Revista Tekhnê
2012, Vol. 9, 33–40

ISSN 1692-8407

Ecualizador gráfico usando sistema de desarrollo con arquitectura ARM

Graphic equalizer using development system with ARM architecture

Yonathan O. Neita
Universidad Distrital Francisco José de Caldas
Yonathaneita@hotmail.com

Holman Montiel Ariza
Universidad Distrital Francisco José de Caldas
hmontiela@udistrital.edu.co

Edwar Jacinto Gómez
Universidad Distrital Francisco José de Caldas
ejacintog@udistrital.edu.co

En el presente artículo se desarrolla un ecualizador gráfico de cinco bandas utilizando un dispositivo con arquitectura ARM y Android 2.3 como sistema operativo. El ecualizador funciona indirectamente como un reproductor de audio. La aplicación diseñada hace uso de librerías nativas de Android, algunas de ellas funcionales a partir del API 9 (Versión 2.3). Para la interacción con el usuario, esta aplicación dispone de dos actividades, la principal donde se captura el audio y donde se establece la variación de la ganancia de cada una de las bandas del ecualizador, y la segunda que muestra los archivos con extensión de audio que se encuentren en la memoria externa del dispositivo.

Palabras clave: ARM, audio, ecualizador, filtros, sistema embebido, sistemas operativos

In this paper, a five-band graphic equalizer using a device with ARM architecture and Android 2.3 as operating system was developed. The equalizer works indirectly as an audio player. The application makes use of native Android libraries, some functional from API 9 (Version 2.3). For the interaction with the user, this application has two activities, the main where the audio is captured and where the variation of the gain of each of the equalizer bands is established, and the second one showing the files with audio format that are in the external memory of the device.

Keywords: ARM, audio, embedded system, equalizer, filters, operating systems

Introducción

Android es el sistema operativo móvil más usado en el mundo, ya existen más de 750 millones de dispositivos activados a nivel mundial. Los dispositivos más comunes que hacen uso de Android son: Sony Ericsson, Motorola, LG y Samsung (Portal TIC, 2014). Sus desarrollos van desde aplicaciones, hasta sistemas multimedia, sistemas Embebidos y telecomunicaciones (Bortot, 2007; Bravo, Cortiguera, y Quintás, 2009; Li, Liu, Liu, y Mei, 2011; Pareja, Falcón, Federico, Mandolesi, y Julián, 2011). Esto trae como consecuencia la tendencia a utilizar Android, Windows o Linux, como Sistemas Operativos (SO) (Lemus y Escobar, 2010; Manrique, 2012; Salinas y Barrero, 2013), con la ventaja de que el modelo de desarrollo de Android es software libre, el cual es el principal producto de la *Open Handset Alliance* (OHA)(OHA, 2007), compañías que se dedican a desarrollar estándares abiertos para dispositivos móviles. Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestre de 2010 (Whitney, 2010).

Fecha recepción del manuscrito: Octubre 17, 2012
Fecha aceptación del manuscrito: Diciembre 10, 2012

Yonathan O. Neita, Facultad Tecnológica, Universidad Distrital Francisco José de Caldas; Holman Montiel Ariza, Facultad Tecnológica, Universidad Distrital Francisco José de Caldas; Edwar Jacinto Gómez, Facultad Tecnológica, Universidad Distrital Francisco José de Caldas.

Esta investigación fue financiada por: Universidad Distrital Francisco José de Caldas.

Correspondencia en relación con el artículo debe ser enviada a: Yonathan O. Neita. Email: Yonathaneita@hotmail.com

El software juega un papel muy importante y fundamental en el desarrollo de varios procesos, como el de negocios de una compañía, tanto para el desarrollador como para el usuario. Es tan importante que un gran grupo de personas se dedican al desarrollo de *plugins* o herramientas que facilitan y agilizan el proceso de desarrollo, *plugins* que se desarrollan en entornos como Eclipse, y para dicho entorno (García-Blanes, 2012; Ochoa, 2006). El objetivo del trabajo es utilizar esta herramienta para desarrollar una aplicación, haciendo uso de *Plugins* creados para Eclipse. De esa manera, contando con todas las facilidades que agilizan los procesos, se desarrolla un sistema de ecualizado utilizando un dispositivo con arquitectura ARM como mecanismo de ejecución e interacción con el usuario.

La instalación de un SO en una tarjeta de desarrollo es un procedimiento sencillo pero de mucho cuidado con los diferentes archivos que se necesitan para la instalación. Por lo general los suministra el fabricante con un manual de inicio rápido donde se especifica el procedimiento para hacerlo. Primero es necesaria una memoria externa tipo SD-CARD donde se guardan los archivos correspondientes según el fabricante, esto por ejemplo en el caso de usar la tarjeta de desarrollo S3C6410 V6.0.

