wireless LAN (Red de Área Local Inalámbrica)
[19] Instituto de Ingenieros Eléctricos y Electrónicos

paquetes de datos dentro de un rango de transmisión. (Mishra, 2004)

1.10.2.- SOCKET

La combinación del número de puerto de la capa de transporte y de la dirección IP[20] de la capa de red asignada al host[21] identifica de manera exclusiva a un proceso en particular que es ejecutado en un dispositivo host específico; este procedimiento, está denominado como socket. (Academy, Aspectos básicos de Networking, 2010)
Por ejemplo, si un usuario quiere realizar una solicitud de página web HTTP[22] que es enviada mediante un servidor web (puerto 80) y que es ejecutada en un host con una dirección IP 192.168.1.2 será destinada al socket 192.168.1.2:80 .(Academy, Aspectos básicos de Networking, 2010)

1.11.- ROUTER

Un router es un dispositivo de telecomunicaciones que permite la conexión entre una red y otra, en otras palabras es el responsable de entregar paquetes a través de diferentes redes. (Academy, Conceptos y protocolos de enrutamiento, 2014)

1.11.1.- DHCP[23]

Es un protocolo de red que permite a los clientes de una red obtener sus parámetros de

[20] Internet Protocol (Protocolo de Internet)
[21] Elemento final de una red de comunicación.
[22] Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto)
[23] Dynamic Host Configuration Protocol (Protocolo de Configuración Dinámica de Host)

configuración automáticamente, es un protocolo de tipo cliente/servidor en el que un servidor posee una lista de direcciones IP dinámicas y las va asignando a los clientes conforme éstas van quedando libres. (Academy, Conceptos y protocolos de enrutamiento, 2014)

1.11.2.- ROUTER INALAMBRICO TP – LINK

El router inalámbrico de la serie TL-WR740N es un dispositivo de conexión de red de cable o inalámbrico que incluye un router para compartir internet y un Switch de 4 puertos. El router inalámbrico es compatible con 802.11 b y g. (TP-LINK, 2014).

Las Especificaciones del router utilizado, están en el **Anexo 9**, la **Figura 11** muestra el router mencionado.

Figura 11.- Vista frontal y posterior del Router TL-WR740N

Fuente.-

http://www.tplink.com/ar/products/details/?categoryid=241&model=TL-WR740N#down

1.12.- MICROCONTROLADOR

Un microcontrolador, es un circuito integrado de alta escala de integración, que posee la arquitectura de

un computador, es decir, contiene CPU[24], memorias RAM[25], EEPROM[26], y circuitos de entrada/salida. (Reyes, 2006)

1.12.1.- FORMATO ARITMÉTICO

Una de las características más importantes en el procesamiento digital es el tipo de aritmética utilizada por el procesador. La mayor parte de los microcontroladores utilizan aritmética de punto fijo, donde los números son representados como enteros o como fracciones entre los valores de -1.0 y +1.0. Otros procesadores en cambio, usan aritmética de punto flotante, donde los valores son representados por una mantisa y un exponente. La mantisa generalmente es una fracción con rango entre -1.0 y +1.0, mientras que el exponente es un numero entero que representa en binario el número de lugares a partir del punto que es desplazado a la izquierda o derecha para obtener el valor representado. (CONSTANTE CASTRO, 2003)

La aritmética en punto flotante, es mucho más flexible que la de punto fijo. En punto flotante, los diseñadores de sistemas tienen acceso a un rango dinámico más amplio. Como resultado, la aritmética de punto flotante, es más fácil de programar que sus correspondientes de punto fijo, pero generalmente son más costosos.

Los procesadores de punto fijo, son utilizados en muchas aplicaciones debido a su bajo costo, en estas aplicaciones son necesarios programas y algoritmos diseñados para determinar el rango dinámico y la

[24] Unidad de Procesamiento Central
[25] Memoria de Acceso Aleatorio
[26] ROM Programable Borrable

precisión. En las aplicaciones en las que el costo es poco importante o bien es necesario un amplio rango dinámico o gran precisión, los procesadores de punto flotante, son los adecuados.

1.12.2.- MICROCONTROLADOR AT32UC3L

El microcontrolador AVR AT32UC3L de Atmel, que muestra la **Figura 12**, es un dispositivo electrónico que funciona a frecuencias de hasta 50 MHz. Posee un núcleo de 32 bits, diseñado para aplicaciones embebidas, con especial énfasis en el bajo consumo de energía, alta densidad de código y alto rendimiento. (ATMEL, www.atmel.com)

Figura 12.- Microcontrolador AT32UC3L
Fuente.- Datasheet AT32UC3L

El procesador cuenta con una unidad de memoria de protección (MPU); para conseguir una capacidad de cálculo superior, pueden ser seleccionadas muchas instrucciones DSP con las que cuenta el microcontrolador. (ATMEL, Atmel) Este y otros microcontroladores, cuentan con las siguientes características:

- Permiten realizar la operación de multiplicación-acumulación dentro de un solo ciclo de trabajo.
- Permiten realizar varios accesos a memoria en un solo ciclo de instrucción, de esta manera el procesador puede buscar una instrucción mientras a la vez, realiza la búsqueda de operando y/o almacena el resultado de una instrucción anterior.
- Para permitir múltiples accesos a memorias, están incluidas memorias multipuerto e incluso bancos de memorias independientes.
- Poseen una o más unidades generadoras de direcciones independientes. Dichas unidades, operan en paralelo con la ejecución de instrucciones aritméticas.
- Disponen de un set de instrucciones que soportan la ejecución de bucles, debido a que los algoritmos DSP implican cálculos repetitivos. (CONSTANTE CASTRO, 2003)

El **Anexo 8** contiene la hoja de datos del microcontrolador AT32UC3L.

1.12.3.- ATMEL STUDIO

Atmel Studio es la plataforma de desarrollo integrada para el desarrollo y la compilación de aplicación para microcontroladores AVR de Atmel; Soporta todos los microcontroladores AVR y SMART; Este entorno entrega un fácil medio para escribir, diseñar y cargar aplicaciones hechas en los lenguajes C/C++ o en código ensamblador. De igual manera este programa permite conectarse a kits de desarrollo Atmel.

Adicionalmente, Atmel Studio cuenta con galerías Atmel, y en línea existen tiendas de aplicaciones que permiten extender aún más la capacidad de desarrollo de los dispositivos. (ATMEL, Atmel)

1.12.4.- AVR DRAGON

El AVR Dragón es un grabador de programas de Atmel, establece un nuevo estándar para las herramientas de desarrollo de bajo costo para dispositivos AVR de 8 bits y 32 bits con capacidad de grabación en chip (OCD[27]). Puede realizarse una grabación en todos los dispositivos con OCD con SPI[28], JTAG.

Un área de desarrollo permite a los diseñadores crear sus propios circuitos. El grabador también soporta NanoTrace, dependiendo del módulo de TOC en el dispositivo AVR. (ATMEL, www.atmel.com)

La **Figura 13** muestra el compilador mencionado.

Figura 13.- Grabador AVR Dragon
Fuente.- Datasheet – grabador AVR Dragon

1.13.- AMPLIFICADOR DE AUDIO

Un amplificador de Audio es un circuito encargado de amplificar la señal entrante para que sea audible en

[27] On Chip Debug
[28] Serial Peripheral Interface

su punto de salida que son los altavoces. (Gallager, 2006) La **Figura 14,** muestra el circuito de un amplificador de 30 watts de potencia por canal.

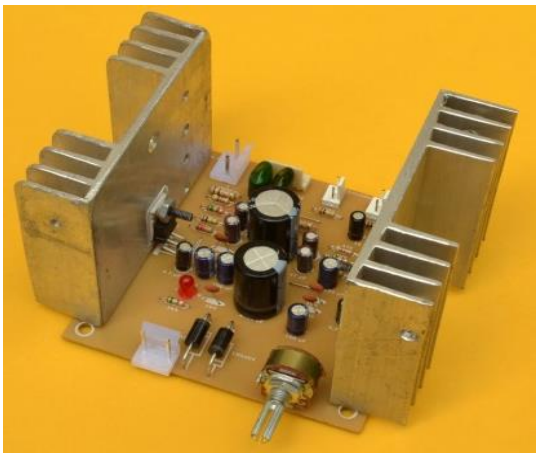


Figura 14.- Amplificador de 30 Watts
Fuente.-
http://construyasuvideorockola.com/imagenes/proyectos/barato/amp_30w_00.jpg

1.13.1.- AMPLIFICADOR TDA2030

El TDA2030 es un circuito integrado monolítico de clase AB. Puede proporcionar 15 w de potencia de salida con una carga de 8 Ohmios. El TDA 2030 ofrece una alta corriente de salida y muy baja distorsión armónica. (ST); En la **Figura 15** está presente la configuración de los pines de este dispositivo.

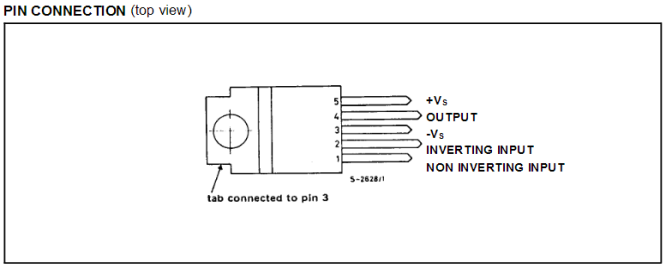


Figura 15.- Configuración de pines del TDA2030
Fuente.- Datasheet TDA2030

El **Anexo 7** contiene la hoja de datos y especificaciones del amplificador TDA2030.

1.14.- ALTAVOZ

Un altavoz, también conocido como parlante, es un transductor electro acústico utilizado en la reproducción de sonido; el proceso de transducción empieza al convertir las ondas eléctricas en energía mecánica y a su vez éstas son transformadas en ondas de frecuencia acústica; según el rango de frecuencias que puede reproducir cada altavoz, pueden ser divididos en woofer, squawker y tweeter. (Gallager, 2006)

Los altavoces son los encargados de producir una señal audible, para lo cual transforma la energía eléctrica recibida en energía acústica, como paso intermedio está la transformación a energía mecánica. (CONSTANTE CASTRO, 2003)

De acuerdo a estas propiedades un altavoz es dividido en las siguientes partes constituyentes:

- Sección electromagnética: Está formada por el imán y la bobina móvil. Aquí llega la señal eléctrica a la bobina móvil que está dentro del campo magnético del imán produciéndose el movimiento de la bobina.
- Sección mecánica: Está constituida por el cono y su suspensión. Sobre el cono está montada la bonina móvil, la cual al desplazarse hace que el cono vibre.
- Sección acústica: es la encargada de transmitir la energía sonora producida por el cono.

En la **Figura 16,** están presentes los elementos mencionados.

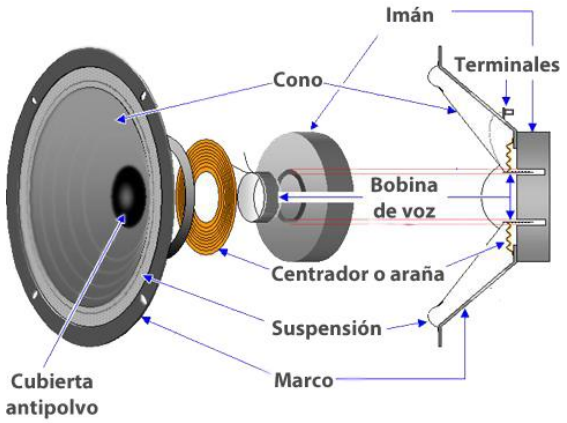


Figura 16.- Elementos de un altavoz
Fuente.- http:www.sound-pixel.com/altavoz.jpg

Para una máxima calidad de reproducción sonora, hay que recurrir al empleo de varios y diversos altavoces, ya que las características constructivas de los mismos los hacen adecuados para una gama de frecuencias e inadecuados para otras. En base a estas características están definidos los altavoces grandes para frecuencias bajas o graves (woofer), altavoces más pequeños para frecuencias medias (squawker) y otros aún más pequeños para las frecuencias más altas o agudas (tweeter). (CONSTANTE CASTRO, 2003)

1.14.1.- WOOFER

Un woofer es un altavoz diseñado para reproducir sonidos de baja frecuencia, generalmente frecuencias entre los 100 y 800 Hz; su nombre es designado así por el nombre ingles del ladrido de perro: woof, en contraste con el nombre usado para los altavoces de altas frecuencias: tweeter. (Gallager, 2006)

Una de las características especiales que tienen los woofer es que su cono no es estático, por eso también son considerados como altavoces de cono flotante, esto ocasiona que el diafragma proporcione un rendimiento excelente para los tonos graves.

1.14.2.- SQUAWKER

Un squawker literalmente significa graznador; es el transductor de un altavoz, encargado de reproducir las frecuencias medias, generalmente entre 300 Hz y 5000 Hz; este tipo de altavoces son usados en sistemas de tres o más vías. (DUIOPS, 2007)

Una de las características de este altavoz es que su cono es estático, por lo cual también son conocidos como altavoces de cono fijo.

1.14.3.- TWEETER

Un tweeter es un altavoz especializado en altas frecuencias, entre 3 KHz a 20 KHz; es decir el rango de sonidos agudos; dado que el oído humano pierde con la edad la capacidad para percibir las frecuencias más altas, estas frecuencias pueden ser apreciadas solamente por personas jóvenes con oídos sanos. (Gallager, 2006)

Los altavoces para agudos, necesitan estar provistos de trompetas, ya que esta sirve para adaptar su impedancia acústica con la del aire; tienen el menor diámetro exterior de los altavoces. (CONSTANTE CASTRO, 2003)

CAPÍTULO II

2.- METODOLOGÍA

En este capítulo está detallada la metodología utilizada para el desarrollo de este trabajo de investigación.

2.1.- TIPO DE ESTUDIO

El tipo de investigación desarrollada es:

2.1.1.- DESCRIPTIVO

La investigación utilizada es de tipo descriptiva, ya que observa y describe el comportamiento de un crossover de tres vías utilizando DSP con parámetros controlados por un dispositivo Android, mediante el desarrollo de ecuaciones matemáticas para generar filtros digitales. Para obtener una respuesta adecuada, es necesario encontrar un equilibrio entre un buen orden de filtro y una buena velocidad.

2.2.- MÉTODO, TÉCNICAS E INSTRUMENTOS

2.2.1.- MÉTODO

El método utilizado en este proyecto, es analítico-deductivo, debido al análisis particular que es realizado en la señal de audio de entrada del crossover y su respuesta a la salida; en la forma de interactuar entre el microcontrolador y la información enviada vía

wi-fi, todo esto previo a que este sistema desempeñe su función eficientemente.

2.2.2.- TÉCNICAS

La técnica utilizada en este procedimiento es la Observación, está caracterizada para recolectar la información de las diferentes situaciones existentes en el crossover de tres vías.

2.2.3.- INSTRUMENTOS

Los instrumentos requeridos en esta investigación son: libros, archivos, páginas web, hojas de datos de dispositivos electrónicos, osciloscopio y el oído humano.

2.3.- POBLACIÓN Y MUESTRA

2.3.1.- POBLACIÓN

La población, es determinada por los datos obtenidos en las pruebas de acuerdo a las distintas frecuencias que son utilizadas en los filtros. Como el rango audible del oído humano está comprendido entre los 100 Hz y los 20 KHz, la población considerada en este proyecto es de 19900 Hz.

2.3.2.- MUESTRA

En este caso, la población es identificable, por lo tanto el número de pruebas es finito, por lo cual la muestra es establecida de acuerdo a la siguiente fórmula:

$$n = \frac{N * Z\infty^2 * p * q}{i^2(N-1) + Z\infty^2 * p * q}$$

Donde:

- n = muestra
- N = Tamaño de la población
- $Z\infty$ = Valor correspondiente a la distribución de Gauss (1,96)
- p = Prevalencia esperada del parámetro a evaluar (0,9 = 90%)
- q = 1 – p (0,1)
- i = error considerado que está previsto cometerse (0,1)

$$n = \frac{19900 * 1{,}96^2 * 0{,}9 * 0{,}1}{0{,}1^2(19900 - 1) + 1{,}96^2 * 0{,}9 * 0{,}1}$$

$$n = \frac{19900 * 3{,}8416 * 0{,}9 * 0{,}1}{0{,}01\ (19899) + 3{,}8416 * 0{,}9 * 0{,}1}$$

$$n = \frac{6880{,}3056}{198{,}99 + 0{,}345744}$$

$$\mathrm{n} = \frac{6880{,}3056}{199{,}335744}$$

$$\mathrm{n} = 34{,}52$$

El número de muestras a tomarse es de 35.

2.3.3.- HIPÓTESIS

"El diseño e implementación de un crossover de tres vías utilizando DSP con parámetros controlados por un dispositivo Android permitirá distribuir eficazmente las componentes de una señal de audio debido a las frecuencias variables"

La hipótesis planteada es de tipo correlacional, ya que involucra a dos tipos de variables (independiente y dependiente) y explica que el crossover implementado no solamente filtrará una señal de audio sino que el usuario puede manipular sus frecuencias de corte de manera inalámbrica.

2.4.- OPERACIONALIZACIÓN DE LAS VARIABLES

Las variables dependientes e independientes están detalladas en las siguientes tablas.

La **Tabla 2** indica la operacionalización de las variables independientes.

VARIABLES	DIMENSIONES	INDICADORES	INSTRUMENTOS
El diseño e implementación de un crossover de tres vías utilizando DSP con parámetros controlados por un dispositivo Android	• diseño de filtros FIR utilizando la convolución y el ventaneo	• Aplicación • Placa con filtros FIR y antena para comunicación inalámbrica • Arreglo de altavoces	• Tablet. • Módulo wi-fly • Android studio • Componentes electrónicos • AWR studio

Tabla 2.- Operacionalización de la variable independiente
Fuente.- El autor

Por su parte, La **Tabla 3** muestra la operacionalización de la variable dependiente.

VARIABLES	DIMENSIONES	INDICADORES	INSTRUMENTOS
Permitirá distribuir eficazmente las componentes de una señal de audio debido a las frecuencias variables.	• Verificación de la banda de paso y frecuencias de corte	• Niveles de voltaje • Frecuencias de corte	• Osciloscopio • Oído humano

Tabla 3.- Operacionalización de la variable dependiente.
Fuente.- El autor

2.5.- PROCEDIMIENTOS

La **Figura 17** muestra el diagrama de bloques del sistema.

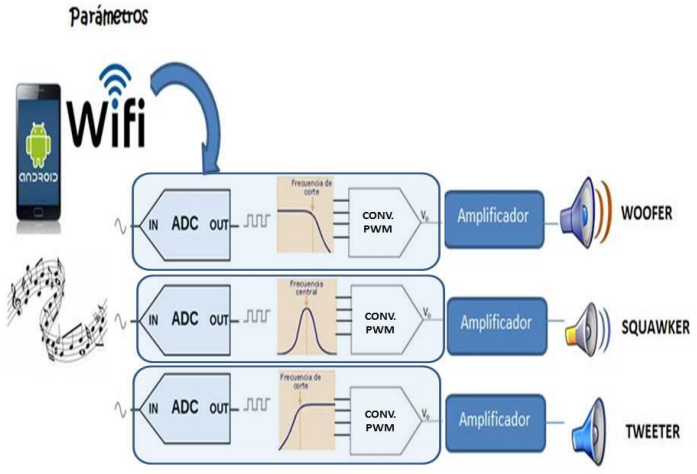


Figura 17.- Diagrama de bloques del sistema
Fuente.- El autor

El procedimiento de este trabajo está dividido las siguientes etapas:

Comunicación

Procesamiento

Amplificación

2.5.1.- COMUNICACIÓN

En esta etapa, está descrita la programación realizada para comunicar al dispositivo Android con el módulo wi-fly.

2.5.1.1.- CONFIGURACIÓN DE LA RED

La comunicación entre estos dispositivos es ejecutada en una red existente debido a que el módulo wi-fly (según criterios de proyectos realizados) es mejor trabajando de esta manera que trabajando en modo ad-hoc, para esto es utilizado el router descrito en la fundamentación teórica (TL-WR740N); donde es realizada la configuración DHCP para este proyecto.

La red a utilizarse tiene como nombre: PRAY, es una red privada clase C, cuya dirección de red es 192.168.0.0 con máscara completa: 255.255.255.0; esta información y su contraseña son configuradas en la tarjeta wi-fly, la dirección física (MAC[29]) de la tarjeta wi-fly es la siguiente: 00:1E:C0:21:26:65 ; en el router se añade esta MAC a su tabla de enrutamiento para que la dirección IP asignada a la tarjeta wi-fly sea siempre la misma, ya que la interfaz Android requiere una dirección IP estática como destinatario para enviar la información.

[29] Media Access Control (Control de Acceso al Medio)

La dirección IP utilizada es 192.168.0.110 para la tarjeta wi-fly, la **Figura 18** muestra la dirección IP reservada para la tarjeta wi-fly en el router.

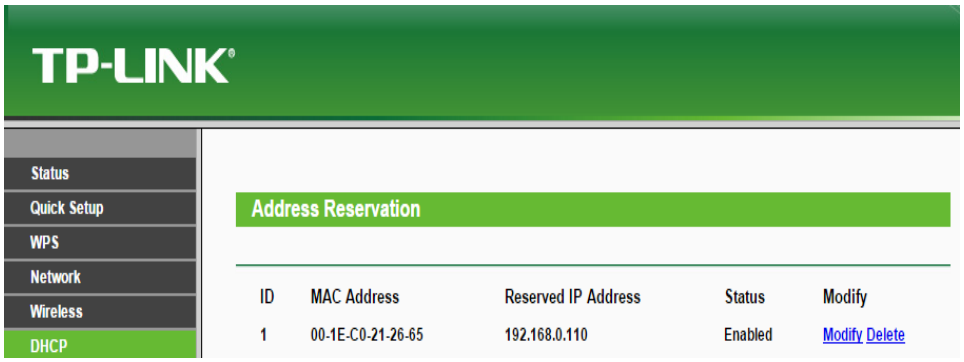


Figura 18.- Direcciones reservadas en el router.
Fuente.- El autor

2.5.1.2.- CONFIGURACIÓN DEL MÓDULO INALÁMBRICO

El módulo inalámbrico Wi-fly XN-RV consta de 20 pines, su alimentación es de 3,3 voltios en el pin 1 y GND[30] está ubicada en el pin 10; sus conexiones UART, transmisor y receptor están presentes en los pines 2 y 3 respectivamente como lo muestra la **Figura 19.**

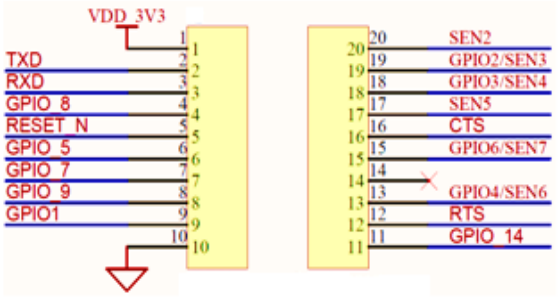


Figura 19.- Pines del módulo wi-fly RN-XV
Fuente.-Datasheet Wi-fly RN-XV

[30] Ground (tierra)

Para configurar el módulo inalámbrico hay que acceder al modo ad-hoc en donde la tarjeta debe tener un valor de 1 lógico en el puerto GPIO9 (pin 8), en la **Tabla 4** están presentes todos los pines del módulo.

Pad Number	Signal Name	Description	Direction
1	VDD_3V3	3.3V regulated power input to the module	POWER
2	UART_TX	UART TX, 8mA drive, 3.3V tolerant	OUT
3	UART_RX	UART RX, 3.3V tolerant	IN
4	GPIO 8	GPIO, 24mA drive, 3.3V tolerant	IN / OUT
5	RESET	**Optional** Module Reset Signal (active low), 100k Pull up, apply pulse of at least 160us, 3.3V Tolerant	INPUT
6	GPIO 5	GPIO, 24mA drive, 3.3V tolerant	OUT
7	GPIO 7	GPIO, 24mA drive, 3.3V tolerant	IN / OUT
8	GPIO 9	Enable Adhoc mode, Restore factory defaults, 8mA drive, 3.3V tolerant	IN / OUT
9	GPIO 1	GPIO, 8mA drive, 3.3V tolerant	IN / OUT
10	GND	Ground	GND
11	GPIO 14	GPIO, 8mA drive, 3.3V tolerant	IN / OUT
12	UART_RTS	UART RTS flow control, 8mA drive, 3.3V tolerant	OUT
13	GPIO 4/SEN 6	GPIO, 24mA drive, 3.3V tolerant/ADC input (**3.3V tolerant).** Defaults to GPIO 4	IN / OUT
14	Not Used		
15	GPIO 6/SEN 7	GPIO, 24mA drive, 3.3V tolerant/ADC input (**3.3V tolerant).** Defaults to GPIO 6	POWER
16	UART_CTS	UART CTS flow control, 3.3V tolerant	IN
17	SENSOR 5	Sensor Interface, Analog input to module, (**3.3V tolerant)**	INPUT
18	GPIO 3/SEN 4	GPIO, 8mA drive, 3.3V tolerant/ADC input (**3.3V tolerant**). Defaults to GPIO 3	IN / OUT
19	GPIO 2/SEN 3	GPIO, 8mA drive, 3.3V tolerant/ADC input (**3.3V tolerant**). Defaults to SEN 3	IN / OUT
20	SEN 2	Sensor Interface, Analog input to module, **3.3V tolerant**	INPUT

Tabla 4.- Distribución de pines del módulo Wi-fly RN-XV
Fuente.- Manual de Usuario del módulo Wi-fly RN-XV

Después de activar el modo ad-hoc en el módulo, el siguiente paso es revisar en el computador las redes disponibles, se muestra una red con el nombre de la tarjeta Wifly-EZX-65, los últimos dígitos son los últimos bytes de su dirección MAC, la **Figura 20** muestra la red a utilizarse.

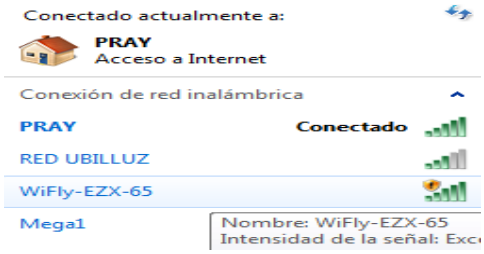


Figura 20.- Red que muestra el módulo en modo ad-hoc
Fuente.- El autor

La wi-fly tiene por defecto la dirección IP 192.168.1.1 y el número de puerto remoto 2000. Al conectarse a la red creada por el módulo, los datos del adaptador de LAN inalámbrica son verificados en el computador entrando al símbolo del sistema con el comando ipconfig.

El ingreso a la configuración del módulo es realizada con el programa PUTTY como cliente telnet, como lo indica la **Figura 21**; éste envía un mensaje con el texto "HELLO" cuando la conexión es exitosa; para ingresar a la configuración es necesario digitar el comando $$$ y después digitar Enter, y el módulo responde con "CDM" como lo indica la **Figura 22**, si esta respuesta es recibida, puede empezar a realizarse la configuración.

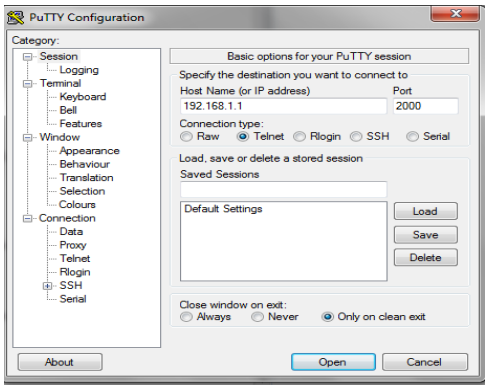


Figura 21.- Conexión por telnet mediante programa PUTTY
Fuente.- El autor

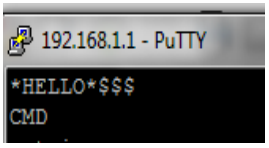


Figura 22.- Saludo y respuesta del módulo

Fuente.- El autor

Los comandos utilizados en la configuración del módulo son: get y set, el primer comando solicita información y el segundo configura el módulo.

2.5.1.2.1.- COMANDO SET IP DHCP <VALOR>

Este comando habilita o deshabilita el modo DHCP, en donde <VALOR> es un número decimal mostrado en la **Tabla 5**, es tomado el valor 1 porque la dirección IP la asigna el router.

MODE	PROTOCOL
0	Turns DHCP off. The module uses its stored static IP address.
1	Turns DHCP on. The module attempts to obtain an IP address and Gateway from the Access point.
2	Enables automatic IP, which is generally used with ad hoc networks.
3	Turns on DHCP cache mode. The module uses a previously set IP address if the lease is not expired.
4	Enables DHCP server in sot AP mode.

Tabla 5.- Valores configurables del comando SET IP DHCP

Fuente.- Manual de usuario Wi-fly RN-XV

2.5.1.2.2. - COMANDO SET WLAN SSID[31] <STRING>

Este comando configura el SSID de la red, en donde <string> es una palabra entre 1 y 32 caracteres; en

[31] Service Set IDentifier (Nombre de la red)

este caso la red en la que la conexión es realizada tiene como nombre "PRAY" por lo tanto el comando final es:

Set wlan ssid PRAY

2.5.1.2.3. – COMANDO SET WLAN PASS <STRING>

Este comando permite ingresar en el módulo wi-fly la contraseña de la red para que en cuanto el módulo haya sido conectado a la red, pueda ingresar a ella y el router le asigne la dirección IP establecida.

2.4.1.2.4. – COMANDO SET WLAN JOIN <VALUE>

Este comando es utilizado para la asociación con la red del router, en donde <value> es una de las opciones mostradas en la **Tabla 6.**

VALUE	POLICY
0	Manual. Do not try to associate with a network automatically.
1	Try to associate with the Access point that matches the stored SSID, passkey, and cannel. If the channel is set to 0, the module will scan for the access point.
2	Associate with ANY Access point that has the security matching the stored authentication mode.
4	Create an ad hoc network using stored SSID, IP address, and netmask. You must set the channel. Unless another ad hoc device can act as DHCP server, set DHCP to 0 (static IP) or use automatic IP.
7	Create a soft AP network using stored the SSID, IP address, netmask, cannel, etc. This model applies only to firmware versions supporting soft AP mode, not ad hoc mode.

Tabla 6.- Valores configurables del comando SET WLAN JOIN
Fuente.- Manual de usuario Wi-fly RN-XV

Con estos comandos es configurado el módulo wi-fly para que logre conectarse en la red.

2.5.1.3.- CONFIGURACIÓN DE LA APLICACIÓN ANDROID

Para esta configuración; es necesario contar con el programa Android Studio, que permite diseñar la aplicación para enviar la información mediante wi-fi. Android Studio cuenta con una interfaz gráfica la cual permite diseñar y ubicar las herramientas de mejor manera y además cuenta con el lenguaje de programación en donde los elementos son configurados.

En la aplicación son añadidos dos "Seekbar", son añadidos también textos estáticos y textos variables para mostrar los cambios producidos en los sliders. La **Figura 23** muestra la interfaz; esto puede ser revisado en Android Studio, accediendo al archivo activity_main.xml; éste es el archivo en donde es almacenada la aplicación de manera gráfica.

Figura 23.- Aplicación para regular las frecuencias.

Fuente.- El autor.

Cuando el proyecto es creado, son generados también los archivos con la extensión java donde se realiza la programación. Esto sucede en el archivo MainActivity.java.

