Sistema de reproducción de archivos mp3

César Ávila Sánchez

Escuela de Ingeniería Electrónica

TEC

Cartago, Costa Rica

cesaravila22@gmail.com

Fabricio Quirós Corella

Escuela de Ingeniería Electrónica

TEC

Cartago, Costa Rica

fabisqc0207@gmail.com

***Resumen***—**El presente documento pretende abarcar todo el proceso de desarrollo ejecutado y los resultados obtenidos por el grupo de trabajo, con el objeto de implementar un sistema capaz de reproducir archivos de audio, con extensión .mp3, tanto de manera local en un ambiente servidor, como en uno remoto o cliente.**

Palabras Claves—BeagleBoard-xM, GStreamer, Pipeline, Qt Creator.

# INTRODUCCIÓN

Como el título de este texto bien lo menciona, asimismo se aclara en el inciso de Resumen respectivo, el objetivo de todo este proyecto básicamente es crear una aplicación que permita la reproducción de audio del tipo mp3, lo cual solamente es posible con un programa que sea capaz de manipular todo los recursos multimedia que incluya una computadora o algún sistema embebido en específico, como lo es, dentro de este contexto, la BeagleBoard-xM.

Es por esta razón que, para efectos de este proyecto, se recurre a una herramienta de software en particular, un framework multimedia, el cual es de carácter libre, que soporta una gama importante de sistemas operativos, procesadores y compiladores, denominado GStreamer, cuya principal funcionalidad consiste en la creación de aplicaciones audiovisuales de todo tipo, tales como aquellas destinadas a la visualización de un video, inclusive reproducción remota a la codificación de cierto tipo de archivos de audio, hasta las que permite efectuar la mezcla de audio y video, por mencionar algunos ejemplos.

Entre las principales características que posee el GStreamer, caben destacar las siguientes: presenta una biblioteca de núcleo compacta que no sobrepasa los 500KB, con aproximadamente 65 000 líneas de código, así como un trasiego de datos extremadamente ligero, lo que implica un alto rendimiento y una acumulación de retardos reducida. De la misma manera, facilita una interfaz de programación de aplicaciones (en inglés, API: *Application Programming Interface*) bastante simple y estable, cuyo acceso es posible realizarlo desde una cantidad considerable de lenguajes, con el objeto de permitir el desarrollo de aplicaciones y complementos (*plugins*),

Otra particularidad importante de la herramienta en cuestión consiste en el hecho que maneja una arquitectura inteligente de complementos, los cuales son cargados dinámicamente y proveen recursos para trabajar con una enorme cantidad de elementos multimedia, además de ello, es posible aumentar las capacidades y la funcionalidad del GStreamer, gracias a la inclusión de nuevos complementos, donde estos están organizados en distintas categorías, tal y como se muestra a continuación.

Primero, se tiene el *gst-plugins-base*, que constituye el conjunto de complementos básico, totalmente soportado por el GStreamer. En segundo lugar, se encuentra el *gst-plugins-good* que incluye todos aquellos complementos bien soportados que emplean licencias libres. Luego, está el *gst-plugins-ugly* que considera todos aquellos plugins que posiblemente presenten inconvenientes en su libre distribución. Y por último, se tiene la categoría de *gst-plugins-bad*, la cual contiene aquel conglomerado de complementos que aún no ha superado rigurosas especificaciones de calidad, impuestas por los desarrolladores de GStreamer.

Como una de las características fundamentales que posee este framework, y que es de suma utilidad a la hora de implementar alguna tarea multimedia en particular, corresponde a la capacidad de la utilización deuna ruta total conformada por una serie de elementos vinculados, los cuales son complementos del GStreamer, donde la denominada ruta constituye el *pipeline* de la función, donde existen infinidad de posibilidades de vinculaciones entre plugins; es decir, hay innumerables pipelines a los que se pueden recurrir y crear.

# DESCRIPCIÓN DEL DISEÑO

**Posibles ideas de esta sección: Incluir descripción y cometarios detallados sobre lo que debe hacer el sistema, lo cual es posible ejemplificarlo mediante un diagrama de bloques. Asimismo, se puede hablar sobre las herramientas que se utilizaron tanto de software como de hardware, también de ideas o planteamientos que fueron necesarios incluir para que se obtuvieran los resultados esperados, esto relativo a las herramientas. Esta sección no debería ser muy extensa y no confundir con la siguiente. De esto se podría encargar Fabricio.**

# PROCESO DE IMPLEMENTACIÓN

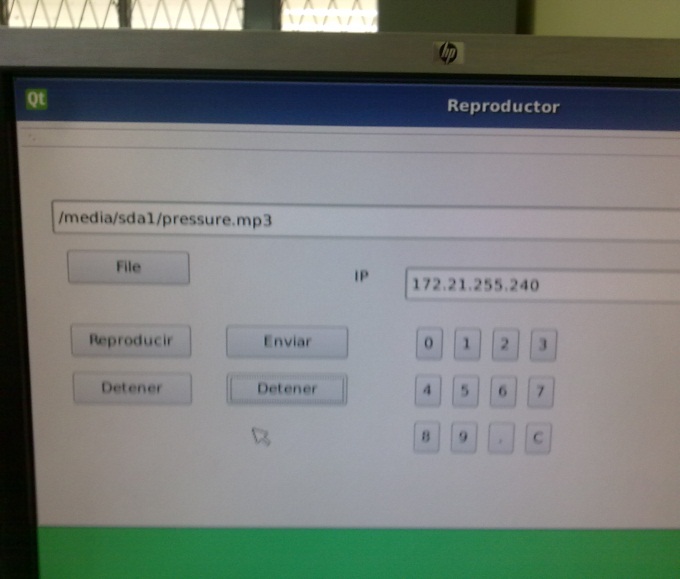
Para implementar nuestra aplicación inicialmente creamos un modelo de compilación cruzada. En este modelo establecimos que el sistema de archivos utilizado por la BeagleBoard-xM se encontraría en la computadora personal; de esta manera nos evitaríamos el problema de modificar continuamente la tarjeta SD. Para crear el sistema de archivos se ejecutó el script llamado setup.sh, el cual nos facilitó esta tarea.