La aplicación cosiste indirectamente en un reproductor de audio con ecualizador gráfico de cinco bandas donde el usuario pueden ajustar la ganancia de cinco frecuencias seleccionadas en el rango de audibles(60 Hz a 14 KHz) y puedan tener una mejor reproducción de sus archivos de una forma sencilla y rápida por medio de la aplicación, la lectura de las pistas de reproducción pueden ser .mp3, .midi, .wav, wma, .cda, .ogg, .ogm, .aac, .ac3, ó .flac y la ruta de acceso siempre está en la memoria externa del dispositivo.

La escogencia de las frecuencias van establecidas con respecto a las normas ISO que especifica las frecuencias centrales a emplear de acuerdo al ancho de banda con el que se trabaje, por lo que se dividen según la octava (intervalos entre frecuencias). En el presente trabajo el ancho de banda va hacer mucho más amplio, es decir la mitad de las bandas al trabajar a intervalos de una octava, de esta manera las cinco frecuencias serán: 63 Hz, 250 Hz, 900 Hz, 3550 Hz y 14000 Hz (ver Fig. 1).

La función del ecualizador permite mejorar la calidad del audio a criterio del usuario, o utilizarlo para fines que sean necesarios al procesamiento de sonido multicanal. Los ecualizadores pueden ser implementados para diferentes propósitos (Fanco, Quintero, y Ardila, 2010; Mendicute, Sobrón, y Prieto, 2008).

Planteamiento del problema

Archivos como grabaciones de voz o música, muchas veces presentan imperfecciones acústicas, tales como, ruidos, zumbidos o frecuencias molestas, además de hacerse necesario evidenciar la respuesta en frecuencia para determinadas necesidades.

Con un reproductor básico no se logra el hecho de acondicionar un tipo de música según el género o al gusto del usuario para lograr diferentes efectos como lo pueden ser: filtrar voces, variar el sonido de determinado instrumento o adecuar el sonido a audífonos o altavoces.

Frec.	1	1/2	1/3	Frec.	1	1/2	1/3	Frec.	1	1/2	1/3
16	*	*	*	160			*	1.600			*
18				180		*		1.800			
20			*	200			*	2.000	*	*	*
22.4		*		224				2.240			
25			*	250	*	*	*	2.500			*
28				280				2.800		*	
31.5	*	*	*	315			*	3.150			*
35.5				355		*		3.550			
40			*	400			*	4.000	*	*	*
45		*		450				4.500			
50			*	500	*	*	*	5.000			*
56				560				5.600		*	
63	*	*	*	630			*	6.300			*
71				710		*		7.100			
80			*	800			*	8.000	*	*	*
90		*		900				9.000			
100			*	1.000	*	*	*	10.000			*
112				1.120				11.200		*	
125	*	*	*	1.250			*	12.500			*
140				1.400		*		14.000			
160			*	1.600			*	16.000	*	*	*
								18.000			(*)

Figura 1. Tabla de frecuencias ISO (ISO, 2013).

Metodología

Para poder realizar una aplicación en Android se debe contar con un entorno que permita realizar el código fuente y estructurarlo, para esto se utilizó la plataforma Eclipse la cual permite realizar dicho código en lenguaje Java. Como primeros pasos se debe instalar algunos programas o *drivers* como el JDK de Android o Java de Oracle, descargar las versiones de Android y configurar la máquina virtual Android que permite ver el funcionamiento y el desarrollo de la aplicación de forma rápida.

Luego de la instalación del software se debe seguir una serie de pasos para el correcto funcionamiento sobre el ordenador donde se esté realizando el desarrollo de la aplicación, el procedimiento completo se puede encontrar en la página de desarrolladores de Android o en libros de desarrollo (Android, 2013; Belmonte, Granell, y Erdozain, 2012).

Sistema de filtrado

El sistema de ecualizado debe ser capaz de modificar la respuesta en frecuencias específicas, y de variar la amplitud en dB de cada una de ellas entre -15 dB y 15 dB en tiempo real, de una señal en tiempo discreto la cual viene dada por el vector correspondiente al archivo de audio.

Los métodos, tipos de parámetros y unidades expuestas por la implementación del ecualizador, están directamente definidos por el OpenSL ES (Khronos, 2013).