Para enlazar los archivos .xml que son los encargados del diseño gráfico y los archivos .java que son los encargados de la programación, existe un comando llamado: **setcontentview;** el cual relaciona lo gráfico con la programación; la **Figura 24** muestra la programación para realizar este lazo.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // Enlazamos la clase Java con el XML de la vista
    setContentView(R.layout.activity_main);
    // Obtenemos los componentes gráficos para usarlos desde el código
    SeekBar sbPasaAltos = ((SeekBar) findViewById(R.id.sbPasaAltos));
    sbPasaAltos.setOnSeekBarChangeListener(this);

    SeekBar sbPasaBajos = ((SeekBar) findViewById(R.id.sbPasaBajos));
    sbPasaBajos.setOnSeekBarChangeListener(this);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

    setSupportActionBar(toolbar);

    cadenaFinal = new char[6]; // cadenaFinal = new char[7];
}
```

Figura 24.- Enlace entre archivos .xml y .java
Fuente.- El autor

Al final del bucle, es creado un buffer llamado **cadenaFinal** el cual contiene 6 bytes que son utilizados para el envío de los datos.

Por defecto los "Seekbar" en Android Studio empiezan en 0, pero en este caso es necesario que uno de ellos empiece en 100 (filtro pasa bajos) y el otro empiece en 2500 (pasa altos); para eso hay que modificar los valores iniciales accediendo al archivo MainActivity.java.

Los **Listener** son los lugares en donde se realiza la programación según la actividad de los elementos de la interfaz, por lo tanto, la configuración para mostrar los datos al cambiar los valores del slider, es realizada en el listener: **OnProgressChanged;** como muestra la **Figura 25.**

```
@Override
public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
    switch (seekBar.getId()) {
        case R.id.sbPasaAltos:
            ((TextView) findViewById(R.id.tvPasaAltosSeekBar)).setText("FILTRO PASA ALTOS: " + (3000 + progress) + " Hz");
            ((TextView) findViewById(R.id.tvPasaAltos)).setText((3000 + progress) + " Hz");
            ///((TextView) findViewById(R.id.ALT)).setText("A" + (30 + i) + " - Char: " + String.valueOf(Character.toChars(30 + i)))
            pasaAltos = 3000 + progress;
            break;

        case R.id.sbPasaBajos:
            ((TextView) findViewById(R.id.tvPasaBajosSeekBar)).setText("FILTRO PASA BAJOS: " + ((100 + progress) + "Hz"));
            Log.d("pogreso", "Progreso: " + progress);
            ((TextView) findViewById(R.id.tvPasaBajos)).setText((100 + progress) + "Hz");
            ///((TextView) findViewById(R.id.BAJ)).setText("B" + (10 + i) + " - Char: " + String.valueOf(Character.toChars(10 + i)))
            pasaBajos = 100 + progress;
            break;
    }
```

Figura 25.- Programación para visualizar los datos de los Seekbar
Fuente.- El autor

Dentro de ese bloque hay dos sentencias las cuales permiten realizar la acción según el slider que es utilizado, en el caso de los valores del filtro pasa altos; son añadidas 2500 unidades al valor del slider, el valor resultante es mostrado bajo la barra del filtro pasa altos, y además el mismo valor es mostrado en la barra del filtro pasa banda. Lo mismo sucede con el filtro pasa bajos al que son añadidas 100 unidades.

El valor es visualizado y a la vez almacenado para enviarlo a través de wi-fi; por lo tanto el valor es guardado en una variable llamada **pasaAltos** para el valor

del filtro pasa altos y en **pasaBajos** para el valor del filtro pasa bajos. También existe la opción de modificar manualmente las frecuencias de corte en un cuadro de texto; en donde además de ingresar manualmente, se muestra un mensaje cuando el número no está dentro del rango permitido.

La **Figura 26** muestra la respectiva programación.

```
@Override
public void onTextChanged(final CharSequence s, int start, int before, int count) {
    try {
        if (Integer.parseInt(s.toString()) > valorMaximoPasaAltos || Integer.parseInt(s.toString()) < valorMinimoPasaAltos) {
            etPasaAltos.setError("El rango debe estar entre " + valorMinimoPasaAltos + " y " + valorMaximoPasaAltos);
        } else {
            timer.cancel();
            timer = new Timer();
            timer.schedule(
                    () -> {
                        try {
                            sbPasaAltos.setProgress(Integer.parseInt(s.toString()) - valorMinimoPasaAltos);
                            onStopTrackingTouch(sbPasaAltos);
                            Log.d("Valor", "sbPasaAltos: " + sbPasaAltos.getProgress());
                            Log.d("Valor", "Valor Ingresado: " + Integer.parseInt(s.toString()));
                            Log.d("Valor", "Valor Minimo Pasa Altos: " + valorMinimoPasaAltos);
                        } catch (Exception ex) {
                            // ex.printStackTrace();
                        }
                    }, 500);
```

Figura 26.- Programación de los cuadros de Texto
Fuente.- El autor

El máximo dato a enviarse por los sliders es 20000 y éste es contenido en dos bytes (en el caso de enviarse esta frecuencia de corte para el filtro pasa altos); son requeridos 2 bytes para cada dato.

La secuencia de dos bytes en alto, no existe en el rango de los números enviados, así que es utilizada como indicador para que el receptor sepa que lo que viene después es la información de los filtros, quedando la trama como lo muestra la **Tabla 7:**

0	1	2	3	4	5
1	1	Byte más significativo	Byte menos significativo	Byte más significativo	Byte menos significativo
INDICADOR		FILTRO PASA BAJOS	FILTRO PASA BAJOS	FILTRO PASA ALTOS	FILTRO PASA ALTOS

Tabla 7.- Trama a enviarse por la interfaz.
Fuente.- El autor

Los datos no son enviados mientras los Seekbar están en progreso, sino cuando dejan de moverse, es por esto, que la generación de la trama es realizada en el listener **onStopTrackingTouch**; como lo muestra la **Figura 27.**

```
cadenaFinal[0] = (byte) 255;
cadenaFinal[1] = (byte) 255;
cadenaFinal[2] = (byte) (pasaBajos >> 8);
cadenaFinal[3] = (byte) (pasaBajos);
cadenaFinal[4] = (byte) (pasaAltos >> 8);
cadenaFinal[5] = (byte) (pasaAltos);
```

Figura 27.- Comandos utilizados en el listener onStopTrackingTouch
Fuente.- El autor

El envío de la trama mediante el dispositivo Android es realizado con la configuración de sockets; dentro de este programa es configurada la dirección IP del destinatario al igual que el puerto a utilizarse, la wi-fly siempre tendrá la dirección IP 192.168.0.110 y el puerto a utilizarse es el puerto 2000 como lo muestra la **Figura 28.**

```java
@Override
protected Boolean doInBackground(char[]... params) {
    Log.d("NetworkCommunication", "NetworkCommunication inBackground: " + params);
    try {
        // Ip, Puerto
        Socket client = new Socket("192.168.0.110", 2000);
        OutputStreamWriter printwriter = new OutputStreamWriter(client
                .getOutputStream(), "UTF-8");
        printwriter.write(params[0]);
        printwriter.flush();
        printwriter.close();
        client.close();
        return true;
    } catch (IOException e) {
        e.printStackTrace();
        return false;
    }
}
```

Figura 28.- Configuración de Socket y dirección IP del destino
Fuente.- El autor

Para finalizar la comunicación, es declarada una sentencia en una tarea asíncrona, esta tarea verifica si el dato llegó a su destino y envía un mensaje de error o de éxito según el caso. La **Figura 29** muestra la programación para dichas sentencias.

```java
@Override
public void taskCompletionResult(boolean result) {
    if (!result) {
        Snackbar snackbar = Snackbar
                .make(findViewById(R.id.clPrincipal), "¡¡Error!! Revise la conexión", Snackbar.LENGTH_LONG);
        snackbar.show();
    } else {
        Snackbar snackbar = Snackbar
                .make(findViewById(R.id.clPrincipal), "Enviado Exitosamente", Snackbar.LENGTH_LONG);
        snackbar.show();
    }
```

Figura 29.- Programación del mensaje de Error / Éxito
Fuente.- El autor

La **Figura 30** muestra el diagrama de flujo del funcionamiento de la aplicación:

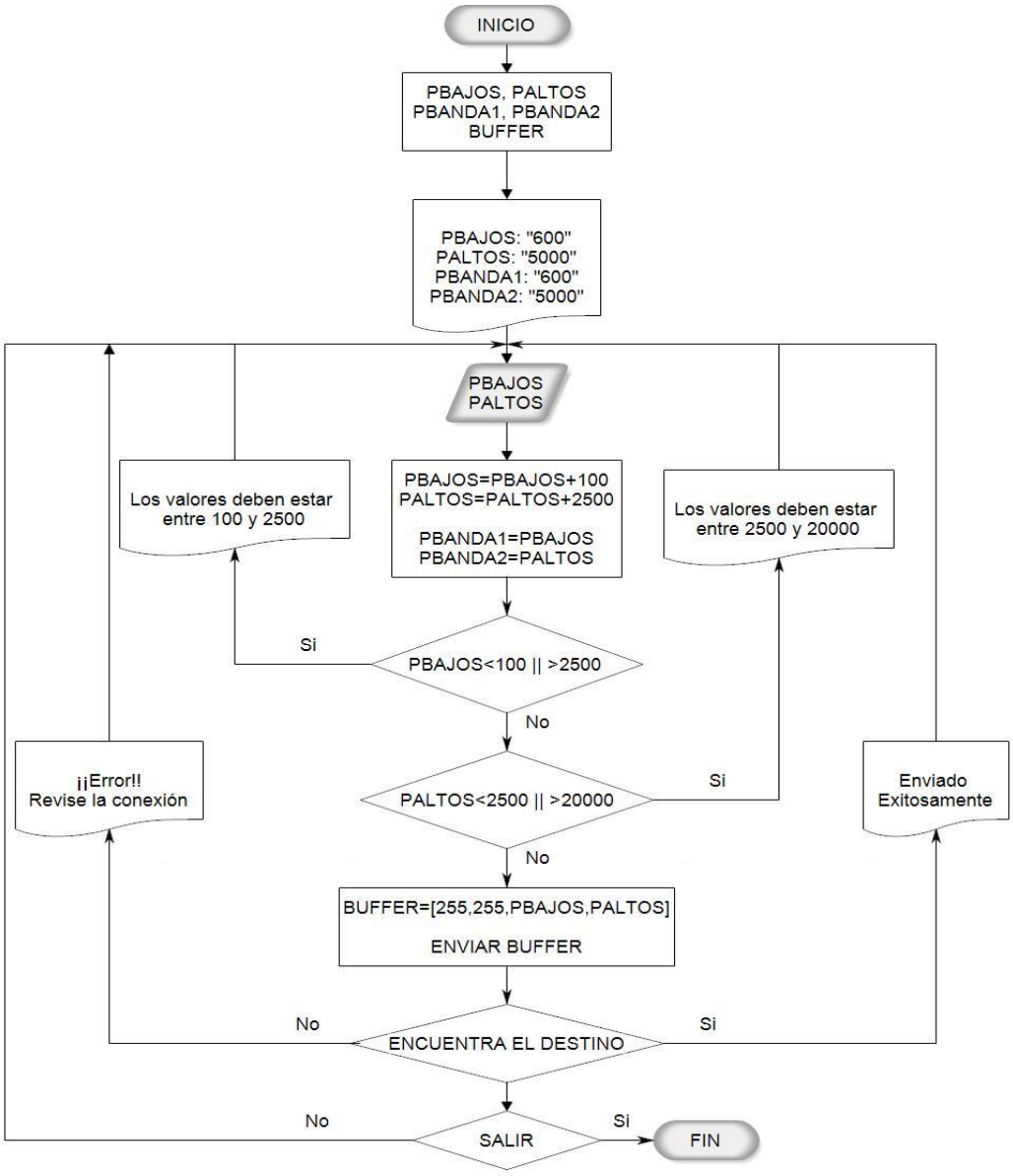


Figura 30.- Diagrama de flujo de la Aplicación Android.
Fuente.- El autor

El **Anexo 4** contiene toda la programación realizada para la Aplicación Android.

2.5.2.- PROCESAMIENTO

En esta etapa, están mostrados todos los pasos a seguir para el muestreo de la señal entrante, su conversión análoga-digital, la recepción de los datos enviados desde la wi-fly, la convolución y demás operaciones para filtrar la señal, la generación del PWM[32] de la señal de salida, y la reconstrucción de la señal digital a una señal analógica.

2.5.2.1.- ALIMENTACIÓN

La alimentación entrante es de una fuente de voltaje de 6.5 Voltios, este voltaje de entrada parte hacia 3 reguladores de voltaje de 5, 3.3 y 1.8 voltios respectivamente para energizar a los diferentes elementos.

Los 5 voltios son utilizados para energizar operacionales que están presenten en la etapa de entrada de la señal de audio; el regulador utilizado para esto es el AMS1117 de 5V.

El voltaje máximo que soporta el módulo wi-fly son 3.3 voltios, para reducir el voltaje hasta esta cantidad, es utilizado el mismo regulador AMS1117, pero para 3.3 voltios.

Finalmente para la reducción hasta 1.8 voltios para energizar los microcontroladores, es utilizado el AMS1117 para 1.8 voltios de salida; la configuración para estos reguladores, es mostrada en la **Figura 31.**

[32] Pulse-Width Modulation (Modulación por Ancho de Pulsos)

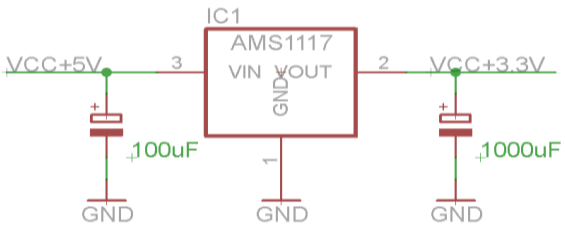


Figura 31.- Configuración del regulador AMS1117.
Fuente.- Datasheet AMS1117

2.5.2.2.- MANIPULACIÓN DE LA SEÑAL DE ENTRADA

Para recibir la señal de entrada, es necesario utilizar un conector de 3,5 mm estéreo; la configuración de este conector, es mostrado en la **Figura 32.**

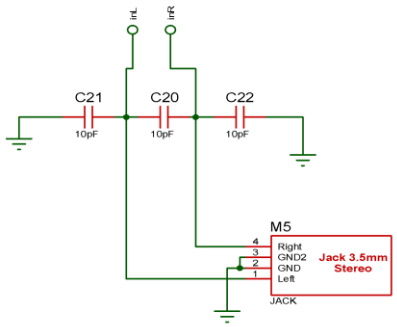


Figura 32.- Configuración del conector de entrada de audio
Fuente.- El autor

Los capacitores son utilizados para atenuar el ruido de la señal entrante.

Una señal estéreo cuenta con tres componentes, la referencia, la componente izquierda (L) y la componente derecha (R); la **Figura 33** muestra la distribución de estas señales en el conector utilizado.

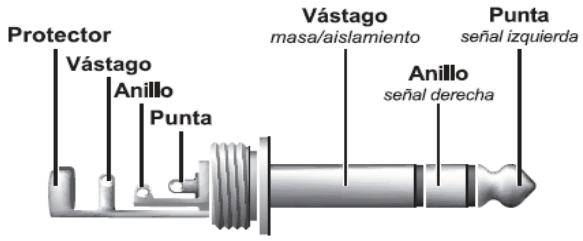
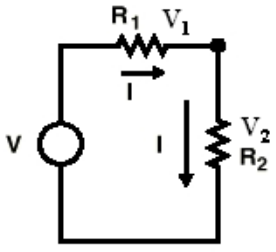


Figura 33.- Distribución de las componentes en un conector estéreo
Fuente.-
http://www.tuelectronica.es/images/tutoriales/audio/conectores_audio/jack_635mm.gif

Por motivos de velocidad del microcontrolador y presupuesto, solo es tomada una de las dos componentes de la señal, en este caso el canal izquierdo (L); ya que el canal con más componentes es éste, cada filtro es cargado en un microcontrolador. De acuerdo con las señales de audio emitidas por los dispositivos electrónicos (computador), es considerado un rango máximo de voltaje pico pico de 4 voltios; mientras que el microcontrolador soporta máximo 1,6 voltios pico; es por esto que es utilizado un divisor de voltaje con las fórmulas que muestra la **Figura 34**:



$$V_2 = \frac{R_2}{R_1 + R_2} V$$

$$V_1 = \frac{R_1}{R_1 + R_2} V$$

Figura 34.- Cálculo de un divisor de voltaje
Fuente.- http://www.electro.com/trabajos40/circuitos-electricos/Image608.gif

Al conocer los voltajes de entrada y salida; se despeja la ecuación y se encuentra una de las resistencias; la resistencia R1 es considerada de 1K; por lo tanto la resistencia R2 es calculada de la siguiente manera:

$$V_2 = \frac{R2}{(R1 + R2)} V_1$$

$$(R1 + R2) = \frac{R2}{V_2} V_1$$

$$(R1 + 1K) = \frac{1K}{1{,}6V} * 4V$$

$$(R1 + 1K) = \frac{4K}{1{,}6}$$

$$R1 = 2{,}5K - 1K$$

$$R1 = 1{,}5\ K$$

Los microcontroladores solamente trabajan en el ciclo positivo, por lo que es necesario utilizar un sujetador; la **Figura 35** muestra el esquema utilizado:

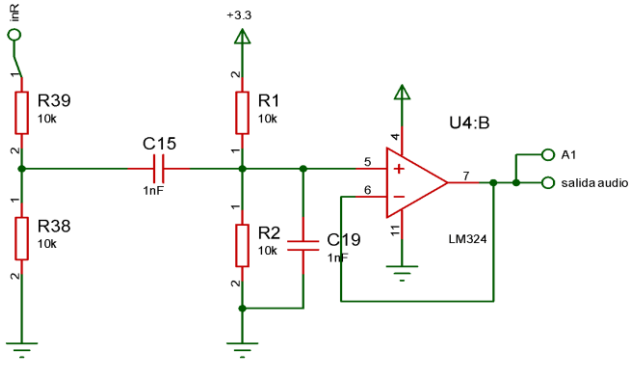


Figura 35.- Divisor de voltaje y sujetador de la señal de audio
Fuente.- El autor

2.5.2.3.- PROGRAMACIÓN

El primer punto es incluir las librerías a ser utilizadas dentro de este proceso; la **Figura 36** muestra la declaración:

```
#include <asf.h>
#include "coeficientes.h"
#include <math.h>
```

Figura 36.- Declaración de las librerías en Atmel Studio
Fuente.- El autor

Existe una diferencia al utilizar los signos: <> y los signos “”; en este caso las comillas indican que es una librería creada por el usuario, en la librería coeficientes, son añadidos los coeficientes utilizados en los filtros al iniciar el programa.

El siguiente paso es declarar las variables utilizadas, la **Figura 37** muestra los comandos utilizados:

```
#define EXAMPLE_TARGET_CLK_ADC_FREQ_HZ 150000
#define BUFFER_LENGTH             1024
#define FILTER_LENGTH             200
#define FILTRO_PASATODO           0
#define FILTRO_PASABAJOS          1
#define FILTRO_PASAALTOS          2
#define FILTRO PASABANDA          3
```

Figura 37.- Variables declaradas.
Fuente.- El autor

El microcontrolador no cuenta con un DAC, en su lugar, reconstruye la señal mediante un PWM de alta frecuencia (150 KHz). Se declaran dos buffer, el primero, donde la señal entrante es almacenada, y el segundo donde son almacenados los coeficientes del filtro. Los valores del 0 al 4 indican las opciones de selección del filtro.

Otros elementos que deben habilitarse; son el puerto USART y el ADC del microcontrolador; para esto son utilizados los comandos mostrados en las **Figuras 38** y **39.**

```
volatile avr32_pwma_t *pwma = &AVR32_PWMA;
int32_t aux;
volatile     uint32_t adc_data;
uint32_t config_status,set_value_status,divi, cpu_speed;
volatile avr32_adcifb_t *adcifb = ADC;
static void tc_init(volatile avr32_tc_t *tc);
static status_code_t adc_init2();
static void tc_irq(void);
ISR(ADCIFB_interrupt_handler, AVR32_ADCIFB_IRQ_GROUP,
ADC_INTERRUPT_PRIORITY);
static void usart_int_handler(void);
volatile float Wc,fc,fp,fs, delta_f, frec_corte, wa, wb, fb, fa;
void coeficientes_filtro_hamming(uint8_t filter);
static void usart_init(void);
```

Figura 38.- Habilitación del ADC.
Fuente. –El autor

```
static void tc_init(volatile avr32_tc_t *tc);
static void tc_irq(void);
static void usart_int_handler(void);
volatile float Wc,fc,fp,fs, delta_f, frec_corte, wa, wb, fb, fa;
void coeficientes_filtro_hamming(uint8_t filter);
static void usart_init(void);
```

Figura 39.- Habilitación del puerto USART
Fuente.- El autor

La **Figura 40** muestra el flujograma de la programación, de allí pueden extraerse algunas consideraciones:

El rango audible del ser humano es hasta los 20 KHz, tomando en cuenta el teorema de muestreo de Nyquist es necesario que como mínimo la frecuencia de muestreo sea 40 KHz; el microcontrolador utiliza una frecuencia de muestreo de 48 KHz; ya que este valor es un estándar en el muestreo de señales de audio digital; por lo cual el tiempo de muestreo es:

$$Ts = \frac{1}{Fs}$$

$$Ts = \frac{1}{48\ KHz}$$

$$Ts = 20{,}83\ us$$

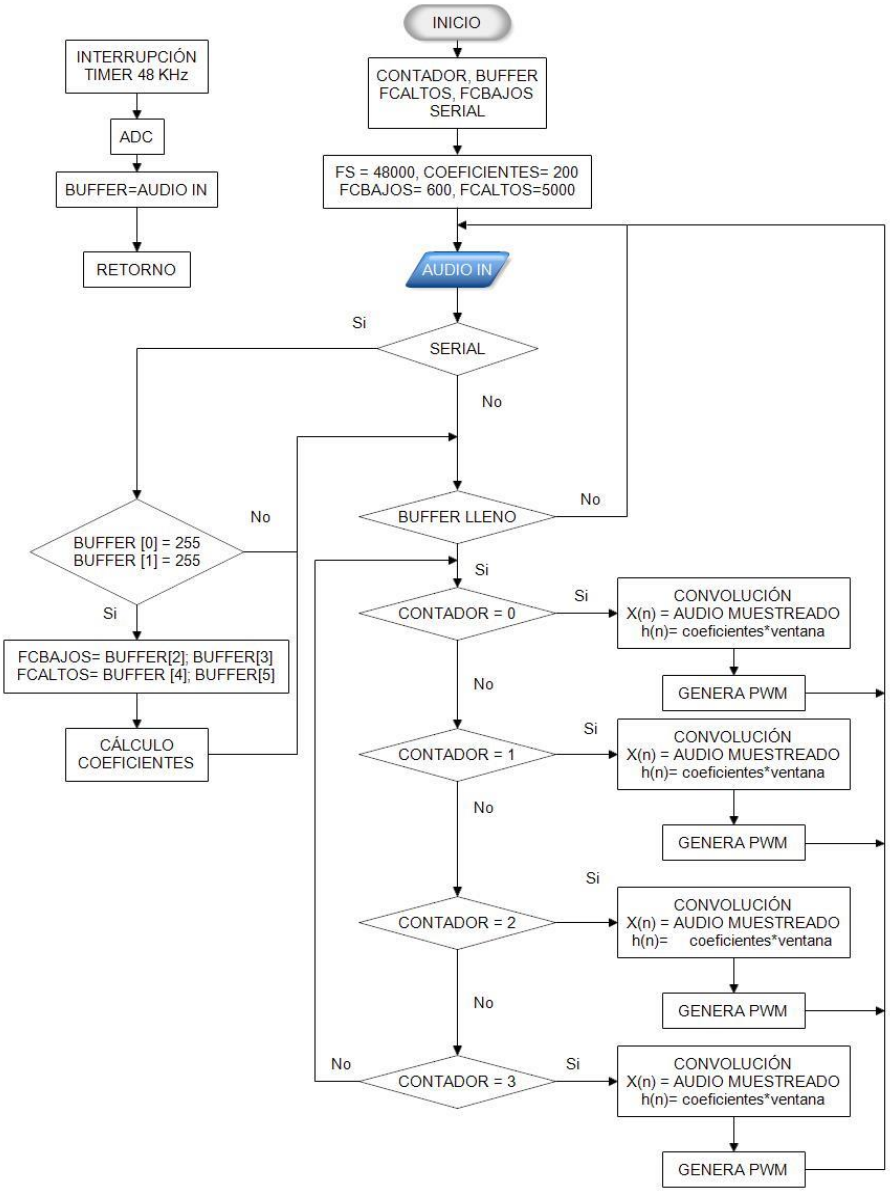


Figura 40.- Diagrama de flujo de los filtros
Fuente.- El autor

El microcontrolador pasa a la interrupción para tomar el dato cada 20 us aproximadamente.

El dato es tomado y almacenado en dos de 1024 bytes, el cual procede a realizar la convolución cuando uno está lleno mientras que el otro continua tomando los datos, la convolución es realizada entre la señal entrante y los coeficientes obtenidos por el microcontrolador; las frecuencias de corte iniciales son 600 Hz y 5000 Hz; el **Anexo 3** muestra los coeficientes utilizados cuando el crossover arranca; cuando el microcontrolador recibe información por el puerto USART; las frecuencias de corte son modificadas; el microcontrolador almacena los valores enviados solamente cuando cumplan la condición que los dos primeros sean "1 lógico. La **Figura 41** muestra la programación de estas condiciones.

```
if (arribo_serial==1){
arribo_serial=0;
if ((puntero_serial==0)&&(dato_serial==255)){
puntero_serial=1;}
else if ((puntero_serial==1)){
if (dato_serial==255){
puntero_serial=2;}
else{
puntero_serial=0;}}
else if (puntero_serial==2){
frecuencia_corte_filtrob=((uint32_t)dato_serial)<<8;
puntero_serial=3;}
else if (puntero_serial==3){
frecuencia_corte_filtrob|=((uint32_t)dato_serial);
puntero_serial=4;}
else if (puntero_serial==4)
{frecuencia_corte_filtroa=((uint32_t)dato_serial)<<8;
puntero_serial=5;}
else if (puntero_serial==5){
frecuencia_corte_filtroa|=((uint32_t)dato_serial);}
```

Figura 41.- Programación para almacenar la trama enviada por el dispositivo Android.
Fuente.- El autor

Después que el microcontrolador recibe la trama y la almacena, se generan los nuevos coeficientes; en el marco teórico están explicadas las fórmulas utilizadas en esta programación, una de ellas es la banda de transición:

$$Banda\ de\ transición = \Delta F * frecuencia\ de\ muestreo$$

La ventana utilizada es la ventana de Hamming, de donde la banda de transición normalizada equivale a:

$$\Delta F = \frac{3{,}3}{N}$$

Siendo N el número de coeficientes utilizados (200); la banda de transición normalizada resulta ser:

$$\Delta F = 0{,}0165$$

Mediante esta fórmula, es hallada la banda de transición que equivale a:

$$banda\ de\ transición = 0{,}0165 * 48000$$

$$banda\ de\ transición = 792\ Hz$$

Con las nuevas frecuencias de corte, son calculados los respectivos coeficientes; matemáticamente son utilizadas ciertas fórmulas según el tipo de filtro.

2.5.2.3.1.- FILTRO PASA BAJOS

En los cálculos de los coeficientes del filtro pasa bajos, son utilizadas las fórmulas:

$$fp = \frac{frecuencia\ de\ corte}{f\ de\ muestreo}$$

$$fc = fp + \frac{\Delta F}{2}$$

$$Wc = 2\pi fc$$

$$\begin{cases} n \neq 0 \rightarrow \dfrac{\sin(n * Wc)}{n * \pi} \\ n = 0 \rightarrow 2\,fc \end{cases}$$

La programación de estas ecuaciones, es mostrada en la **Figura 42.**

```
if (filter==1) // filtro pasa bajos
       {
       delta_f = 3.3/FILTER_LENGTH;
       frec_corte=(float)frecuencia_corte_filtrob;
       fs= 48000;
       fp=frec_corte/fs;
       fc = fp+delta_f/2;
       Wc = 2*CST_PI*fc;

       for (int n=-FILTER_LENGTH/2; n<=FILTER_LENGTH/2;n++)
       {
       if (n!=0)coeficientes [n+FILTER_LENGTH/2] =  sin(n*Wc)/(n*CST_PI);
       else coeficientes [n+FILTER_LENGTH/2] =  2*fc;
       coeficientes_filtro[n+FILTER_LENGTH/2]= DSP16_Q(coeficientes
[n+FILTER_LENGTH/2]);
             }
```

Figura 42.- Generación de los coeficientes del filtro pasa bajos
Fuente.- El autor

2.5.2.3.2.- FILTRO PASA ALTOS.

En los cálculos de los coeficientes del filtro pasa altos, son utilizadas las fórmulas:

$$fp = \frac{frecuencia\ de\ corte}{f\ de\ muestreo}$$

$$fc = fp - \frac{\Delta F}{2}$$

$$Wc = 2\pi fc$$

$$\begin{cases} n \neq 0 \ \rightarrow -\dfrac{\sin(n * Wc)}{n * \pi} \\ n = 0 \ \rightarrow 1 - 2\,fc \end{cases}$$

La programación de estas ecuaciones, es mostrada en la **Figura 43.**

```
if (filter==2)  //  filtro pasa altos
      {
      delta_f = 3.3/FILTER_LENGTH;
      frec_corte=(float)frecuencia_corte_filtroa;
      fs= 48000;

            fp=frec_corte/fs;
            fc = fp-delta_f/2;
            Wc = 2*CST_PI*fc;

            for (int n=-FILTER_LENGTH/2; n<=FILTER_LENGTH/2;n++)
            {
                  if (n!=0)coeficientes [n+FILTER_LENGTH/2] =  -
sin(n*Wc)/(n*CST_PI);
                  else coeficientes [n+FILTER_LENGTH/2] =  1-2*fc;
                  coeficientes_filtro[n+FILTER_LENGTH/2]= DSP16_Q(coeficientes
[n+FILTER_LENGTH/2]);}
      }
```

Figura 43.- Generación de los coeficientes del filtro pasa altos.
Fuente.- El autor

2.5.2.3.3.- FILTRO PASA BANDA.

En los cálculos de los coeficientes del filtro pasa banda, son utilizadas las fórmulas:

$$fp = \frac{frecuencia\ de\ corte}{f\ de\ muestreo}$$

$$fb = fp - \frac{\Delta F}{2}$$

$$Wb = 2\pi fb$$

$$fa = fp + \frac{\Delta F}{2}$$

$$Wa = 2\pi fa$$

$$\begin{cases} n \neq 0 \rightarrow \dfrac{sin\,(n * Wa) - \sin(n * wb)}{n * \pi} \\ n = 0 \rightarrow 2 * (fa - fb) \end{cases}$$

La programación de estas ecuaciones, es mostrada en la **Figura 44.**

```
if (filter==3) // filtro pasa banda
       { delta_f = 3.3/FILTER_LENGTH;
       frec_corte=(float)frecuencia_corte_filtrob;
if ((frecuencia_corte_filtrob<=600)){
frecuencia_corte_filtrob=600;}
                fs= 48000;
                fp=frec_corte/fs;
                fb = fp-delta_f/2;
                wb = 2*CST_PI*fb;
                frec_corte=(float)frecuencia_corte_filtroa;
                fs= 48000;
                fp=frec_corte/fs;
                fa = fp+delta_f/2;
                wa = 2*CST_PI*fa;
for (int n=-FILTER_LENGTH/2; n<=FILTER_LENGTH/2;n++){
if (n!=0)coeficientes [n+FILTER_LENGTH/2] =  sin(n*wa)/(n*CST_PI)-sin(n*wb)/(n*CST_PI);
else coeficientes [n+FILTER_LENGTH/2] =  2*(fa-fb);
coeficientes_filtro[n+FILTER_LENGTH/2]= DSP16_Q(coeficientes [n+FILTER_LENGTH/2]);
```

Figura 44.- Generación de los coeficientes del filtro pasa banda.
Fuente.- El autor

En el marco teórico está indicado que la convolución es una operación matemática entre h(n) y x(n).

Al usar el método de ventaneo, h(n) es el resultado del producto entre hd(n) y la ventana w(n)

Los coeficientes del filtro son los elementos de hd(n); Para adquirir estos valores es necesario contar con las nuevas frecuencias de corte, la programación lo muestra la **Figura 45.**

