Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Desarrollo de una aplicación de referencia para transmisiones de multidifusión con la tarjeta LeopardBoard DM365

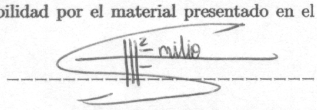
Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura

Marco Emilio Madrigal Solano

3 de mayo de 2011

Declaratoria de autenticidad

Por este medio declaro que el contenido del presente reporte final de proyecto de graduación ha sido realizado por mi persona, utilizando cuando ha sido necesario referencias bibliográficas y añadiendo aportes personales.
En aquellos casos que se hiciera uso de material o ideas referentes a la bibliografía, este se ha indicado de forma explícita mediante las correspondientes citas bibliográficas.
Como consecuencia de lo anterior asumo completa responsabilidad por el material presentado en el documento final del reporte final de proyecto de graduación.



Marco Emilio Madrigal Solano

1-1315-0600

INSTITUTO TECNOLOGICO DE COSTA RICA

ESCUELA DE INGENIERIA ELECTRONICA

PROYECTO DE GRADUACIÓN

TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Ing. William Marín Moreno

Profesor lector

Ing. Johan Carvajal Godínez

Profesor lector

Ing. Aníbal Coto Cortés

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, 28 de Abril, 2011

Resumen

El presente proyecto tiene como objetivo desarrollar un sistema de vigilancia de bebé que sirva de base para el desarrollo de programas que utilicen transmisiones multimedia por multidifusión. La solución presentada se basa en el uso de una tarjeta de evaluación LeopardBoard DM365 la cual posee un SoC TMS320DM365 de Texas Instrument.

En general el sistema desarrollado es capaz de medir el ruido ambiente y determinar cuándo su nivel presenta un aumento significativo en la banda de 3 a 4KHz. Al detectar dicha variación se realiza una multidifusión del audio y video desde el nodo servidor hacia uno o más nodos cliente.

Con el objetivo de llevar a cabo el proyecto, se hizo uso de la plataforma multimedia GStreamer para la codificación y decodificación del audio y el video, así como la transferencia de estos a través de un mecanismo multidifusión basado en los protocolos RTP/UDP/IP. Además se realizó el diseño e implementación de un sistema de análisis de audio basado en un filtro digital IIR elíptico de orden 10 y un algorítmo de FFT DIF radix-2. La aplicación se llevó a cabo con el uso de QT para el desarrollo de la interfáz gráfica y se hizo uso del demonio de GStreamer (GSTD) para la ejecución de las tuberías o *"pipelines"* de GStreamer, la comunicación entre ambas partes se realizó mediante DBus.

Los resultados obtenidos mostraron un desempeño adecuado al lograr obtener un reconocimiento de llanto en un 97 % de las pruebas y una cancelación del 100 % ante pruebas de diferentes ambientes comunes. Sin embargo se obtuvo un consumo de CPU de un 99 % al transmitir videos de escenarios en movimiento lo cual representa un punto crtítico en el desarrollo de aplicaciones más complejas o transmisiones de videos de ambientes con mayor movimiento.

Palabras clave: multidifusión, GStreamer, Fourier, filtro digital.

Abstract

This project's main objective is to develope a baby monitor system in order to be used as a starter point for the development of new programs that use multicast transmissions. The solution presented is based on the LeopardBoard DM365 evaluation board which uses an SoC TMS320DM365 from Texas Instrument.

The system developed is able to measure the local audio and detect when its level rises over a limit in the bandwith of 3 and 4 Khz. When this level is reached it will be started a multicast transmission of the audio and video from the server to one or more clients.

In order to finish the project, the GStreamer framework was used for the video and audio encoding/decoding and also for the network transmission using the RTP/UDP/IP protocols. Besides that, it was developed an audio analisys system based in the implementation of an IIR Ellyptic filter of order 10 and a FFT DIF radix-2 algorithm. The application was made by using QT for the GUI and GStreamer daemon (GSTD) for the pipeline setup, both of them interconnected throught DBus.

The results showed a good performance, getting a 97 % of match with crying bebies' sounds and a 100 % of cancellation for common environment's sounds. However, it was gotten a 99 % of CPU consuption when a video of a moving stage is transmitted, this is a critical point for more complex applications and videos that need the capture of video with more movement .

Keywords: multicast, GStreamer, Fourier, digital filter.

Agradecimientos

Este trabajo ha sido resultado principalmente del apoyo de mi familia; mi madre Rosalinda Solano Soto, mi padre Manuel Francisco Madrigal Córdoba quien a pesar de no encontrarse con nosotros al final de este proyecto siempre fue un apoyo incondicional, y mis hermanos Manuel Francisco Madrigal Solano y Rosibel Madrigal Solano. Todos ellos quienes sin duda alguna me ofrecieron siempre su apoyo y me enseñaron a ser quien ahora soy dejando una profunda huella en mi. Agradezco también a mi novia Andrea de los Angeles Vargas Rodríguez quien me apoyó en los buenos y malos momentos y que me alentó a seguir adelante en los instantes de flaqueza. Por otra parte agradezco al Ingeniero Aníbal Coto Cortés quien gentilmente se ofreció a ayudar en la guía y asesoramiento del proyecto. Por último agradezco a todos mis amigos y profesores que de una u otra forma confiaron en mi y me brindaron su apoyo a lo largo de mi recorrido hasta aquí.

No basta que rodees una gran montaña, súbela sin mirar abajo con paso firme que una vez arriba no solo tendrás el camino libre hacia adelante sino que también podrás ver lo que solo aquellos que se arriesgan y luchan ven al superar un obstáculo.

Marco Emilio Madrigal Solano
Cartago, 3 de mayo de 2011

A mi padre...

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Uso de transmisiones de multidifusión para la difusión de contenido multimedia

Hoy en día el uso creciente de dispositivos electrónicos y el avance en sus capacidades de procesamiento han hecho posible su utilización para aplicaciones de transmisión y reproducción de contenido multimedia. En muchos casos se ofrecen servicios de difusión de este tipo de contenido a una gran cantidad de usuarios, en este contexto se ha hecho necesario el desarrollo de métodos de difusión de contenidos multimedia que permitan distribuir un mismo conjunto de datos a varios clientes con un mínimo de complejidad y de forma eficiente.

El creciente desarrollo de aplicaciones y servicios de difusión multimedia ha llevado a la creación de métodos de transmisión masiva de contenidos tales como la *multidifusión* (*multicast*). Este método de transmisión de datagramas es uno de los más utilizados para la difusión de contenidos multimedia a multiples terminales debido a sus características de transmisión y asociación así como su fléxibilidad para el uso con diversos protocolos de red.

A pesar del gran potencial que ofrece el uso de métodos de transmisión de multidifusión, el conocimiento necesario para el desarrollo de aplicaciones que lo utilicen no es de fácil obtención. Es por esta razón que muchas empresas han optado por realizar investigaciones en dicha área con el fin de ofrecer el conocimiento adquirido como parte de sus servicios.

Actualmente para empresas dedicadas al desarrollo de programas para dispositivos electrónicos es de gran importancia el desarrollo de conocimiento en el campo de transmisiones de contenido multimedia ya que esto les permite ingresar en un nuevo mercado. Una de estas empresas la constituye RidgeRun Ltda., dedicada al desarrollo de aplicaciones para sistemas empotrados que utilizan los SoC de las familias Davinci y OMAP de Texas Intrument. Las exigencias de sus clientes ha llevado a la empresa a interesarse por la investigación en técnicas de difusión multimedia por red con el fin de cumplir las espectativas esperadas y ofrecer un conjuno de servicios más amplio en el desarrollo de sistemas empotrados.

Uno de los requerimientos de la empresa es la realización de una aplicación de referencia que permita obtener el conocimiento necesario en transmisiones multimedia por medio de multidifusión, esto permitiría su utilización como base en el desarrollo de proyectos que utilicen esta característica. Dicha aplicación consiste en un sistema de vigilancia de bebé que permite analizar el nivel de audio ambiente e iniciar una transmisión de multidifusión hacia uno o más nodos cliente una véz se detecte un aumento considerable en el nivel de ruido en la banda de 3 a 4KHz, considerando dicho evento como un posible llanto del niño.

En el presente documento se detalla el trabajo realizado, para el caso específico que se plantea se hace uso de una tarjeta de evaluación *LeopardBoard DM365* la cual cuenta con un SoC TMS320DM365, con un microprocesador ARM9 y aceleración por hardware para procesamiento de video lo cual lo convierte en una poderosa herramienta en el desarrollo de dispositivos multimedia.

1.2. Objetivo del proyecto

Desarrollar una aplicación de referencia que permita el desarrollo de aplicaciones de multidifusión con la tarjeta LeopardBoardDM365. Dicho programa deberá contar con las siguientes características:

- Detección de niveles de audio para señales en una banda de frecuencia entre los 3kHz y los 4kHz.
- Capacidad de asociación a un grupo específico de multidifusión.
- Capacidad de transmisión simultánea de audio y video en multidifusión.
- Estructura modular que permita la reutilización de código.

Además de lo descrito antes, se debe realizar un estudio de ancho de banda con el fin de determinar los requerimientos del canal de comunicación y los formatos y medios de transmisión a utilizar. Todo lo anterior deberá ser comprobado mediante un banco de pruebas que permita dar fe del correcto funcionamiento del sistema desarrollado.

1.3. Estructura del documento

En el Capítulo 2 se exponen las bases teóricas que atañen al desarrollo de la solución implementada, en el Capítulo 3 se muestra el proceso de implementación de la solución y se exponen sus principales características. En el Capítulo 4 se realiza un breve estudio de las principales mediciones realizadas al sistema implementado con el fin de verificar sus características de operación y contrarrestarlas con las especificaciones teóricas esperadas. En el Capítulo 5 se exponen las principales conclusiones y recomendaciones referentes al proyecto que permitan una posterior mejora del sistema.

Capítulo 2

Marco teórico

2.1. Principios de redes y multidifusión

En esta sección se realiza una breve descripción de los principales conceptos relacionados con redes de computadores y transmisiones de datos mediante multidifusión.

2.1.1. Protocolo IP para transmisión de datos en red

El protocolo IP es uno de los protocolos de la capa de red de mayor uso y más específicamente el protocolo IPv4. En dicho protocolo, cada dispositivo asociado a una misma red posee un número o dirección única dentro de la red que permite diferenciarlo del resto. La dirección IP se compone de cuatro octetos[1] separados por puntos que representan el número de la red y el número de computador cliente o más comúnmente llamado *host* [1]. En la figura 2.1 se muestra un esquema básico de la estructura de una dirección IPv4 en formato decimal con puntos. Cabe destacar que la distribución mostrada (2 octetos para el número de red y 2 octetos para el número de cliente) es solamente con motivo de ejemplo, el número de bits designados para la red y el cliente puede variar.