Para el diseño del filtro de tipo *Finite Impulse Response* (FIR) se tiene en cuenta que este tipo de filtro obtiene su

respuesta a partir de las entradas presentes y anteriores, para un filtro de orden N se tiene (ecuaciones 1 y 2):

$$Y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_{N-1} x(n-N+1) \quad (1)$$

$$Y(n) = \sum_{k=0}^{N-1} b_k x(n-k) \quad (2)$$

Ahora, frente al estímulo impulso, la respuesta $Y(n)$ se puede escribir como la convolución entre dicha respuesta y la entrada $x(n\text{-}k)$, por lo tanto la expresión puede reescribirse como (ecuación 3):

$$Y(n) = \sum_{k=0}^{N-1} h(k) \times x(n-k) \quad (3)$$

Donde $h(k)$ corresponde a los coeficientes de la respuesta impulso (Martínez, Gómez, Gómez, y Vila, 2010).

En la ecuación 4 se muestra la transformada Z a la expresión 3, la cual juega el mismo papel en procesado digital de señales que la Transformada de Laplace en análisis de sistemas continuos, lo que permite representar analizar y diseñar señales de sistemas discretos en el tiempo.

$$H(z) = \sum_{k=0}^{N-1} h_k z^{-k} = h_0 + h_1 z^{-1} + \cdots + h_{N-1} z^{-(N-1)} \quad (4)$$

Esta transformada será útil para el análisis en frecuencia, la ecuación 4 se puede escribir como (ecuaciones 5 y 6):

$$H(w) = h_0 + h_1 e^{-iw} + \cdots + h_{n-1} e^{-i(n-1)w} \quad (5)$$

$$H(w) = \sum_{k=0}^{n-1} h_t cos(tw) - \sum_{k=0}^{n-1} h_t sin(tw) \quad (6)$$

La respuesta en frecuencia es la evaluación de la Transformada Z en la circunferencia de radio unidad. Para obtener la respuesta en frecuencia, la transformada debe converger en dicha circunferencia, la definición de esta respuesta se muestra en la ecuación 5.

La respuesta en frecuencia modifica la amplitud y fase de la señal, y por ende se tendrán estas dos representaciones (Soria, 2005).

El tipo de atenuaciones máximas y mínimas se obtienen a partir del rizo (λ_1 y λ_2), siendo estos valores la mínima y máxima amplitud requerida, en las ecuaciones 7 y 8 se muestran las relaciones de atenuación con la magnitud de la respuesta en frecuencia.

$$\frac{1}{\lambda_1} \leq |H(w)| \leq \lambda_1 \quad (7)$$

$$|H(w)| \leq \lambda_2 \quad (8)$$

Y el valor en decibeles:

$$\text{Máximo: } 20\, log_{10}\, \lambda_1 \ [dB] \quad (9)$$

$$\text{Mínimo: -20 } log_{10}\, \lambda_2 \ [dB] \quad (10)$$

La estructura básica de un filtro tipo FIR es como se muestra en la Fig. 2, este tipo de filtros son estables en sentido lineal, causal e invariantes en el tiempo.

Las frecuencias centrales (bandas) para las cuales está definido cada uno de los filtros del ecualizador son: 63 Hz, 250 Hz, 900 Hz, 3550 Hz y 14000 Hz.

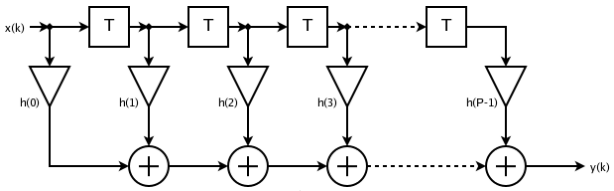

Figura 2. Estructura FIR (Duiops, 2013).

Haciendo uso de la herramienta CAD (FIRToolbox) de la empresa Mediatronix, se obtienen los coeficientes del filtro. En la Fig. 3 se muestra un filtro pasa banda de frecuencia central 14 KHz, orden 50 y frecuencia de muestro de 48 KHz la cual está configurada por defecto en el módulo de audio. Para las demás bandas se efectúa el mismo procedimiento.

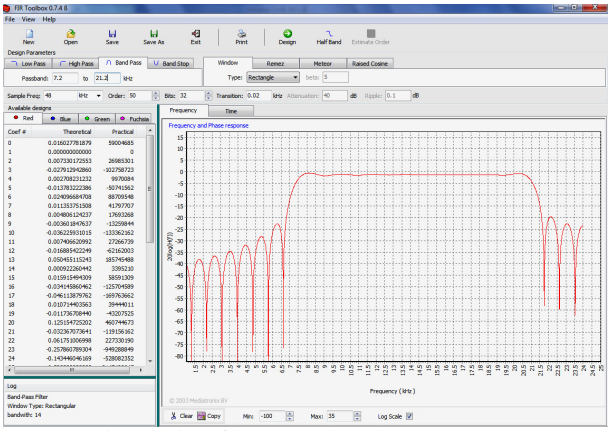

Figura 3. Simulación filtro pasa banda con Fc = 14 KHz.