A continuación, y como deseábamos reproducir formato mp3, debíamos asegurarnos que esto fuera posible en el BeagleBoard-xM. Instalamos para este caso gst-plugins-bad, asegurándonos de instalar también las dependencias de mad (mp3 decoder). Las cuales eran libmad y libid3tag.

Luego de encontrar el pipeline adecuado de Gstreamer para reproducir mp3, procedimos a implementarlo en C. Y a continuación buscamos el pipeline para enviar, vía UDP, un flujo de música. Haciendo uso del pipeline anterior y usando el mismo formato de implementación en C utilizado para la reproducción de mp3, logramos crear un programa capaz de enviar y otro capaz de recibir los datos de la canción seleccionada.

Sin embargo, para facilitar la interacción del usuario con el programa, se implementó una interfaz gráfica para ambos programas (tanto el servidor como el cliente). Estas interfaces se realizaron con Qtcreator.

El servidor, que se puede observar en la figura1, debía de contar con un buscador de archivos mp3 y un selector de IP al que se debía de enviar el flujo. Para tal fin se crearon botones que podrían seleccionarse a través de un mouse o una pantalla táctil que permitieran al usuario tanto seleccionar el archivo como digitar la IP del cliente.



*Figura1. Interfaz gráfica del servidor*

Por otro lado, para el cliente se creó una interfaz más simple que solo le permite indicar cuando desea recibir o cuando no los datos desde el servidor. Dicha interfaz se puede observar en la figura2.



*Figura2. Interfaz gráfica del cliente.*

Por último, se realizaron los pasos necesarios para que nuestro programa servidor se mostrara al encender la BeagleBoard-xM. Esto fue posible evitando que el script que iniciaba matrix-gui (Interfaz gráfica por defecto de nuestro ambiente de desarrollo) se ejecutara. Básicamente se logró renombrando dicho script ubicado en /etc/rc5.d. A continuación, con ayuda del comando ln, se linkeo un archivo llamado S99reproductor a un script creado por nosotros que ejecutaba el programa servidor. Es importante mencionar que la ubicación de los scripts que se utilizan en la Beagle se encuentran en /etc/init.d dentro del sistema de archivos. Para finalizar, el cliente solo se ubicaba dentro de la computadora para tal fin y se corría el programa creado.

# RESULTADOS OBTENIDOS

Al concluir nuestro proyecto se logró cumplir con las especificaciones y la ejecución de los comandos básicos: enviar, reproducir, detener, entre otros. Se cree que una aplicación de este tipo podría ser útil en un futuro, siempre y cuando se trabaje un poco más en los detalles (tales como manejo de errores e interfaz gráfica). Esto ya que la implementación es bastante sencilla y no fue “debuggeada” extensivamente.

Una de las limitantes encontradas en el programa fue que al ejecutar el pipeline, ya sea para enviar o para reproducir, el programa se quedaba ejecutando cierto proceso que al cerrar la aplicación se mantenía corriendo incluso si ya no se veía la interfaz gráfica.

Por otro lado, intentamos ejecutar nuestra aplicación en una pantalla HDMI y no lo logramos, solo fue posible probarla haciendo uso de un monitor con entrada DVI. En cuanto a esto, no estuvimos seguros de cuál fue la razón. Pero se cree tiene que ver de alguna manera con la resolución de la pantalla utilizada.

Para finalizar, en cuanto a la manera en que se oía la reproducción de la música, está fue bastante aceptable. Descubrimos que algunas veces al enviar vía UDP, se oía un tanto intermitente por cierto espacio de tiempo, pero no sucedía así siempre. Probablemente este detalle se deba a que existe algún problema de la red, ya que no se daba con frecuencia.

# CONCLUSIONES

**Posibles ideas: Esta sección es más que clara, así que no hay mucho que agregar al respecto, solamente decir que deben ser concisas y relacionadas con las observaciones realizadas en los resultados y concisas con los objetivos de este proyecto, los cuales, a mi criterio, fueron trabajar con el GStreamer, la Beagle, el Qt y el GStreamer, conocer todas las posibilidades de la Beagle y los ambientes de trabajo. Identificar todas implicaciones y particularidades del GStreamer y el Qt, y de aquí lo que sea. De esto se podría encargar Fabricio**

# Referencias

1. Cypress Semiconductor Corporation, “PSoC 5: CY8C55 Family Datasheet”, pp. 3–42, Febrero 2012.
2. Cypress Semiconductor Corporation, “PSoC Creator 2.1”, <http://www.cypress.com/?id=2494>, recuperado el 24 de noviembre del 2012.
3. Cypress Semiconductor Corporation, “PSoC Designer”, <http://www.cypress.com/?id=2522>, recuperado el 24 de noviembre del 2012.
4. Cypress Semiconductor Corporation, “PSoC Programmer 3.16”, <http://www.cypress.com/?id=2522>, recuperado el 24 de noviembre del 2012.
5. R. Jiménez, Universidad de Huelva, “Diseño de Sistemas Empotrados: Introducción” <http://www.uhu.es/raul.jimenez/EMPOTRADO/introduccion.pdf>, recuperdado el 20 de noviembre del 2012.