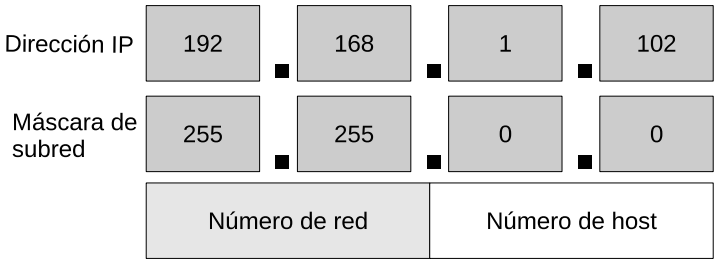


Figura 2.1: Estructura básica de una dirección IP.

Como puede observarse, los bits más significativos (no necesariamente octetos enteros) corresponden al número de red, un identificador necesario para saber si un paquete entrante o saliente corresponde a un emisor o receptor en la misma red o si corresponde a una red externa. Los bits restantes corresponden al número de cliente. La máscara de subred es de igual forma un conjunto de octetos que define qué sección de la dirección IP corresponde al número de subred y cual al número de cliente. De esta forma, una red puede tener como número de red 251.101.1 (con una máscara 255.255.255.0) y asignar los

[1] En redes es común designar a los bytes como octetos, por lo que una dirección IPv4 se compone de un total de 32 bits.

ocho bits menos significativos al número de host, es decir, un computador en dicha red puede tener la dirección IP 251.101.1.56. En el caso de ejemplo tratado, solamente se cuenta con ocho bits para asignar el número de host, esto permite tener como máximo 256 computadores en dicha red[2].

La cantidad de bits asignados en una dirección IP para el número de red y por ende, la cantidad máxima de computadores en dicha red depende directamente de la máscara de subred. Debe considerarse que actualmente se utiliza la metodología CIDR (*Classless Inter-Domain Routing*) la cual permite asignar cualquier cantidad de bits de la dirección IP para el número de red, esto permite utilizar de forma eficáz los números de direcciones en una red de acuerdo a la cantidad de computadores miembro.

Una excepción particular de direccionamiento es el utilizado para transmisiones de multidifusión, en dicho caso se hace uso del conjunto de direcciones IP entre 224.0.0.0 y 239.255.255.255.

2.1.2. Multidifusión

En una transmisión de red punto a punto o *unicast* existe un computador emisor que envía un paquete de datos a través de la red hacia otro computador receptor especificando su dirección IP, sin embargo existen casos donde es necesario enviar un mismo paquete de datos desde un servidor hacia uno o más computadores clientes, un ejemplo de esto es cuando se desea enviar audio o video simultáneamente a muchos computadores. En este contexto el uso de protocolos de transmisión convencionales se convierte en un problema, ya que por cada computador cliente sería necesario reenviar el mismo paquete de datos lo que se traduce en una mayor saturación de la red. Es en dichos casos donde surgen los términos de multidifusión IP *(multicast)* y difusión (*broadcast*).

En general multidifusión es el nombre que se le atribuye a un método de transmisión de datagramas que permite enviar datos desde un único dispositivo hacia múltiples clientes sin la necesidad de conocer quien recibe los datos [1], en la figura 2.2 se muestra la idea básica de una transmisión de multidifusión, en ésta puede observarse como los datos son dirigidos hacia todos los computadores clientes que se encuentran dentro del *grupo de multidifusión*

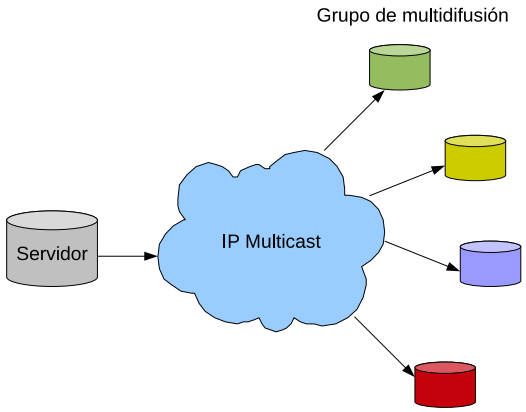


Figura 2.2: Ejemplo de una red de multidifusión IP

Es común utilizar indistintamente los términos multidifusión y difusión, sin embargo ambos difieren ligeramente. En una transmisión de multidifusión el servidor envía los datagramas a la red y estos son capturados y procesados por aquellos clientes que estén asociados al canal de multidifusión, se dice por tanto que dicho computador pertenece al grupo de multidifusión. En el caso de la difusión o *broadcast*, los datagramas se envían de igual forma a la red pero son capturados por todo computador que se encuentre conectado a la misma.

[2]Realmente no se hace uso de las 256 direcciones ya que las direcciones 0 y 255 se encuentran reservadas para otros usos.

Las características de subscripción y de transmisión de datos de la multidifusión la convierte en uno de los métodos más utilizados en aplicaciones de difusión multimedia como video conferencias o video por demanda.

Para pertenecer a un grupo de multidifusión, un computador cliente deberá asociarse a un puerto determinado en una dirección IP dada, destinada a transmisiones en dichos grupos.

2.1.3. Principales protocolos utilizados en multidifusión

Debido a la variedad de aplicaciones en las que se hace uso de transmisiones de multidifusión, se ha desarrollado un conjunto de protocolos que, combinados entre sí y con protocolos convencionales, ofrece una amplia variedad de mecanismos de difusión de datos (principalmente multimedia).

Bettahar[2] ofrece un esquema de los principales protocolos de difusión para video (igualmente utilizables en audio), dicho esquema se muestra en la figura 2.3.

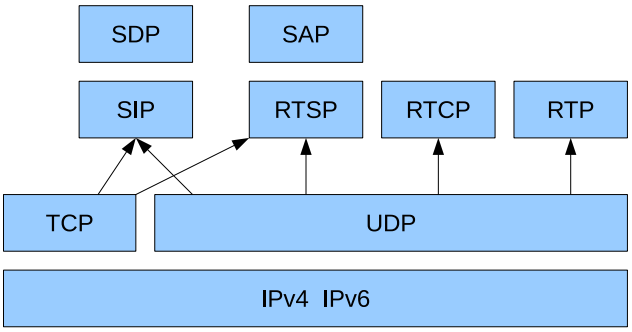


Figura 2.3: Protocolos usados en streaming multimedia.

Como puede observarse en la figura 2.3, es común utilizar un protocolo de red IP como base en una multidifusión multimedia, sobre éste suelen combinarse otros protocolos de transporte y aplicación que ofrecen una mayor flexibilidad y características determinadas.

- *Protocolos de transporte:* sobre los protocolos de red IP es usual utilizar TCP (*Transmission Control Protocol*) o UDP (*User Datagram Protocol*) como protocolos de transporte. A pesar de que ambos pertenecen a la misma capa, difieren sustancialmente en sus características de transmisión. Por otra parte también es común el uso conjunto de estos protocolos con otros tales como RTP y RTCP. En el cuadro 2.1 se detallan las principales características de los protocolos mencionados.

  A pesar de la menor fiabilidad del protocolo UDP en comparación con TCP, es este el de mayor uso en aplicaciones de multidifusión, debido a que en lo que respecta al rendimiento de transferencia multimedia es mucho más manejable la pérdida esporádica de un paquete de datos que el retraso de la transmisión, dado que se cuenta con los algoritmos de retransmisión y detección de errores. Un usuario se sentirá molesto en mayor medida por un retraso en la reproducción de un video que por una súbita distorsión, además muchos sistemas actuales poseen algoritmos que atenúan el efecto de la pérdida de paquetes con técnicas tan simples como la repetición de cuadros anteriores (en el caso de un video).

  Por otra parte, usualmente las transmisiones RTP se llevan a cabo en conjunto con RTCP, de esta forma mientras se realiza la difusión de los datos a través de RTP, RTCP proporciona estadísticas de la calidad del servicio que son por lo general utilizadas como retroalimentación para la modificación de las características de la transmisión.

- *Protocolos de aplicación:* sobre los protocolos de transporte suelen utilizarse además otros protocolos de aplicación tales como RTSP y SIP, cuyas principales características se resumen en el cuadro 2.2.

Protocolo	Descripción
TCP	- Protocolo de Control de Transmisión. - Orientado a conexión. - Detección de errores. - Reordenamiento de paquetes. - Retransmisión de datos.
UDP	- Protocolo de Datagrama de Usuario. - No es orientado a conexión (no necesita conocer el receptor de los paquetes). - No posee control sobre el estado en que llegan los paquetes. - No posee detección de errores. - No permite el reordenamiento de paquetes.
RTP	- Protocolo de Transporte en Tiempo Real. - Transporte de datos. - Se encarga de la sincronización. - Ofrece la opción de detección de errores. - No cuenta con opción de calidad de servicio (QoS).
RTCP	- Protocolo de Control en Tiempo Real. - Mensajes de control. - Permite obtener información de la calidad de servicio (estadísticas de transmisión).

Cuadro 2.1: Protocolos de transmisión comúnmente utilizados en aplicaciones de multidifusión.

Protocolo	Descripción
RTSP	- Protocolo de Difusión en Tiempo Real. - Actúa como un control remoto de red que permite controlar varios flujos de medios sincronizados.
SIP	- Protocolo de Inicio de Sesión. - Orientado a iniciar, modificar y terminar sesiones multimedia interactivas.

Cuadro 2.2: Protocolos de aplicación comúnmente utilizados en aplicaciones de multidifusión.

- *Protocolos de descripción:* en ocasiones es utilizado un protocolo de descripción de sesión tal como SDP (*Session Description Protocol*). Este protocolo describe las características de una sesión determinada: el tipo de dato multimedia (audio o video), la frecuencia de muestreo, la duración, la codificación, entre otros; esto con el objetivo de anunciar una sesión, invitar a la misma u ofrecer un control sobre esta [2]. En sí, este protocolo solamente describe el formato de los datos a transmitir y no constituye un protocolo de transmisión de los datos.

- *Protocolos de anunciación:* en conjunto con el SDP usualmente se utiliza un Protocolo de Anunciación de Sesión o SAP (*Session Announcement Protocol*) el cual consiste en una multidifusión periódica a un grupo y puerto determinado que se utiliza como medio de anunciación de los servicios de difusión de medios existentes y sus características.

Una alternativa utilizada en las transmisiones multimedia es el uso del formato de encapsulamiento TS o *MPEG-2 Transport Stream*[3]. En este mecanismo es posible transmitir varios contenidos multimedia y de programa en un solo flujo de datos. Aunque originalmente fue utilizado como formato de transmisión para difusiones de alta calidad en MPEG-2, actualmente es utilizado también para la transmisión de contenidos en formato MPEG-4. La transmisión se lleva a cabo en paquetes de 188 bytes, alternando en los paquetes contenidos de diferentes tipos (audio, video o información). A pesar de que este método proporciona una mayor sincronía entre los contenidos multimedia tiene la desventaja de generar un mayor estrés en la transmisión y es más sensible a la pérdida de paquetes.

2.2. GStreamer

2.2.1. ¿Qué es GStreamer?