Para la reproducción del audio se hace uso del módulo de audio con el que cuenta la tarjeta de desarrollo. El integrado WM9713, el cual tiene un conversor análogo digital (DAC) estéreo que logra la reproducción de audio de alta calidad a un bajo consumo. El volumen de la señal de salida del DAC es controlado por un PGA (amplificador de ganancia programable), este ofrece controles separados para graves y agudos con ganancias programables y características de

filtros. Esta función opera en datos de audio digital antes de que se pase a los DACs de audio. En la Fig. 4 se muestra el diagrama de bloques de dicho integrado.

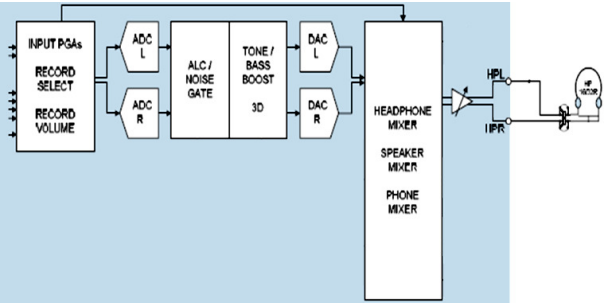

Figura 4. Diagrama de bloques del circuito integrado WM9713 (Cirrus Logic, 2011).

Arquitectura Android

Android tiene una arquitectura con estructura en capas, donde la base es una versión modificada del núcleo de Linux que tiene como objetivos, el primero la manipulación del hardware por medio de drivers, y el segundo el poder acceder al hardware de las capas posteriores de manera regulada. A continuación se cuenta con una capa formada por un conjunto de librerías.

La capa principal consiste en una máquina virtual denominada *Dalvik* la cual se encarga de la compilación de las aplicaciones, delegando en el sistema operativo el aislamiento de procesos. Finalmente existe una capa denominada *Framework*, que es la encargada de hacer posible el manejo de las librerías nativas de Android utilizando lenguaje Java (ver Fig. 5).

Figura 5. Arquitectura Android (Lemus y Escobar, 2010).

Máquina Virtual

Haciendo uso de la herramienta Android Virtual Device Manager (AVD) en Eclipse, se crea un nuevo dispositivo virtual con los parámetros que se deseen. Para el desarrollo del ecualizador gráfico se usó la versión 2.3 de Android, ya que algunas librerías no están disponibles para todas las versiones, y además porque es la versión más comercial en la actualidad.

Los parámetros del dispositivo en dicha máquina virtual son: Pantalla de 7 pulgadas, RAM de 800 MHz, sistema ARM y una memoria externa de 2 GB (Fig. 6).

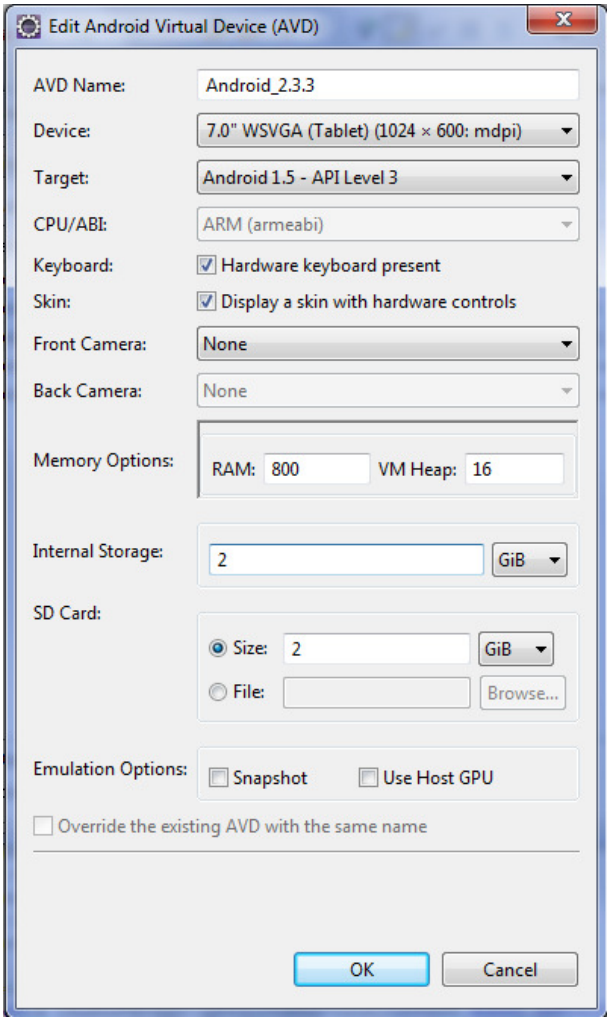

Figura 6. Máquina virtual para la versión 2.3 de Android.