```
dsp16_gen_step(hamming_coef, FILTER_LENGTH, DSP16_Q(1.), DSP16_Q(1.),
0);
dsp16_win_hamm(hamming_coef, hamming_coef, FILTER_LENGTH);
dsp16_vect_dotmul(coeficientes_filtro,coeficientes_filtro,hamming_coef
,FILTER_LENGTH);
      for (int n=0; n<FILTER_LENGTH;n++)
      {

      coeficientes[n]=(double)(coeficientes_filtro[n])/pow(2,15);
      }
```

Figura 45.- Producto entre la ventana Hamming y los coeficientes de los filtros
Fuente.- El autor

Después de encontrar los valores de h(n); sigue la convolución con la señal de entrada y la generación del PWM; los comandos lo muestra la **Figura 46.**

```
divi = div_ceil((sysclk_get_pba_hz()) , 120000000);
      genclk_enable_config(AVR32_SCIF_GCLK_PWMA,GENCLK_SRC_RC120M,divi
);
      gpio_enable_module_pin(led1,pwm_led);
      gpio_configure_pin(led, GPIO_DIR_OUTPUT | GPIO_INIT_LOW);
      gpio_configure_pin(button,GPIO_DIR_INPUT|GPIO_PULL_UP);

      config_status = pwma_config_enable(pwma, 480000,
      120000000,0);
      set_value_status = pwma_set_channels_value(pwma,(1<<25),50);
```

Figura 46.- Generación del PWM
Fuente.- El autor

El **Anexo 2** contiene toda la programación de los filtros FIR utilizados.

2.5.2.4.- OBTENCIÓN DE LA SEÑAL DE SALIDA

Después que es generado el PWM, es necesario atravesar la señal por un filtro R-C para obtener nuevamente una señal analógica similar a la original.

El marco teórico indica que la ecuación de la frecuencia de corte es:

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

Para evitar que la señal obtenida tenga demasiado ruido, se considera que la frecuencia de corte sea 10 veces menor que la frecuencia del PWM; por lo tanto:

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

$$\frac{150000}{10} = \frac{1}{2\pi RC}$$

Asumiendo un capacitor de 0,1 uf, hay que resolver la ecuación y encontrar el valor de la resistencia:

$$15000 = \frac{1}{2\pi R(0{,}1uf)}$$

$$R = \frac{1}{2\pi(1500000)(0{,}1uf)}$$

$$R = \frac{1}{94247{,}7796 * (0{,}0000001)}$$

$$R = \frac{1}{0{,}00942477}$$

$$R = 106{,}103$$

Finalmente para obtener la señal de salida, es necesario que pase por un circuito seguidor para pre-amplificar la corriente de salida. La **Figura 47** muestra el diagrama del circuito RC y la etapa de pre-amplificación.

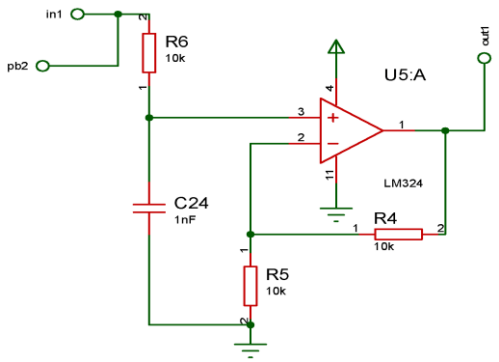


Figura 47.- Circuito RC / Pre-amplificador
Fuente.- El autor

Para el acople de los conectores de la señal de salida, es usado un conector hembra de 3,5 mm; son utilizados también capacitores para acoplar la señal y filtrar el ruido. La **Figura 48** muestra esta configuración.

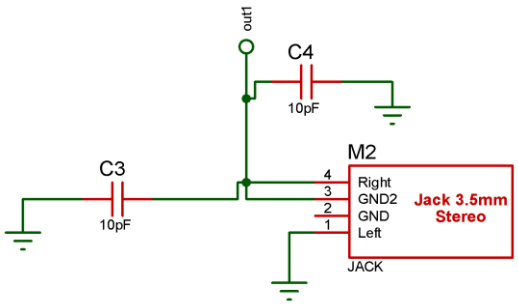


Figura 48.- Configuración de los conectores de salida
Fuente.- El autor

Son añadidos también pulsadores, para cambiar el tipo de filtro de cada microcontrolador; en la placa también es añadido el módulo wi-fly con su respectiva configuración. Todos estos elementos forman parte de una placa especialmente diseñada para el procesamiento de audio, esta placa fue realizada por parte de www.egm-

robotics.com; empresa la cual recibe el diseño de la circuitería y la realiza en una placa especializada para audio con microcomponentes. El diseño de la placa utilizada, lo muestra la **Figura 49.**

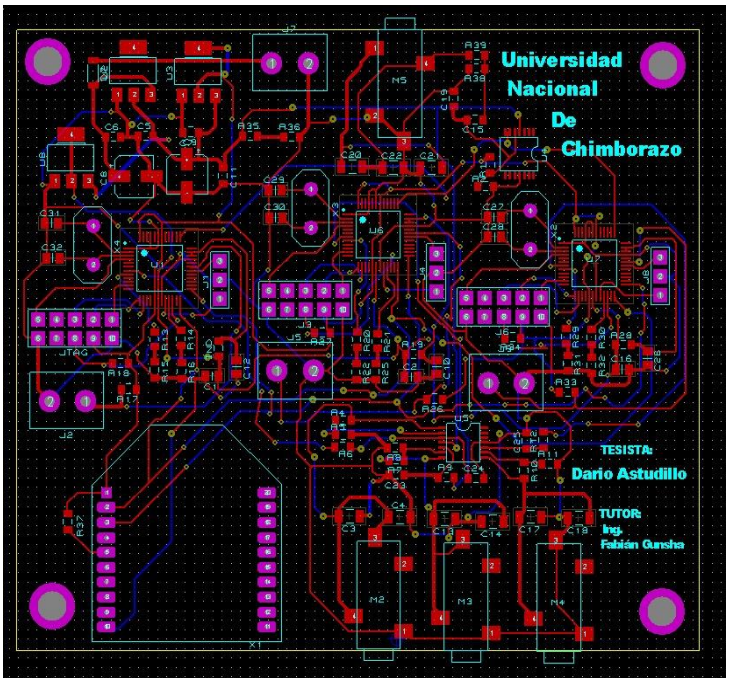


Figura 49.- Placa final del crossover de tres vías
Fuente.- El autor

2.5.3.- AMPLIFICACIÓN

Dentro de la etapa de amplificación, son tomados en cuenta algunos pasos a seguir, a continuación están detalladas las actividades realizadas en esta etapa.

2.5.3.1.- IMPLEMENTACIÓN DEL AMPLIFICADOR

Después que la señal pasa por la digitalización y el filtrado, como resultado son obtenidas tres salidas de audio, las cuales son las frecuencias bajas, las frecuencias medias, y las frecuencias altas; cada señal de salida es conectada a un amplificador de audio estéreo

de 18 watts de potencia por canal, el **Anexo 6** muestra las características detalladas de este amplificador; la **Figura 50** muestra el circuito descrito.

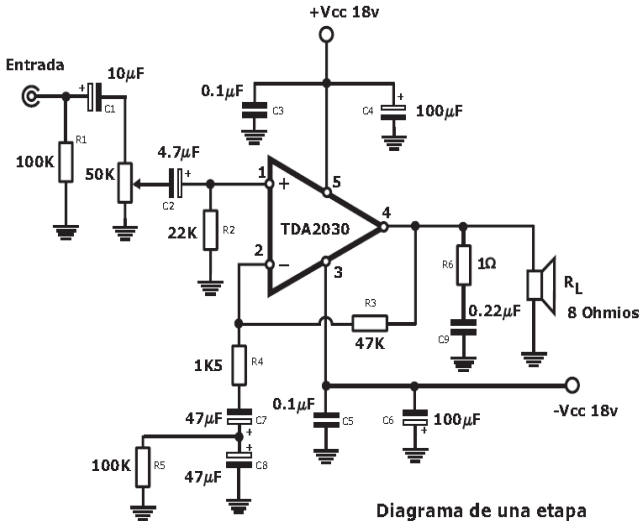


Figura 50.- Diagrama electrónico del amplificador de Audio
Fuente.- http://4.bp.blogspot.com/-RKylDggulsc/TnnmLVvJ7KI/AAAAAAAAAG4/QegHI0W43qQ/s1600/amp_30w.JPG

El elemento encargado de la amplificar la corriente es el TDA2030, el cual fue detallado en la fundamentación teórica. La **Figura 51** muestra la placa concluida.

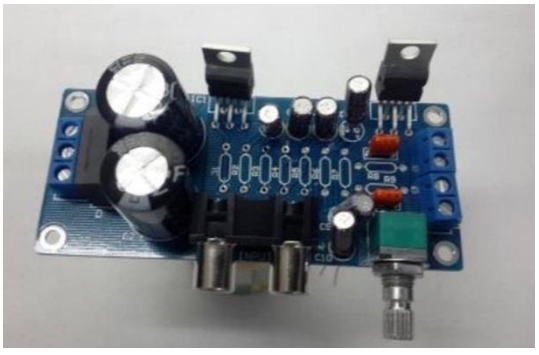


Figura 51.- Amplificador de audio terminado
Fuente.- El autor

Otros de los elementos presentes en este diseño son: un conector de plug estéreo hembra, por donde es adquirida la señal de entrada hacia el amplificador, las salidas hacia los altavoces, y un potenciómetro doble, el cual es el encargado de regular la potencia de salida de cada canal hacia los altavoces.

Como son tres tipos de altavoces, son utilizados 3 amplificadores de 18 watts, uno para cada par de altavoces.

2.5.3.2.- DISEÑO Y EQUIPAMENTO DE LAS CAJAS PARA ALTAVOCES

Para este proyecto, fueron diseñadas dos cajas para altavoces, dentro de las cuales están los tres tipos de altavoces para las frecuencias respectivas, sus amplificadores y su fuente de alimentación que en este caso es un transformador de 10 voltios de corriente alterna; entre otras cosas detalladas a continuación.

CAJA.- La caja de los altavoces consta de 6 tablas de 1 cm de profundidad aproximadamente, cubiertas de fieltro, cuyas dimensiones se muestran en la **Tabla 8.**

UBICACIóN	**DIMENSIóN (Cm)**
Frente	60 x 30
Trasera	62 x 30
Lateral Izquierda	60 x 25
Lateral Derecha	60 x 25
Arriba	27 x 30
Abajo	27 x 30

Tabla 8.- Dimensiones de la caja para altavoces
Fuente.- El autor

En la tabla frontal, se realizan los agujeros donde serán ubicados los altavoces, el sonido no sale solamente por la parte frontal del altavoz, también sale por la parte trasera aunque en menor potencia, por lo cual los sistemas de altavoces deben contar con agujeros de desfogue, en este caso existen dos agujeros en cada caja. La **Figura 52** muestra las dimensiones de los orificios para los altavoces.

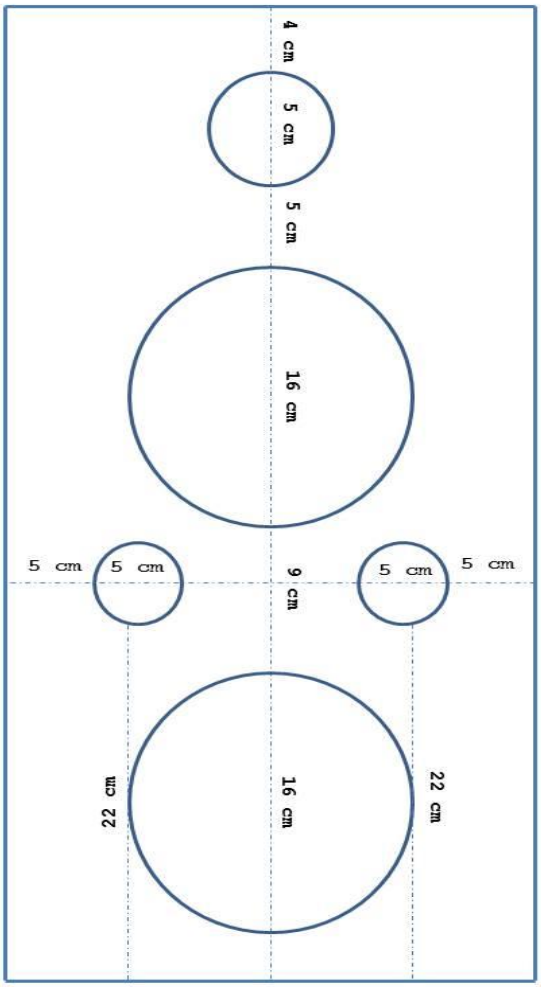


Figura 52.- Dimensiones de los orificios de la Tabla Frontal
Fuente.- El autor

ALTAVOCES.-Son utilizados altavoces pasivos[33] ; los cuales son clasificados de la siguiente manera:

Para las bajas frecuencias, fueron utilizados dos woofer de seis pulgadas, marca American Xtreme, serie DW-166; de 6 ohmios de resistencia, y con una potencia máxima de 250 watts; la **Figura 53** muestra el woofer a utilizarse.

Figura 53.- Woofer – American Xtreme
Fuente.- www.americanxtreme.com

Para las frecuencias medias, son utilizados dos squawker de seis pulgadas, marca American Xtreme, serie AX-630; de 4 ohmios de resistencia, y con una potencia máxima de 30 watts; la **Figura 54** muestra el squawker a utilizarse.

Figura 54.- Squawker – American Xtreme
Fuente.- www.americanxtreme.com

[33] No necesitan alimentación directa

Finalmente, para las frecuencias altas, fueron utilizados dos tweeters de 3 pulgadas, marca Acoustic, serie KP105; cuatro ohmios de resistencia; potencia máxima de 75 watts. La **Figura 55** muestra el tweeter que utiliza este proyecto.

Figura 55.- Tweeter – Acoustic
Fuente.- http://www.telerepuestos.com/787-home_default/kp307m.jpg

Los elementos están distribuidos de la siguiente manera:

- Dentro de una caja: los altavoces respectivos y el transformador de 10 voltios; las **Figuras 56, 57,58** muestran el proceso de construcción de las cajas.

Figura 56.- Perforaciones de la caja para altavoces
Fuente.- El autor

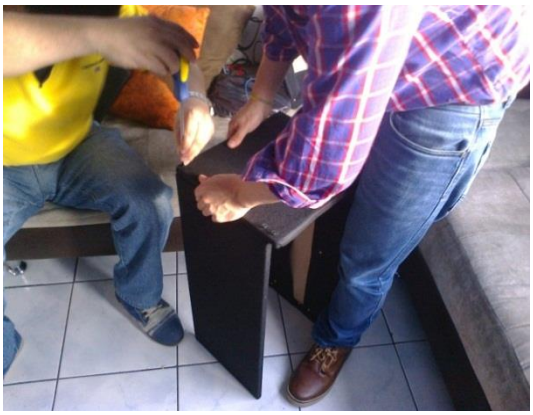


Figura 57.- Ensamblaje de la caja para altavoces
Fuente.- El autor

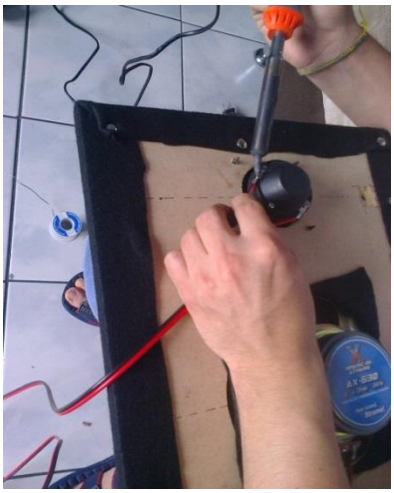


Figura 58.- Instalación de los altavoces en la Tabla frontal
Fuente.- El autor

Después de haber hecho la instalación de los altavoces, se ubica el transformador de 10 voltios, el cual alimenta a los amplificadores; en la Tabla posterior, son colocados 3 conectores RCA los cuales van enlazados a los tres altavoces de la caja, además de dos conectores banano hembra para la alimentación. La **Figura 59** muestra los altavoces y el transformador de la primera caja.

Figura 59.- Elementos internos de la caja de altavoces
Fuente.- El autor

- Dentro de la segunda caja, son ubicados los altavoces, son instalados los tres amplificadores responsables de enviar la señal hacia los altavoces; las **Figuras 60,61** muestran el procedimiento para la instalación de esta caja. El circuito del amplificador cuenta con un led que indica cuando el amplificador está encendido; y también con un potenciómetro doble el cual está encargado de regular el volumen de los altavoces.

Figura 60.- Ubicación de los elementos en la Tabla frontal
Fuente.- El autor

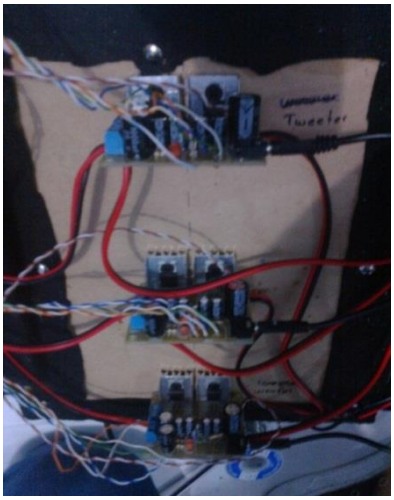


Figura 61.- Ubicación de los amplificadores.
Fuente.- El autor

En la Tabla posterior, son colocados 3 conectores RCA los cuales van enlazados desde cada amplificador a los tres altavoces de la otra caja, son conectados tres cables "auxiliares" estéreo, los cuales reciben la señal de audio filtrada enviada por el crossover. La **Figura** 62 muestra las dos cajas de altavoces finalizadas.

Figura 62.- Cajas de altavoces finalizadas
Fuente.- El autor

De esta manera, están concluidas todas las etapas de diseño e implementación del Crossover de tres vías.

2.6.- PROCESAMIENTO Y ANÁLISIS

La **Figura 63** muestra el proceso seguido durante la investigación, se presenta un resumen de las etapas que conforman el proyecto, también están presentes, las herramientas y equipos utilizados dentro de cada una de las etapas.

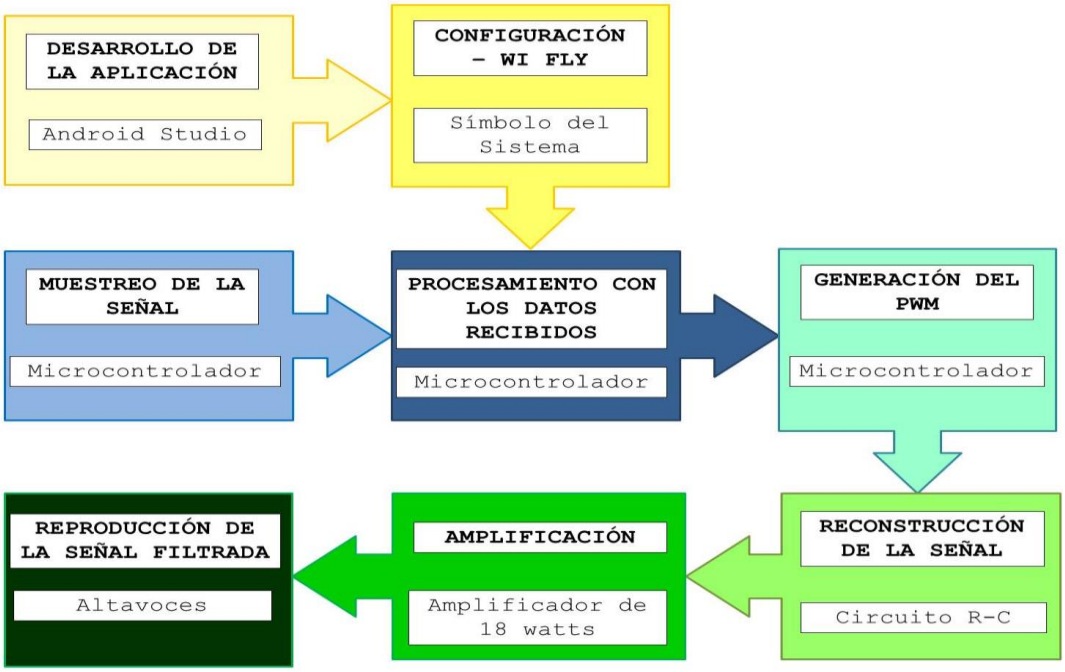


Figura 63.- Diagrama de bloques del Procedimiento

Fuente.- El autor

2.6.1.- ENVÍO DE LOS DATOS

El dispositivo con sistema operativo Android es comunicado con la antena receptora wi-fly a través de la red domiciliaria, éstos dos dispositivos entablan comunicación entre sí siempre y cuando estén en la misma red, sin importan si dicha red no tiene acceso a internet. De esta manera son adquiridos los datos que

después utilizará el microcontrolador para el diseño de los filtros respectivos; la **Figura 66** muestra el diagrama del proceso para que la información llegue desde el dispositivo hasta el microcontrolador.

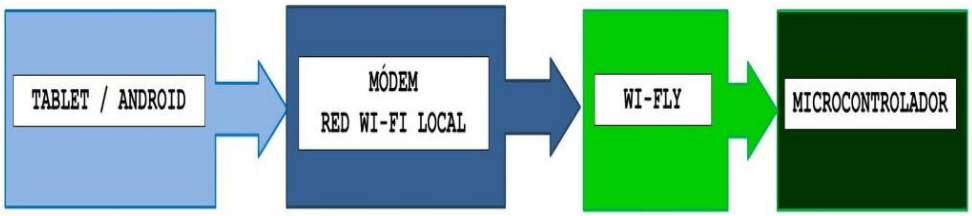


Figura 64.- Etapas para la comunicación entre el dispositivo Android y la Wi-fly
Fuente.- El autor

2.6.2.- PROCESAMIENTO DE LA SEÑAL DE AUDIO

El microcontrolador está encargado de recibir la señal de audio entrante y procesarla para que atraviese los filtros y al final sea reconstruida nuevamente, esto es realizado mediante el proceso matemático de la convolución, en la convolución son necesarios los datos tomados del muestreo de la señal entrante, además de los coeficientes del filtro, al finalizar, es generado un PWM con las operaciones realizadas, para después ser enviado a un circuito R-C el cual está encargado de reconstruir la señal del PWM a una señal analógica similar a la original.

La **Figura 65**, muestra los pasos que sigue el microcontrolador para obtener la señal resultante.

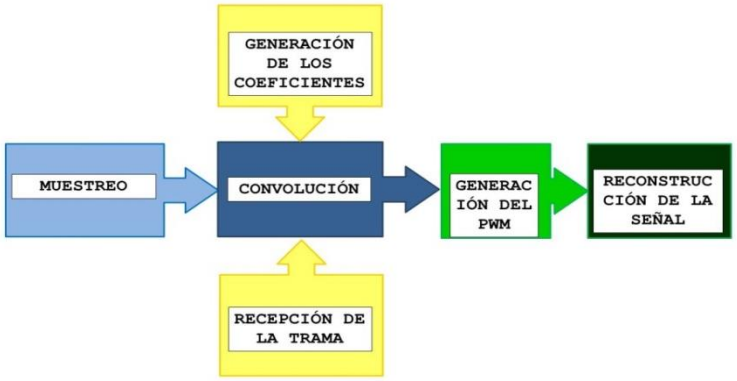


Figura 65.- Etapas para el procesamiento de la Señal de Audio
Fuente.- El autor

2.6.3.- AMPLIFICACIÓN DE LA SEÑAL DE SALIDA

Después que los microcontroladores reconstruyen la señal muestreada, se utiliza una etapa de amplificación en vista que los altavoces utilizados no cuentan con alimentación sino que su alimentación es realizada a través de los pulsos que envía un amplificador externo.

2.7.- COMPROBACIÓN DE LA HIPÓTESIS

El procedimiento mostrado a continuación, sigue un conjunto de pasos para proceder a afirmar o negar la hipótesis planteada.

Las pruebas realizadas, siguen un esquema de aseveración dentro de una media poblacional definida.

2.7.1.- PLANTEAMIENTO DE LA HIPÓTESIS

2.7.1.1.- HIPÓTESIS NULA (H0)

El diseño e implementación de un crossover de tres vías utilizando DSP con parámetros controlados por un dispositivo Android no permitirá distribuir

eficazmente las componentes de una señal de audio debido a las frecuencias variables.

2.7.1.2.- HIPÓTESIS ALTERNATIVA (H1)

El diseño e implementación de un crossover de tres vías utilizando DSP con parámetros controlados por un dispositivo Android permitirá distribuir eficazmente las componentes de una señal de audio debido a las frecuencias variables.

2.7.2.- ESTABLECIMIENTO DEL NIVEL DE SIGNIFICANCIA.

Para obtener un resultado confiable para la investigación, es escogido un nivel de significancia de 0,05; lo que representa un nivel de 0,95 de confianza.

2.7.3.- DETERMINACIÓN DEL VALOR ESTADÍSTICO DE PRUEBA

Para la comprobación de la hipótesis, es utilizado el método estadístico de Chi-cuadrado mediante la prueba de ajuste, en la que son tomados en cuenta el número de éxitos realizados en las pruebas; en este caso el número de veces que los filtros permitieron el paso de las respectivas frecuencias.

Para la aceptación de la hipótesis nula, o de la hipótesis alternativa, se debe tomar en cuenta la relación del resultado con los siguientes parámetros:

H0 es aceptada si: $X_E^2 \leq X_C^2$

H1 es aceptada si: $X_E^2 > X_C^2$

Las pruebas fueron realizadas con 35 muestras, por lo cual fueron tomados 35 valores dentro del rango de 100 y

20000 Hz; dichos valores fueron tomados uniformemente. La **Tabla 9** muestra los valores recogidos según las frecuencias y el orden del filtro, y también muestra los valores de entrada de dichas frecuencias.

MUESTRA	FRECUENCIA (Hz)	VOLTAJE IDEAL (V)			VOLTAJE REAL (V)		
		PASA BAJOS	PASA BANDA	PASA ALTOS	PASA BAJOS	PASA BANDA	PASA ALTOS
1	100	1,25	0	0	1,2	0,064	0,16
2	670	1,25	0	0	1,16	0,264	0,16
3	1240	1,25	1,25	0	1,04	1,09	0,16
4	1810	0	1,25	0	0,2	1	0,16
5	2380	0	1,25	0	0,16	0,84	0,16
6	2950	0	1,25	0	0,16	0,76	0,16
7	3520	0	1,25	0	0,16	0,64	0,4
8	4090	0	0	1,25	0,16	0,44	0,45
9	4660	0	0	1,25	0,16	0,12	0,56
10	5230	0	0	1,25	0,16	0,12	0,52
11	5800	0	0	1,25	0,16	0,12	0,44
12	6370	0	0	1,25	0,16	0,12	0,48
13	6940	0	0	1,25	0,16	0,16	0,48
14	7510	0	0	1,25	0,16	0,16	0,312
15	8080	0	0	1,25	0,16	0,12	0,376
16	8650	0	0	1,25	0,16	0,16	0,344
17	9220	0	0	1,25	0,16	0,16	0,352
18	9790	0	0	1,25	0,16	0,16	0,312
19	10360	0	0	1,25	0,16	0,12	0,376
20	10930	0	0	1,25	0,16	0,2	0,328
21	11500	0	0	1,25	0,16	0,24	0,352

22	12070	0	0	1,25	0,16	0,28	0,296
23	12640	0	0	1,25	0,2	0,28	0,352
24	13210	0	0	1,25	0,2	0,36	0,272
25	13780	0	0	1,25	0,24	0,36	0,272
26	14350	0	0	1,25	0,4	0,32	0,296
27	14920	0	0	1,25	0,24	0,36	0,216
28	15490	0	0	1,25	0,48	0,36	0,28
29	16060	0	0	1,25	0,44	0,36	0,248
30	16630	0	0	1,25	0,2	0,24	0,24
31	17200	0	0	1,25	0,28	0,28	0,272
32	17770	0	0	1,25	0,28	0,36	0,2
33	18340	0	0	1,25	0,2	0,36	0,264
34	18910	0	0	1,25	0,2	0,28	0,312
35	19480	0	0	1,25	0,2	0,24	0,336

Tabla 9.- Datos tomados de las pruebas con los filtros
Fuente.- El autor

La tabla anterior es reducida de la siguiente manera, un promedio de los datos tomados por cada filtro en su banda pasante, y el promedio de los datos tomados por cada filtro en la banda de rechazo; teniendo como resultado la **Tabla 10.**

VALORES REALES	PASA BAJOS	PASA BANDA	PASA ALTOS
BANDA PASANTE	1,133333333	0,866	0,340642857
BANDA DE RECHAZO	0,2075	0,266387097	0,194285714

Tabla 10.- Promedio de los voltajes medidos
Fuente.- El autor

Esta tabla es comparada con la tabla de los valores esperados en cada caso, para después proceder al cálculo del Chi cuadrado; la **Tabla 11** muestra los siguientes valores esperados:

	PASA BAJOS	PASA BANDA	PASA ALTOS
BANDA PASANTE	1,25	1,25	1,25
BANDA DE RECHAZO	0,01	0,01	0,01

Tabla 11.-Tabla de niveles de voltaje esperados en los filtros
Fuente.- El autor

Tomando en cuenta los datos obtenidos, y los principios del método estadístico, se considera que:

Número de pruebas (n) = 35

Grado de libertad (G1) = (cantidad de filas – 1)*(cantidad de columnas – 1) = (2-1)*(3-1)=1*2= 2

Nivel de significancia = 0,95

P= 1-nivel de significancia = 0,05

Chi-cuadrado estadístico = $X_E^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{Ei}$

La **Tabla 12** muestra las operaciones realizadas en Excel para encontrar en valor calculado de Chi-cuadrado:

		PASA BAJO	PASA BANDA	PASA ALTOS
MEDIDOS	**BANDA PASANTE**	1,1333	0,866	0,3406
	BANDA DE RECHAZO	0,2075	0,2663	0,1942
ENVIADOS	**BANDA PASANTE**	1,25	1,25	1,25
	BANDA DE RECHAZO	0,01	0,01	0,01
X	**MEDIDOS - ESPERADOS**	-0,1166	-0,384	-0,9093
	MEDIDOS - ESPERADOS	0,1975	0,2563	0,1842

Y	X AL CUADRADO	0,0136	0,1474	0,8269
		0,0390	0,0657	0,0339
Y/X		0,0108	0,1179	0,6615
		3,9006	6,5734	3,3961
SUMATORIA (Y/X)		**14,66057981**		

Tabla 12.- Cálculo del Chi-cuadrado en Excel
Fuente.- El autor

DISTRIBUCION DE χ^2

Grados de libertad	Probabilidad										
	0,95	0,90	0,80	0,70	0,50	0,30	0,20	0,10	0,05	0,01	0,001
1	0,004	0,02	0,06	0,15	0,46	1,07	1,64	2,71	3,84	6,64	10,83
2	0,10	0,21	0,45	0,71	1,39	2,41	3,22	4,60	5,99	9,21	13,82
3	0,35	0,58	1,01	1,42	2,37	3,66	4,64	6,25	7,82	11,34	16,27
4	0,71	1,06	1,65	2,20	3,36	4,88	5,99	7,78	9,49	13,28	18,47
5	1,14	1,61	2,34	3,00	4,35	6,06	7,29	9,24	11,07	15,09	20,52
6	1,63	2,20	3,07	3,83	5,35	7,23	8,56	10,64	12,59	16,81	22,46
7	2,17	2,83	3,82	4,67	6,35	8,38	9,80	12,02	14,07	18,48	24,32
8	2,73	3,49	4,59	5.53	7,34	9,52	11,03	13,36	15,51	20,09	26,12
9	3,32	4,17	5,38	6,39	8,34	10,66	12,24	14,68	16,92	21,67	27,88
10	3,94	4,86	6,18	7,27	9,34	11,78	13,44	15,99	18,31	23,21	29,59
	No significativo								Significativo		

Tabla 13.- Distribución de X^2

Fuente.-
https://cristina92sm.files.wordpress.com/2011/05/tabla-chi-cuadrado.jpg

2.7.4.- APROBACIÓN O RECHAZO DE LA HIPÓTESIS

Para aprobar o rechazar la hipótesis, se toma el valor del Chi-cuadrado crítico según la **Tabla 13**; con la probabilidad de aceptación de la hipótesis nula de 0,05 y un grado de libertad 2; es determinado el valor de 5,99. Al comprobar X^2_E con respecto a X^2_C; la Hipótesis Alternativa H1 es aceptada, en vista que X^2_E es mayor;

concluyendo, la hipótesis aceptada es la siguiente: *"El diseño e implementación de un crossover de tres vías utilizando DSP con parámetros controlados por un dispositivo Android permitirá distribuir eficazmente las componentes de una señal de audio debido a las frecuencias variables"*

CAPÍTULO III

3.- RESULTADOS

En este capítulo, están detalladas las pruebas realizadas entre el dispositivo Android y la antena wi-fly; también están descritas las diferentes respuestas a la salida, conforme a la variación de los filtros utilizados y de las frecuencias de corte enviadas de manera inalámbrica.

3.1.- PRUEBA DE CONECTIVIDAD ENTRE WI-FLY Y DISPOSITIVO ANDROID.

Para la prueba de conectividad entre los dos dispositivos, es necesario realizar un ping, para esto se ingresa al módulo wi-fly con el comando telnet; teniendo en cuenta que el puerto que utiliza el módulo para la comunicación es el puerto 2000. La **Figura 66** muestra la respuesta.

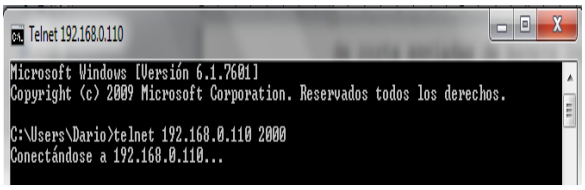


Figura 66.- Conexión al módulo wi-fly vía telnet
Fuente.- El autor

Después de ingresar a la wi-fly; sigue la prueba de comunicación entre el módulo y el dispositivo Android,

para lo cual es utilizado el comando ping. La **Figura 67** muestra el resultado de aplicar este comando.

Figura 67.- Trama enviada desde módulo wi-fly hacia dispositivo Android
Fuente.- El autor

3.2.- PRUEBA DE FUNCIONAMIENTO DE LOS FILTROS

Las pruebas realizadas para notar la eficiencia del filtro, fueron hechas con algunas consideraciones según el tipo de filtro a utilizar; la **Tabla 14** muestra los datos utilizados para las respectivas pruebas. Las gráficas de la respuesta según los valores medidos están presentes en el **Anexo 1.**

ORDEN DEL FILTRO	200
VOLTAJE PICO PICO – SEÑAL DE ENTRADA	1,5 V
FRECUENCIA DE CORTE – PASA BAJOS (PASA BANDA FRECUENCIA INFERIOR)	1,2 KHz
FRECUENCIA DE CORTE – PASA ALTOS (PASA BANDA FRECUENCIA SUPERIOR)	3,8 KHz

Tabla 14.- Valores utilizados en las pruebas de los filtros