GStreamer es una plataforma o *framework* para la manipulación de flujos multimedia, permite desarrollar aplicaciones de manejo de audio y video e incluso otros tipos de flujo de datos mediante el uso de complementos *(plugins)* que ejecutan funciones específicas [4]. En términos generales, según se describe en [4], GStreamer cuenta con la siguientes características:

- Una API[3] para aplicaciones multimedia.
- Una arquitectura de complementos.
- Una arquitectura de tuberías *(pipelines).*
- Un mecanismo de manipulación de tipos de medios y negociación.
- Más de 150 complementos.
- Un conjunto de herramientas.

2.2.2. Conceptos básicos de GStreamer

Como se ha mencionado, GStreamer se basa en el uso de complementos para la creación de flujos multimedia. Una unidad básica que cumple una función dada (lectura/escritura de archivos, decodificador, entre otras) es denominada un *elemento*. En general un elemento no es más que la clase de objeto de mayor importancia en GStreamer. Los elementos pueden ser utilizados en conjunto para formar una *tubería* a través de la cual se realiza el flujo de datos deseado. Los conceptos citados hasta el momento y otros que se explicarán a continuación se representan en la Figura 2.4.

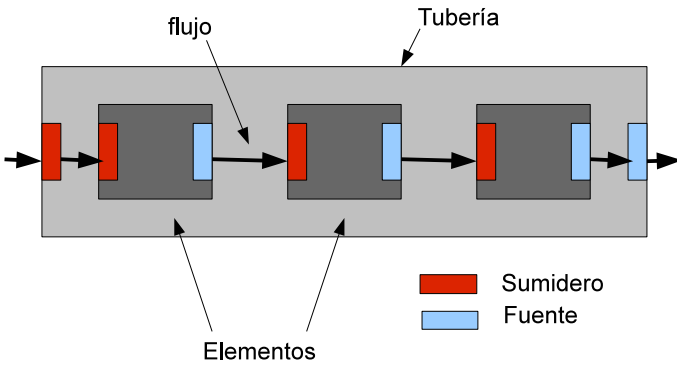


Figura 2.4: Diagrama básico de una tubería de GStreamer.

Como es posible observar en la figura 2.4, GStreamer se basa en un concepto con cierto nivel de abstracción, en el cual no importa la implementación de cada elemento, solo es importante su función y los tipos de datos que procesa. Los elementos se enlazan entre sí mediante el uso de *caminos* (*pads*) los cuales pueden ser de dos tipos: *fuente (source)* o *sumidero (sink),* dependiendo de la dirección del flujo de datos. Cada camino tiene asociado uno o varios tipos de datos específicos que soporta y que determinan con qué otros elementos pueden enlazarse.

[3]Una API o Interfaz de Programación de Aplicaciones es un conjunto de reglas y especificaciones que un programa debe cumplir para acceder a los recursos del programa que implementa la API.

En la Figura 2.4 también es posible observar los rasgos de un enfoque orientado a objetos, donde un conjunto de éstos (elementos) conforman un objeto más complejo (tubería) que hereda algunas de sus propiedades básicas (una tubería posee fuentes y sumideros y es considerada igual que un elemento).

Tal y como se mencionó, al enlazar dos elementos, sus caminos deben coincidir en el tipo de dato. Un camino específico puede aceptar uno o varios tipos de datos con rangos de propiedades determinados (dimensión, número de cuadros por segundo, entre otros), a este conjunto de datos se le denominan las *capacidades (capabilities* en inglés) del camino. Al encadenar dos elementos se lleva a cabo un proceso de *negociación* de las capacidades con la finalidad de encontrar un conjunto de propiedades que sea soportado por ambos.

En general, las tuberías son representadas mediante conjuntos de elementos y sus propiedades, separados por el símbolo "*!*". Un ejemplo de tubería es el siguiente: *alsasrc ! alsasink;* en este caso el elemento *alsasrc* captura audio desde el dispositivo de captura estándar y lo redirige al elemento *alsasink* el cual reproduce los datos que recibe de la tubería a través de la salida estándar de audio.

2.3. Transformada rápida de Fourier (FFT)

En esta sección se muestra de forma básica la obtención de un método de representación en frecuencia de una señal digital: la transformada discreta de Fourier (DFT). Se explicarán los inconvenientes computacionales de la DFT y se mostrará un algoritmo computacional más eficaz para su cálculo: la transformada rápida de Fourier (FFT) y los principales métodos de implementación utilizados para su cálculo. Es necesario comprender estos conceptos con el fin de entender las bases computacionales y las ventajas y desventajas de los algorítmos propuestos y de esta forma comprender el porqué de la elección realizada para la implementación del proyecto.

2.3.1. La transformada discreta de Fourier (DFT)

Una señal contínua (ya sea periódica o no) puede ser representada en el dominio de la frecuencia haciendo uso de las Series de Fourier o la Transformada de Fourier. En esta sección se mostrará un método para representar en el dominio de la frecuencia una señal discreta.

Si se supone una señal discreta $x(n)$ y se calcula su Transformada de Fourier se obtiene la siguiente expresión:

$$X(j\omega) = \int_{-\infty}^{\infty} x(n)e^{-j\omega t}dn \tag{2.1}$$

Sin embargo, puesto que $x(n)$ sólo existe para valores puntuales de n, la integral puede reemplazarse por una sumatoria de la siguiente forma:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega t} \tag{2.2}$$

De esta forma 2.2 representa el espectro en frecuencia de la señal discreta que se compone de una serie de réplicas del espectro de la señal original. A pesar que 2.2 es el cálculo exacto del espectro de $x(n)$, dicha expresión presenta dos inconvenientes de implementación tal y como se menciona en [5]:

- No es posible tener un número infinito de muestras n en el tiempo.
- No es posible obtener un cálculo con todos los valores de frecuencia posibles.

Estas limitaciones llevan a reformular 2.2 para obtener

$$X(\omega_k) = \sum_{n=0}^{N-1} x(n)e^{-j\omega_k n} \; para\, k = 0, 1, 2, ...\,;\; con\, \omega_k = \tfrac{2\pi k}{N} \tag{2.3}$$

donde ω_k es el espaciamiento de frecuencia utilizado y N el número de puntos utilizados en el cálculo. A 2.3 se le denomina *Transformada Discreta de Fourier* o *DFT* por sus siglas en Inglés.

Es necesario distinguir que N no corresponde siempre al número de muestras de la señal procesada, $x(n)$ puede tener una longitud de muestras L menor a N (rellenando con ceros el resto de puntos de la transformada) y no se afectaría el cálculo mas si se obtendría mayor resolución en el espectro obtenido [6, 5]. Por otro lado, incrementar el número de puntos de frecuencia K no mejorará el detalle del espectro obtenido, solamente rellenará con mayor cantidad de puntos la misma curva; es por esta razón que usualmente suele elegirse $K = N$.

El hecho de tener que truncar el número de muestras de la señal a L tiene efectos negativos sobre el espectro obtenido debido a que se generan lóbulos menores a lo largo del mismo. Este efecto usualmente es atenuado al usar una función ventana, $w(n)$, para el truncamiento de forma que se calcule la DFT del producto de $x(n)$ y la ventana. En la sección 2.4 se explicará de forma más detallada el efecto del truncamiento en las muestras de una señal.

2.3.2. Un algoritmo de cálculo más eficaz: FFT

Si bien la DFT permite calcular de forma práctica el espectro en frecuencia de una señal discreta, su cálculo conlleva realizar N multiplicaciones complejas en N puntos de frecuencia distintos, lo que eleva el orden del algoritmo a $O(n^2)$, lo cual para casos donde $n = N$ es muy grande implica un tiempo de cálculo apreciable. Debido a esto en 1965 Cooley y Tukey presentaron un nuevo algoritmo que permitía calcular la DFT de forma mucho más rápida, disminuyendo el orden del algoritmo a $O(nlogn)$ [7], dicho algoritmo general se conoce como *Transformada Rápida de Fourier* o *FFT* por sus siglas en inglés. Actualmente existen gran variedad de algoritmos de FFT, sin embargo todos se basan en el mismo principio, en el presente documento se hará énfasis en el algoritmo *radix-2* de Cooley-Tukey.

Algoritmo Cooley-Tukey (radix-2)

El algoritmo propuesto por Cooley y Tukey en 1965 permite realizar el cálculo de la DFT de una forma más eficiente haciendo uso del enfoque *divide y vencerás.* En las siguientes subsecciones se presentarán dos versiones del algoritmo *radix-2,* una implementación específica del algoritmo, orientada al cálculo eficiente de secuencias de datos de longitud 2^p.

FFT radix-2 DIF

El algoritmo radix-2 DIF (*Decimation In Frequency*) se obtiene al aplicar el enfoque divide y vencerás en la definición de la DFT en 2.3. De esta forma es posible dividir la sumatoria en dos sumatorias de longitud $N/2$ de la siguiente forma:

$$X(k) = \sum_{n=0}^{N/2-1} x(n)W_N^{kn} + \sum_{n=N/2}^{N-1} x(n)W_N^{kn} \tag{2.4}$$

donde $W_N = e^{j2\pi/N}$ y es llamado factor de fase o factor *Twiddle* [8, 6].

Si se cambia el índice de la segunda sumatoria en 2.4 y considerando que $W_N^{kN/2} = (-1)^k$ se obtiene

$$X(k) = \sum_{n=0}^{N/2-1} \left[x(n) + (-1)^k x(n + \frac{N}{2}) \right] W_N^{kn} \tag{2.5}$$

asumiendo valores pares de k y recordando que $(W_N)^2 = W_{N/2}$

$$X(2k) = \sum_{n=0}^{N/2-1} f_{par} W_{N/2}^{kn} \quad para\, k = 0, 1, ..., N/2 - 1 \tag{2.6}$$

por otra parte, para valores de k impares se tiene que

$$X(2k+1) = \sum_{n=0}^{N/2-1} f_{impar} W_{N/2}^{kn} \quad para\, k = 0, 1, ..., N/2 - 1 \tag{2.7}$$

donde

$$f_{par} = x(n) + x(n + \frac{N}{2}) \tag{2.8}$$

$$f_{impar=} \left[x(n) + x(n + \frac{N}{2}) \right] W_N^n \tag{2.9}$$

El algoritmo propuesto puede utilizarse de forma recursiva reduciendo a la mitad cada vez el número de puntos de la transformada hasta reducirse a una DFT de 2 puntos. El hecho que se obtengan los valores de la transformada para frecuencias pares e impares de forma separada a partir de la secuencia de datos ordenada conlleva al uso de un algoritmo de ordenamiento de los datos resultantes [6, 8], es esta característica la que le da su nombre al algoritmo. En la figura 2.5 se muestra una representación gráfica del algoritmo para $N = 8$, puede observarse como a partir de una secuencia de datos ordenada se realizan una serie de operaciones *mariposa* para finalmente obtener los valores de DFT en orden de bit invertido.