Diagrama código fuente

Todo el desarrollo del código fuente está escrito en lenguaje java sobre Eclipse como entorno integrado, la elaboración de la aplicación consiste en dos actividades, la principal donde se encuentran las funciones de reproducción y ecualización, la segunda donde se encuentra la lista de

archivos de audio, esto conlleva a la idea de una proyección general de la actividad (ver figura 8), en esta podemos ver las barras de selección de volumen y ganancia de cada una de las frecuencias y los botones del reproductor, también se encuentra un botón en la parte superior derecha el cual sirve para visualizar la lista de archivos de audio que se encuentran en la raíz de la SD-CARD.

El desarrollo de la programación de la aplicación es de manera secuencial. La selección de archivos se basa en el filtrado, de tal forma que no se puedan visualizar archivos con extensiones diferentes a formatos de audio anteriormente mencionados. Por lo tanto es una selección sencilla de pista y reproducción, como se resume en las Fig. 7 y 8.

Figura 7. Proyección del entorno.

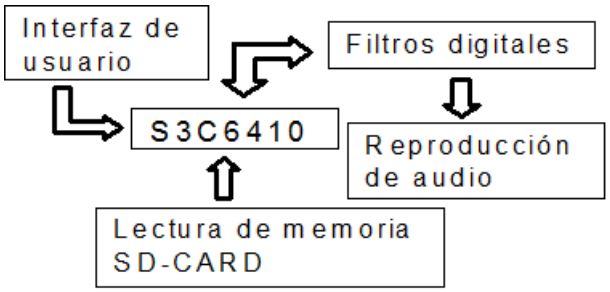

Figura 8. Diagrama de bloques.

- Interfaz: Se dispone de seis barras modificables para intensidad de volumen y ganancia de cada una de las cinco frecuencias y tres botones que sirven para pausar, reproducir y seleccionar la pista de audio de los archivos que se encuentren en la raíz de la SD-CARD.
- Lectura SD-CARD: Los archivos de audio que pueden ser visualizados en la lista de la segunda actividad corresponderán a la ruta de la memoria externa con extensiones de audio anteriormente mencionadas.
- Filtros: Bloque encargado de recibir las variaciones establecidas por el usuario por medio de la Interfaz para luego aplicarlas a la pista de reproducción actual, dichas variaciones pueden ser realizadas en cinco frecuencias específicas mencionadas anteriormente.
- Reproducción de audio: La reproducción de audio se efectúa automáticamente al iniciar la aplicación, que por defecto es el primer archivo que se encuentra en la lista; el usuario tiene la posibilidad de pausar y reanudar dicha reproducción.
- Manual - manejo del módulo de audio: El manual es diseñado con el fin de que las personas profundicen en la creación de aplicaciones basadas en Android, así como el manejo del módulo de audio, utilizando como ejemplo la creación de un Ecualizador gráfico de audio de cinco bandas.

Resultados

Máquina virtual

Las pruebas iniciales se hacen usando la máquina virtual, ya que el tiempo entre pruebas se reduce bastante. Además se pueden corregir errores los cuales también ocurrirán al momento de probar la aplicación en un dispositivo real.

Para evidenciar el funcionamiento el sistema, fue probado con varios archivos de música escogidos aleatoriamente. Dichos archivos se deben adicionar a la memoria SD-CARD, en este caso se debe tener configurada la máquina virtual para que reconozca una memoria externa del tamaño que se desee, para este trabajo de 2 GB.

El programa está encargado de filtrar los archivos que no posean una extensión de audio compatible con las mencionadas anteriormente, para comprobar este filtrado fue necesario adicionar a la memoria externa otro tipo de archivo con diferente extensión y probar que al momento de abrir la lista de archivos efectivamente el archivo que no es común sea filtrado y por ende no se muestre en dicha lista.