Fuente.- El autor

3.2.1.- FILTRO PASA BAJOS

Tomando en cuenta las consideraciones mencionadas anteriormente, los resultados fueron los siguientes:

- Con una frecuencia de entrada de 100 Hz, el filtro permite el paso de la señal; la señal de salida es mostrada en la **Figura 68.**

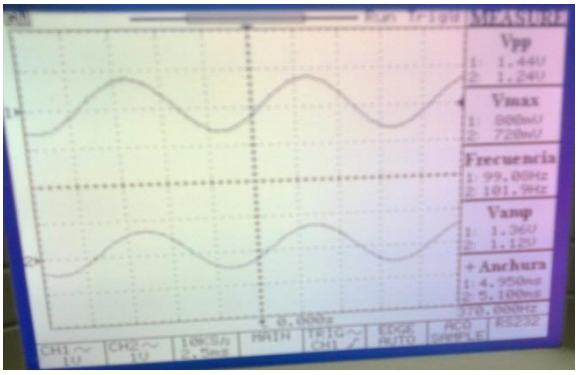


Figura 68.-Señales de entrada y salida del filtro pasa bajos (banda pasante)
Fuente.- El autor

- Con una señal de entrada de 1.2 KHz (valor de la frecuencia de corte), el filtro permite el paso de la Señal.
- La salida presenta una atenuación considerable; Siendo la señal de entrada aproximada de 1,5 V; la salida tiene en valor de 1 V. Las señales medidas son mostradas en la **Figura 69.**

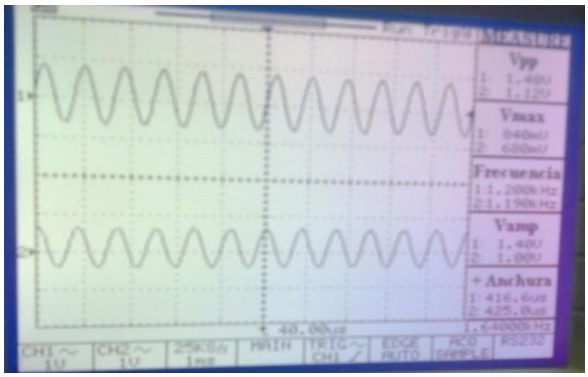


Figura 69.- Señales de entrada y salida del filtro pasa bajos (en la frecuencia de corte)
Fuente.- El autor

- Con una señal de entrada de 1,3 KHz, el filtro aún deja pasar la señal; pero la atenúa, teniendo como amplitud de entrada 1,5 V y amplitud de salida 0,96 V.

La **Figura 70** muestra los resultados del filtro en esta prueba.

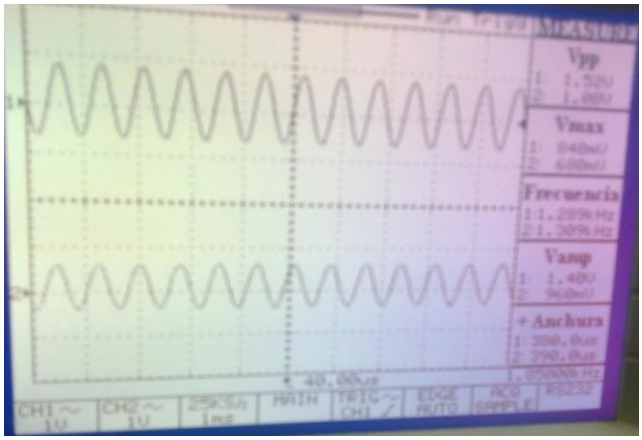


Figura 70.- Señales de entrada y salida del filtro pasa bajos (banda de transición)
Fuente.- El autor

Finalmente, es realizada la prueba del filtro fuera de su banda pasante, la frecuencia utilizada es de 3 KHz; como resultado, la señal de salida es nula. La **Figura 71** muestra las señales de entrada y salida de este caso.

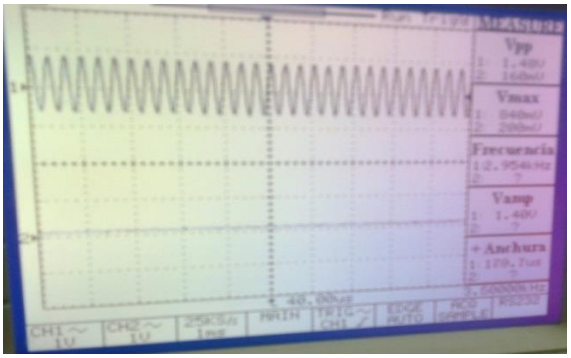


Figura 71.- Señales de entrada y salida del filtro pasa bajos (banda de rechazo)
Fuente.- El autor

3.2.2.- FILTRO PASA ALTOS

Los resultados fueron los siguientes:

- Con una señal de entrada de 100 Hz, el filtro no permite el paso de la señal; las señales utilizadas son mostradas en la **Figura 72.**

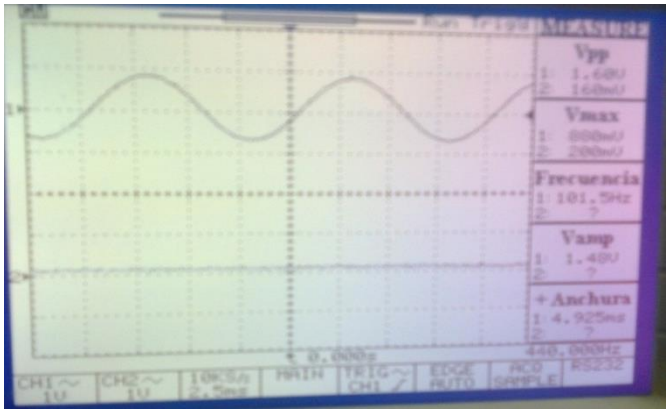


Figura 72.- Señales de entrada y salida del filtro pasa altos (banda de rechazo)
Fuente.- El autor

- Con una señal de entrada de 3,7 KHz, el filtro empieza a dejar pasar la señal.

La **Figura 73** muestra los resultados de esta prueba.

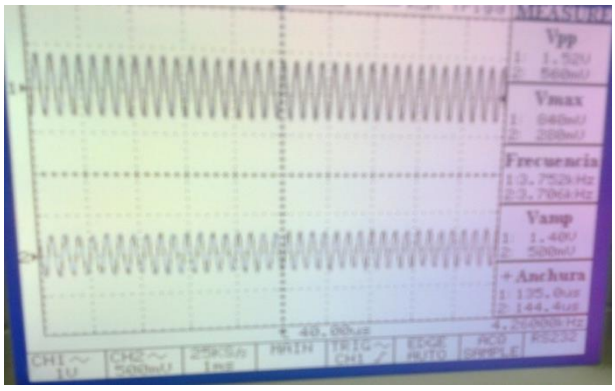


Figura 73.- Señales de entrada y salida del filtro pasa altos (Banda de transición)
Fuente.- El autor

- Utilizando la frecuencia de entrada del mismo valor que la frecuencia de corte (3,8 KHz); el filtro permite el paso de esta señal, las señales resultantes, se muestran en la **Figura 74.**

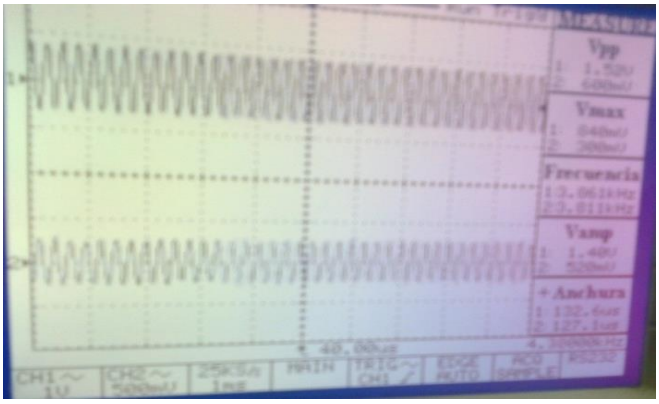


Figura 74.- Señales de entrada y salida del filtro pasa altos (frecuencia de corte)
Fuente.- El autor

- La última prueba realizada con este filtro es dentro de su banda de paso, la señal de entrada es de 5 KHz; los resultados, son mostrados en la Figura **75.**

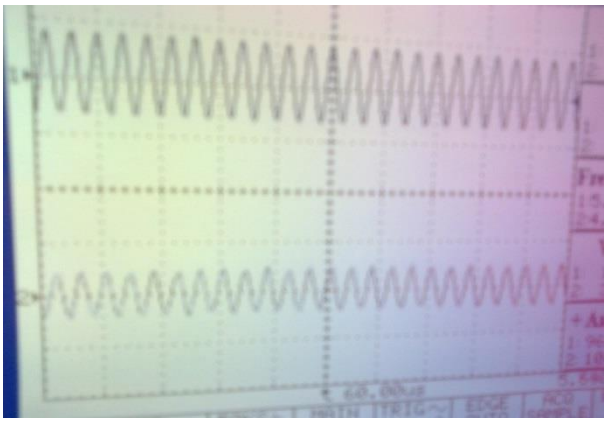


Figura 75.- Señales de entrada y salida del filtro pasa altos (banda de paso)
Fuente.- El autor

3.2.3.- FILTRO PASA BANDA

Tomando en cuenta las consideraciones mencionadas anteriormente, los resultados fueron los siguientes:

- Con una señal de entrada de 100 Hz, el filtro no permite el paso de dicha frecuencia; las señales en cuestión, son mostradas en la **Figura 76.**

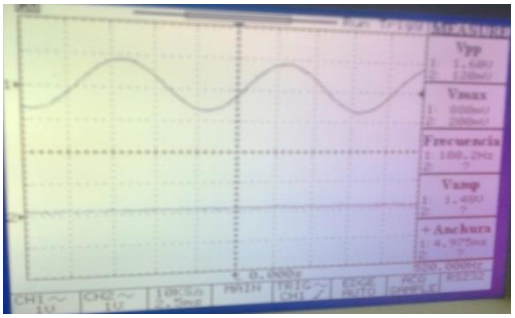


Figura 76.- Señales de entrada y salida del filtro pasa banda (banda de rechazo inferior)
Fuente.- El autor

- Con una señal de entrada igual a la frecuencia de corte inferior (1.2 KHz), el filtro permite el paso de la señal con un poco de atenuación, la **Figura 77** muestra las señales medidas.

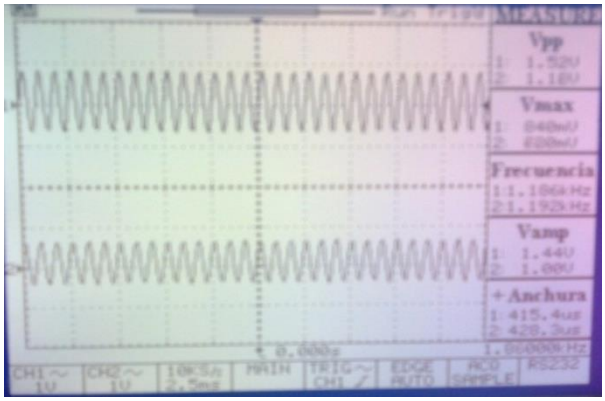


Figura 77.-Señales de entrada y salida del filtro pasa banda (frecuencia de corte inferior)
Fuente.- El autor

- Con una señal de entrada de 2 KHz, dentro de la banda pasante, el filtro permite el paso de la señal. En la **Figura 78** están las señales de entrada y salida de este caso.

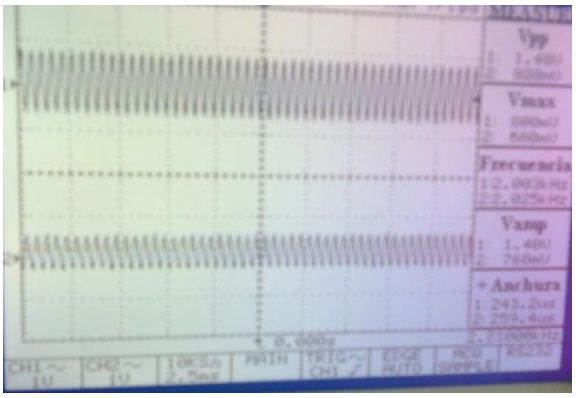


Figura 78.- Señales de entrada y salida del filtro pasa banda (banda de paso)
Fuente.- El autor

- Con una señal de entrada igual a la frecuencia de corte superior (3,8 KHz); el filtro empieza a atenuar la señal significativamente, las señales las muestra la **Figura 79.**

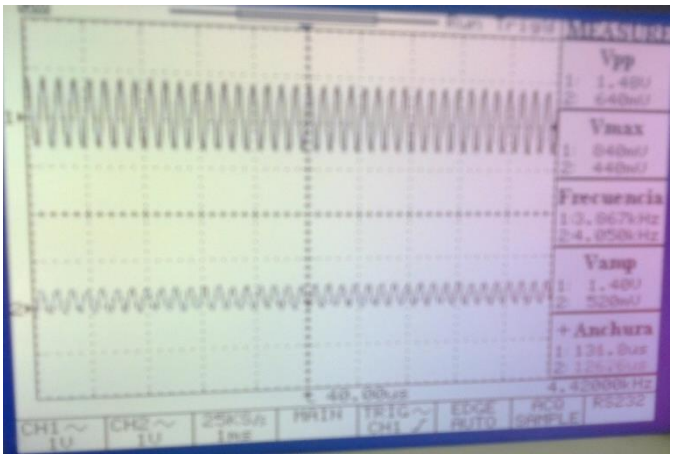


Figura 79.- Señales de entrada y salida del filtro pasa banda (frecuencia de corte superior)
Fuente.- El autor

- La última prueba realizada, es con una frecuencia de 5 KHz, en este caso el filtro atenúa completamente la señal, ya que está en su banda de rechazo. La Figura **80** muestra las señales de este caso.

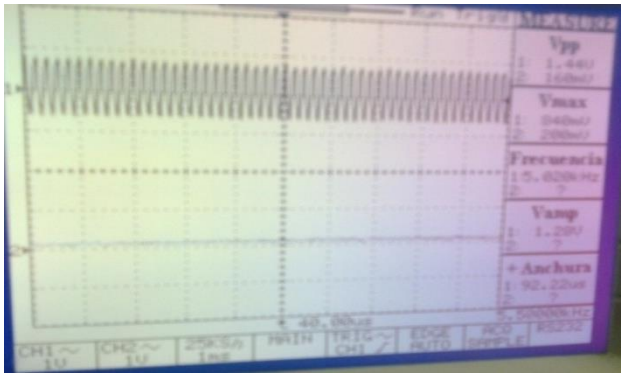


Figura 80.- Señales de entrada y salida del filtro pasa banda (banda de rechazo superior)
Fuente.- El autor

Las pruebas realizadas en esta etapa, y las muestras tomadas para la comprobación de la hipótesis, muestran que los filtros diseñados cumplen su propósito. La **Figura 81** muestra el sistema concluido y funcionando.

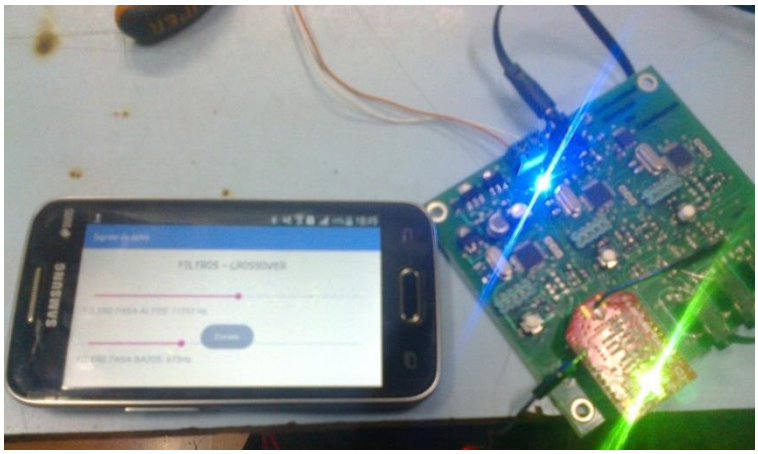


Figura 81.- Sistema en operación
Fuente.- El autor

3.3. – ANÁLISIS FINANCIERO

La **Tabla 15**, muestra el costo total del proyecto diseñado e implementado, donde el costo total, justifica evidentemente los beneficios que ofrece este proyecto.

CANTIDAD	DETALLE	COSTO UNITARIO	COSTO TOTAL
3	Amplificador de Audio de 18 watts	$ 20,00	$ 60,00
3	Cable auxiliar Estéreo	$ 5,00	$ 15,00
1	Cable DB9	$ 10,00	$ 10,00
1	Caja de tachuelas	$ 3,00	$ 3,00
1	Caja de tornillos MDF	$ 5,00	$ 5,00
1	Cemento de contacto	$ 3,00	$ 3,00
4	Conector Banano - hembra	$ 0,50	$ 2,00
4	Conector Banano - macho	$ 0,50	$ 2,00
6	Conector RCA hembra	$ 0,50	$ 3,00
1	Elementos varios para la placa	$ 20,00	$ 20,00
2	Manija de metal	$ 2,50	$ 5,00
10	Metro de cable de audio	$ 0,80	$ 8,00
3	Metro de cable gemelo	$ 0,50	$ 1,50
10	Metro de cable UTP	$ 0,50	$ 5,00
5	Metro de fieltro	$ 3,00	$ 15,00
3	Microcontrolador AT32UC3L	$ 30,00	$ 90,00
1	Módulo Wi fly RN-XV	$ 90,00	$ 90,00
1	Placa electrónica	$ 100,00	$ 100,00
8	Soportes de goma	$ 0,50	$ 4,00
2	Squawker American Xtreme	$ 15,00	$ 30,00
12	Tabla MDF	$ 2,00	$ 24,00
1	Transformador de 10 voltios CA	$ 5,00	$ 5,00
2	Tweeter Acoustic	$ 2,00	$ 4,00
4	Bisagra	$ 0,50	$ 2,00
2	Woofer American Xtreme	$ 15,00	$ 30,00
TOTAL			**$ 536,50**

Tabla 15.- Presupuesto total del proyecto
Fuente.- El autor

CAPÍTULO IV

4.- DISCUSIÓN

El desarrollo de este crossover, está basado en el Procesamiento de Señales Digitales (DSP) el cual permite manipular una señal analógica de manera amplia, para este propósito es necesario contar con un dispositivo que permita cumplir con dicho procesamiento, en este caso el microcontrolador AT32UC3L17 de 32 bits, de la familia de Atmel.

Los microcontroladores deben utilizar la Convolución de señales, lo que hace que sean consumidos muchos recursos; es por lo que deben contar con mucha velocidad.

La frecuencia de muestreo, es una frecuencia estándar utilizada en el muestreo de señales de audio, esta frecuencia es de 48 MHz.

En cuestión del orden de los filtros, el utilizado es un filtro de orden 200; podría utilizarse un mayor orden, pero esto afectaría de manera negativa a la etapa de la convolución, haciendo que los datos en la señal de salida resulten entrecortados.

En cuanto a la antena receptora escogida, la Wi fly RN-XV; han sido revisados algunos trabajos de titulación y su denominador común en cuestión de comunicaciones vía wi-fi ha sido la antena propuesta, en vista que su costo no es elevado y su funcionamiento es adecuado.

El desarrollo de la aplicación está realizado en el entorno Android Studio, ya que estadísticamente la

mayoría de personas cuentan con un teléfono o dispositivo móvil con el sistema operativo Android, además este programa permite diseñar una propia aplicación según los requerimientos que sean necesarios.

Los amplificadores de audio utilizados, son diseños establecidos para este propósito, los cuales pueden ser adquiridos en la mayoría de tiendas electrónicas. Su potencia estándar es de 18 watts por canal.

Con respecto a los altavoces utilizados, hay que considerar utilizar altavoces que soporten al menos un 35 % más que la potencia suministrada por los amplificadores para evitar saturaciones del sonido.

El diseño de las cajas fue realizado tomando en cuenta diseños similares encontrados en la web en vista que el diseño de la caja de altavoces afecta significativamente al sonido resultante; fueron cubiertas con fieltro porque este es un material que mejora la acústica, además que cubre la madera de manera estética.

De esta manera, y culminado todo el diseño, el crossover resultante es capaz de cumplir con los requerimientos planteados, de manera eficiente, sin pérdida de datos, y con un procesamiento de la información eficiente.

CAPÍTULO V

5.- CONCLUSIONES Y RECOMENDACIONES

5.1.- CONCLUSIONES

- La comunicación entre la aplicación y el crossover es más estable en una red inalámbrica local comparada con la comunicación mediante una red AD-HOC.
- La frecuencia de muestreo debe ser como mínimo el doble de la máxima frecuencia de entrada, mientras la frecuencia de muestreo es mucho mayor, la señal se recompone de manera más precisa; concluyendo que con una frecuencia de muestreo de 48 Khz se obtiene una señal de salida adecuada.
- La relación entre la banda de transición de un filtro es inversamente proporcional al número de coeficientes utilizados, por lo cual si se quiere conseguir un filtro próximo a un filtro ideal, es necesario aumentar el número de coeficientes, pero esto hace que el microcontrolador se demore más en el proceso de la generación de éstos por lo cual es necesario encontrar un término medio entre calidad del filtro y velocidad de procesamiento, por lo cual se tomó en cuenta un filtro de orden 200.
- El crossover utiliza filtros FIR, mediante la ventana de Hamming debido a que su banda de transición no es extensa y la amplitud del rizado en la banda de rechazo es mínima; de acuerdo al orden de los filtros

utilizados, la banda de transición está comprendida en 800 Hz aproximadamente.

- En la aplicación, la variación de frecuencias del filtro pasa bajos está comprendida entre los 100 y los 1200 Hz; mientras que en el filtro pasa altos el rango se encuentra entre los 2 y los 20 KHz; siendo las frecuencias de corte del filtro pasa banda las mismas frecuencias del filtro pasa bajos y pasa altos; se utiliza este rango ya que los altavoces son más estables reproduciendo ese rango de frecuencias respectivamente.
- La señal de salida es reconstruida mediante un PWM de alta frecuencia (150 Khz) y un filtro pasa bajos para eliminar las frecuencias altas, el crossover cuenta con una etapa de pre-amplificación la cual permite que la señal de salida pueda escucharse por audífonos, además las señales de salida pueden conectarse a amplificadores estéreo de 18 watts para que puedan ser escuchadas mediante altavoces woofer, squawker y tweeter.

5.2.- RECOMENDACIONES

- La comunicación entre una interfaz Android y el módulo wi-fly puede realizarse en una red privada clase C, hay que tomar en cuenta que el puerto de comunicación del módulo es el 2000. En la aplicación Android, es necesario definir los permisos de acceso completo a la red y a internet, ya que estos permisos permiten a la aplicación enviar los datos a utilizarse, si estos

permisos no son declarados, no puede realizarse el envío respectivo.

- Si se considera usar una frecuencia de muestreo mucho mayor a 48 Khz para recomponer las frecuencias altas de mejor manera, es aconsejable tomar en cuenta un microcontrolador con mayor velocidad de procesamiento que el utilizado en este proyecto.
- Hay que considerar que, al aumentar el orden de los filtros, la banda de transición se reduce, pero esto se da a costa de la velocidad del procesamiento, por lo cual es adecuado buscar un equilibrio entre el orden de los filtros y la banda de transición.
- Cuando sea necesaria una banda de transición más estrecha, se pueden tomar en cuenta las ventanas planteadas en el marco teórico; pero cabe señalar que dichas ventanas tienen mayor rizado con respecto a la ventana de Hamming.
- Para contar con una mejor calidad de los bajos, una opción a considerar es usar una cuarta vía con la utilización de un altavoz subwoofer, el cual reproduce frecuencias menores a los 120 Hz.
- Si es requerido un sistema de altavoces con mayor potencia (utilizando otro amplificador) hay que considerar que los altavoces soporten 35 % más de la potencia suministrada por el amplificador, esto evita que los altavoces sufran deterioros debido al gran consumo de corriente en su bobina.

CAPÍTULO VI

6.- PROPUESTA

6.1.- TÍTULO DE LA PROPUESTA

"DISEÑO E IMPLEMENTACION DE UN CROSSOVER DE TRES VÍAS UTILIZANDO DSP CON PARÁMETROS CONTROLADOS POR UN DISPOSITIVO ANDROID"

6.2.- INTRODUCCIÓN

Para la adecuada comunicación entre el dispositivo Android y el módulo wi-fly, es necesario analizar la programación de los sockets en Android Studio y es necesario también programar el módulo para que de esta manera esté conectado en la misma red que el dispositivo Android y pueda recibir la información enviada.

Para el funcionamiento adecuado del crossover, es necesario tomar en cuenta el muestreo de la señal de audio entrante, el orden del filtro, la convolución de esta señal y la velocidad de procesamiento del microcontrolador establecido.

Con el estudio anticipado sobre las tecnologías de comunicación inalámbrica y la velocidad de procesamiento de los dispositivos, podrán determinarse cuáles son los elementos adecuados para utilizar en el proyecto de manera positiva y corregir así los errores que puedan presentarse en las etapas posteriores al desarrollo de este tema.

6.3.- OBJETIVOS

6.3.1.- OBJETIVO GENERAL

- Diseñar e implementar un crossover de tres vías, utilizando DSP, con parámetros controlados por un dispositivo Android para distribuir las componentes de una señal de audio de acuerdo a las frecuencias variables.

6.3.2.- OBJETIVOS ESPECÍFICOS

- Establecer un envío de datos eficiente desde la aplicación Android hacia el crossover.
- Realizar el muestreo y la digitalización de una señal de audio con una frecuencia de muestreo adecuada para el sistema.
- Determinar un orden del filtro adecuado, tomando en cuenta las características del procesador.
- Diseñar filtros FIR mediante el método de ventaneo, utilizando una ventana eficiente.
- Establecer rangos de frecuencias de corte adecuados según las características de los altavoces a utilizarse.
- Convertir la señal digitalizada y procesada en una señal analógica audible.

6.4.- FUNDAMENTACIÓN CIENTÍFICO-TEÓRICA

Para la implementación del crossover de tres vías, están considerados diversos componentes que cumplen un rol importantes dentro de este desarrollo, estos componentes están detallados a continuación:

- **Software Android Studio**

Android, es un sistema operativo que fue pensado inicialmente para teléfonos móviles. La diferencia de este sistema operativo es que es fundamentado en Linux, un núcleo de sistema operativo libre, multiplataforma y gratuito.

Este sistema permite programar aplicaciones en una variación del lenguaje de programación Java, llamada Dalvik. El sistema operativo, proporciona todas las interfaces y requerimientos necesarios para desarrollar aplicaciones que accedan a las funciones del móvil de una forma sencilla en un lenguaje de programación muy utilizado como lo es Java.

- **Módulo wi-fly RN-XV**

El módulo inalámbrico RN-XV de Roving Networks, es una solución wi-fi certificada, está basado en el módulo wi-fi RN 171 que incorpora un radio con protocolo 802.11 b/g con procesador de 32 bits, stack TCP/IP, reloj de tiempo real, acelerador criptográfico, unidad de manejo de energía, y una unidad de manejo de sensores analógicos.

- **Procesamiento Digital de Señales (DSP)**

Los DSP's son microcontroladores que permiten el procesamiento digital de señales, es decir la manipulación matemática de señales representadas digitalmente, el procesamiento digital de señales, es una tecnología cuyas aplicaciones están creciendo rápidamente, como en el caso de las comunicaciones inalámbricas, procesamiento de audio, video y control industrial.

6.5.- DESCRIPCIÓN DE LA PROPUESTA

La propuesta está fundamentada en el Procesamiento Digital de Señales; para filtrar una señal de audio mediante la convolución realizada dentro de tres microcontroladores, además de contar con la capacidad de variar las frecuencias de corte de los filtros establecidos mediante una interfaz Android controlada por el usuario. Después del procesamiento matemático, la señal filtrada es reconstruida mediante la generación de un PWM de alta frecuencia; para después pasar por un circuito R-C y así obtener la señal de salida analógica. Como punto final, la señal de salida analógica pasa por un proceso de amplificación de la señal para después ser reproducida en tres tipos de altavoces.

6.6.- DISEÑO ORGANIZACIONAL

A continuación está detallada la estructura organizacional de la unidad administrativa del proyecto. El diagrama organizacional del proyecto, lo muestra la **Figura 82.**

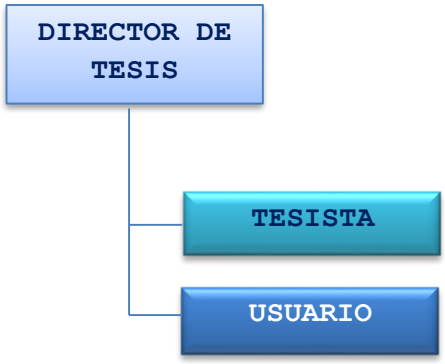


Figura 82.- Diagrama del sistema organizacional
Fuente.- El autor

6.7.- MONITOREO Y EVALUACIÓN DE LA PROPUESTA

En lo que comprende al monitoreo y evaluación de la propuesta, fueron realizadas algunas pruebas de comunicación entre la interfaz Android y el módulo wi-fly; también fueron tomadas pruebas en los diferentes filtros del crossover con diferentes frecuencias de prueba, verificando la eficiencia de los filtros según su tipo y su orden; las pruebas realizadas, y las etapas del proyecto, están documentadas, con el fin de en un futuro perfeccionar los dispositivos y establecer las limitaciones que podrían presentarse.

CAPÍTULO VII

7.- BIBLIOGRAFÍA

Academy, C. N. (2010). Aspectos básicos de Networking.

Academy, C. N. (2014). Conceptos y protocolos de enrutamiento.

ALVARADO MOYA, J. P. (2006). *Procesamiento Digital de Señales.*

ATMEL. (s.f.). *Atmel*. Obtenido de http://www.atmel.com/tools/atmelstudio.aspx

ATMEL. (s.f.). *www.atmel.com*. Obtenido de www.atmel.com

Clavijo Mendoza, J. R. (2011). *Diseño y Simulación de sistemas microcontrolados en lenguaje C.* Colombia: ISBN.

CONSTANTE CASTRO, L. I. (2003). *Filtros crossover utilizando el módulo EZ-kit ADSP-2181.* Quito.

CUENCA, David, GOMEZ, Eduardo. (2001). *Tecnología Básica del Sonido II.* España: Paraninfo.

Developers, A. (2014). *http://developer.android.com/intl/es/tools/studio/index.html*. Recuperado el 2014

DUIOPS. (2007). *DUIOPS*. Obtenido de http://www.duiops.net/hifi/enciclopedia/squawker.htm

E. Soria Olivas, M. M. (2003). *Tratamiento Digital de Señales. Problemas y ejercicios resueltos.* Madrid: Prentice Hall.

Gallager, M. (2006). *Acoustic Design for the Home Studio.* Thomson Course Technology.

https://www.android.com/play/. (s.f.).

Java. (10 de Noviembre de 2014). *https://www.java.com.es/download/faq/develop.xml*.

López, J. G. (2000). *Procesamiento Digital de Señales.*

Mishra, A. R. (2004). *Fundamental of Networking.* England.

Networks, R. (8 de Agosto de 2011). *www.rovingnetworks.com*.

Reyes, C. (2006). *Microcontroladores PIC Programación en Basic.* Quito: RISPERGRAF.

ST. (s.f.). *TDA2030*. Obtenido de http://www.itisff.it/dip_eln/tda2030.pdf

TP-LINK. (2014). *TP-LINK , the reliable choice*. Obtenido de http://www.tp-link.ec/products/details/?model=TL-WR740N

CAPÍTULO VIII

ANEXOS

ANEXO 1.- GRÁFICAS DE LOS VALORES TOMADOS

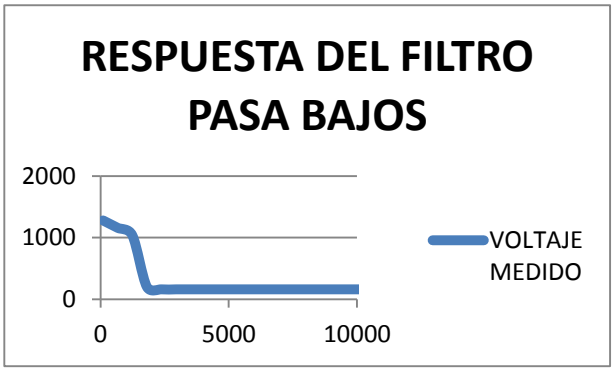


Figura 83.- Respuesta en el dominio de la Frecuencia; Filtro Pasa Bajos
Fuente.- El autor

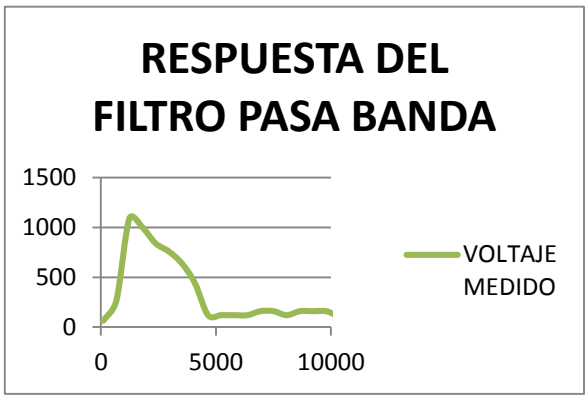


Figura 84.- Respuesta en el dominio de la Frecuencia; Filtro Pasa Banda
Fuente.- El autor

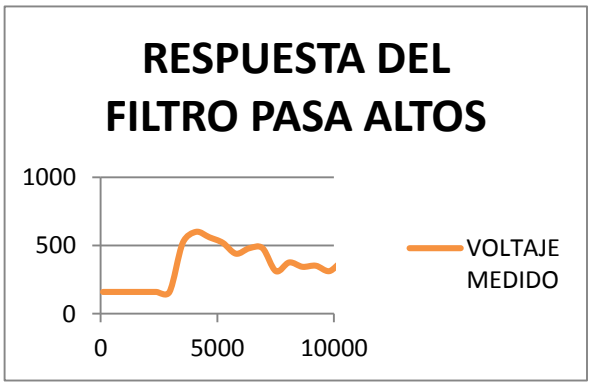


Figura 85.-Respuesta en el dominio de la Frecuencia; Filtro Pasa Banda
Fuente.- El autor

ANEXO 2.- CÓDIGO DE LOS FILTROS