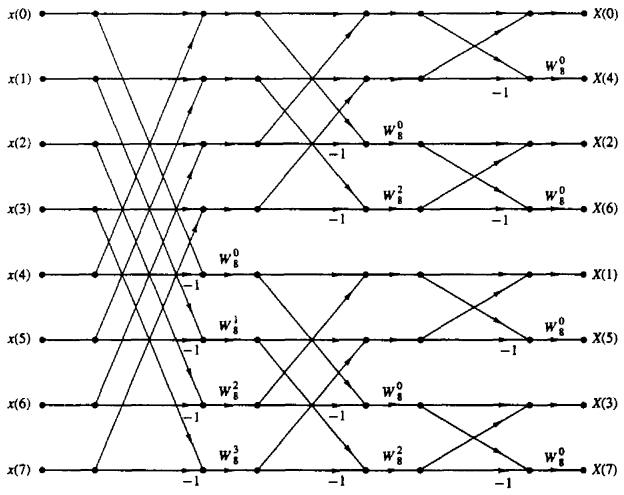


Figura 2.5: Representación gráfica del algoritmo DIF radix-2.

FFT radix-2 DIT

El algoritmo radix-2 DIT (*Decimation In Time*) es similar conceptualmente al DIF discutido anteriormente, con la diferencia que en este caso se aplica el concepto divide y vencerás a la secuencia de datos de entrada. De esta forma dada una secuencia de datos $x(n)$ con $x_p(n) = x(2n')$ y $x_i(n) = x(2n' + 1)$ las correspondientes secuencias de elementos pares e impares de $x(n)$, respectivamente, donde $n' = 0, 1, ..., N/2 - 1$ se tiene de 2.3 que

$$X(k) = \sum_{n=0}^{N/2-1} x_p(n) W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x_i(n) W_{N/2}^{nk} \tag{2.10}$$

En 2.10 se puede observar que cada sumatoria corresponde a la transformada discreta de la función par/impar, por lo cual puede reescribirse como

$$X(k) = X_p(k) + W_N^k X_i(k) \quad para\, k = 0, 1, 2, ..., N/2 - 1 \tag{2.11}$$

Como es posible observar en 2.11, el cálculo de la FFT DIT tal y como se ha expresado solamente suministra los valores de DFT para $N/2$ muestras de datos, sin embargo, considerando que $X_p(k)$ y $X_i(k)$ tienen periodo $N/2$ y utilizando la propiedad del factor de fase $W_N^{k+N/2} = -W_N^k$ se tiene que

$$X(k + N/2) = X_p(k) - W_N^k X_i(k) \quad para\, k = 0, 1, 2, ..., N/2 - 1 \tag{2.12}$$

de esta forma 2.11 y 2.12 suministran la transformada discreta de todos los N valores de entrada.

En la figura 2.6 se muestra un gráfico del algoritmo aplicado a una secuencia de 8 muestras, como es posible observar, los valores de entrada son reordenados mientras que la salida en frecuencia se obtiene automáticamente en orden, es por esta razón que se le da el nombre DIT al algoritmo.

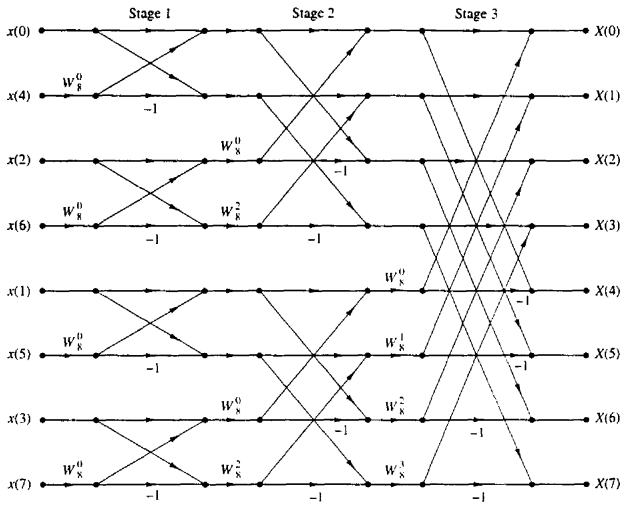


Figura 2.6: Representación gráfica del algoritmo DIT radix-2.

FFT radix-2 para señales reales

Usualmente el algoritmo radix-2 en cualquiera de sus dos versiones (DIF o DIT) asume que los datos de entrada son complejos, sin embargo en numerosas aplicaciones dichos datos son reales (tal es el caso de señales de audio). En estos casos es posible calcular la DFT usando los algoritmos propuestos con la simple adición de un pequeño nivel de lógica adicional. Para mayor referencia respecto al tema puede consultarse [6].

Existen dos métodos comunes para obtener la FFT de un conjunto de muestras reales utilizando los algoritmos antes descritos:

- Cálculo de la FFT de dos secuencias reales: en este caso, se considera que la secuencia de datos compleja de entrada del algoritmo es $x(n) = x_1(n) + jx_2(n)$ donde x_1 y x_2 son dos secuencias reales de las que se tiene interés conocer su transformada (ambas de longitud N). Debido a que la DFT es una operación lineal se tiene que:

$$X(k) = X_1(k) + jX_2(k) \tag{2.13}$$

y por tanto puede obtenerse el valor de las transformadas respectivas mediante

$$X_1(k) = \frac{1}{2}\left[X(k) + X*(N-k)\right] \tag{2.14}$$

$$X_2(k) = \frac{1}{2j}\left[X(k) - X*(N-k)\right] \tag{2.15}$$

- Cálculo de la FFT de una secuencia real de longitud 2N: si solamente se desea la DFT de una secuencia real m de longitud $2N$ es posible utilizar un algoritmo convencional con $x(n) = m_p(n) + jm_i(n)$ como señal de entrada, donde m_p y m_i son las muestras pares e impares de m. Utilizando 2.14 y 2.15 se pueden calcular los valores de M_p y M_i, para luego obtener

$$M(k) = M_p(k) + W_{2N}^k X_i(k) \quad para\, k = 0, 1, 2, ..., N-1 \tag{2.16}$$

$$M(k+N) = M_p(k) - W_{2N}^k X_i(k) \quad para\, k = 0, 1, 2, ..., N-1 \tag{2.17}$$

Complejidad y otros algoritmos

Tal y como se menciona en [6, 8], el algoritmo radix-2 posee un nivel de complejidad $O(Nlog_2N)$, comparando este resultado con el nivel $O(N^2)$ de la DFT ordinaria se tiene que la relación de número de ejecuciones necesarias esta dado por

$$\frac{N^2}{Nlog_2N} = \frac{N}{log_2N} = \frac{2^p}{p} \tag{2.18}$$

como puede observarse, para el caso de ejemplo de una transformada de 1024 puntos ($p = 10$) se obtiene una reducción de cerca 100 veces en la velocidad de ejecución del algoritmo (sin considerar aspectos de implementación) lo cual lo hace realmente eficiente en la medida que el número de muestras por transformar aumenta.

Si bien a lo largo del texto se ha mencionado solamente el algoritmo radix-2, cabe destacar que la propuesta hecha por Cooley y Tukey no se limita a un número de muestras de la forma 2^p sino que es igualmente aplicable para un número n^p de muestras. En dichos casos es posible realizar implementaciones más eficientes usualmente denominadas *radix-n* que permitan un cálculo mucho más eficiente y veloz aprovechando las características de la secuencia de entrada.

En muchas ocasiones, es usual mezclar algoritmos para obtener aun mayor eficiencia en la computación de la FFT, por ejemplo el utilizar radix-2 y radix-4 en un mismo algoritmo. A este esquema se le denomina *radix-partido* o *split-radix* [6].

2.4. Introducción a los filtros digitales

En esta sección se hace una pequeña referencia a los conceptos básicos de filtros digitales, principalmente se buscará establecer las principales ventajas y desventajas de los filtros IIR y FIR. Los conceptos mostrados permiten comprender la elección del filtro IIR Elíptico utilizado en el proyecto como parte del sistema de detección de audio.

2.4.1. Filtros digitales de respuesta infinita (IIR)

Un filtro IIR o filtro de respuesta infinita al impulso (*Infinite Impulse Response*) tiene una respuesta en frecuencia de la forma

$$H(\omega) = \frac{B(\omega)}{A(\omega)} = e^{-j\omega N_0} \frac{\sum_{k=0}^{M} b_k e^{-j\omega k}}{\sum_{k=0}^{N} a_k e^{-j\omega k}} \tag{2.19}$$

donde N_0 es una constante entera.

Como puede observarse, este tipo de filtro posee una respuesta en frecuencia fraccional lo que le da un carácter de retroalimentación respecto a valores de salida previos. Se dice que el filtro es de orden N y que deben realizarse N cálculos iniciales antes de obtener los valores de salida reales debido a que son necesarios N valores de salida pasados en memoria para su cálculo, a este tiempo se le suele llamar *tiempo de transición* del filtro [9]. Los filtros IIR poseen además una respuesta al impulso *h(n)* infinita, es decir, poseen un número infinito de componentes no nulas (de ahí su nombre).

Usualmente suele realizarse su diseño mediante el uso del método de *transformada bilinear,* en el cual se parte de la función de transferencia de un filtro analógico que cumple con los requerimientos deseados y se realiza un mapeo de dicha función al tiempo discreto para obtener la función de transferencia del filtro digital [5, 7].

Filtros IIR clásicos

Si bien un filtro IIR puede ser diseñado a partir de casi cualquier filtro analógico, el extenso uso en aplicaciones de ciertos diseños ha hecho que su implementación se realice en torno a un selecto grupo de tipos de filtro clásicos, dejando para aplicaciones más específicas diseños más elaborados. En términos generales se puede hacer referencia a cuatro tipos principales de filtros IIR clásicos [7]: Butterworth, Chevyshev I, Chevyshev II y Elíptico.

En la Figura 2.7 se muestra un ejemplo de la magnitud de la respuesta de cada uno de los filtros mencionados. Como puede observarse, la diferencia radica principalmente en las características de

selectividad y rizo de cada uno de los filtros. En general, la selectividad del filtro para un mismo orden N se incrementa de acuerdo al tipo de filtro de la siguiente forma: Butterworth, Chevyshev y Elíptico.