Con respecto al funcionamiento del reproductor y del filtrado de las diferentes frecuencias, se probó que si las barras de las frecuencias 63 Hz y 250 Hz están al máximo (15 dB) y las barras de las frecuencias 900 Hz, 3550 Hz y 14000 Hz están en el mínimo (-15 dB). Solamente suenan los bajos del archivo en reproducción y viceversa para el funcionamiento de los altos.

El funcionamiento mencionado anteriormente se evidencia en la siguiente sección donde se muestra un segmento de la señal original y el cambio que se produce al momento de modificar la ganancia en cada una de las bandas.

Dispositivo S3C6410 V6.0

No todos los errores o defectos de la aplicación se pueden evidenciar con la implementación de la misma en la máquina virtual. Para ello se instaló adecuadamente la aplicación en un dispositivo, y como primer paso se vio que diferencias de funcionamiento existen al momento del uso de la aplicación. Una de ellas fue el giro de la aplicación al momento de que el dispositivo reconoció que se cambió de posición, este reconocimiento lo hacen

muchos de los dispositivos haciendo uso de su acelerómetro. En este caso la aplicación fue diseñada verticalmente y si se gira el dispositivo horizontalmente la aplicación por defecto cambia su orientación, esto es un problema ya que el espacio que maneja la aplicación verticalmente no es el mismo que maneja horizontalmente. Para dar solución a este inconveniente se pueden tomar dos diferentes soluciones, la primera es hacer un diseño de aplicación diferente para cada una de las posiciones, es decir un diseño de interfaz para cuando este vertical y otra interfaz para cuando este horizontal y enseguida direccionar el programa para que tome el diseño adecuado dependiendo de la posición del dispositivo. La segunda es hacer que el programa bloquee el cambio de orientación de la aplicación, es decir que si el dispositivo cambia de orientación, la aplicación permanezca siempre en su orientación original.

Específicamente para la aplicación de este ecualizador se escogió la segunda opción para que la aplicación siempre quede en orientación vertical, ya que en la orientación horizontal es complicada la ubicación de las barras, por el espacio que necesitan y por comodidades al momento de analizar cada una de las variables del programa.

Se realizaron diferentes pruebas con veinte archivos de audio, tomando una cantidad de muestras significativas por un periodo de tiempo, evidenciando el cambio que se genera al momento de modificar la ganancia en cada una de las bandas, en las Fig. 9, 10, 11, 12 y 13 se muestran las variaciones de la señal original con respecto a las señales que fueron modificadas por el acondicionamiento de los filtros a 250 Hz y a 14000 Hz entre -15 dB y 15 dB.

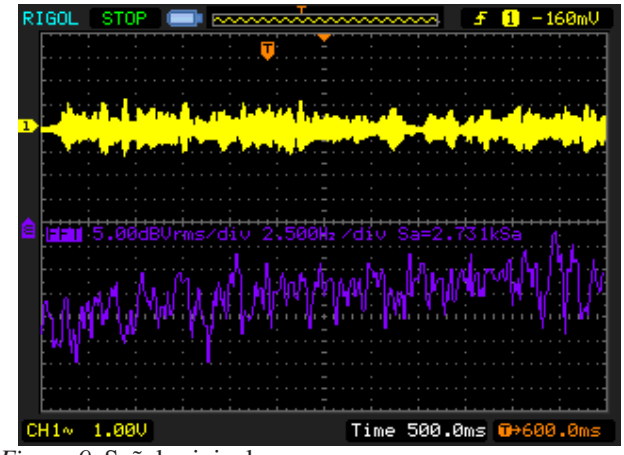

Figura 9. Señal original.

La pista que presentó de mejor manera las variaciones fue *Aherusia- Archangels.mp3*, ya que el inicio de la pista es solamente instrumental y se logran apreciar las diferencias en la reproducción del archivo procesado.

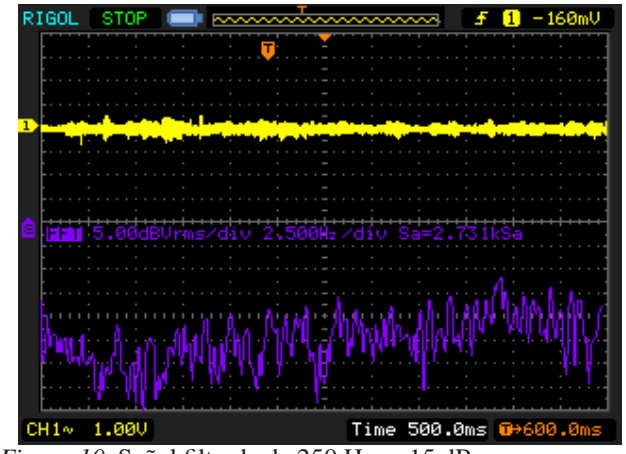

Figura 10. Señal filtrada de 250 Hz a -15 dB.