```
#include <asf.h>
#include "coeficientes.h"
#include <math.h>
#define EXAMPLE_TARGET_CLK_ADC_FREQ_HZ  1500000
#define BUFFER_LENGTH             1024
#define FILTER_LENGTH             200
#define FILTRO_PASATODO           0
#define FILTRO_PASABAJOS          1
#define FILTRO_PASAALTOS          2
#define FILTRO_PASABANDA          3

volatile      int8_t seleccion_buffer_in=0;

volatile U32 puntero_in=0;
volatile      int8_t status_end_buff_in=0;
static A_ALIGNED dsp16_t buff_datos_in1[BUFFER_LENGTH+FILTER_LENGTH];
static A_ALIGNED dsp16_t buff_datos_in2[BUFFER_LENGTH+FILTER_LENGTH];

//////////////////////////////////////////////////
volatile      int8_t seleccion_buffer_out=0;
volatile U32 puntero_out=0;
static A_ALIGNED dsp16_t buff_datos_out1[BUFFER_LENGTH+0];
static A_ALIGNED dsp16_t buff_datos_out2[BUFFER_LENGTH+0];
//////////////////////////////////////////////////////////
static A_ALIGNED dsp16_t buff_datos[BUFFER_LENGTH+FILTER_LENGTH];

//////////////////////////////////////////////////////////
static A_ALIGNED dsp16_t hamming_coef[FILTER_LENGTH];
static A_ALIGNED dsp16_t coeficientes_filtro[FILTER_LENGTH];
static float coeficientes[FILTER_LENGTH];
//////////////////////////////////////////////////

volatile      int32_t       arribo_serial=0;
volatile      int32_t dato_serial=0;
volatile      int32_t puntero_serial=0;
volatile      int32_t buffer_filtro=0;
volatile      int32_t       tipo_filtro=0;

volatile      int32_t frecuencia_corte_filtro=0;
volatile      int32_t frecuencia_corte_filtroa=0;
volatile      int32_t frecuencia_corte_filtrob=0;
////////////////////////////////////////////////
volatile static bool update_timer = true;

volatile avr32_pwma_t *pwma = &AVR32_PWMA;

int32_t aux;
volatile      uint32_t adc_data;
uint32_t config_status,set_value_status,divi, cpu_speed;
volatile avr32_adcifb_t *adcifb = ADC;

static void tc_init(volatile avr32_tc_t *tc);
static status_code_t adc_init2();
static void tc_irq(void);
```

```
ISR(ADCIFB_interrupt_handler, AVR32_ADCIFB_IRQ_GROUP, ADC_INTERRUPT_PRIORITY);
static void usart_int_handler(void);
volatile float Wc,fc,fp,fs, delta_f, frec_corte, wa, wb, fb, fa;
void coeficientes_filtro_hamming(uint8_t filter);
static void usart_init(void);

int main (void){
	sysclk_init();
	board_init();
	tc_init(EXAMPLE_TC);
	adc_init2();
	usart_init();
	cpu_speed = sysclk_get_cpu_hz();
	delay_init(sysclk_get_cpu_hz());
	gpio_configure_pin(led, GPIO_DIR_OUTPUT | GPIO_INIT_LOW);
	gpio_configure_pin(led2, GPIO_DIR_OUTPUT | GPIO_INIT_LOW);
	gpio_configure_pin(led3, GPIO_DIR_OUTPUT | GPIO_INIT_LOW);
	divi = div_ceil((sysclk_get_pba_hz()) , 120000000);
	genclk_enable_config(AVR32_SCIF_GCLK_PWMA,GENCLK_SRC_RC120M,divi);
	gpio_enable_module_pin(led1,pwm_led);
	gpio_configure_pin(led, GPIO_DIR_OUTPUT | GPIO_INIT_LOW);
	gpio_configure_pin(button,GPIO_DIR_INPUT|GPIO_PULL_UP);
	config_status = pwma_config_enable(pwma, 480000,	120000000,0);
	set_value_status = pwma_set_channels_value(pwma,(1<<25),50);
	INTC_init_interrupts();
	INTC_register_interrupt(&tc_irq, EXAMPLE_TC_IRQ,
EXAMPLE_TC_IRQ_PRIORITY);
INTC_register_interrupt(&ADCIFB_interrupt_handler, AVR32_ADCIFB_IRQ,
ADC_INTERRUPT_PRIORITY);
	INTC_register_interrupt(&usart_int_handler, AVR32_USART2_IRQ,
AVR32_INTC_INT3);
	AVR32_USART2.ier=AVR32_USART_IER_RXRDY_MASK;
	frecuencia_corte_filtrob=600;
	frecuencia_corte_filtroa=5000;
	tipo_filtro=2;
	coeficientes_filtro_hamming(tipo_filtro);
	Enable_global_interrupt();
		while (1){
		if (!gpio_get_pin_value(button))
		{ gpio_set_pin_high(led);
		delay_s(1);
		tipo_filtro++;
		if (tipo_filtro==4)
		{tipo_filtro=0;}
		coeficientes_filtro_hamming(tipo_filtro);}
		else
		gpio_set_pin_low(led);
		if (status_end_buff_in==1){
		status_end_buff_in=0;

	for (int i = 0; i < FILTER_LENGTH; i++) {
	buff_datos_in1[i] = buff_datos_in2[BUFFER_LENGTH + i];}
dsp16_filt_fir(buff_datos,buff_datos_in1,BUFFER_LENGTH + FILTER_LENGTH -
1,coeficientes_filtro,FILTER_LENGTH);
for (int16_t i=0;i<(BUFFER_LENGTH);i++){
```

```
        if (tipo_filtro==0) buff_datos_out1[i]=buff_datos_in1[i+FILTER_LENGTH];
        else if (tipo_filtro==1)   buff_datos_out1[i]=buff_datos[i];
        else if (tipo_filtro==2)buff_datos_out1[i]=buff_datos[i]+512;
        else if (tipo_filtro==3)buff_datos_out1[i]=buff_datos[i]+512;} }
        else if (status_end_buff_in==2){
        status_end_buff_in=0;
        for (int i = 0; i < FILTER_LENGTH; i++) {
        buff_datos_in2[i] =  buff_datos_in1[BUFFER_LENGTH + i];}

dsp16_filt_fir(buff_datos,buff_datos_in2,BUFFER_LENGTH + FILTER_LENGTH -
1,coeficientes_filtro,FILTER_LENGTH);
for (int16_t i=0;i<(BUFFER_LENGTH);i++){
if (tipo_filtro==0) buff_datos_out2[i]=buff_datos_in2[i+FILTER_LENGTH];
else if (tipo_filtro==1)buff_datos_out2[i]=buff_datos[i];
else if (tipo_filtro==2)buff_datos_out2[i]=buff_datos[i]+512;
else if (tipo_filtro==3)buff_datos_out2[i]=buff_datos[i]+512;}
asm("nop");}
int received_byte;
if (arribo_serial==1){
arribo_serial=0;
if ((puntero_serial==0)&&(dato_serial==255))
puntero_serial=1;
else if ((puntero_serial==1))
if (dato_serial==255)
puntero_serial=2;
else
puntero_serial=0;
else if (puntero_serial==2){
frecuencia_corte_filtrob=((uint32_t)dato_serial)<<8;
puntero_serial=3;}
else if (puntero_serial==3){
frecuencia_corte_filtrob|=((uint32_t)dato_serial);
puntero_serial=4;}
else if (puntero_serial==4){
frecuencia_corte_filtroa=((uint32_t)dato_serial)<<8;
puntero_serial=5;}
else if (puntero_serial==5){
frecuencia_corte_filtroa|=((uint32_t)dato_serial);
if (tipo_filtro==1){
if ((frecuencia_corte_filtrob<=24000)){
gpio_toggle_pin(led);
coeficientes_filtro_hamming((uint8_t)tipo_filtro);
usart_serial_putchar(EXAMPLE_USART,(uint8_t)(frecuencia_corte_filtrob>>8));
usart_serial_putchar(EXAMPLE_USART,(uint8_t)frecuencia_corte_filtrob);
usart_serial_putchar(EXAMPLE_USART,(uint8_t)(frecuencia_corte_filtroa>>8))
usart_serial_putchar(EXAMPLE_USART,(uint8_t)frecuencia_corte_filtroa);}}
if (tipo_filtro==2){
if (frecuencia_corte_filtroa>=1000){
gpio_toggle_pin(led);
coeficientes_filtro_hamming((uint8_t)tipo_filtro);
usart_serial_putchar(EXAMPLE_USART,(uint8_t)(frecuencia_corte_filtrob>>8));
usart_serial_putchar(EXAMPLE_USART,(uint8_t)frecuencia_corte_filtrob);
usart_serial_putchar(EXAMPLE_USART,(uint8_t)(frecuencia_corte_filtroa>>8));
usart_serial_putchar(EXAMPLE_USART,(uint8_t)frecuencia_corte_filtroa);}}
if (tipo_filtro==3) {
```

```
if ((frecuencia_corte_filtrob<=24000)&&(frecuencia_corte_filtroa>=1000)) {
if ((frecuencia_corte_filtrob<=600)) {
frecuencia_corte_filtrob=600;}
gpio_toggle_pin(led);
coeficientes_filtro_hamming((uint8_t)tipo_filtro);
usart_serial_putchar(EXAMPLE_USART,(uint8_t)(frecuencia_corte_filtrob>>8));
usart_serial_putchar(EXAMPLE_USART,(uint8_t)frecuencia_corte_filtrob);
usart_serial_putchar(EXAMPLE_USART,(uint8_t)(frecuencia_corte_filtroa>>8));
usart_serial_putchar(EXAMPLE_USART,(uint8_t)frecuencia_corte_filtroa);}}
puntero_serial=0;}}}}
void coeficientes_filtro_hamming(uint8_t filter) {
dsp16_gen_step(hamming_coef, FILTER_LENGTH, DSP16_Q(1.), DSP16_Q(1.), 0);
dsp16_win_hamm(hamming_coef, hamming_coef, FILTER_LENGTH);
if (filter==0){
delta_f = 3.3/FILTER_LENGTH;
frec_corte=(float)frecuencia_corte_filtrob;
fs= 48000;
frec_corte=24000;
fp=frec_corte/fs;
fc = fp+delta_f/2;
Wc = 2*CST_PI*fc;}
if (filter==1){
delta_f = 3.3/FILTER_LENGTH;
frec_corte=(float)frecuencia_corte_filtrob;
fs= 48000;
fp=frec_corte/fs;
fc = fp+delta_f/2;
Wc = 2*CST_PI*fc;
for (int n=-FILTER_LENGTH/2; n<=FILTER_LENGTH/2;n++){
if (n!=0)coeficientes [n+FILTER_LENGTH/2] =  sin(n*Wc)/(n*CST_PI);
else coeficientes [n+FILTER_LENGTH/2] =  2*fc;
coeficientes_filtro[n+FILTER_LENGTH/2]= DSP16_Q(coeficientes
[n+FILTER_LENGTH/2]);}}
if (filter==2){
delta_f = 3.3/FILTER_LENGTH;
frec_corte=(float)frecuencia_corte_filtroa;
fs= 48000;
fp=frec_corte/fs;
fc = fp-delta_f/2;
Wc = 2*CST_PI*fc;
for (int n=-FILTER_LENGTH/2; n<=FILTER_LENGTH/2;n++){
if (n!=0)coeficientes [n+FILTER_LENGTH/2] =  -sin(n*Wc)/(n*CST_PI);
else coeficientes [n+FILTER_LENGTH/2] =  1-2*fc;
coeficientes_filtro[n+FILTER_LENGTH/2]= DSP16_Q(coeficientes
[n+FILTER_LENGTH/2]);}}
if (filter==3){
delta_f = 3.3/FILTER_LENGTH;
frec_corte=(float)frecuencia_corte_filtrob;
fs= 48000;
fp=frec_corte/fs;
fb = fp-delta_f/2;
wb = 2*CST_PI*fb;
frec_corte=(float)frecuencia_corte_filtroa;
fs= 48000;
             fp=frec_corte/fs;
```

```
            fa = fp+delta_f/2;
            wa = 2*CST_PI*fa;
            for (int n=-FILTER_LENGTH/2; n<=FILTER_LENGTH/2;n++){
if (n!=0)coeficientes [n+FILTER_LENGTH/2] =  sin(n*wa)/(n*CST_PI)-
sin(n*wb)/(n*CST_PI);
else coeficientes [n+FILTER_LENGTH/2] =  2*(fa-fb);
coeficientes_filtro[n+FILTER_LENGTH/2]= DSP16_Q(coeficientes
[n+FILTER_LENGTH/2]);}}
dsp16_vect_dotmul(coeficientes_filtro,coeficientes_filtro,hamming_coef,FILTER_L
ENGTH);
for (int n=0; n<FILTER_LENGTH;n++){
coeficientes[n]=(double)(coeficientes_filtro[n])/pow(2,15);}
asm("nop");}
static void tc_init(volatile avr32_tc_t *tc){
static const tc_waveform_opt_t waveform_opt = {
.channel  = EXAMPLE_TC_CHANNEL,
.bswtrg   = TC_EVT_EFFECT_NOOP,
.beevt    = TC_EVT_EFFECT_NOOP,
.bcpc     = TC_EVT_EFFECT_NOOP,
.bcpb     = TC_EVT_EFFECT_NOOP,
.aswtrg   = TC_EVT_EFFECT_NOOP,
.aeevt    = TC_EVT_EFFECT_NOOP,
.acpc     = TC_EVT_EFFECT_NOOP,
.acpa     = TC_EVT_EFFECT_NOOP,
.wavsel   = TC_WAVEFORM_SEL_UP_MODE_RC_TRIGGER,
.enetrg   = false,
.eevt     = 0,
.eevtedg  = TC_SEL_NO_EDGE,
.cpcdis   = false,
.cpcstop  = false,
.burst    = false,
.clki     = false,
.tcclks   = TC_CLOCK_SOURCE_TC2};
static const tc_interrupt_t tc_interrupt = {
            .etrgs = 0,
            .ldrbs = 0,
            .ldras = 0,
            .cpcs  = 1,
            .cpbs  = 0,
            .cpas  = 0,
            .lovrs = 0,
            .covfs = 0};
tc_init_waveform(tc, &waveform_opt);
aux = (sysclk_get_pba_hz() / 2 / 48000);
tc_write_rc(tc, EXAMPLE_TC_CHANNEL, aux);
tc_configure_interrupts(tc, EXAMPLE_TC_CHANNEL, &tc_interrupt);
tc_start(tc, EXAMPLE_TC_CHANNEL);}
static status_code_t adc_init2(){
static const gpio_map_t ADCIFB_GPIO_MAP = {
{EXAMPLE_ADCIFB_PIN, EXAMPLE_ADCIFB_FUNCTION}};
adcifb_opt_t adcifb_opt = {
.resolution = AVR32_ADCIFB_ACR_RES_10BIT,
.shtim  = ADC_SAMPLE_HOLD_TIME,
.ratio_clkadcifb_clkadc = (sysclk_get_pba_hz()) / ADC_FREQUENCY,
            .startup = ADC_STARTUP_TIME,
```

```
.sleep_mode_enable = false};
gpio_disable_pin_pull_up(EXAMPLE_ADCIFB_PIN);
gpio_enable_module(ADCIFB_GPIO_MAP,
sizeof(ADCIFB_GPIO_MAP) / sizeof(ADCIFB_GPIO_MAP[0]));
sysclk_enable_pba_module(SYSCLK_ADCIFB);
if (adcifb_configure(adcifb, &adcifb_opt)) {
return ERR_TIMEOUT;}
if (adcifb_configure_trigger(adcifb, AVR32_ADCIFB_TRGMOD_CM, 0)) {
return ERR_BUSY;}
adcifb_channels_enable(adcifb, EXAMPLE_ADCIFB_CHANNEL_MASK);
cpu_irq_disable();
irq_initialize_vectors();
adcifb_enable_analog_compare_mode(adcifb);
adcifb_enable_compare_gt_interrupt(adcifb);
return STATUS_OK;}
static void usart_init(void) {
const static usart_options_t usart_opt = {
.baudrate     = EXAMPLE_USART_BAUD,
.channelmode = USART_NORMAL_CHMODE,
.charlength   = 8,
.paritytype   = USART_NO_PARITY,
.stopbits     = USART_1_STOPBIT,};
gpio_map_t COMPORT_GPIO_MAP ={
{ USART0_RXD_PIN, USART0_RXD_FUNCTION },
{ USART0_TXD_PIN, USART0_TXD_FUNCTION }};
gpio_enable_module(COMPORT_GPIO_MAP,
sizeof(COMPORT_GPIO_MAP) / sizeof (COMPORT_GPIO_MAP[0]));
usart_init_rs232(EXAMPLE_USART, &usart_opt,
sysclk_get_peripheral_bus_hz(EXAMPLE_USART));}
static void tc_irq(void){
 int32_t dato_Salida;
 tc_read_sr(EXAMPLE_TC, EXAMPLE_TC_CHANNEL);
 update_timer = true;
 gpio_toggle_pin(led2);
if (seleccion_buffer_in==0){
buff_datos_in1[puntero_in++]=adc_data;
if (puntero_in==(BUFFER_LENGTH+FILTER_LENGTH)){
puntero_in=FILTER_LENGTH;
seleccion_buffer_in=1;
status_end_buff_in=1;}}
else{
buff_datos_in2[puntero_in++]=adc_data;
if (puntero_in==(BUFFER_LENGTH+FILTER_LENGTH)){
puntero_in=FILTER_LENGTH;
seleccion_buffer_in=0;
status_end_buff_in=2;}}
if (seleccion_buffer_out==0){
dato_Salida = buff_datos_out1[puntero_out++];
if (puntero_out==(BUFFER_LENGTH+0)){
puntero_out=0;
seleccion_buffer_out=1;}}
else{
dato_Salida = buff_datos_out2[puntero_out++];
if (puntero_out==(BUFFER_LENGTH+0)){
puntero_out=0;
```

```
seleccion_buffer_out=0;}}
aux = dato_Salida&0x03ff;
aux= aux>>2;
set_value_status = pwma_set_channels_value(pwma,(1<<25),aux);
asm ("mov r12, 0");
asm ("nop");}
ISR(ADCIFB_interrupt_handler, AVR32_ADCIFB_IRQ_GROUP, ADC_INTERRUPT_PRIORITY){
adc_data = adcifb_get_last_data(adcifb);
adc_data &=0x03ff;
adcifb_clear_compare_gt_interrupt_flag(adcifb);
adcifb->ISR;}
static void usart_int_handler(void){
arribo_serial=1;
usart_read_char(EXAMPLE_USART, &dato_serial);}
```

ANEXO 3.- LIBRERÍA – COEFICIENTES

```
#ifndef COEFICIENTES_H_
#define COEFICIENTES_H_
A_ALIGNED dsp16_t filter_coef[3 - 1][32] = {{
            DSP16_Q(-0.0184183),
            DSP16_Q(-0.0207874),
            DSP16_Q(-0.0231255),
            DSP16_Q(-0.0254103),
            DSP16_Q(-0.0276199),
            DSP16_Q(-0.0297327),
            DSP16_Q(-0.0317281),
            DSP16_Q(-0.0335863),
            DSP16_Q(-0.0352887),
            DSP16_Q(-0.0368182),
            DSP16_Q(-0.0381593),
            DSP16_Q(-0.0392986),
            DSP16_Q(-0.0402242),
            DSP16_Q(-0.0409269),
            DSP16_Q(-0.0413995),
            DSP16_Q(0.9583631),
            DSP16_Q(-0.0416369),
            DSP16_Q(-0.0413995),
            DSP16_Q(-0.0409269),
            DSP16_Q(-0.0402242),
            DSP16_Q(-0.0392986),
            DSP16_Q(-0.0381593),
            DSP16_Q(-0.0368182),
            DSP16_Q(-0.0352887),
            DSP16_Q(-0.0335863),
            DSP16_Q(-0.0317281),
            DSP16_Q(-0.0297327),
            DSP16_Q(-0.0276199),
            DSP16_Q(-0.0254103),
            DSP16_Q(-0.0231255),
            DSP16_Q(-0.0207874),
            DSP16_Q(-0.0184183),}, {
            DSP16_Q(0.0184183),
            DSP16_Q(0.0207874),
            DSP16_Q(0.0231255),
            DSP16_Q(0.0254103),
            DSP16_Q(0.0276199),
            DSP16_Q(0.0297327),
            DSP16_Q(0.0317281),
            DSP16_Q(0.0335863),
            DSP16_Q(0.0352887),
            DSP16_Q(0.0368182),
            DSP16_Q(0.0381593),
            DSP16_Q(0.0392986),
            DSP16_Q(0.0402242),
            DSP16_Q(0.0409269),
            DSP16_Q(0.0413995),
            DSP16_Q(0.0416369),
            DSP16_Q(0.0416369),
            DSP16_Q(0.0413995),
            DSP16_Q(0.0409269),
```

```
                DSP16_Q(0.0402242),
                DSP16_Q(0.0392986),
                DSP16_Q(0.0381593),
                DSP16_Q(0.0368182),
                DSP16_Q(0.0352887),
                DSP16_Q(0.0335863),
                DSP16_Q(0.0317281),
                DSP16_Q(0.0297327),
                DSP16_Q(0.0276199),
                DSP16_Q(0.0254103),
                DSP16_Q(0.0231255),
                DSP16_Q(0.0207874),
                DSP16_Q(0.0184183),}};
#endif
```

ANEXO 4.- CÓDIGO DE LA APLICACIÓN

```
package com.example.dario.sliders;

import android.os.Bundle;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.SeekBar;
import android.widget.TextView;
import java.util.Arrays;
import java.util.Timer;
import java.util.TimerTask;

public class MainActivity extends AppCompatActivity implements
SeekBar.OnSeekBarChangeListener, TaskDelegate {
private EditText etPasaAltos;
private EditText etPasaBajos;
private int pasaAltos;
private int pasaBajos;
private byte[] cadenaFinal;

private int valorMinimoPasaAltos = 2500;
private int valorMinimoPasaBajos = 100;
private int valorMaximoPasaAltos = 20000;
private int valorMaximoPasaBajos = 1500;
private int valorInicialPasaAltos = 2500;
private int valorInicialPasaBajos = 500;
private Timer timer;

private int contadorEnvios = 0;
    @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
final SeekBar sbPasaAltos = ((SeekBar)
findViewById(R.id.sbPasaAltos));
final SeekBar sbPasaBajos = ((SeekBar)
findViewById(R.id.sbPasaBajos));
sbPasaBajos.setOnSeekBarChangeListener(this);
sbPasaAltos.setOnSeekBarChangeListener(this);
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
timer = new Timer();
cadenaFinal = new byte[6];
etPasaAltos = (EditText) findViewById(R.id.etPasaAltos);
etPasaAltos.addTextChangedListener(new TextWatcher() {
    @Override
public void beforeTextChanged(CharSequence s, int start, int count,
```

```
int after) {}
     @Override
public void onTextChanged(final CharSequence s, int start, int
before, int count) {
try {
if (Integer.parseInt(s.toString()) > valorMaximoPasaAltos ||
Integer.parseInt(s.toString()) < valorMinimoPasaAltos) {
etPasaAltos.setError("El rango debe estar entre " +
valorMinimoPasaAltos + " y " + valorMaximoPasaAltos);} else {
timer.cancel();
timer = new Timer();
timer.schedule(
new TimerTask() {
@Override
public void run() {
try {
sbPasaAltos.setProgress(Integer.parseInt(s.toString()) -
valorMinimoPasaAltos);
onStopTrackingTouch(sbPasaAltos);
Log.d("Valor", "sbPasaAltos: " + sbPasaAltos.getProgress());
Log.d("Valor", "Valor Ingresado: "+Integer.parseInt(s.toString()));
Log.d("Valor", "Valor Minimo Pasa Altos: " + valorMinimoPasaAltos);}
catch (Exception ex) {}}}, 500);}}
catch (Exception ex) {}}
            @Override
public void afterTextChanged(Editable s) {}});
etPasaBajos = (EditText) findViewById(R.id.etPasaBajos);
etPasaBajos.addTextChangedListener(new TextWatcher() {
            @Override
public void beforeTextChanged(CharSequence s, int start, int count,
int after) {}
            @Override
public void onTextChanged(final CharSequence s, int start, int
before, int count) {
try {
if (Integer.parseInt(s.toString()) > valorMaximoPasaBajos ||
Integer.parseInt(s.toString()) < valorMinimoPasaBajos) {
etPasaBajos.setError("El rango debe estar entre " +
valorMinimoPasaBajos + " y " + valorMaximoPasaBajos);}
else {
timer.cancel();
timer = new Timer();
timer.schedule(
new TimerTask() {
          @Override
public void run() {
try {
sbPasaBajos.setProgress(Integer.parseInt(s.toString()) -
valorMinimoPasaBajos);
onStopTrackingTouch(sbPasaBajos);
Log.d("Valor", "sbPasaBajos: " + sbPasaBajos.getProgress());
Log.d("Valor", "Valor Ingresado: "+Integer.parseInt(s.toString()));
Log.d("Valor", "Valor Minimo Pasa Bajos: " + valorMinimoPasaBajos);
```

```
} catch (Exception ex) {}}}, 500);}
} catch (Exception ex) {
}}
      @Override
public void afterTextChanged(Editable s) {}});
sbPasaAltos.setProgress(valorInicialPasaAltos);
sbPasaBajos.setProgress(valorInicialPasaBajos);}
    @Override
public boolean onCreateOptionsMenu(Menu menu) {
        return true;}
    @Override
public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_settings) {
return true;}
return super.onOptionsItemSelected(item);}
    @Override
public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
switch (seekBar.getId()) {
case R.id.sbPasaAltos:
((TextView) findViewById(R.id.tvPasaAltosSeekBar)).setText("FILTRO
PASA ALTOS: " + (valorMinimoPasaAltos + progress) + " Hz");
((TextView)findViewById(R.id.tvPasaAltos)).setText((valorMinimoPasaA
ltos + progress) + " Hz");
pasaAltos = valorMinimoPasaAltos + progress;
etPasaAltos.setText("" + pasaAltos);
etPasaAltos.setSelection(etPasaAltos.getText().length());
contadorEnvios++;
break;
case R.id.sbPasaBajos:
((TextView) findViewById(R.id.tvPasaBajosSeekBar)).setText("FILTRO
PASA BAJOS: " + ((valorMinimoPasaBajos + progress) + " Hz"));
if (valorMinimoPasaBajos + progress <= 600) {
((TextView) findViewById(R.id.tvPasaBajos)).setText(600 + " Hz");}
else {
((TextView)findViewById(R.id.tvPasaBajos)).setText((valorMinimoPasaB
ajos + progress) + " Hz");}
pasaBajos = valorMinimoPasaBajos + progress;
etPasaBajos.setText("" + pasaBajos);
etPasaBajos.setSelection(etPasaBajos.getText().length());
contadorEnvios++;
break;}}
    @Override
public void onStartTrackingTouch(SeekBar seekBar) {}
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        cadenaFinal[0] = (byte) 255;
        cadenaFinal[1] = (byte) 255;
        cadenaFinal[2] = (byte) (pasaBajos >> 8);
        cadenaFinal[3] = (byte) (pasaBajos);
        cadenaFinal[4] = (byte) (pasaAltos >> 8);
        cadenaFinal[5] = (byte) (pasaAltos);
```