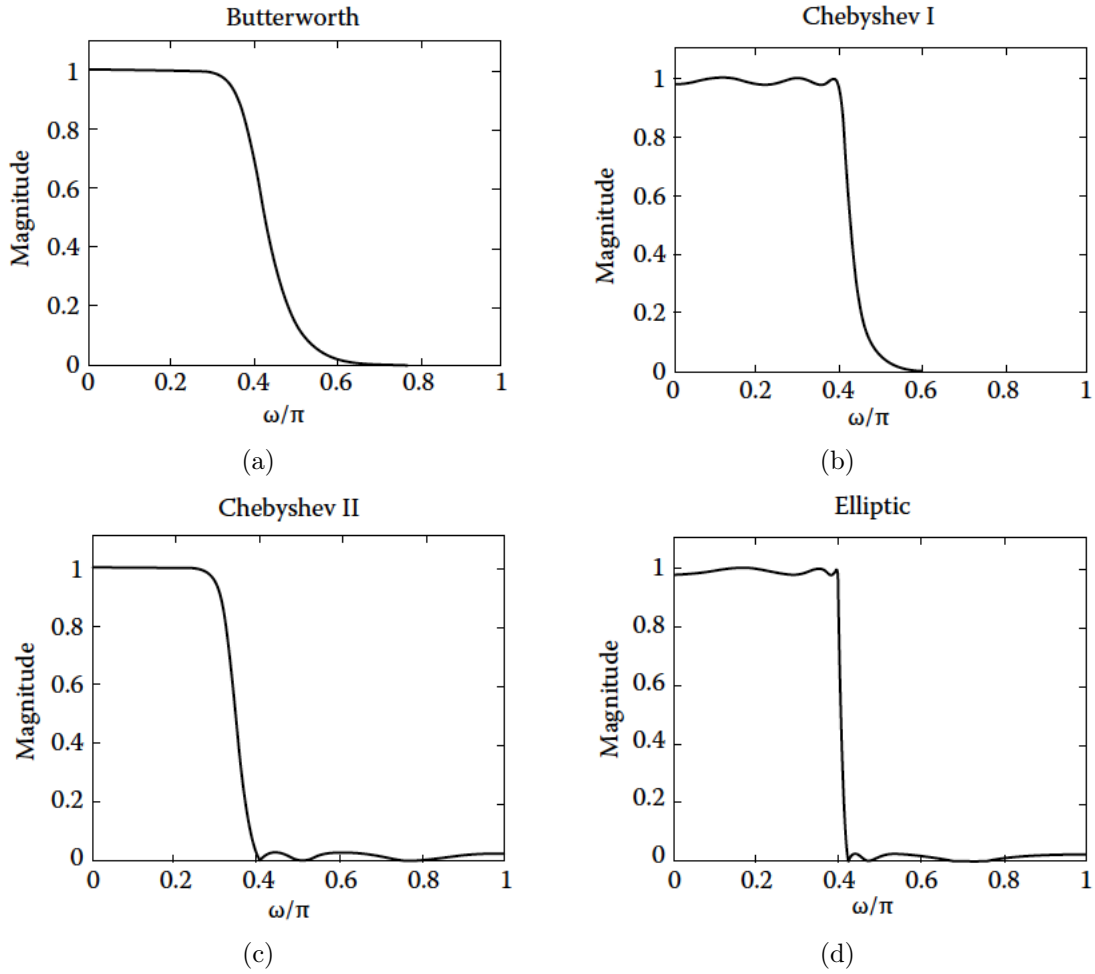


Figura 2.7: Tipos de filtros IIR clásicos.

Estabilidad y otras características de los filtros IIR

No importa cuál sea el método de diseño utilizado, los filtros IIR se caracterizan por proporcionar respuestas de frecuencia con características de magnitud realmente aceptables con un orden de filtro pequeño, sin embargo al ser diseñados como sistemas causales poseen una respuesta de fase no lineal no apta para aplicaciones donde la forma de la señal es un factor clave, en dichos casos es posible recurrir a técnicas de diseño que permiten obtener filtros IIR de fase cero con la desventaja de que la complejidad del filtro se incrementa considerablemente [7].

Otra característica de importancia en el diseño de filtros IIR es su estabilidad, tal y como puede observarse en 2.19, la respuesta en frecuencia de un filtro, y por ende su función de transferencia, poseen polos, los cuales determinan los requerimientos de estabilidad del filtro. En general un filtro IIR es estable si sus polos no están ubicados en el circulo unitario y además es causal si éstos se encuentran dentro de dicho círculo [7]. Debido a los requisitos de estabilidad, los filtros IIR son sensibles a la cuantización de sus coeficientes ya que este proceso podría ubicar los polos del filtro fuera de su región de convergencia.

2.4.2. Filtros digitales de respuesta finita (FIR)

Los filtros FIR o filtros de *respuesta finita al impulso* (por sus siglas en inglés: *Finite Impulse Response*) tienen una respuesta en frecuencia de la forma

$$H(\omega) = \sum_{n=N_1}^{N_2} h(n)e^{-j\omega n} \tag{2.20}$$

donde N_1 y N_2 son índices enteros correspondientes a las muestras de *h(n)*. Su nombre se debe al hecho de que este tipo de filtros posee una respuesta al impulso con un número de elementos distintos de cero finito.

Si se observa 2.20, es posible notar que la respuesta en frecuencia para un filtro FIR consiste en la serie de Fourier de la repuesta real truncada o acotada dentro de un número de valores de n finito. Este truncamiento genera un desvío de la respuesta en frecuencia respecto a su valor real que es incrementado cuanto menos componentes espectrales se utilicen, este fenómeno es conocido generalmente como *fenómeno de Gibbs.* Lo anterior puede representarse como que para un filtro FIR su respuesta al impulso esta dada por un número finito de valores de n.

Debido a que el truncamiento de los coeficientes representa un punto crítico en las características de la respuesta del filtro, se han desarrollado una serie de técnicas que permitan realizar dicho proceso con el mínimo efecto sobre la respuesta en frecuencia del filtro. En términos generales puede verse dicho truncamiento como el efecto de aplicar una ventana o máscara sobre la respuesta al impulso del filtro de la siguiente forma

$$h(n) = h_{ideal}(n) \bullet W(n) \quad n = 0, \pm 1, \pm 2, ..., \pm N \tag{2.21}$$

donde $W(n)$ corresponde a la función de la ventana aplicada. Las características de $W(n)$ determinan los efectos sobre la respuesta al impulso resultante, de tal forma que una ventana rectangular generará una mayor deformación de $h(n)$ que una ventana cuyas características supriman de forma más suave los coeficientes de $h_{ideal}(n)$. Algunas de las funciones ventana comúnmente utilizadas son: Rectangular, Triangular o Barlett, von Hann, Hamming, Blackman y Kaiser.

Una desventaja del uso de ventanas como las mencionadas es la existencia de un compromiso entre lo pronunciado de la banda de transición y la ganancia de la banda de rechazo, de esta forma su elección y características se encuentra atada a los requerimientos del diseño.

Existen implementaciones más avanzadas de diseño de este tipo de filtros como la Parks-McClellan que optimiza la respuesta en frecuencia del mismo haciendo que su magnitud presente lóbulos laterales de valor máximo constante en lugar de tener una magnitud decreciente como el el caso de los métodos de ventana antes mencionados.

Estabilidad y otras características de los filtros FIR

Debido a que la función de transferencia de los filtros FIR no posee polos, este tipo de filtros son estables siempre y sus características permiten su implementación causal de fase lineal, de gran importancia en aplicaciones donde es necesario preservar la forma de la señal procesada.

Una desventaja importante de este tipo de filtros respecto a los de respuesta infinita es que para su implementación requieren de un número mucho mayor de coeficientes, sin embrago sus características permiten su cálculo mediante convolución lo cual puede implementarse de una forma más eficaz si se hace uso de la FFT.

2.4.3. Filtros Bi-Quad

Un filtro Bi-Quad o *Bicuadrático* es un filtro IIR lineal de segundo orden que posee dos polos y dos ceros. En 2.22 se muestra la función de transferencia de un filtro Bi-Quad. Debido a que los filtros IIR de orden alto son sensibles a la cuantización de sus coeficientes, usualmente se implementan a base de etapas bicuadráticas lo que les proporciona una mayor estabilidad ante errores de cuantización.

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{2.22}$$

2.5. DBus

En la presente sección se explican los conceptos básicos de D-Bus, para un mayor detalle se recomienda revisar [10, 11]. D-Bus se refiere al nombre dado a un método de *comunicación entre procesos*

(Inter-process Communication) o IPC. Como todo sistema de IPC, D-Bus busca permitir la comunicación entre varios procesos con el fin de comunicar mensajes, ejecutar métodos externos o compartir información. Existen muchos otros proyectos de IPC tales como COBRA, MBUS y DCOM, sin embargo D-Bus es uno de los más utilizados actualmente en el desarrollo de aplicaciones para sistemas GNU/Linux y con un uso aún más extendido en los ambientes de escritorio KDE.

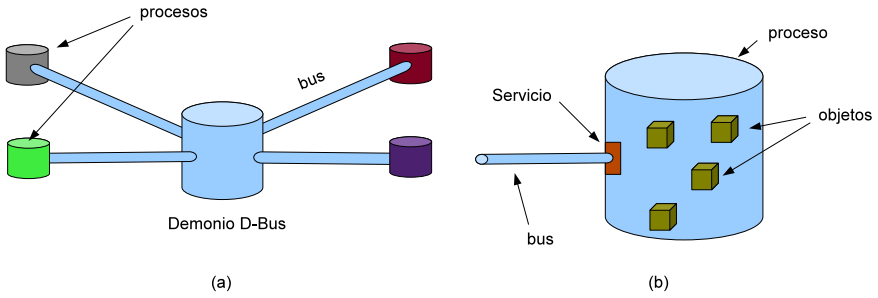


Figura 2.8: D-Bus. (a) Ejemplo de la topología de conexión entre varios procesos. (b) Representación gráfica de un proceso en un bus.

En la Figura 2.8 se muestran algunos conceptos básicos de D-Bus, los cuales se explican brevemente a continuación:

- Demonio de D-Bus: el demonio de D-Bus o *D-Bus daemon* es una aplicación que se encarga de administrar el tráfico de mensajes entre procesos, en general puede considerarse como un enrutador o *router* de red que enruta los diversos mensajes hacia sus respectivos procesos destino. En un sistema pueden existir varias instancias de este proceso para controlar el tráfico de mensajes entre aplicaciones, cada una puede ofrecer diferentes características o limitantes orientados a distintos tipos de aplicaciones.

- Servicio: según se explica en [11], puede verse al demonio de D-Bus como la bocina de un aro de bicicleta y cada aplicación que lo requiera se conecta a ésta a través de un *bus* (véase la Figura 2.8 (a)). Cada bus posee un nombre específico denominado *bus name* o *service.* Este nombre permite hacer referencia al bus de la aplicación desde un proceso externo conectado al mismo demonio. El nombre del bus puede ser genérico y único asignado por el demonio o puede ser asignado de forma que sea interpretable por los seres humanos, ésta ultima forma se denomina *well-known names.* El nombre del servicio suele escribirse de forma punteada de la siguiente forma: *nombre.de.un.servicio*

- Objetos: un proceso existente suele tener una o más instancias de objetos que realizan funciones específicas (véase Figura 2.8 (b)). Dentro del proceso dichos objetos son denominados *objetos nativos (native objects).* Algunos de estos objetos pueden ser accedidos por otros procesos mediante D-Bus, en dicho caso la referencia a éstos se hace utilizando una *ruta de objeto (object path).* Un *object path* se especifica de forma similar a como se hace referencia a un directorio en un sistema GNU/Linux, por ejemplo una ruta de objeto puede especificarse como sigue: */My/Object/Path*

- Métodos y señales: cuando se ha asociado ya con un determinado servicio y objeto, es posible que dicho objeto ofrezca una serie de métodos y señales. Usualmente un método se accede desde un proceso externo, enviando un mensaje con el nombre del método y sus parámetros y con la posibilidad de retorno de un valor; en casos en que se hace referencia a un método no existente se retorna una señal de error. Una señal por otro lado son similares a una transferencia de *broadcast* ya que es enviada por el proceso generador hacia todos los procesos existentes en el bus.