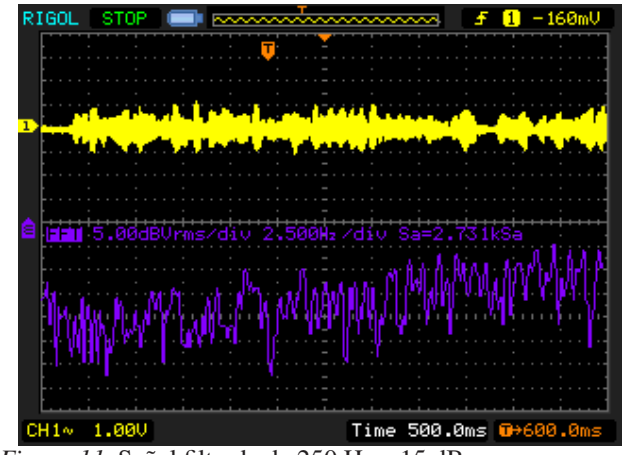

Figura 11. Señal filtrada de 250 Hz a 15 dB.

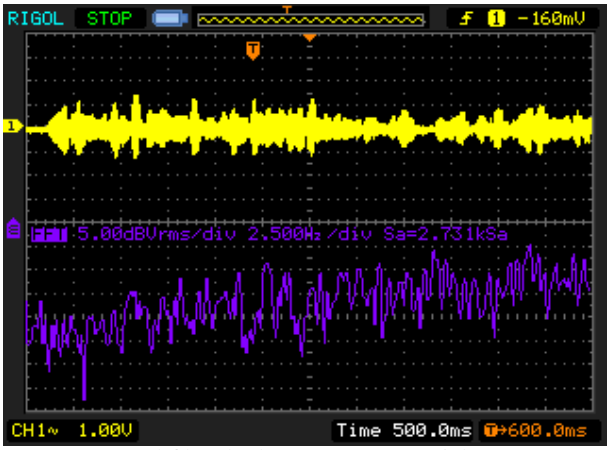

Figura 12. Señal filtrada de 14000 Hz a -15 dB.

Futuras mejoras

Como futura mejora se propone desarrollar un *Widget* (*Spinner*) para que el usuario pueda seleccionar entre los diferentes géneros musicales, ya que cada uno de

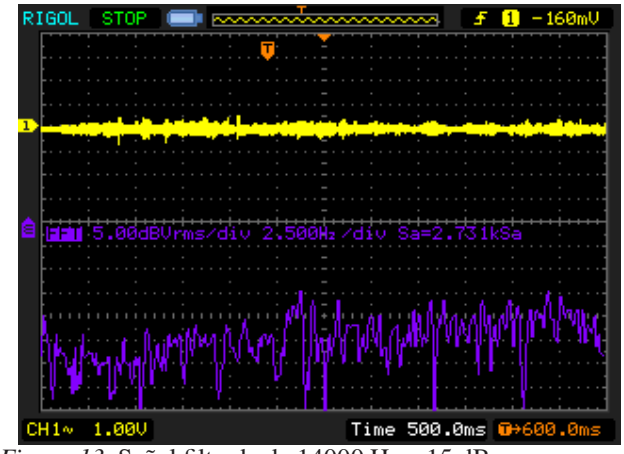

Figura 13. Señal filtrada de 14000 Hz a 15 dB.

estos maneja instrumentos diferentes y por tanto las variaciones de frecuencia también lo serán, al poder ajustar automáticamente las ganancias de las frecuencias dependiendo del género el usuario podrá tener una mejor apreciación de la música sin necesidad de que el mismo se encargue de ajustar cada una de las barras; también se espera tomar las muestras de sonido en tiempo real de la voz (grabación) para luego ecualizarla y graficar las diferentes amplitudes en cada una de las bandas.

Para ecualizar dicha grabación en tiempo real será necesario ajustar la entrada de audio del dispositivo al espacio acústico para eliminar ruidos o armónicos no deseados y que disminuye la calidad del audio, para graficar las diferentes amplitudes se debe sincronizar la reproducción actual del dispositivo con el entorno gráfico para pintar las variaciones en cada una de las bandas.

Conclusiones

En el presente artículo se presentó la implementación de un ecualizador gráfico de audio. Este tipo de aplicación resulta particularmente practica para la reproducción de archivos de audio en frecuencias especificar a criterio del usuario, de dicha manera estos parámetros son modificables.