```
Log.d("Conversion", "Conversion: " + Arrays.toString(cadenaFinal));
new NetworkCommunicationTask(this).execute(cadenaFinal);}
  @Override
public void taskCompletionResult(boolean result) {
if (!result) {
Snackbar snackbar = Snackbar
.make(findViewById(R.id.clPrincipal), "¡¡Error!! Revise la
conexión", Snackbar.LENGTH_LONG);
snackbar.show();}
else {
Snackbar snackbar = Snackbar
.make(findViewById(R.id.clPrincipal), "Enviado Exitosamente",
Snackbar.LENGTH_LONG);
snackbar.show();}}}
```

ANEXO 5.- DATASHEET WI-FLY

Features

- Drop in Wi-Fi solution for existing systems currently using 802.15.4 modules
- Based on Roving Networks' robust RN-171 Wi-Fi module
- Based on pseudo-standard footprint
- Onboard TCP/IP stack provides internet access to every node
- No custom profiles needed
- Ultra-low power: 4uA sleep, 40mA Rx, 180 mA Tx at 10dBm
- ConFigurable transmit power: 0dBm to +12dBm
- Hardware interface: TTL UART
- Data rate: 464Kbps using hardware flow control
- Through hole board simplifies system integration
- 8 general purpose digital I/O
- 3 analog sensor interfaces.
- Real-time clock for wakeup and time stamping
- Complete TCP/IP networking stack
- Wi-Fi Alliance certified for WEP, WPA and WPA2-PSK
- WPS push button for easy conFiguration
- FCC / CE/ ICS certified and RoHS compliant.

Applications

- Industrial metering
- HVAC control
- Room temperature sensors
- Pump conFiguration and control
- Telemetry
- PV/Solar controllers
- Robotics

Description

The RN-XV is a 802.11 b/g solution especially designed for customer who want to migrate their existing 802.15.4 architecture to a more standard TCP/IP based platform without having to redesign their existing hardware.

Based on a pseudo standard footprint often found in embedded applications, the RN-XV module from Roving Networks allows for Wi-Fi connectivity using 802.11 b/g standards in legacy and existing designs that may have been based upon 802.15.4 standard.

The RN-XV module is based upon Roving Networks' robust RN-171 Wi-Fi module and incorporates 802.11 b/g radio, 32 bit SPARC processor, TCP/TP stack, real-time clock, crypto accelerator, power management unit and analog sensor interface.

The module offers additional functionality through its onboard programmable GPIOs (10) and ADCs (8). The ADCs provide 14-bit resolution while the GPIOs can be configured to provide standard functionality or status signaling to a host microcontroller to reduce the need for serial polling between the Wi-Fi module and host microcontroller.

The module is pre-loaded with firmware to simplify integration and minimize development time of your application. In the simplest conFiguration, the hardware only requires four connections (PWR, TX, RX and GND) to create a wireless data connection.

Overview

- Host Data Rate up to 460Kbps over UART
- Intelligent built-in power management with programmable wakeup events (timers and I/O)
- Real time clock for time stamping, auto-sleep and auto-wakeup modes
- ConFiguration over WiFi or UART using simple ASCII commands
- Over the air firmware upgrade via FTP
- Secure WiFi authentication: WEP, WPA-PSK (TKIP), WPA2-PSK
- Built in networking applications DHCP, DNS, ARP, ICMP UDP, Telnet, FTP, HTML client
- 802.11 b/g power save and roaming functions
- ConFigurable transmit power: -2dBm to 12dBm
- WPS push button mode for easy and secure wireless setup
- Built-in HTML web client to send GPIO, UART and sensor data to remote web servers

Environmental Conditions

Parameter	Value
Temperature Range (Operating)	-40 ^{o}C - 85 ^{o}C
Temperature Range (Storage)	-40^{o}C - 85 ^{o}C
Relative Humidity (Operating)	≤90%
Relative Humidity (Storage)	≤90%

Electrical Characteristics

Supply Voltage	Min	Typ.	Max.	Unit
Input Power	3.0	3.3	3.7	V
Power consumption				
Sleep		4		uA
Standby (doze)		15		mA
Connected (idle, RX)		40		mA
Connected (TX)		180 at 10dBm		mA

Analog Sensor Inputs

Parameter	Value
Sense 0,1,2,3 wakeup detect threshold	500mV
AD sense 0-4 measurement range	0-400mV
Precision	14 bits = 12uV
Accuracy	5% un-calibrated, .01% calibrated
Minimum conversion time	35uS (5kHz over Wi-Fi)
Sensor Power (pin 33) output resistance 3.3V	10 ohms, max current = 50mA

Radio Characteristics

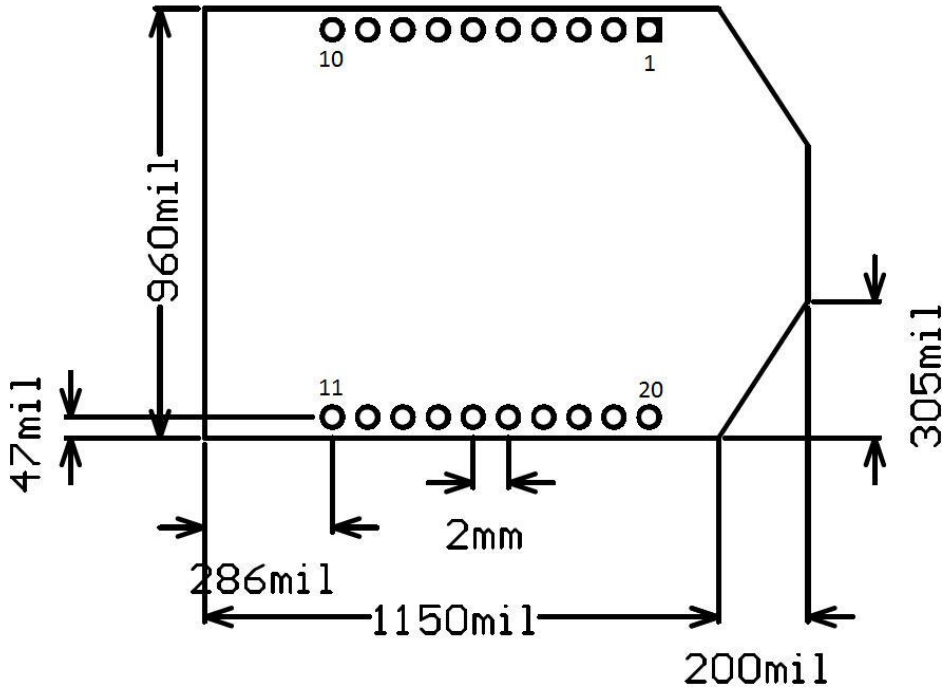
Parameter	Specifications
Frequency	2402 ~ 2480MHz
Modulation	802.11b compatibility : DSSS(CCK-11, CCK-5.5, DQPSK-2, DBPSK-1) 802.11g : OFDM (default)
Channel intervals	5MHz
Channels	1 - 14
Transmission rate (over the air)	1 – 11Mbps for 802.11b / 6 – 54Mbps for 802.11g
Receive sensitivity	-83dBm typ.
Output level (Class1)	0dBm to +12dBm (software conFigurable)

Transmit Power

Output Power	802.11 b (2Mbps) Current in mA*	802.11 g (24Mbps) Current in mA*
0	120	135
2	130	150
4	170	190
6	175	200
8	180	210
10	185	225
12	190	240

*** Measured at 3.3VDC input. The power consumption is the average power, active during actual power consumption**

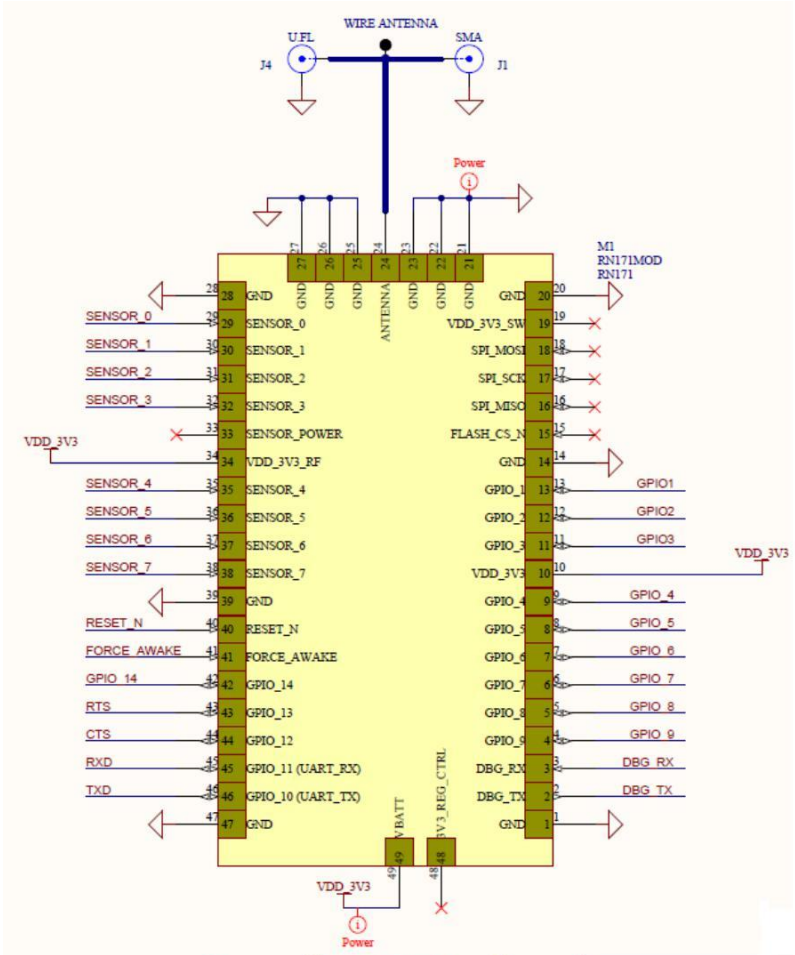
Physical Dimensions and pin out table

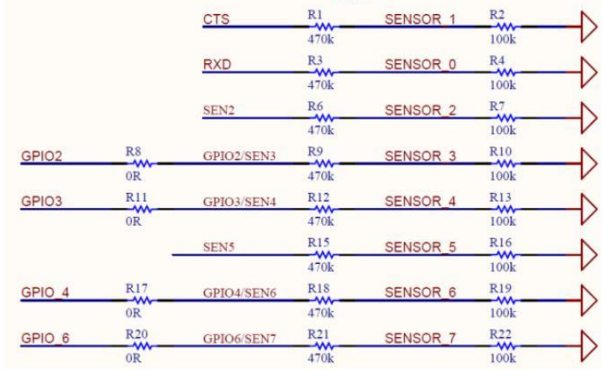


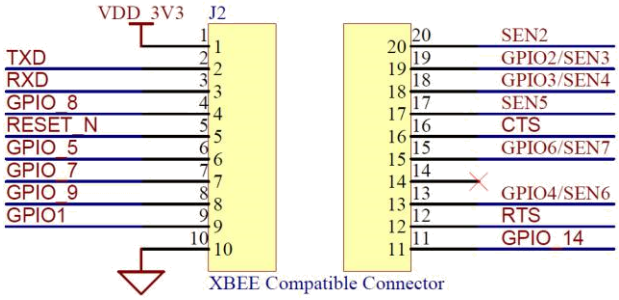
1 mil = 0.0254 millimeters

Pad Number	Signal Name	Description	Optional Function	Direction
1	VDD_3V3	3.3V regulated power input to the module		POWER
2	UART_TX	UART TX, 8mA drive, 3.3V tolerant		OUT →
3	UART_RX	UART RX, 3.3V tolerant		IN ←
4	GPIO 8	GPIO, 24mA drive, 3.3V tolerant		IN / OUT
5	RESET	**Optional** Module Reset Signal (active low), 100k Pull up, apply pulse of at least 160us, 3.3V Tolerant		INPUT
6	GPIO 5	GPIO, 24mA drive, 3.3V tolerant	Data TX/RX	OUT
7	GPIO 7	GPIO, 24mA drive, 3.3V tolerant		IN / OUT
8	GPIO 9	Enable Adhoc mode, Restore factory defaults, 8mA drive, 3.3V tolerant		IN / OUT
9	GPIO 1	GPIO, 8mA drive, 3.3V tolerant		IN / OUT
10	GND	Ground		GND
11	GPIO 14	GPIO, 8mA drive, 3.3V tolerant		IN / OUT
12	UART_RTS	UART RTS flow control, 8mA drive, 3.3V tolerant		OUT →
13	GPIO 4/SEN 6	GPIO, 24mA drive, 3.3V tolerant/ADC input , (**3.3V tolerant).** Defaults to GPIO 4	Association Status	IN / OUT
14	Not Used			No Connect
15	GPIO 6/SEN 7	GPIO, 24mA drive, 3.3V tolerant/ADC input , (**3.3V tolerant).** Defaults to GPIO 6	Connection Status	POWER
16	UART_CTS	UART CTS flow control, 3.3V tolerant		IN ←
17	SENSOR 5	Sensor Interface, Analog input to module, (**3.3V tolerant)**		INPUT
18	GPIO 3/SEN 4	GPIO, 8mA drive, 3.3V tolerant/ADC input (**3.3V tolerant**). Defaults to GPIO 3		IN / OUT
19	GPIO 2/SEN 3	GPIO, 8mA drive, 3.3V tolerant/ADC input (**3.3V tolerant**). Defaults to SEN 3		IN / OUT
20	SEN 2	Sensor Interface, Analog input to module, **3.3V tolerant**		INPUT

Module Schematic







Pins 13, 15, 18 and 19 on the RN-XV connector can be configured as GPIOs or as sensor inputs depending on the installed resistors.

When configured as GPIOs, ONLY Column 1 resistors MUST be installed. When configured as sensor inputs, ONLY Column 2 and Column 3 resistors MUST be stuffed.

Default conFiguration:

Sensor inputs – Pins 19 and 20
GPIO – Pins 13, 15 and 18

Design Concerns

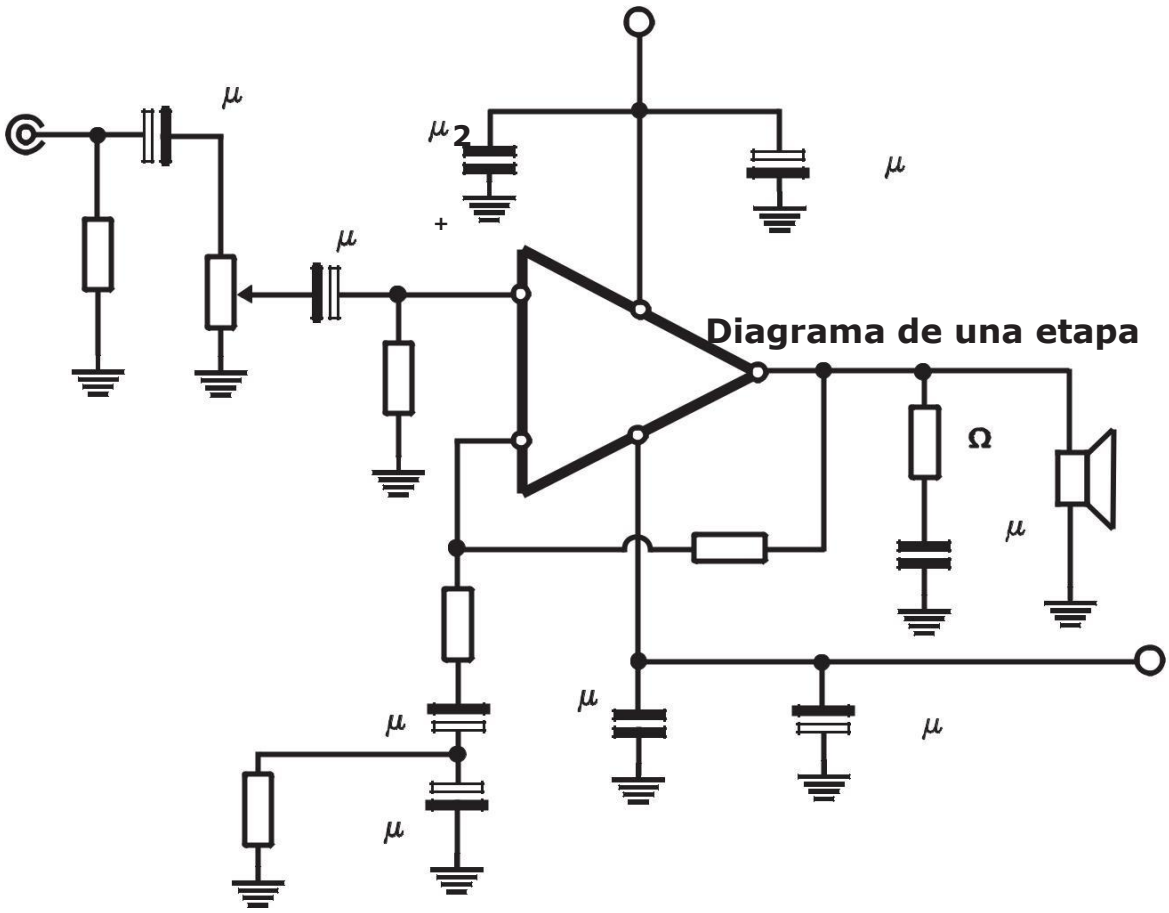
1. **Powering the RN-XV module:** Apply ONLY 3.3V±10% regulated power to pin 1 (VDD) and pin 10 (Ground). The module does not have any voltage regulator on board and hence MUST be powered from a regulated 3.3V power supply.

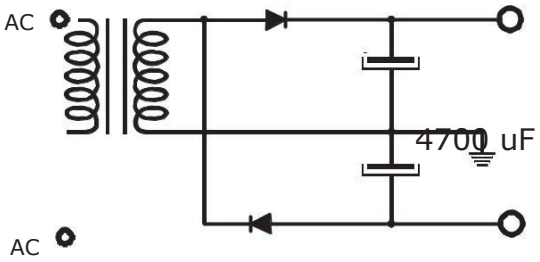
2. **Antenna:** The RN-XV ships with a wire antenna. Custom antenna conFigurations such as chip, U.FL. and SMA connector are available with extended lead times

Ordering Information

Part Number	Description
RN-XV-W	Standard conFiguration, industrial Temperature (- 40 to + 85 C) 802.15.4 replacement solution with ¼ inch wire antenna
RN-XV-U	Custom conFiguration, industrial Temperature (- 40 to + 85 C) 802.15.4 replacement solution with U.FL. connector
RN-XV-S	Custom conFiguration, industrial Temperature (- 40 to + 85 C) 802.15.4 replacement solution with SMA connector
RN-XV-C	Custom conFiguration, industrial Temperature (- 40 to + 85 C) 802.15.4 replacement solution with chip antenna

Amplificador de 30W estereo



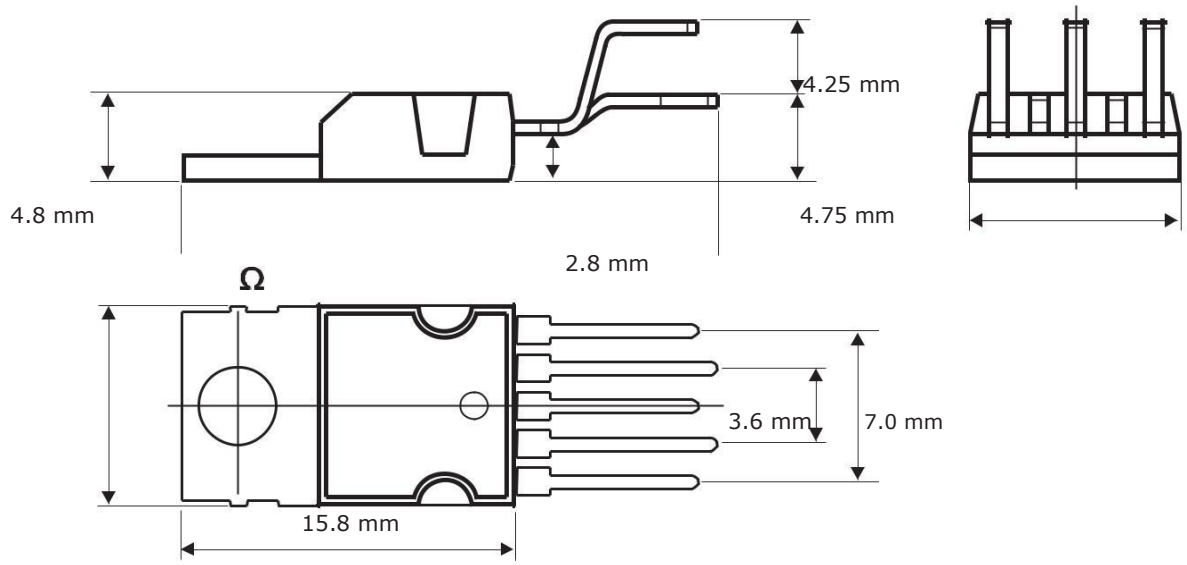


Este amplificador estereo es ideal para principiantes, por su sencillez y bajo costo. Utiliza el integrado **TDA2030** que es un circuito integrado monolítico de clase AB. Puede proporcionar 15W de potencia de salida con una carga de 8 Ohmios.

El TDA2030 ofrece una alta corriente de salida y muy baja distorsión armónica. Además, el dispositivo incorpora un sistema de protección de corto circuito que limita automáticamente la potencia disipada para mantener el punto de trabajo de los transistores de salida, asegurando su funcionamiento. También incluye un sistema térmico de apagado convencional.

Puede usar este amplificador con parlantes de 60W en adelante a 8 Ohmios.

TDA2030

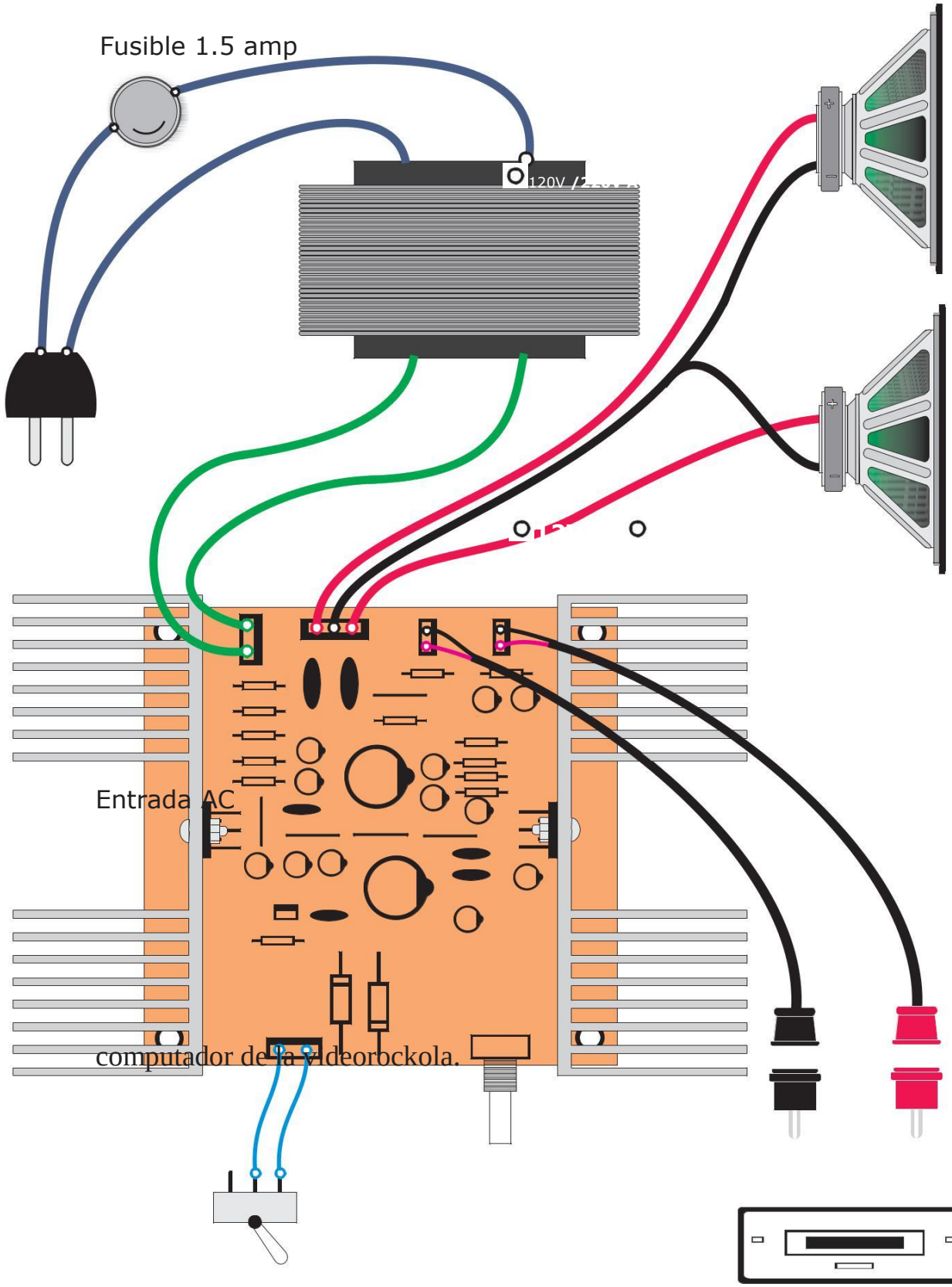


Valores sugeridos para un buen funcionamiento

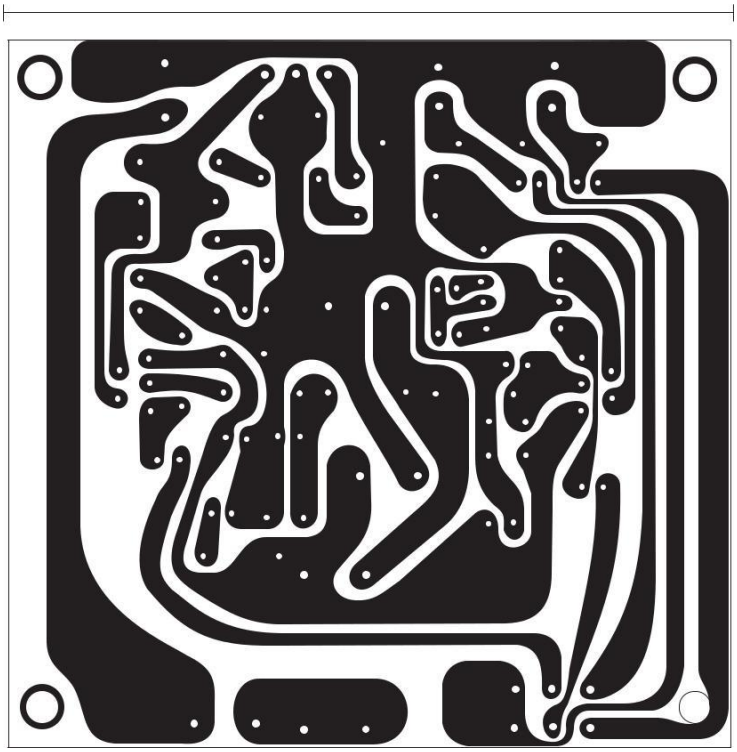
Los valores recomendados para los componentes externos son los que se muestran en el diagrama esquemático. Los valores modificables, están en la siguiente tabla y le puede ayudar al diseñador a personalizar el circuito. Los componentes que no se encuentran en la tabla, no se pueden modificar.

Componente	Valor sugerido	Función	Valor mayor que el propuesto	Valor menor que el propuesto
R1,R2	100K,22K	Impedancia de entrada	Aumenta la impedancia de entrada	Disminuye la impedancia de entrada
R3	47K	Ganancia de retroalimentación	Aumenta la ganancia	Disminuye la ganancia
R4	1.5K	Ganancia de retroalimentación	Disminuye la ganancia	Aumenta la ganancia
R5	100K	Estabilización de frecuencia	Peligro de Oscilaciones	
R6	1	Red de Zobel	Peligro de Oscilaciones	
C1	10 uF	Desacople de entrada	Aumenta el pop al encender	Recorte de las frecuencias bajas
C2	4.7 uF	Desacople	Aumenta el pop al encender	Recorte de las frecuencias bajas
C3, C5	0.1 uF	Derivación del voltaje de alimentación		Peligro de Oscilaciones
C4, C6	100 uF	Derivación del voltaje de alimentación		Peligro de Oscilaciones
C7,C8	47 uF	Derivación del pedestal de ganancia	Peligro de Oscilaciones	Recorte de las frecuencias bajas

Diagrama de conexión



Lista de materiales



Integrados
2 TDA 2030

Resistencias
2 R 1 ohmio 1/2w (café, negro, dorado)
2 R 47K 1/4w (amarillo, violeta, naranja)
4 R 100K 1/4w (café, negro, amarillo)
3 R 1K5 1/4w (café, verde, rojo)
2 R22K 1/4w (rojo, rojo, naranja)

Condensadores
2 C 4700 uF /16v
4 C 47 uF /16v
4 C 100 uF /16v
2 C 4.7 uF /16v
2 C 10 uF /16v
2 C 0.22 uF (224) /100v (poliester)
4 C 0.1 uF (104) /50v (Cerámico)

Varios
2 Diodos 1N5404 o superior
1 potenciometro 50K doble
1 led rojo
☐ interruptor
☐ conectores de 3 pines grande
☐ conectores de 3 pines pequeño
☐ conector de 2 pines pequeño
1. conector de 6 pines pequeño
1 transformador de 12 voltios 4 amperios
1 terminal para bafle de presión
1 regleta RCAx2
2 disipadores de aluminio

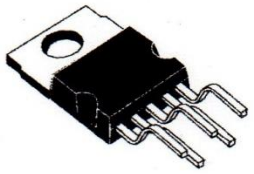
ANEXO 7.- DATASHEET TDA 2030

TDA2030

14W Hi-Fi AUDIO AMPLIFIER

DESCRIPTION

The TDA2030 is a monolithic integrated circuit in Pentawatt® package, intended for use as a low frequency class AB amplifier. Typically it provides 14W output power (d = 0.5%) at 14V/4Ω; at ± 14V or 28V, the guaranteed output power is 12W on a 4Ω load and 8W on a 8Ω (DIN45500).
The TDA2030 provides high output current and has very low harmonic and cross-over distortion.
Further the device incorporates an original (and patented) short circuit protection system comprising an arrangement for automatically limiting the dissipated power so as to keep the working point of the output transistors within their safe operating area. A conventional thermal shut-down system is also included.



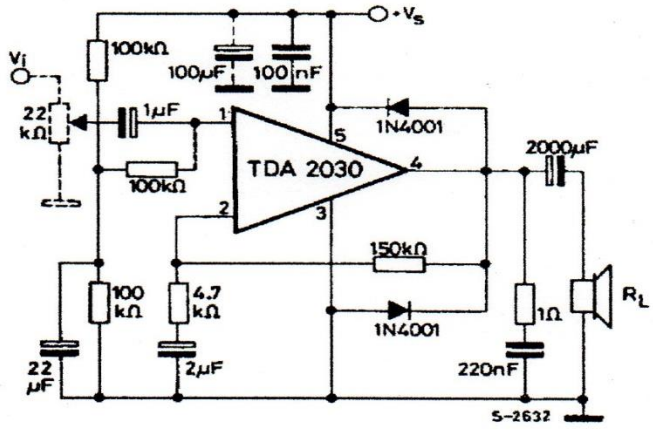
Pentawatt

ORDERING NUMBERS : TDA2030H
TDA2030V

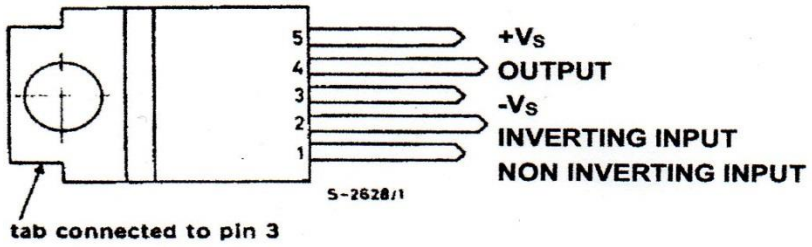
ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_s	Supply voltage	± 18 (36)	V
V_i	Input voltage	V_s	
V_i	Differential input voltage	± 15	V
I_o	Output peak current (internally limited)	3.5	A
P_{tot}	Power dissipation at T_{case} = 90°C	20	W
T_{stg}, T_j	Stoprage and junction temperature	-40 to 150	°C

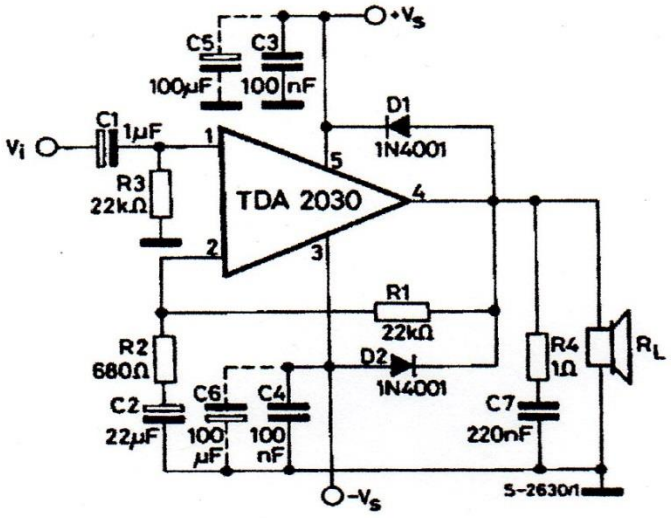
TYPICAL APPLICATION



PIN CONNECTION (top view)



TEST CIRCUIT



THERMAL DATA

Symbol	Parameter		Value	Unit
$R_{th\ j\text{-}case}$	Thermal resistance junction-case	max	3	°C/W

ELECTRICAL CHARACTERISTICS (Refer to the test circuit, $V_s = \pm 14V$, $T_{amb} = 25°C$ unless otherwise specified) for single Supply refer to fig. 15 Vs = 28V

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
V_s	Supply voltage		± 6 12		± 18 36	V
I_d	Quiescent drain current	$V_s = \pm 18V$ (Vs = 36V)		40	60	mA
I_b	Input bias current			0.2	2	µA
V_{os}	Input offset voltage			± 2	± 20	mV
I_{os}	Input offset current			± 20	± 200	nA
P_o	Output power	d = 0.5% G_v = 30 dB f = 40 to 15,000 Hz $R_L = 4\Omega$ $R_L = 8\Omega$	 12 8	 14 9		 W W
		d = 10% G_v = 30 dB f = 1 KHz $R_L = 4\Omega$ $R_L = 8\Omega$		 18 11		 W W
d	Distortion	P_o = 0.1 to 12W $R_L = 4\Omega$ G_v = 30 dB f = 40 to 15,000 Hz		 0.2	 0.5	 %
		P_o = 0.1 to 8W $R_L = 8\Omega$ G_v = 30 dB f = 40 to 15,000 Hz		 0.1	 0.5	 %
B	Power Bandwidth (-3 dB)	G_v = 30 dB P_o = 12W $R_L = 4\Omega$		10 to 140,000		Hz
R_i	Input resistance (pin 1)		0.5	5		MΩ
G_v	Voltage gain (open loop)			90		dB
G_v	Voltage gain (closed loop)	f = 1 kHz	29.5	30	30.5	dB
e_N	Input noise voltage	B = 22 Hz to 22 KHz		3	10	µV
i_N	Input noise current			80	200	pA
SVR	Supply voltage rejection	$R_L = 4\Omega$ G_v = 30 dB R_g = 22 kΩ V_{ripple} = 0.5 V_{eff} f_{ripple} = 100 Hz	40	50		dB
I_d	Drain current	P_o = 14W $R_L = 4\Omega$ P_o = W $R_L = 8\Omega$		 900 500		 mA mA

Figure 1. Output power vs. supply voltage

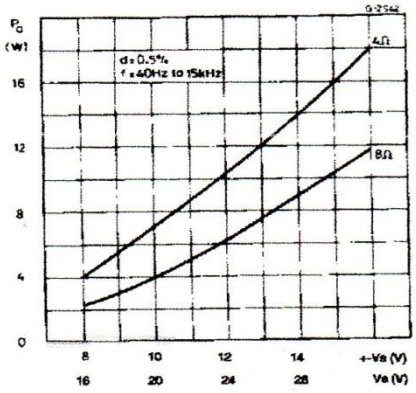


Figure 2. Output power vs. supply voltage

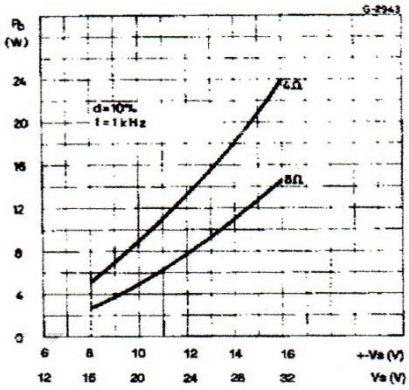


Figure 3. Distortion vs. output power

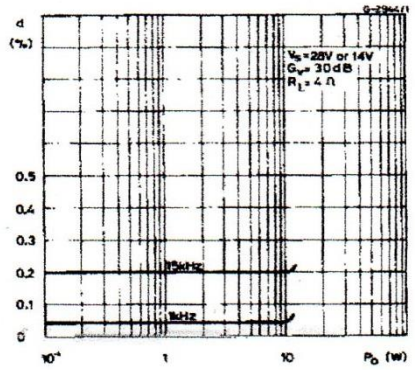


Figure 4. Distortion vs. output power

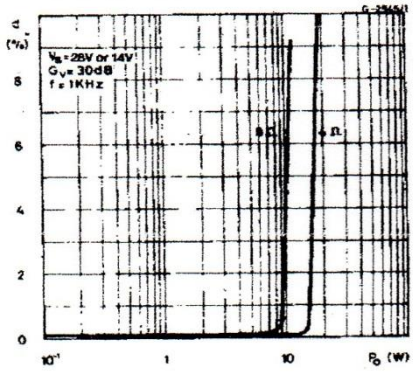


Figure 5. Distortion vs. output power

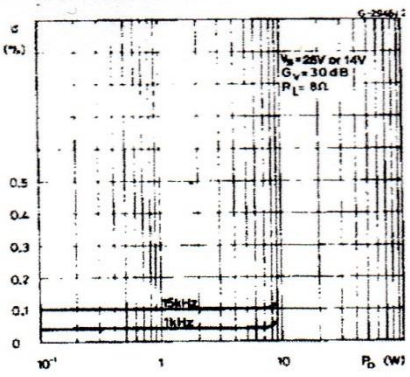


Figure 6. Distortion vs. frequency

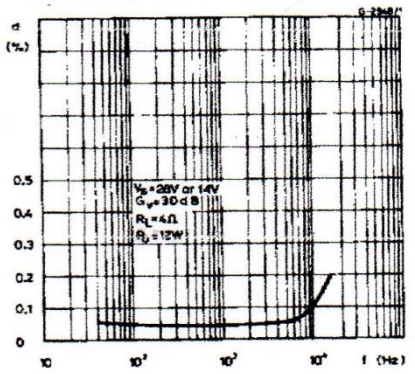


Figure 7. Distortion vs. frequency

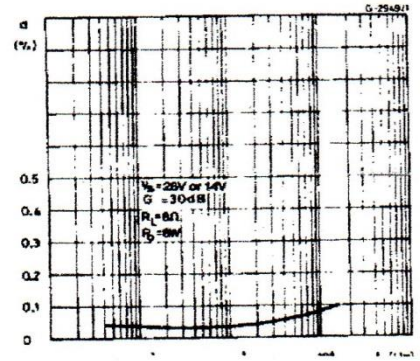


Figure 8. Frequency response with different values of the rolloff capacitor C8 (see fig. 13)

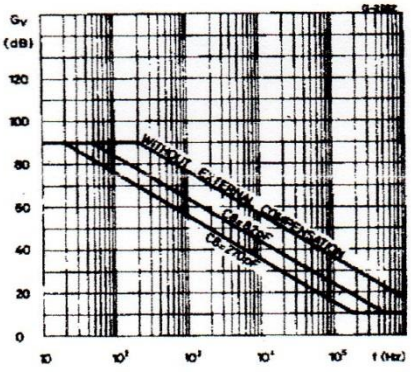


Figure 9. Quiescent current vs. supply voltage

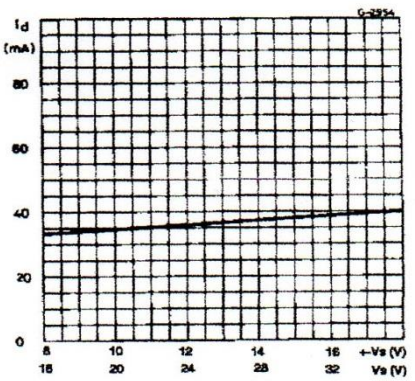


Figure 10. Supply voltage rejection vs. voltage gain

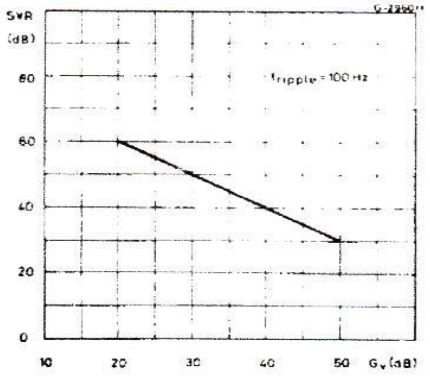


Figure 11. Power dissipation and efficiency vs. output power

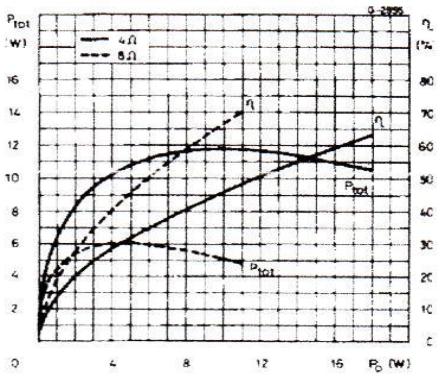
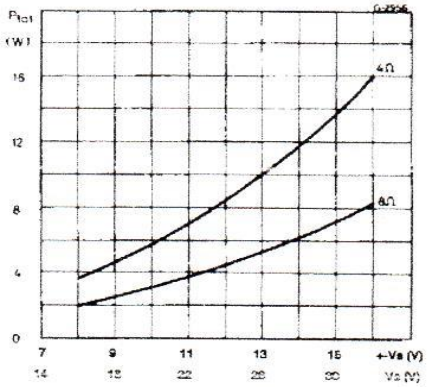


Figure 12. Maximum power dissipation vs. supply voltage (sine wave operation)



APPLICATION INFORMATION

Figure 13. Typical amplifier with split power supply

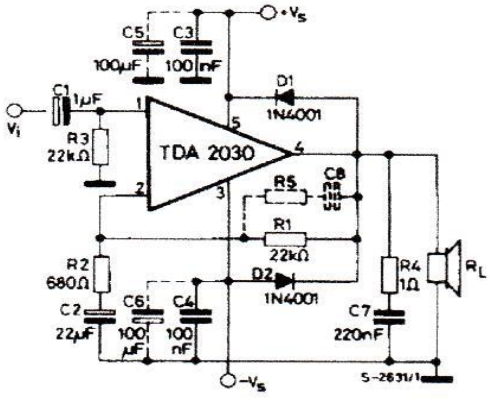
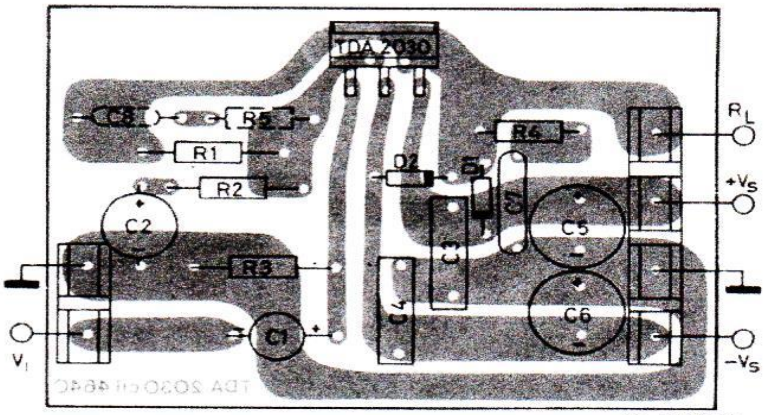


Figure 14. P.C. board and component layout for the circuit of fig. 13 (1 : 1 scale)



CS-0088 / 1

APPLICATION INFORMATION (continued)

Figure 15. Typical amplifier with single power supply

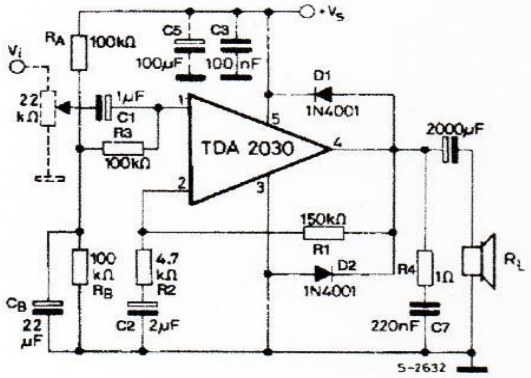


Figure 16. P.C. board and component layout for the circuit of fig. 15 (1 : 1 scale)

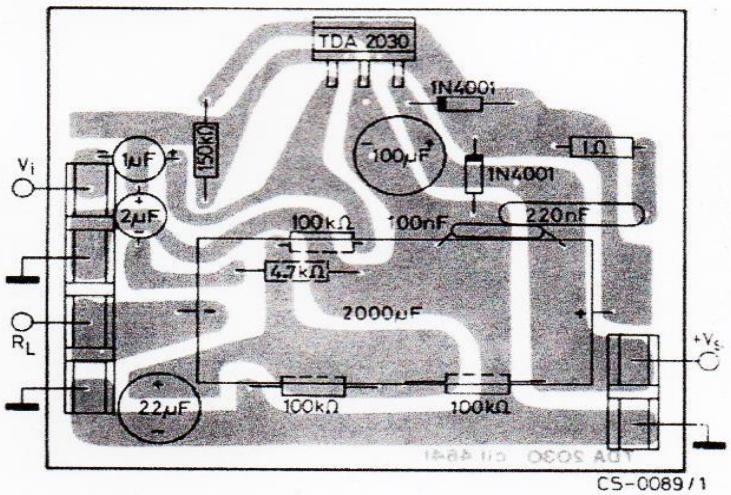
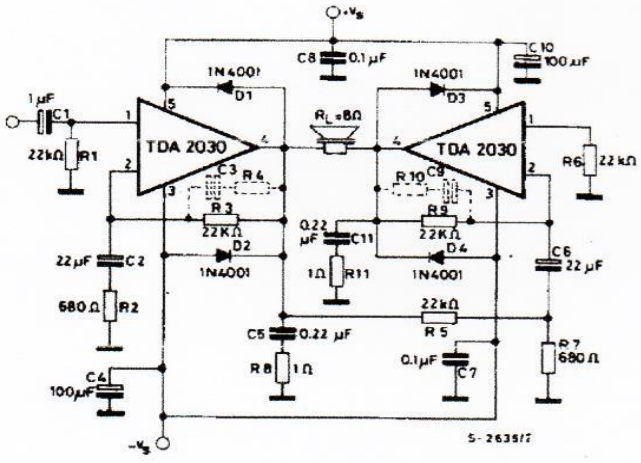


Figure 17. Bridge amplifier configuration with split power supply (P_o = 28W, V_s = ±14V)



PRACTICAL CONSIDERATIONS

Printed circuit board