- Interfase: una interfase especifica los métodos y señales disponibles en un objeto dado. Pueden existir una o más interfaces por objeto y permiten acceder sus métodos o señales. Para especificar un método o señal a través de una interfaz dada se utiliza la notación punteada, a manera de ejemplo supóngase que se desea acceder el método *destroyAll* a través de la interfaz *ejemplo.de.interfaz,* lo anterior se haría a través de la siguiente expresión *ejemplo.de.interfaz.destroyAll.*
- Objeto proxy: un objeto proxy o *proxy object* corresponde a una implementación nativa de un objeto que hace referencia a un objeto en otro proceso.

Capítulo 3

Desarrollo de una aplicación de referencia

Tal y como se mencionó en el Capítulo 1, las aplicaciones multimedia que hacen uso de la difusión masiva han llevado al desarrollo de aplicaciones de multidifusión que permitan una entrega de contenidos de forma eficiente a múltiples terminales. Con el fin de adquirir el conocimiento necesario en esta tecnología, RidgeRun ha optado por la investigación y desarrollo en ésta área y de esta manera ingresar en este nuevo mercado, sin embargo hasta la fecha, no cuenta con conocimiento alguno ni experiencia en el desarrollo de aplicaciones de este tipo.

Con motivo de resolver el problema expuesto, RidgeRun requirió la realización de una aplicación de referencia que sirva de base para el desarrollo de futuras aplicaciones de multidifusión. Siguiendo las sugerencias de la empresa se determinó desarrollar un sistema de vigilancia de bebé que permitiera distribuir el audio y video de forma simultánea, de un punto de acceso situado en el mismo lugar donde se encuentra el niño hacia uno o más nodos cliente. En la figura 3.1 se expone la idea general de la aplicación descrita. Como es posible observar el sistema se constituye de un nodo principal o servidor, el cual debe medir el sonido ambiente con el objetivo de detectar un nivel de audio significativo dentro de un rango de frecuencias específico (que permita detectar el llanto de un niño); una vez detectado deberá iniciar la multidifusión del audio y video de forma simultánea hacia los nodos inscritos en el grupo de multidifusión. Cada uno de los nodos inscritos debe reproducir el audio y video de forma simultánea.

El diseño de la aplicación puede dividirse en dos secciones: el diseño del sistema de análisis de audio para la detección del llanto del niño, y el diseño del sistema de transmisión multidifusión. Una vez desarrollados ambos puntos su implementación en conjunto dependerá de si se trata de un nodo servidor o cliente y también de las capacidades de la tarjeta utilizada (en este caso la LeopardBoard DM365).

En las siguientes secciones se explica el proceso de diseño e implementación de los diferentes componentes del proyecto.

3.1. Equipo utilizado

El equipo utilizado consta de una tarjeta de evaluación LeopardBoard DM365. Esta tarjeta posee un SoC (*System on Chip*) TMS320DM365 de la familia Davinci de Texas Instrument con un procesador ARM 9.

Entre las principales características con las que cuenta ésta tarjeta se encuentran:

- Dos motores co-procesadores de imagen y video (HDVICP, MJCP).
- Soporte de procesamiento de video: reconocimiento facial, filtro de ruido, auto enfoque, balance automático de blancos, etc.
- Periféricos: EMAC, USB 2.0 OTG, DDR2/NAND, UART's, SPI, Ethernet, SD, entre otros.

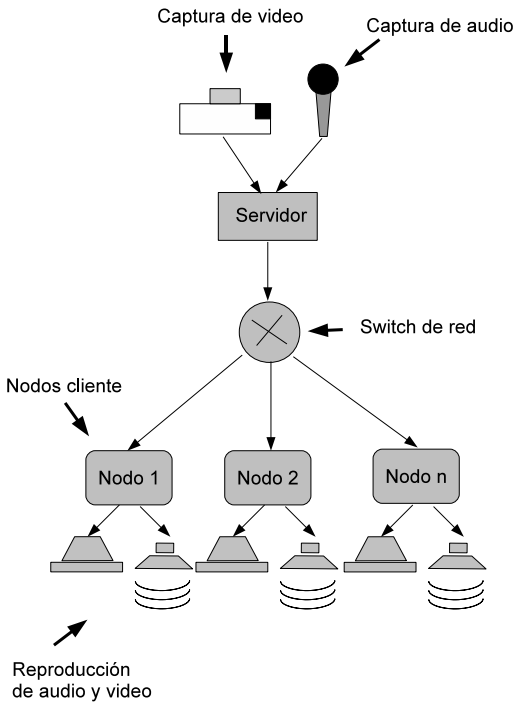


Figura 3.1: Diagrama general de la aplicación

- Soporta los paquetes de codecs: H.264, MPEG-4, MPEG-2, MJPEG y VC1/WMV9.
- Soporte de salida de video compuesto, componente y DVI[1].
- Puertos de entrada y salida de audio.
- Interfaz para módulos de cámara.

En conjunto con la tarjeta, también fue utilizado un sensor de CMOS *MT9P031* de Aptina de 5 Megapixeles, esto debido a que a la fecha de realización del proyecto, solamente se contaba con soporte para éste sensor.

3.2. Diseño del sistema de detección de audio

El sistema de detección de audio es el encargado del senso y detección de niveles de audio que permitan determinar el momento en que exista un posible llanto de bebé. Más que un sistema de reconocimiento de sonido se trata de un promediador que permite determinar niveles de ruido considerables dentro de una banda de frecuencias dada. El sistema puede describirse de la forma mostrada en la figura 3.2. En primer lugar se encuentra un sistema de captura de muestras de audio, seguido por un filtro digital para limitar las señales de audio a una banda de frecuencia deseada. Seguido del filtro se encuentra una etapa de FFT que permite obtener las magnitudes de las componentes espectrales, y por último un promediador que determine el valor promedio del espectro de la señal. A continuación se explican los principales elementos de diseño utilizados en la implementación del sistema.

[1]Requiere de adaptador

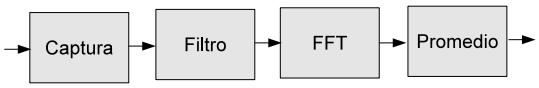


Figura 3.2: Diagrama de bloques del sistema de detección de audio.

3.2.1. Etapa de captura de muestras de audio

La captura de audio se lleva a cabo mediante el uso de la biblioteca STK (*Synthesis ToollKit*) [12]. Esta biblioteca de código libre ofrece utilidades para el desarrollo de aplicaciones de procesamiento de audio, entre ellas una utilidad para captura de muestras de audio, de esta forma las muestras obtenidas son almacenadas en un arreglo de 512 campos.

La frecuencia de muestreo se estableció en 44100Hz, lo que permite capturar señales de audio en casi todo el espectro audible y además permite ignorar los efectos de no tener un filtro anti-alias en el sistema.

3.2.2. Diseño e implementación del filtro digital

Según expone Huron[13], el llanto de un niño se encuentra usualmente entre los 3kHz y los 4kHz debido a que en este rango el ser humano percibe los sonidos con una mayor sensibilidad. De esta forma, para el diseño del sistema de detección de audio no es necesario considerar todas las frecuencias. Para lograr lo anterior fue necesario implementar un filtro digital pasa banda con una banda de paso ubicada en la banda de 3-4 KHz.

El filtro elegido fue un filtro IIR Elíptico, esto debido a las características mencionadas en la sección 2.4 y a las razones que se presentan a continuación:

- Un filtro IIR implementable posee un orden mucho menor que un equivalente FIR: al realizar pruebas de diseño se encontró que un filtro IIR promedio requería de un orden aproximadamente diez veces menor que un filtro FIR con las mismas especificaciones. Este es un factor de gran relevancia cuando el sistema no posee una capacidad de procesamiento dedicada como por ejemplo una unidad DSP.

- Ofrece la selectividad necesaria para la aplicación: debido a que la aplicación requería la detección de posibles peligros, no es necesario obtener una gran selectividad de audio ya que son preferibles falsos positivos en el sistema.

Una vez elegido el filtro por diseñar, se procedió a establecer sus especificaciones iniciales, representadas en la figura 3.3. *Fs1* y *Fs2* representan las frecuencias límite de la banda de rechazo mientras que *Fp1* y *Fp2* representan las frecuencias límite de la banda de paso. *Amax* es la atenuación máxima permitida en la banda de paso y *Amin* es la atenuación mínima permitida en la banda de rechazo. En la segunda columna del cuadro 3.1 se muestran los valores teóricos iniciales elegidos para el diseño del filtro, se ha elegido un espaciamiento de 500Hz en la banda de transición ubicando las frecuencias de paso en los límites de audio requeridos (3-4kHz). Se seleccionó una atenuación baja en la banda de paso con motivo de permitir lo más posible las señales de audio en dichas frecuencias y una atenuación mínima de 50dB para suprimir el resto de frecuencias en la banda de rechazo.

Una vez establecidas las especificaciones iniciales se procedió al diseño del filtro con la ayuda del programa de diseño de filtros digitales DISPRO ™[14]. Se decidió establecer el orden del filtro en 10 con lo cual se obtuvieron las especificaciones mostradas en la columna 3 del cuadro 3.1. Finalmente

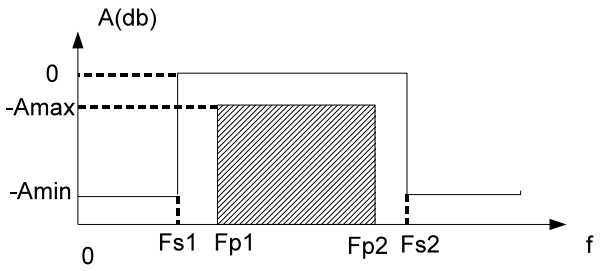


Figura 3.3: Especificación de un filtro pasa banda.

Especificación	Valor inicial	Valor ajustado
Amax	-0.1 dB	-0.002 dB
Amin	-50 dB	-50 dB
Fs1	2.5 KHz	2.5 KHz
Fp1	3 KHz	3 KHz
Fp2	4 KHz	4 KHz
Fs2	4.5 KHz	4.5 KHz

Cuadro 3.1: Especificaciones de diseño del filtro IIR Elíptico. (Frecuencia de muestreo 44100Hz)

se obtuvieron los coeficientes que se muestran en el cuadro 3.2. Como es posible observar, el filtro se conforma de cinco etapas Bi-Quad.

Etapa	b_0	b_1	b_2	a_1	a_2
1	0.2185225	-0.406858	0.2185225	-1.80889	0.9807354
2	0.6534409	-1.04187	0.6534409	-1.651065	0.9741636
3	0.4148288	-0.786630	0.4148288	-1.750345	0.9349073
4	0.3450008	-0.508111	0.3450008	-1.633529	0.9203182
5	0.08539221	0	-0.08539221	-1.670159	0.896928

Cuadro 3.2: Coeficientes obtenidos para el filtro IIR Elíptico.