La aplicación no solo es funcional en la tarjeta de desarrollo sino en cualquier dispositivo con SO Android Versión 2.3 y posterior, pero no para dispositivos con una versión anterior a esta, ya que algunas librerías son funcionales a partir de esta versión.

Esta aplicación sirve como guía de aprendizaje para el desarrollo de aplicaciones Android y arquitecturas ARM o sistemas embebidos que se puedan llevar a cabo en los ámbitos educativos principalmente en la Universidad Distrital Francisco José de Caldas.

Referencias

Android. (2013). *Android developers.* Descargado de `http://developer.android.com`

Belmonte, O., Granell, C., y Erdozain, M. (2012). *Desarrollo de proyectos informáticos con tecnología Java*. Commons Atribución-NoComercial-CompartirIgual 3.0 Unported.

Bortot, P. (2007). *Implementación y desarrollo de un sistema de medición de flujo, comprensibilidad y densidad de gas multicomponente en un procesador ARM y un microcontrolador* (Tesis de Master no publicada). Universidad Central de Venezuela.

Bravo, A., Cortiguera, H., y Quintás, J. (2009). *Minix 3 sobre arquitectura ARM* (Tesis de Master no publicada). Universidad Complutense de Madrid.

Cirrus Logic. (2011). Datasheet wm9713 (3.3 ed.) [Manual de software informático].

Duiops. (2013). *Filtros FIR.* Descargado de `http://www.duiops.net/hifi/enciclopedia/images/FIR_estr.gif`

Fanco, R., Quintero, E., y Ardila, W. (2010). Ecualizador de 3 bandas basado en filtros activos de segundo orden. *Scientia Et Technica*, *16*(45), 124-129.

García-Blanes, I. (2012). *Diseño y evaluación de un complemento de refactorización de C++ secuencial a Intel TBB para el entorno Eclipse* (Tesis de Master no publicada). Univeridad Carlos III de Madrid.

ISO. (2013). *Tabla de frecuencias iso.* Descargado de `www.vu-rms.com/apuntes/EQs.pdf`

Khronos. (2013). *La norma para la aceleración de audio embebido.* Descargado de `www.khronos.org/opensles/`

Lemus, C., y Escobar, J. (2010). Desarrollo de aplicaciones para tecnologías móviles. *Revista Tecnológica*, 34-36. (ITCA-FEPADE)

Li, Q., Liu, Y., Liu, Y., y Mei, Y. (2011). The design and implementation of audio analyzer based on ARM. En *2011 international conference on mechatronics and automation (icma)* (p. 2238-2242).

Manrique, A. (2012). *Implementación de una aplicación móvil sobre sistema operativo Android para la gestión remota del brazo mitsubishi RV-M1 para el grupo de investigación Teletecno.* (Universidad Distrital Francisco José de Caldas, Ingeniería Telecomunicaciones)

Martínez, M., Gómez, L., Gómez, A., J. Serrano, y Vila, J. (2010). *Filtros digitales.* (Notas de curso)

Mendicute, M., Sobrón, I., y Prieto, R., P. Isasi. (2008). Design, simulation and implementation of a channel equalizer for dvb-t on-channel repeaters. En *3rd iaria international conference on systems and networks communications* (p. 11-16).

Ochoa, J. (2006). *Módulo de toma de manejo de tiempos*

para el proyecto de Plugin de Eclipse para Changeset (Tesis de Master no publicada). Universidad de los Andes.

OHA. (2007). *Industry leaders announce open platform for mobile devices.* Descargado de www.openhandsetalliance.com/press_110507.html

Pareja, P., Falcón, A., Federico, M., Mandolesi, P., y Julián, P. (2011). Sistemas de medición remoto basado en dispositivos FPGA. En *Congreso argentino de sistemas embebidos case* (p. 5-9).

Portal TIC. (2014). *Android o ios: ¿quién dominó el mercado de los smartphones en 2014?* Descargado de www.europapress.es/portaltic/software/noticia-android-ios-quien-domino-mercado_smartphones-2014-20150225175533.html

Salinas, H., y Barrero, A. (2013). *Desarrollo de una aplicación en plataforma Android para la manipulación del brazo robótico Mitsubishi RV-M1 vía inalámbrica.* (Universidad Distrital Francisco José de Caldas, Tecnología Electrónica)

Soria, E. (2005). *Introducción al procesado digital de señales.* (Notas de clase)

Whitney, L. (2010). *Android hits top spot in U.S. smartphone maket.* Descargado de www.cnet.com/news/android-hits-top-spot-in-u-s-smartphone_market/