The layout shown in Fig. 16 should be adopted by the designers. If different layouts are used, the ground points of input 1 and input 2 must be well decoupled from the ground return of the output in which a high current flows.

Assembly suggestion

No electrical isolation is needed between the package and the heatsink with single supply voltage configuration.

Application suggestions

The recommended values of the components are those shown on application circuit of fig. 13. Different values can be used. The following table can help the designer.

Component	Recomm. value	Purpose	Larger than recommended value	Smaller than recommended value
R1	22 kΩ	Closed loop gain setting	Increase of gain	Decrease of gain (*)
R2	680 Ω	Closed loop gain setting	Decrease of gain (*)	Increase of gain
R3	22 kΩ	Non inverting input biasing	Increase of input impedance	Decrease of input impedance
R4	1 Ω	Frequency stability	Danger of osccilat. at high frequencies with induct. loads	
R5	≅ 3 R2	Upper frequency cutoff	Poor high frequencies attenuation	Danger of oscillation
C1	1 µF	Input DC decoupling		Increase of low frequencies cutoff
C2	22 µF	Inverting DC decoupling		Increase of low frequencies cutoff
C3, C4	0.1 µF	Supply voltage bypass		Danger of oscillation
C5, C6	100 µF	Supply voltage bypass		Danger of oscillation
C7	0.22 µF	Frequency stability		Danger of oscillation
C8	$\cong \frac{1}{2\pi\ B\ R1}$	Upper frequency cutoff	Smaller bandwidth	Larger bandwidth
D1, D2	1N4001	To protect the device against output voltage spikes		

(*) Closed loop gain must be higher than 24dB

SINGLE SUPPLY APPLICATION

Component	Recomm. value	Purpose	Larger than recommended value	Smaller than recommended value
R1	150 kΩ	Closed loop gain setting	Increase of gain	Decrease of gain (*)
R2	4.7 kΩ	Closed loop gain setting	Decrease of gain (*)	Increase of gain
R3	100 kΩ	Non inverting input biasing	Increase of input impedance	Decrease of input impedance
R4	1 Ω	Frequency stability	Danger of osccilat. at high frequencies with induct. loads	
R_A/R_B	100 kΩ	Non inverting input Biasing		Power Consumption
C1	1 µF	Input DC decoupling		Increase of low frequencies cutoff
C2	22 µF	Inverting DC decoupling		Increase of low frequencies cutoff
C3	0.1 µF	Supply voltage bypass		Danger of oscillation
C5	100 µF	Supply voltage bypass		Danger of oscillation
C7	0.22 µF	Frequency stability		Danger of oscillation
C8	$\cong \frac{1}{2\pi\ B\ R1}$	Upper frequency cutoff	Smaller bandwidth	Larger bandwidth
D1, D2	1N4001	To protect the device against output voltage spikes		

(*) Closed loop gain must be higher than 24dB

ANEXO 8.- DATASHEET AT32UC3L

Features

- High-performance, Low-power 32-bit Atmel® AVR® Microcontroller
  - Compact Single-cycle RISC Instruction Set Including DSP Instructions
  - Read-modify-write Instructions and Atomic Bit Manipulation
  - Performance
    - Up to 64DMIPS Running at 50MHz from Flash (1 Flash Wait State)
    - Up to 36DMIPS Running at 25MHz from Flash (0 Flash Wait State)
  - Memory Protection Unit (MPU)
    - Secure Access Unit (SAU) providing User-defined Peripheral Protection
- picoPower® Technology for Ultra-low Power Consumption
- Multi-hierarchy Bus System
  - High-performance Data Transfers on Separate Buses for Increased Performance
  - 12 Peripheral DMA Channels Improve Speed for Peripheral Communication
- Internal High-speed Flash
  - 64Kbytes, 32Kbytes, and 16Kbytes Versions
  - Single-cycle Access up to 25MHz
  - FlashVault Technology Allows Pre-programmed Secure Library Support for End User Applications
  - Prefetch Buffer Optimizing Instruction Execution at Maximum Speed
  - 100,000 Write Cycles, 15-year Data Retention Capability
  - Flash Security Locks and User-defined Configuration Area
- Internal High-speed SRAM, Single-cycle Access at Full Speed
  - 16Kbytes (64Kbytes and 32Kbytes Flash), or 8Kbytes (16Kbytes Flash)
- Interrupt Controller (INTC)
  - Autovectored Low-latency Interrupt Service with Programmable Priority
- External Interrupt Controller (EIC)
- Peripheral Event System for Direct Peripheral to Peripheral Communication
- System Functions
  - Power and Clock Manager
  - SleepWalking Power Saving Control
  - Internal System RC Oscillator (RCSYS)
  - 32KHz Oscillator
  - Multipurpose Oscillator and Digital Frequency Locked Loop (DFLL)
- Windowed Watchdog Timer (WDT)
- Asynchronous Timer (AST) with Real-time Clock Capability
  - Counter or Calendar Mode Supported
- Frequency Meter (FREQM) for Accurate Measuring of Clock Frequency
- Six 16-bit Timer/Counter (TC) Channels
  - External Clock Inputs, PWM, Capture, and Various Counting Capabilities
- 36 PWM Channels (PWMA)
  - 8-bit PWM with a Source Clock up to 150MHz
- Four Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
  - Independent Baudrate Generator, Support for SPI
  - Support for Hardware Handshaking
- One Master/Slave Serial Peripheral Interfaces (SPI) with Chip Select Signals
  - Up to 15 SPI Slaves can be Addressed
- Two Master and Two Slave Two-wire Interface (TWI), 400kbit/s I^2C-compatible
- One 8-channel Analog-to-digital Converter (ADC) with up to 12 Bits Resolution
  - Internal Temperature Sensor

32-bit Atmel AVR Microcontroller

AT32UC3L064
AT32UC3L032
AT32UC3L016

32099–01/2012

- Eight Analog Comparators (AC) with Optional Window Detection
- Capacitive Touch (CAT) Module
  - Hardware-assisted Atmel® AVR® QTouch® and Atmel® AVR® QMatrix Touch Acquisition
  - Supports QTouch and QMatrix Capture from Capacitive Touch Sensors
- QTouch Library Support
  - Capacitive Touch Buttons, Sliders, and Wheels
  - QTouch and QMatrix Acquisition
- On-chip Non-intrusive Debug System
  - Nexus Class 2+, Runtime Control, Non-intrusive Data and Program Trace
  - aWire Single-pin Programming Trace and Debug Interface Muxed with Reset Pin
  - NanoTrace Provides Trace Capabilities through JTAG or aWire Interface
- 48-pin TQFP/QFN/TLLGA (36 GPIO Pins)
- Five High-drive I/O Pins
- Single 1.62-3.6 V Power Supply

1. Description

The Atmel® AVR® AT32UC3L016/32/64 is a complete system-on-chip microcontroller based on the AVR32 UC RISC processor running at frequencies up to 50MHz. AVR32 UC is a high-performance 32-bit RISC microprocessor core, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption, high code density, and high performance.

The processor implements a Memory Protection Unit (MPU) and a fast and flexible interrupt controller for supporting modern and real-time operating systems. The Secure Access Unit (SAU) is used together with the MPU to provide the required security and integrity.

Higher computation capability is achieved using a rich set of DSP instructions.

The AT32UC3L016/32/64 embeds state-of-the-art picoPower technology for ultra-low power consumption. Combined power control techniques are used to bring active current consumption down to 165µA/MHz, and leakage down to 9nA while still retaining a bank of backup registers. The device allows a wide range of trade-offs between functionality and power consumption, giving the user the ability to reach the lowest possible power consumption with the feature set required for the application.

The Peripheral Direct Memory Access (DMA) controller enables data transfers between peripherals and memories without processor involvement. The Peripheral DMA controller drastically reduces processing overhead when transferring continuous and large data streams.

The AT32UC3L016/32/64 incorporates on-chip Flash and SRAM memories for secure and fast access. The FlashVault technology allows secure libraries to be programmed into the device. The secure libraries can be executed while the CPU is in Secure State, but not read by non-secure software in the device. The device can thus be shipped to end customers, who will be able to program their own code into the device to access the secure libraries, but without risk of compromising the proprietary secure code.

The External Interrupt Controller (EIC) allows pins to be configured as external interrupts. Each external interrupt has its own interrupt request and can be individually masked.

The Peripheral Event System allows peripherals to receive, react to, and send peripheral events without CPU intervention. Asynchronous interrupts allow advanced peripheral operation in low power sleep modes.

The Power Manager (PM) improves design flexibility and security. The Power Manager supports SleepWalking functionality, by which a module can be selectively activated based on peripheral events, even in sleep modes where the module clock is stopped. Power monitoring is supported by on-chip Power-on Reset (POR), Brown-out Detector (BOD), and Supply Monitor (SM). The device features several oscillators, such as Digital Frequency Locked Loop (DFLL), Oscillator 0 (OSC0), and system RC oscillator (RCSYS). Either of these oscillators can be used as source for the system clock. The DFLL is a programmable internal oscillator from 40 to 150MHz. It can be tuned to a high accuracy if an accurate refernce clock is running, e.g. the 32KHz crystal oscillator.

The Watchdog Timer (WDT) will reset the device unless it is periodically serviced by the software. This allows the device to recover from a condition that has caused the system to be unstable.

The Asynchronous Timer (AST) combined with the 32KHz crystal oscillator supports powerful real-time clock capabilities, with a maximum timeout of up to 136 years. The AST can operate in counter mode or calendar mode.

The Frequency Meter (FREQM) allows accurate measuring of a clock frequency by comparing it to a known reference clock.

The device includes six identical 16-bit Timer/Counter (TC) channels. Each channel can be independently programmed to perform frequency measurement, event counting, interval measurement, pulse generation, delay timing, and pulse width modulation.

The Pulse Width Modulation controller (PWMA) provides 8-bit PWM channels which can be synchronized and controlled from a common timer. One PWM channel is available for each I/O pin on the device, enabling applications that require multiple PWM outputs, such as LCD backlight control. The PWM channels can operate independently, with duty cycles set independently from each other, or in interlinked mode, with multiple channels changed at the same time.

The AT32UC3L016/32/64 also features many communication interfaces, like USART, SPI, and TWI, for communication intensive applications. The USART supports different communication modes, like SPI Mode and LIN Mode.

A general purpose 8-channel ADC is provided, as well as eight analog comparators (AC). The ADC can operate in 10-bit mode at full speed or in enhanced mode at reduced speed, offering up to 12-bit resolution. The ADC also provides an internal temperature sensor input channel. The analog comparators can be paired to detect when the sensing voltage is within or outside the defined reference window.

The Capacitive Touch (CAT) module senses touch on external capacitive touch sensors, using the QTouch technology. Capacitive touch sensors use no external mechanical components, unlike normal push buttons, and therefore demand less maintenance in the user application. The CAT module allows up to 17 touch sensors, or up to 16 by 8 matrix sensors to be interfaced. One touch sensor can be configured to operate autonomously without software interaction, allowing wakeup from sleep modes when activated.

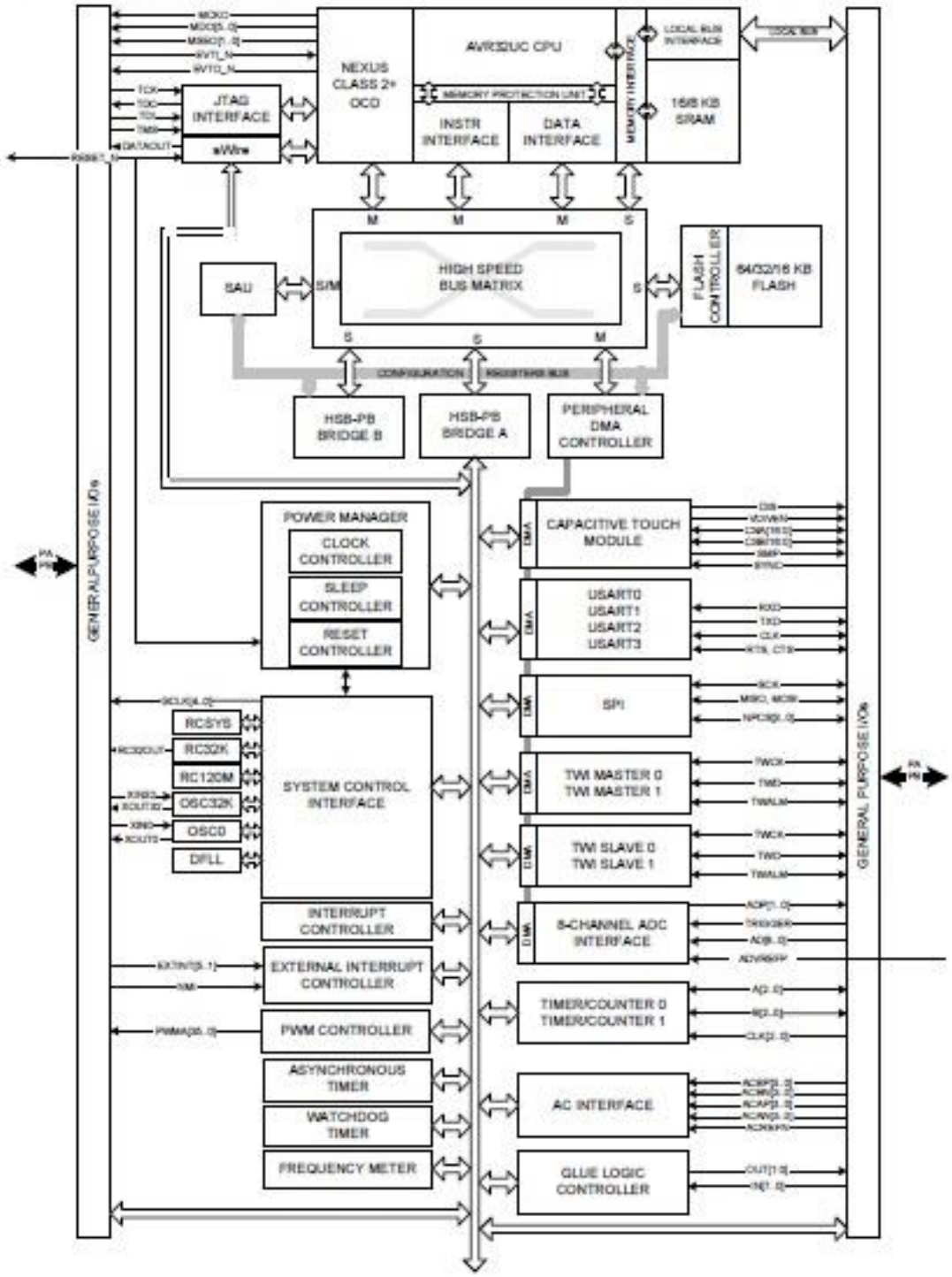
Atmel offers the QTouch library for embedding capacitive touch buttons, sliders, and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys as well as Adjacent Key Suppression® (AKS®) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop, and debug your own touch applications.

The AT32UC3L016/32/64 integrates a class 2+ Nexus 2.0 On-chip Debug (OCD) System, with non-intrusive real-time trace and full-speed read/write memory access, in addition to basic run-time control. The NanoTrace interface enables trace feature for aWire- or JTAG-based debuggers. The single-pin aWire interface allows all features available through the JTAG interface to be accessed through the RESET pin, allowing the JTAG pins to be used for GPIO or peripherals.

2. Overview

2.1 Block Diagram

Figure 2-1. Block Diagram



2.2 Configuration Summary

Table 2-1. Configuration Summary

Feature	AT32UC3L064	AT32UC3L032	AT32UC3L016
Flash	64KB	32KB	16KB
SRAM	16KB	16KB	8KB
GPIO	36		
High-drive pins	5		
External Interrupts	6		
TWI	2		
USART	4		
Peripheral DMA Channels	12		
Peripheral Event System	1		
SPI	1		
Asynchronous Timers	1		
Timer/Counter Channels	6		
PWM channels	36		
Frequency Meter	1		
Watchdog Timer	1		
Power Manager	1		
Secure Access Unit	1		
Glue Logic Controller	1		
Oscillators	Digital Frequency Locked Loop 40-150MHz (DFLL) Crystal Oscillator 3-16MHz (OSC0) Crystal Oscillator 32KHz (OSC32K) RC Oscillator 120MHz (RC120M) RC Oscillator 115kHz (RCSYS) RC Oscillator 32kHz (RC32K)		
ADC	8-channel 12-bit		
Temperature Sensor	1		
Analog Comparators	8		
Capacitive Touch Module	1		
JTAG	1		
aWire	1		
Max Frequency	50MHz		
Packages	TQFP48/QFN48/TLLGA48		

3. Package and Pinout

3.1 Package

The device pins are multiplexed with peripheral functions as described in Section 3.2.

Figure 3-1. TQFP48/QFN48 Pinout

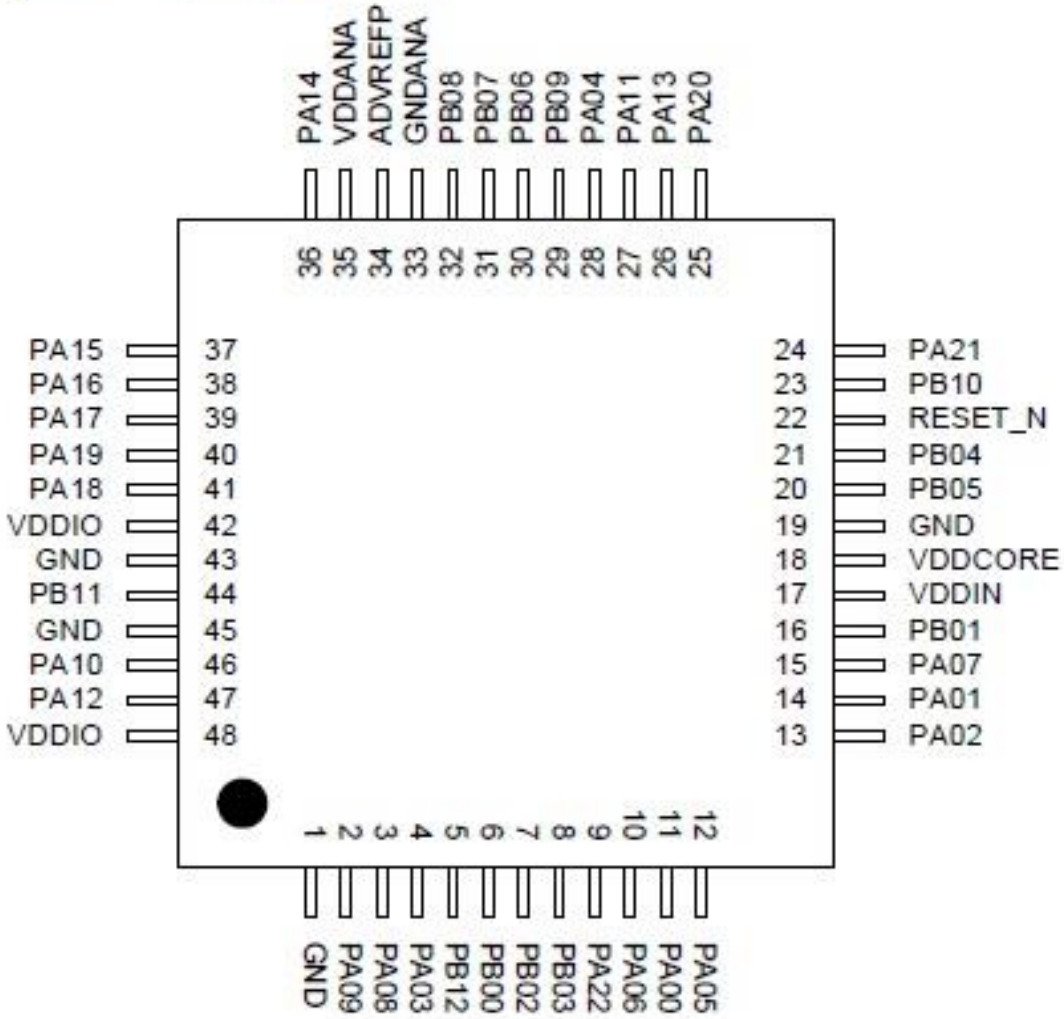
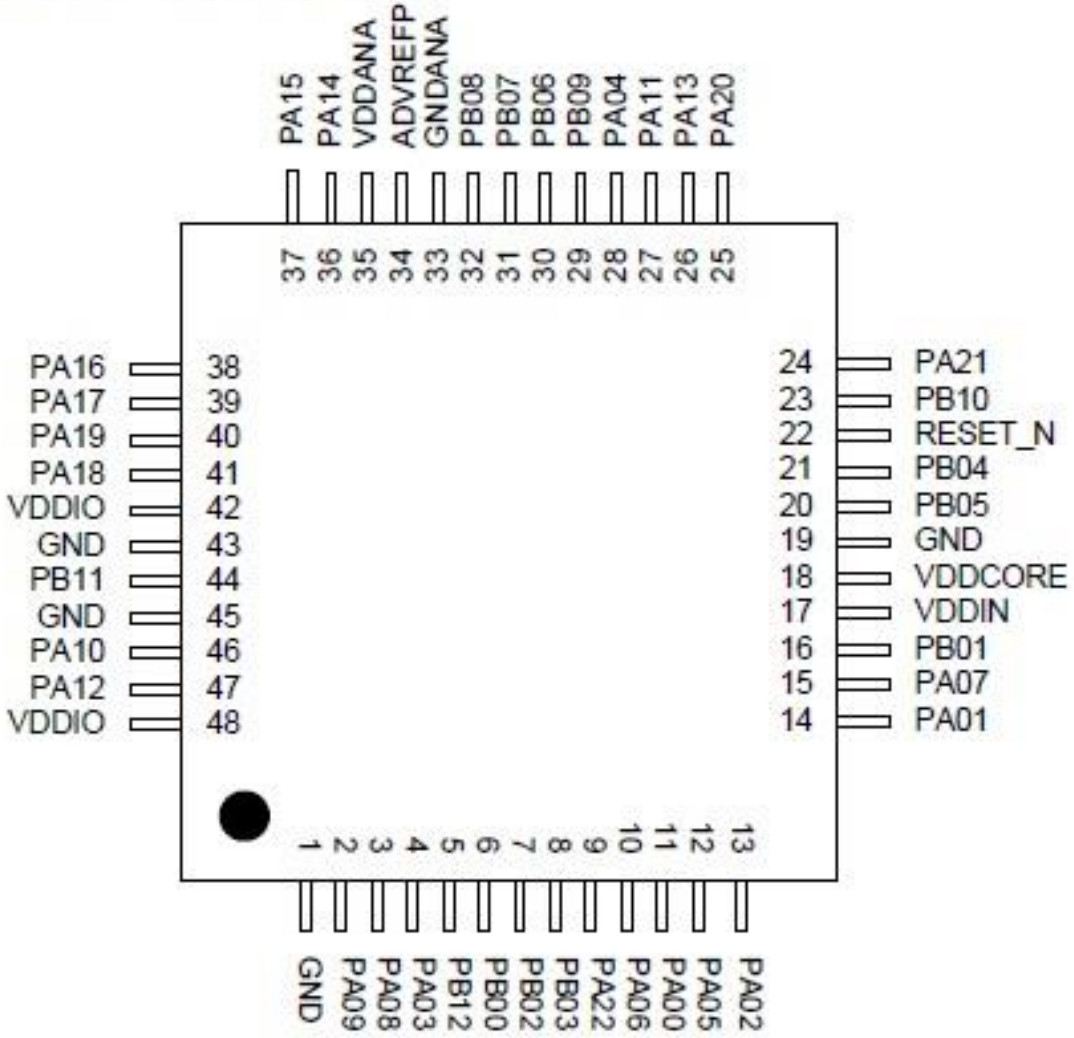


Figure 3-2. TLLGA48 Pinout



3.2 Peripheral Multiplexing on I/O lines

3.2.1 Multiplexed signals

Each GPIO line can be assigned to one of the peripheral functions. The following table describes the peripheral signals multiplexed to the GPIO lines.

Table 3-1. GPIO Controller Function Multiplexing

48-pin	PIN	GPIO	Supply	Pin Type	GPIO Function							
					A	B	C	D	E	F	G	H
11	PA00	0	VDDIO	Normal I/O	USART0 TXD	USART1 RTS	SPI NPCS[2]		PWMA PWMA[0]		SCIF GCLK[0]	CAT CSA[2]
14	PA01	1	VDDIO	Normal I/O	USART0 RXD	USART1 CTS	SPI NPCS[3]	USART1 CLK	PWMA PWMA[1]	ACIFB ACAP[0]	TWIMS0 TWALM	CAT CSA[1]
13	PA02	2	VDDIO	High-drive I/O	USART0 RTS	ADCIFB TRIGGER	USART2 TXD	TC0 A0	PWMA PWMA[2]	ACIFB ACBP[0]	USART0 CLK	CAT CSA[3]
4	PA03	3	VDDIO	Normal I/O	USART0 CTS	SPI NPCS[1]	USART2 TXD	TC0 B0	PWMA PWMA[3]	ACIFB ACBN[3]	USART0 CLK	CAT CSB[3]
28	PA04	4	VDDIO	Normal I/O	SPI MISO	TWIMS0 TWCK	USART1 RXD	TC0 B1	PWMA PWMA[4]	ACIFB ACBP[1]		CAT CSA[7]
12	PA05	5	VDDIO	Normal I/O (TWI)	SPI MOSI	TWIMS1 TWCK	USART1 TXD	TC0 A1	PWMA PWMA[5]	ACIFB ACBN[0]	TWIMS0 TWD	CAT CSB[7]
10	PA06	6	VDDIO	High-drive I/O, 5V tolerant	SPI SCK	USART2 TXD	USART1 CLK	TC0 B0	PWMA PWMA[6]		SCIF GCLK[1]	CAT CSB[1]
15	PA07	7	VDDIO	Normal I/O (TWI)	SPI NPCS[2]	USART2 RXD	TWIMS1 TWALM	TWIMS0 TWCK	PWMA PWMA[7]	ACIFB ACAN[0]	EIC EXTINT[0]	CAT CSB[2]
3	PA08	8	VDDIO	High-drive I/O	USART1 TXD	SPI NPCS[2]	TC0 A2	ADCIFB ADP[0]	PWMA PWMA[8]			CAT CSA[4]
2	PA09	9	VDDIO	High-drive I/O	USART1 RXD	SPI NPCS[3]	TC0 B2	ADCIFB ADP[1]	PWMA PWMA[9]	SCIF GCLK[2]	EIC EXTINT[1]	CAT CSB[4]
46	PA10	10	VDDIO	Normal I/O	TWIMS0 TWD		TC0 A0		PWMA PWMA[10]	ACIFB ACAP[1]	SCIF GCLK[2]	CAT CSA[5]
27	PA11	11	VDDIN	Normal I/O					PWMA PWMA[11]			
47	PA12	12	VDDIO	Normal I/O		USART2 CLK	TC0 CLK1	CAT SMP	PWMA PWMA[12]	ACIFB ACAN[1]	SCIF GCLK[3]	CAT CSB[5]
26	PA13	13	VDDIN	Normal I/O	GLOC OUT[0]	GLOC IN[7]	TC0 A0	SCIF GCLK[2]	PWMA PWMA[13]	CAT SMP	EIC EXTINT[2]	CAT CSA[0]
36	PA14	14	VDDIO	Normal I/O	ADCIFB AD[0]	TC0 CLK2	USART2 RTS	CAT SMP	PWMA PWMA[14]		SCIF GCLK[4]	CAT CSA[6]
37	PA15	15	VDDIO	Normal I/O	ADCIFB AD[1]	TC0 CLK1		GLOC IN[6]	PWMA PWMA[15]	CAT SYNC	EIC EXTINT[3]	CAT CSB[6]
38	PA16	16	VDDIO	Normal I/O	ADCIFB AD[2]	TC0 CLK0		GLOC IN[5]	PWMA PWMA[16]	ACIFB ACREFN	EIC EXTINT[4]	CAT CSA[8]