En la figura 3.4 se muestra el gráfico de polos y ceros del filtro obtenido con el programa DISPRO^{TM}. Los polos se encuentran dentro del círculo unitario, tal y como se requiere para que el filtro sea estable y causal, sin embargo su proximidad al límite del círculo unitario requiere tener cuidado con el redondeo de los coeficientes para evitar que los polos se desplacen fuera del círculo haciendo que el filtro se torne inestable.

Una vez obtenidos los coeficientes del filtro se llevó a cabo su implementación en C++ con el uso de la biblioteca STK (*Synthesis ToollKit*) [12]. Ésta biblioteca cuenta con un conjunto de funciones para la implementación de etapas de filtros bicuadráticos utilizada en este caso para implementar el filtro diseñado.

3.2.3. Implementación del algoritmo de FFT

Una vez obtenido el filtro, se obtuvo la transformada discreta de Fourier de las muestras de audio con el fin de tener una idea de la magnitud de las componentes espectrales y de esta forma, una idea de la entropía de la señal. Para la obtención de dicha transformada se hizo uso del algoritmo FFT DIF radix-2 debido a las razones que se mencionan a continuación:

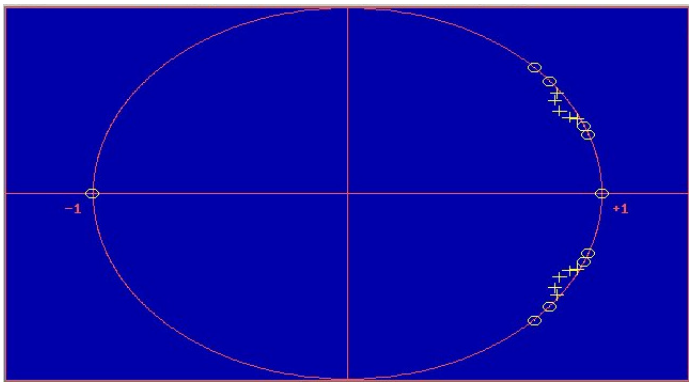


Figura 3.4: Gráfico de polos y ceros del filtro diseñado.

- El algoritmo de FFT radix-2 permite generar una biblioteca que puede ser reutilizada de forma más general en otras aplicaciones en contraste con un algoritmo radix-n superior.

- Al elegir un algoritmo DIF es posible omitir el reordenamiento de las muestras resultantes debido a que el interés se centra en su magnitud y no en el orden de las componentes espectrales, esto permite eliminar dicha carga de procesamiento en el sistema.

El algoritmo desarrollado se encuentra basado en la implementación propuesta en [15], sin embargo se ha omitido el algoritmo de reordenamiento de las componentes espectrales y se ha adaptado para el cálculo de la FFT de un conjunto de muestras reales. Para el cálculo de las muestras se ha utilizado el algoritmo para dos conjuntos de muestras reales planteado en la sección 2.3.2 considerando el segundo conjunto como un arreglo de ceros. Esta modificación se llevó a cabo debido a que requiere un menor procesamiento de datos en comparación con el segundo método propuesto.

Con motivo de disminuir la carga de procesamiento y elevar la velocidad del cálculo, se ha creado un archivo de cabecera con valores precalculados de los factores Twiddle lo cual permite su acceso en una estructura a modo de tabla.

El algoritmo fue modificado además de forma tal que el arreglo resultante sea la magnitud de las componentes espectrales, por lo que también se trata de un conjunto de datos real.

La implementación se llevó a cabo en C++ y para una FFT de 512 puntos.

3.2.4. Obtención del nivel promedio de audio

Una vez obtenidas las magnitudes de las componentes espectrales de la señal de audio, se buscó representar las variaciones en el nivel de la señal mediante el cálculo de la media aritmética del conjunto de datos. Considerando que en cada análisis se realiza una captura de N muestras de audio, las cuales permiten obtener N valores espectrales, es posible calcular la media de dichos valores de la siguiente forma:

$$\overline{X} = \frac{1}{N} \sum_{n=0}^{N-1} X(n) \tag{3.1}$$

De esta forma se obtiene una medida de la entropía de la señal en el tiempo ya que el promedio calculado será mucho mayor a frecuencias dentro de la banda deseada que fuera de esta.

3.3. Diseño del sistema de transmisión multidifusión

En esta sección se expone el proceso de diseño e implementación de las tuberías de GStreamer para la transmisión del contenido multimedia utilizando multidifusión. En primer lugar se describirá el proceso de selección y diseño de los protocolos y formatos utilizados así como las tuberías de GStreamer, por último se explicará la implementación de éstos en la tarjeta LeopardBoard DM365.

3.3.1. Diseño de las tuberías de GStreamer para multidifusión

Para el diseño de las tuberías a utilizar fue necesario considerar tres aspectos importantes: el conjunto de protocolos de transmisión, los formatos multimedia a utilizar y el ancho de banda necesario para la transmisión.

La decisión de utilizar GStreamer fue debido a dos razones principales, la primera consiste en el hecho que GStreamer cuenta con un conjunto de elementos para la transferencia de contenidos multimedia a través de red (muchos de los cuales se encuentran en el grupo de complementos *good* o *base* por lo cual poseen una gran estabilidad). La segunda razón se debe al hecho que la empresa RidgeRun cuenta con una reputación de ofrecer un soporte experto y calificado en GStreamer.

Mecanismos de transmisión multidifusión

En la sección 2.1 se expusieron algunos de los principales conjuntos de protocolos utilizados en la transmisión punto a multipunto, la selección de un conjunto específico se encuentra ligada principalmente a las especificaciones de la aplicación. Para el proyecto se decidió utilizar el mecanismo RTP/UDP/IP para la transferencia de multidifusión debido a las características de estabilidad, confiabilidad y sincronización obtenidas en pruebas de diseño, las cuales se detallan en el Apéndice A.

No se consideró el uso de los protocolos de descripción de sesión (SDP) o de anunciación (SAP) debido a que la aplicación no requiere de la existencia de diversas transmisiones de contenidos sino que solo existe una transmisión de audio y video conocida. También se descartó el uso de RTSP debido a que no es necesario el control remoto sobre el contenido multimedia difundido.

Formatos multimedia utilizados

Una vez seleccionado el medio, el siguiente paso fue la elección de los formatos multimedia a utilizar. La tarjeta LeopardBoard DM365 cuenta con una API de GStreamer similar a la utilizada en computadores ordinarios, sin embargo RidgeRun ha desarrollado una serie de complementos dedicados para la arquitectura ARM. En general la tarjeta cuenta con dos conjuntos de complementos de video especializados para los formatos H264 y MPEG-4 mientras que para audio solo se cuenta con complementos convencionales tales como *mu-law* y *a-law* principalmente (estos formatos de baja compresión ofrecen un desempeño realmente estable en aplicaciones de transferencia red).

En el cuadro 3.3 se muestra el consumo de CPU aproximado en un computador personal[2] de los formatos mencionados, como puede observarse, tanto MPEG-4 en el caso de video como *mu-law* para audio ofrecieron un mejor desempeño con un consumo de CPU menor.

Considerando el video como la principal fuente de consumo de recursos se decidió realizar un segundo conjunto de pruebas en la tarjeta para los dos formatos de video antes mencionados con lo cual se obtuvo el resultado que se muestra en el cuadro 3.4 como es posible observar, MPEG-4 requirió entre un 40 % hasta un 60 % menos recursos que H264.

Formato	CPU servidor	CPU cliente
H264	48 %	15 %
MPEG-4	42 %	10 %
mu-law	0.7 %	1 %
Vorbis	6 %	2 %

Cuadro 3.3: Consumo de CPU para diversos formatos multimedia.

Formato	CPU servidor	CPU cliente
H264	61 %	65 %
MPEG-4	26 %	40 %

Cuadro 3.4: Consumo CPU de formatos de video en la LeopardBoard DM365.

[2]Las características del computador utilizado son las siguientes: CPU AMD Semprom 140 2.7GHz y 2GB de memoria RAM.

En el Apéndice B se detallan las tuberías de prueba utilizados en la mediciones mencionadas.

Consumo de ancho de banda

Un tercer aspecto a considerar es el consumo de ancho de banda compartido de la transmisión multimedia. En general es preferible reducir al máximo este parámetro ya que un uso considerable de ancho de banda puede interferir con otras transmisiones en la misma red. Para el caso de la aplicación en cuestión, se utilizó el sensor de cámara *mt9p031* con una resolución de 640 x 480 píxeles. La escogencia del sensor fue principalmente debido a que en el momento de desarrollo del proyecto la tarjeta LeopardBoard DM365 solamente contaba con soporte para éste.

En el cuadro 3.5 se muestran los resultados obtenidos de las pruebas de ancho de banda realizadas con el programa NetMeter^{TM}[16]. Como puede observarse MPEG-4 requirió de un menor consumo de ancho de banda en comparación con H264. Para un mayor detalle de las pruebas realizadas véase el apéndice B. Cabe destacar que las mediciones mostradas corresponden a un formato de video con una resolución de 640x40 píxeles, un aumento en la resolución elevaría de igual forma el ancho de banda requerido para la transmisión.

Formato	Ancho de banda consumido
H264 (640x480)	2.2Mbps
MPEG-4 (640x480)	1.5Mpbs
mu-law	80.1kbps

Cuadro 3.5: Consumo de ancho de banda de algunos formatos multimedia para transmisiones punto a multipunto.

Tuberías de multidifusión utilizadas

Una vez finalizadas las pruebas de desempeño fue posible realizar una elección de la estructura de las tuberías por crear. Se decidió utilizar RTP/UDP/IP como mecanismo de transmisión en conjunto con los complementos de MPEG-4 y *mu-law* de GStreamer en su versión 0.10.28 debido a que ofrecieron el menor consumo de CPU y ancho de banda en las pruebas realizadas.

En la figura 3.5-(a) y 3.5-(b) se muestra el esquema general de una tubería de emisión y recepción de una transmisión multimedia, respectivamente. En general pueden describirse las tuberias en cinco bloques básicos de acuerdo a sus funciones. En ambos casos debe existir un elemento fuente de datos, los cuales son filtrados a través de un conjunto de capacidades y procesados ya sea para su codificación o decodificación. El empaquetamiento RTP para su transmisión y su envío a la red se realiza en la etapa final de la tubería servidor mientras que el proceso inverso de recepción se lleva a cabo en la etapa inicial de la tubería cliente.

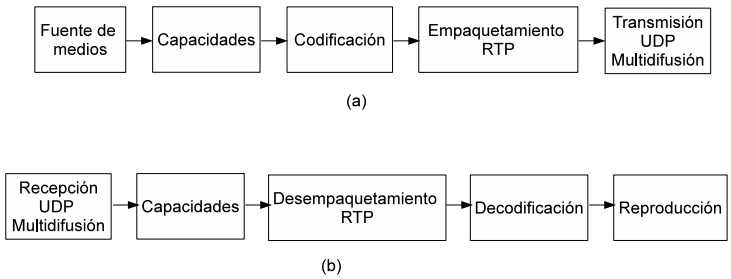


Figura 3.5: Diagrama de bloques básico de una tubería (a) servidor y (b) cliente para transmisiones de red.