39	PA17	17	VDDIO	Normal I/O (TWI)		TC0 A1	USART2 CTS	TWIMS1 TWD	PWMA PWMA[17]	CAT SMP	CAT DIS	CAT CSB[8]
41	PA18	18	VDDIO	Normal I/O	ADCIFB AD[4]	TC0 B1		GLOC IN[4]	PWMA PWMA[18]	CAT SYNC	EIC EXTINT[5]	CAT CSB[0]

Table 3-1. GPIO Controller Function Multiplexing

40	PA19	19	VDDIO	Normal I/O	ADCIFB AD[5]		TC0 A2	TWIMS1 TWALM	PWMA PWMA[19]		CAT SYNC	CAT CSA[10]
25	PA20	20	VDDIN	Normal I/O	USART2 TXD		TC0 A1	GLOC IN[3]	PWMA PWMA[20]	SCIF RC32OUT		CAT CSA[12]
24	PA21	21	VDDIN	Normal I/O (TWI, 5V tolerant SMBus)	USART2 RXD	TWIMS0 TWD	TC0 B1	ADCIFB TRIGGER	PWMA PWMA[21]	PWMA PWMAOD[21]	SCIF GCLK[0]	CAT SMP
9	PA22	22	VDDIO	Normal I/O	USART0 CTS	USART2 CLK	TC0 B2	CAT SMP	PWMA PWMA[22]	ACIFB ACBN[2]		CAT CSB[10]
6	PB00	32	VDDIO	Normal I/O	USART3 TXD	ADCIFB ADP[0]	SPI NPCS[0]	TC0 A1	PWMA PWMA[23]	ACIFB ACAP[2]	TC1 A0	CAT CSA[9]
16	PB01	33	VDDIO	High-drive I/O	USART3 RXD	ADCIFB ADP[1]	SPI SCK	TC0 B1	PWMA PWMA[24]		TC1 A1	CAT CSB[9]
7	PB02	34	VDDIO	Normal I/O	USART3 RTS	USART3 CLK	SPI MISO	TC0 A2	PWMA PWMA[25]	ACIFB ACAN[2]	SCIF GCLK[1]	CAT CSB[11]
8	PB03	35	VDDIO	Normal I/O	USART3 CTS	USART3 CLK	SPI MOSI	TC0 B2	PWMA PWMA[26]	ACIFB ACBP[2]	TC1 A2	CAT CSA[11]
21	PB04	36	VDDIN	Normal I/O (TWI, 5V tolerant SMBus)	TC1 A0	USART1 RTS	USART1 CLK	TWIMS0 TWALM	PWMA PWMA[27]	PWMA PWMAOD[27]	TWIMS1 TWCK	CAT CSA[14]
20	PB05	37	VDDIN	Normal I/O (TWI, 5V tolerant SMBus)	TC1 B0	USART1 CTS	USART1 CLK	TWIMS0 TWCK	PWMA PWMA[28]	PWMA PWMAOD[28]	SCIF GCLK[3]	CAT CSB[14]
30	PB06	38	VDDIO	Normal I/O	TC1 A1	USART3 TXD	ADCIFB AD[6]	GLOC IN[2]	PWMA PWMA[29]	ACIFB ACAN[3]	EIC EXTINT[0]	CAT CSB[13]
31	PB07	39	VDDIO	Normal I/O	TC1 B1	USART3 RXD	ADCIFB AD[7]	GLOC IN[1]	PWMA PWMA[30]	ACIFB ACAP[3]	EIC EXTINT[1]	CAT CSA[13]
32	PB08	40	VDDIO	Normal I/O	TC1 A2	USART3 RTS	ADCIFB AD[8]	GLOC IN[0]	PWMA PWMA[31]	CAT SYNC	EIC EXTINT[2]	CAT CSB[12]
29	PB09	41	VDDIO	Normal I/O	TC1 B2	USART3 CTS	USART3 CLK		PWMA PWMA[32]	ACIFB ACBN[1]	EIC EXTINT[3]	CAT CSB[15]
23	PB10	42	VDDIN	Normal I/O	TC1 CLK0	USART1 TXD	USART3 CLK	GLOC OUT[1]	PWMA PWMA[33]		EIC EXTINT[4]	CAT CSB[16]
44	PB11	43	VDDIO	Normal I/O	TC1 CLK1	USART1 RXD		ADCIFB TRIGGER	PWMA PWMA[34]	CAT VDIVEN	EIC EXTINT[5]	CAT CSA[16]
5	PB12	44	VDDIO	Normal I/O	TC1 CLK2		TWIMS1 TWALM	CAT SYNC	PWMA PWMA[35]	ACIFB ACBP[3]	SCIF GCLK[4]	CAT CSA[15]

See Section 3.3 for a description of the various peripheral signals.

Refer to "Electrical Characteristics" on page 770 for a description of the electrical properties of the pin types used.

3.2.1.1 *TWI, 5V Tolerant, and SMBUS Pins*

Some Normal I/O pins offer TWI, 5V Tolerant, and SMBUS features. These features are only available when either of the TWI functions or the PWMAOD function in the PWMA are selected for these pins.

Refer to the "TWI Pin Characteristics(1)" on page 778 for a description of the electrical properties of the TWI, 5V Tolerant, and SMBUS pins.

3.2.2 Peripheral Functions

Each GPIO line can be assigned to one of several peripheral functions. The following table describes how the various peripheral functions are selected. The last listed function has priority in case multiple functions are enabled on the same pin.

Table 3-2. Peripheral Functions

Function	Description
GPIO Controller Function multiplexing	GPIO and GPIO peripheral selection A to H
Nexus OCD AUX port connections	OCD trace system
aWire DATAOUT	aWire output in two-pin mode
JTAG port connections	JTAG debug port
Oscillators	OSC0, OSC32

3.2.3 JTAG Port Connections

If the JTAG is enabled, the JTAG will take control over a number of pins, irrespectively of the I/O Controller configuration.

Table 3-3. JTAG Pinout

48-pin	Pin Name	JTAG Pin
11	PA00	TCK
14	PA01	TMS
13	PA02	TDO
4	PA03	TDI

3.2.4 Nexus OCD AUX Port Connections

If the OCD trace system is enabled, the trace system will take control over a number of pins, irrespectively of the I/O Controller configuration. Two different OCD trace pin mappings are possible, depending on the configuration of the OCD AXS register. For details, see the AVR32 UC Technical Reference Manual.

Table 3-4. Nexus OCD AUX Port Connections

Pin	AXS=1	AXS=0
EVTI_N	PA05	PB08
MDO[5]	PA10	PB00
MDO[4]	PA18	PB04
MDO[3]	PA17	PB05
MDO[2]	PA16	PB03

Table 3-4. Nexus OCD AUX Port Connections

Pin	AXS=1	AXS=0
MDO[1]	PA15	PB02
MDO[0]	PA14	PB09
EVTO_N	PA04	PA04
MCKO	PA06	PB01
MSEO[1]	PA07	PB11
MSEO[0]	PA11	PB12

3.2.5 Oscillator Pinout

The oscillators are not mapped to the normal GPIO functions and their muxings are controlled by registers in the System Control Interface (SCIF). Please refer to the SCIF chapter for more information about this.

Table 3-5. Oscillator Pinout

48-pin	Pin	Oscillator Function
3	PA08	XIN0
46	PA10	XIN32
26	PA13	XIN32_2
2	PA09	XOUT0
47	PA12	XOUT32
25	PA20	XOUT32_2

3.2.6 Other Functions

The functions listed in Table 3-6 are not mapped to the normal GPIO functions. The aWire DATA pin will only be active after the aWire is enabled. The aWire DATAOUT pin will only be active after the aWire is enabled and the 2_PIN_MODE command has been sent. The WAKE_N pin is always enabled. Please refer to Section 6.1.4 on page 40 for constraints on the WAKE_N pin.

Table 3-6. Other Functions

48-pin	Pin	Function
27	PA11	WAKE_N
22	RESET_N	aWire DATA
11	PA00	aWire DATAOUT

3.3 Signal Descriptions

The following table gives details on signal name classified by peripheral.

Table 3-7. Signal Descriptions List

Signal Name	Function	Type	Active Level	Comments
Analog Comparator Interface - ACIFB				
ACAN3 - ACAN0	Negative inputs for comparators "A"	Analog		
ACAP3 - ACAP0	Positive inputs for comparators "A"	Analog		
ACBN3 - ACBN0	Negative inputs for comparators "B"	Analog		
ACBP3 - ACBP0	Positive inputs for comparators "B"	Analog		
ACREFN	Common negative reference	Analog		
ADC Interface - ADCIFB				
AD8 - AD0	Analog Signal	Analog		
ADP1 - ADP0	Drive Pin for resistive touch screen	Output		
TRIGGER	External trigger	Input		
aWire - AW				
DATA	aWire data	I/O		
DATAOUT	aWire data output for 2-pin mode	I/O		
Capacitive Touch Module - CAT				
CSA16 - CSA0	Capacitive Sense A	I/O		
CSB16 - CSB0	Capacitive Sense B	I/O		
DIS	Discharge current control	Analog		
SMP	SMP signal	Output		
SYNC	Synchronize signal	Input		
VDIVEN	Voltage divider enable	Output		
External Interrupt Controller - EIC				
NMI	Non-Maskable Interrupt	Input		
EXTINT5 - EXTINT1	External interrupt	Input		
Glue Logic Controller - GLOC				
IN7 - IN0	Inputs to lookup tables	Input		
OUT1 - OUT0	Outputs from lookup tables	Output		
JTAG module - JTAG				
TCK	Test Clock	Input		
TDI	Test Data In	Input		
TDO	Test Data Out	Output		
TMS	Test Mode Select	Input		

Table 3-7. Signal Descriptions List

Power Manager - PM				
RESET_N	Reset	Input	Low	
Pulse Width Modulation Controller - PWMA				
PWMA35 - PWMA0	PWMA channel waveforms	Output		
PWMAOD35 - PWMAOD0	PWMA channel waveforms, open drain mode	Output		Not all channels support open drain mode
System Control Interface - SCIF				
GCLK4 - GCLK0	Generic Clock Output	Output		
RC32OUT	RC32K output at startup	Output		
XIN0	Crystal 0 Input	Analog/ Digital		
XIN32	Crystal 32 Input (primary location)	Analog/ Digital		
XIN32_2	Crystal 32 Input (secondary location)	Analog/ Digital		
XOUT0	Crystal 0 Output	Analog		
XOUT32	Crystal 32 Output (primary location)	Analog		
XOUT32_2	Crystal 32 Output (secondary location)	Analog		
Serial Peripheral Interface - SPI				
MISO	Master In Slave Out	I/O		
MOSI	Master Out Slave In	I/O		
NPCS3 - NPCS0	SPI Peripheral Chip Select	I/O	Low	
SCK	Clock	I/O		
Timer/Counter - TC0, TC1				
A0	Channel 0 Line A	I/O		
A1	Channel 1 Line A	I/O		
A2	Channel 2 Line A	I/O		
B0	Channel 0 Line B	I/O		
B1	Channel 1 Line B	I/O		
B2	Channel 2 Line B	I/O		
CLK0	Channel 0 External Clock Input	Input		
CLK1	Channel 1 External Clock Input	Input		
CLK2	Channel 2 External Clock Input	Input		
Two-wire Interface - TWIMS0, TWIMS1				
TWALM	SMBus SMBALERT	I/O	Low	
TWCK	Two-wire Serial Clock	I/O		
TWD	Two-wire Serial Data	I/O		
Universal Synchronous/Asynchronous Receiver/Transmitter - USART0, USART1, USART2, USART3				

Table 3-7. Signal Descriptions List

CLK	Clock	I/O		
CTS	Clear To Send	Input	Low	
RTS	Request To Send	Output	Low	
RXD	Receive Data	Input		
TXD	Transmit Data	Output		

Note: 1. ADCIFB: AD3 does not exist.

Table 3-8. Signal Description List, Continued

Signal Name	Function	Type	Active Level	Comments
Power				
VDDCORE	Core Power Supply / Voltage Regulator Output	Power Input/Output		1.62V to 1.98V
VDDIO	I/O Power Supply	Power Input		1.62V to 3.6V. VDDIO should always be equal to or lower than VDDIN.
VDDANA	Analog Power Supply	Power Input		1.62V to 1.98V
ADVREFP	Analog Reference Voltage	Power Input		1.62V to 1.98V
VDDIN	Voltage Regulator Input	Power Input		1.62V to 3.6V (1)
GNDANA	Analog Ground	Ground		
GND	Ground	Ground		
Auxiliary Port - AUX				
MCKO	Trace Data Output Clock	Output		
MDO5 - MDO0	Trace Data Output	Output		
MSEO1 - MSEO0	Trace Frame Control	Output		
EVTI_N	Event In	Input	Low	
EVTO_N	Event Out	Output	Low	
General Purpose I/O pin				
PA22 - PA00	Parallel I/O Controller I/O Port 0	I/O		
PB12 - PB00	Parallel I/O Controller I/O Port 1	I/O		

1. See Section 6.1 on page 36

3.4 I/O Line Considerations

3.4.1 JTAG Pins

The JTAG is enabled if TCK is low while the RESET_N pin is released. The TCK, TMS, and TDI pins have pull-up resistors when JTAG is enabled. The TCK pin always has pull-up enabled during reset. The TDO pin is an output, driven at VDDIO, and has no pull-up resistor. The JTAG pins can be used as GPIO pins and multiplexed with peripherals when the JTAG is disabled. Please refer to Section 3.2.3 on page 11 for the JTAG port connections.

3.4.2 PA00

Note that PA00 is multiplexed with TCK. PA00 GPIO function must only be used as output in the application.

3.4.3 RESET_N Pin

The RESET_N pin is a schmitt input and integrates a permanent pull-up resistor to VDDIN. As the product integrates a power-on reset detector, the RESET_N pin can be left unconnected in case no reset from the system needs to be applied to the product.

The RESET_N pin is also used for the aWire debug protocol. When the pin is used for debugging, it must not be driven by external circuitry.

3.4.4 TWI Pins PA21/PB04/PB05

When these pins are used for TWI, the pins are open-drain outputs with slew-rate limitation and inputs with spike filtering. When used as GPIO pins or used for other peripherals, the pins have the same characteristics as other GPIO pins. Selected pins are also SMBus compliant (refer to Section 3.2 on page 9). As required by the SMBus specification, these pins provide no leakage path to ground when the AT32UC3L016/32/64 is powered down. This allows other devices on the SMBus to continue communicating even though the AT32UC3L016/32/64 is not powered.

After reset a TWI function is selected on these pins instead of the GPIO. Please refer to the GPIO Module Configuration chapter for details.

3.4.5 TWI Pins PA05/PA07/PA17

When these pins are used for TWI, the pins are open-drain outputs with slew-rate limitation and inputs with spike filtering. When used as GPIO pins or used for other peripherals, the pins have the same characteristics as other GPIO pins.

After reset a TWI function is selected on these pins instead of the GPIO. Please refer to the GPIO Module Configuration chapter for details.

3.4.6 GPIO Pins

All the I/O lines integrate a pull-up resistor. Programming of this pull-up resistor is performed independently for each I/O line through the GPIO Controller. After reset, I/O lines default as inputs with pull-up resistors disabled, except PA00. PA20 selects SCIF-RC32OUT (GPIO Function F) as default enabled after reset.

3.4.7 High-Drive Pins

The five pins PA02, PA06, PA08, PA09, and PB01 have high-drive output capabilities. Refer to Section 32. on page 770 for electrical characteristics.

3.4.8 RC32OUT Pin

3.4.8.1 *Clock output at startup*

After power-up, the clock generated by the 32kHz RC oscillator (RC32K) will be output on PA20, even when the device is still reset by the Power-On Reset circuitry. This clock can be used by the system to start other devices or to clock a switching regulator to rise the power supply voltage up to an acceptable value.

The clock will be available on PA20, but will be disabled if one of the following conditions are true:

- PA20 is configured to use a GPIO function other than F (SCIF-RC32OUT)
- PA20 is configured as a General Purpose Input/Output (GPIO)
- The bit FRC32 in the Power Manager PPCR register is written to zero (refer to the Power Manager chapter)

The maximum amplitude of the clock signal will be defined by VDDIN.

Once the RC32K output on PA20 is disabled it can never be enabled again.

3.4.8.2 *XOUT32_2 function*

PA20 selects RC32OUT as default enabled after reset. This function is not automatically disabled when the user enables the XOUT32_2 function on PA20. This disturbs the oscillator and may result in the wrong frequency. To avoid this, RC32OUT must be disabled when XOUT32_2 is enabled.

3.4.9 ADC Input Pins

These pins are regular I/O pins powered from the VDDIO. However, when these pins are used for ADC inputs, the voltage applied to the pin must not exceed 1.98V. Internal circuitry ensures that the pin cannot be used as an analog input pin when the I/O drives to VDD. When the pins are not used for ADC inputs, the pins may be driven to the full I/O voltage range.

ANEXO 9.- ESPECIFICACIONES DEL ROUTER TL-WR740N

CARACTERÍSTICAS DE HARDWARE	
Interfaz	4 Puertos LAN 10/100Mbps 1 Puerto WAN 10/100Mbps
Botón	Botón WPS/Reset
Antena	Fija Omnidireccional de 5dBi
Fuente de Alimentación Externa	5VDC/0.6A
Estándares Inalámbricos	IEEE 802.11n*, IEEE 802.11g, IEEE 802.11b
Dimensiones (W X D X H)	6.9 x 4.6 x 1.3 pulgadas. (174 x 118 x 33 mm)
CARACTERÍSTICAS INALÁMBRICAS	
Frequency	2.4-2.4835GHz
Signal Rate	11n: Hasta 150Mbps(dinámica) 11g: Hasta 54Mbps(dinámica) 11b: Hasta 11Mbps(dinámica)
Reception Sensitivity	130M: -68dBm@10% PER 108M: -68dBm@10% PER 54M: -68dBm@10% PER 11M: -85dBm@8% PER 6M: -88dBm@10% PER 1M: -90dBm@8% PER
Transmit Power	CE: <20dBm(2.4GHz) FCC: <30dBm
Wireless Functions	Habilitar /Deshabilitar Radio Inalámbrica, Puente WDS , WMM, Estadísticas Inalámbricas
Wireless Security	64/128/152-bit WEP / WPA / WPA2,WPA-PSK / WPA2-PSK
CARACTERÍSTICAS DE SOFTWARE	
Quality of Service	WMM, Control de Banda Ancha
WAN Type	IP Dinámica/IP Estática/PPPoE/ PPTP(Acceso dual)/L2TP(Acceso Dual)/BigPond
Management	Control de Acceso Administración Local Administración Remota
DHCP	Servidor, Cliente, Lista de Cliente DHCP, Reservación de Direcciones
Port Forwarding	Servidor Virtual,Puerto Triggering, UPnP, DMZ
Dynamic DNS	DynDns, Comexe, NO-IP
VPN Pass-Through	PPTP, L2TP, IPSec (ESP Head)

Access Control	Control Parental , Control de Admnistración Local , Lista de Host, Acceso a Agenda, Reglas de Administración
Firewall Security	DoS, SPI Firewall IP Address Filter/MAC Address Filter/Domain Filter IP and MAC Address Binding
Protocols	Support IPv4 and IPv6
Guest Network	2.4GHz Guest Network x1
OTHERS	
Certification	CE, FCC, RoHS
Package Contents	TL-WR740N 1 antena fija omnidireccional Fuente de Alimentación CD de Instalación Cable Ethernet Guía de Instalación Rápida
System Requirements	Microsoft® Windows® 98SE, NT, 2000, XP, Vista™ or Windows 7, Windows8/ 8.1/10 MAC® OS, NetWare®, UNIX® or Linux
Environment	Temperatura de Funcionamiento: 0℃~40℃ (32°F~104°F) Temperatura de Almacenamiento: -40℃~70℃ (-40°F~158°F) Humedad de Funcionamiento: 10%~90% sin condensación Humedad de Almacenamiento: 5%~90% sin condensación