En general se desarrollaron cuatro descripciones de tuberías: dos de video (servidor y cliente) y dos de audio (servidor y cliente), los cuales se detallan a continuación:

- Servidor de video

La tubería utilizada para el servidor de video es el siguiente:

"v4l2src always-copy=FALSE ! video/x-raw-yuv,format=(fourcc)NV12, width=640, height=480, framerate=(fraction)30/1 ! dmaiaccel ! queue ! dmaienc_mpeg4 ! rtpmp4vpay ! udpsink host=$(multicast_group) port=$(port_number) auto-multicast=true"

Como es posible observar, la tubería inicia con el elemento *v4l2src* el cual es utilizado para capturar los cuadros de video desde el sensor de cámara de la tarjeta. A éste elemento se le desactivó la propiedad *always-copy* lo cual evita que se realice una copia del buffer de captura a un segundo buffer para su procesamiento, esto permite reducir el consumo de recursos del sistema. El siguiente elemento en la cadena está constituido por las capacidades deseadas para la captura de video, en este caso se desea obtener un formato de video en NV12 con una resolución de 640x480 píxeles y a una velocidad de captura de 30 cuadros por segundo. Este flujo es enlazado a través de un acelerador de hardware para procesamiento de video representado por el elemento *dmaiaccel* y dirigido hacia una cola (*queue*) que permite almacenar un buffer de datos mientras son procesados por el codificador de video *dmaienc_mpeg4*. Una vez el video ha sido codificado este es empaquetado en un formato RTP por el elemento *rtpmp4vpay* para su posterior envío por la red. Finalmente el elemento *udpsink* se encarga de enviar los paquetes de datos a través de la red sobre un protocolo de transmisión UDP/IP; las propiedades *host* y *port* especifican el grupo de multidifusión y número de puerto al que se desea transmitir la información, respectivamente. La propiedad *auto-multicast* permite la asociación y desasociación automática a un grupo de multidifusión.

- Cliente de video

El cliente de video fue realizado con base en la siguiente tubería:

udpsrc port=$(port_number) multicast-group=$(multicast_group) auto-multicast=true timeout=$(time) ! application/x-rtp, media=(string)video, clock-rate=(int)90000, encoding-name=(string)MP4V-ES, profile-level-id=(string)5, config=(string)000001b005000001b50ecf0000010000000120086e0002ea6600b7c52c999cba98514043c1463, payload=(int)96, ssrc=(guint)1226930637, clock-base=(guint)344275943, seqnum-base=(guint)15648 ! rtpmp4vdepay ! queue ! dmaidec_mpeg4 ! TIDmaiVideoSink videoOutput=component sync=false accelFrameCopy=true videoStd=720P_60 noCopy=true"

Como puede observarse, para la etapa de recepción de red se ha utilizado el elemento *udpsrc* al cual se le han establecido las propiedades de *musticast-group* y *port* los valores del grupo de multidifusión y puerto al que se desea inscribir el cliente, además se ha activado la propiedad de *auto-multicast* para la asociación automática al grupo. La propiedad *timeout* del elemento *udpsrc* emite una señal en el sistema cada vez que no sean recibidos datos desde la red durante el tiempo especificado (dado en microsegundo). Seguido del elemento de recepción se encuentran las capacidades de los datos que se esperan recibir, en general se especifican características tales como la codificación del video y la base de tiempo para su reproducción; estas capacidades son suministradas por la tubería del servidor y deben ser utilizadas como filtro en el cliente para un correcto funcionamiento. Una vez establecidas las capacidades se desempaquetan los datos RTP mediante el uso del elemento *rtpmp4vdepay* y se pasan a través de una cola (*queue*) hacia el decodificador de MPEG-4 (*dmaidec_mpeg4*). Ya decodificado el video, este es desplegado en pantalla a través de la salida componente de la tarjeta con el uso del elemento *TIDmaiVideoSink*; dicho elemento esta diseñado para la arquitectura y permite activar algunas propiedades como la aceleración de copia de cuadros y el uso directo del buffer sin copia, algunas otras propiedades como el tipo de salida de video y resolución también son establecidas.

- Servidor de audio

La tubería de audio del servidor es la siguiente:

"alsasrc ! queue ! audioconvert ! mulawenc ! queue ! rtppcmupay ! udpsink host=$(multicast_group) port=$(port_number) auto-multicast=true"

De forma similar al servidor de video, el servidor de audio inicia con el elemento *alsasrc* el cual es usado para capturar el audio desde el dispositivo estándar, en el caso de la LeopardBoard DM365 la entrada de *line-in*. Seguido al *alsasrc* se encuentra una cola para el almacenaje de muestras capturadas mientras estas son convertidas por el elemento *audioconvert* al formato de audio necesario para poder ser codificadas a *mu-law* por el elemento *mulawenc*. Una vez encodificadas las muestras de audio, éstas son empaquetadas en RTP por el elemento *rtppcmupay* para posteriormente ser enviadas a la red por el elemento *udpsink* (cuyas propiedades se han establecido de igual forma que en el servidor de video).

- Cliente de audio

En el caso del cliente, los datos son recibidos y reproducidos mediante la siguiente tubería:

"udpsrc port=$(port_number) multicast-group=$(multicast_group) timeout=$(time) auto-multicast=true ! application/x-rtp, media=(string)audio, clock-rate=(int)8000, encoding-name=(string)PCMU, payload=(int)0 ! rtppcmudepay ! queue ! mulawdec ! audioconvert ! alsasink sync=false"

El cliente de audio se compone del elemento *udpsrc* para la recepción de los datagramas transmitidos (configurado de igual forma al cliente de video). Se utilizan nuevamente capacidades para filtrar el tipo de datos por recibir y sus propiedades. Una vez se reciben los datagramas del servidor estos son desempaquetados por el elemento *rtppcmudepay* para luego pasar a una cola que suministre un flujo constante de datos al decodificador de audio *mulawdec*. Una vez se ha decodificado el contenido éste pasa por el elemento *audioconvert* para darle el formato de audio adecuado para ser reproducido por *alsasink*, sumidero de audio que utiliza la salida estándar de audio del driver de ALSA.

Con el objetivo de realizar una transmisión simultánea del audio y video, se utilizó un mismo grupo de multidifusión para ambas tuberías pero se varió el puerto de red al que se envían los datos, de esta forma ambas transferencias pueden ser procesadas por separado.

3.3.2. Implementación de las tuberías en la LeopardBoard DM365

Una vez obtenidas las tuberías fue necesario realizar su implementación en la tarjeta LeopardBoard DM365, para ello se hizo uso del proyecto de código libre *GStreamer Daemon (GSTD)* [17]. *GStreamer daemon* o demonio de GStreamer es una aplicación escrita en VALA y desarrollada para ejecutar tuberías de GStreamer como procesos separados similar a la aplicación *gst-launch* que es instalada por defecto con GStreamer. La principal diferencia de GSTD respecto de *gst-launch* es su capacidad de controlar el estado de la tubería y sus propiedades en tiempo de ejecución y además permite la comunicación con otras aplicaciones a través de DBus.

En la figura 3.6 puede observarse un diagrama general de los elementos que componen GSTD y su interacción con otros procesos. GSTD es capaz de controlar varias tuberías ejecutándose al mismo tiempo y modificar sus estados (play, pause, null, entre otros) , además GSTD puede redirigir mensajes desde las tuberías hacia el bus de sistema DBus y puede recibir comandos desde otras aplicaciones de igual forma. Además de los estados, GSTD admite la modificación de las propiedades de los elementos de una tubería específica en tiempo de ejecución.

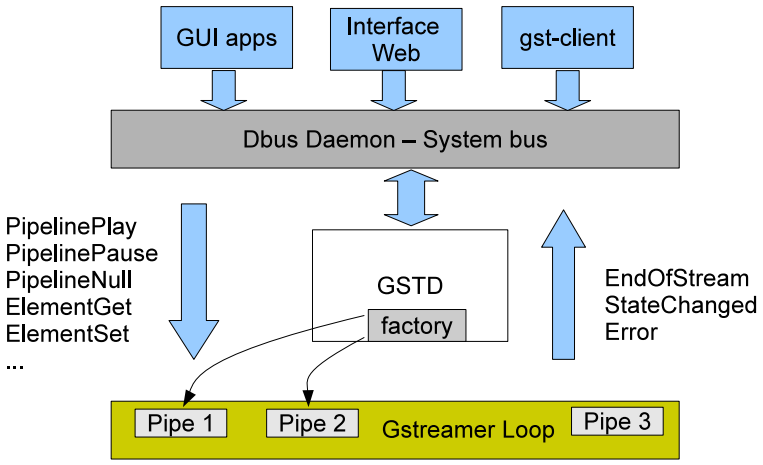


Figura 3.6: Diagrama general de funcionamiento de GSTD.

Como puede observarse en la figura 3.6, GSTD cuenta con la capacidad de reenviar tres señales desde las tuberías hacia el sistema: *EndOfStream, StateChanged, Error*. Debido a los requerimientos de diseño de la aplicación, fue necesario realizar un parche de código a la aplicación para añadir el soporte para la señal de *timeout* del elemento *udpsrc*, la justificación de dicha modificación será evidente en la siguiente sección.

Al iniciar GSTD éste crea un servicio llamado *com.ridgerun.gstreamer.gstd* a través del cual se tiene acceso a dos objetos: */com/ridgerun/gstreamer/gstd/factory* para la creación de tuberías y */com/ridgerun/gstreamer/gstd/pipeline* para el control de las tuberías creadas. De esta forma la implementación se hizo a través de una biblioteca de C++ con acceso a los objetos del servicio mediante el uso de objetos proxy, dicha biblioteca interactúa con GSTD a través de DBus para crear las tuberías y cambiar algunas de las propiedades de estos. Este esquema permite que la aplicación siga respondiendo mientras las tuberías se encuentran en ejecución en otro proceso.

La biblioteca creada se encuentra programada en una clase de C++ con la adición del uso de señales para la advertencia de eventos lo cual la hace fácilmente integrable con proyectos desarrollados en QT^{TM}.

3.4. Diseño e implementación del nodo servidor

En la figura 3.7 se muestra el esquema general de la implementación realizada para el nodo servidor. En general pueden establecerse tres bloques principales que conforman la aplicación: la interfaz gráfica realizada en QT^{TM}, el algoritmo de detección de audio y la biblioteca de multidifusión.

La interfaz gráfica fue desarrollada en QT^{TM}y compilada utilizando *QT Embedded 4.6.0*. En el Apéndice C se puede encontrar el detalle de las opciones de la interfaz .

La biblioteca de multidifusión se encuentra en el mismo hilo de proceso de la interfaz gráfica y proporciona el medio de comunicación entre la aplicación y GSTD a través del bus del sistema. Las operaciones necesarias para crear, ejecutar y destruir las tuberías de multidifusión (entre otras operaciones) se realizan a través de ésta biblioteca.

Por otro lado, el algoritmo de detección de audio se ha incluido como un hilo de ejecución separado al resto de la aplicación, esto debido a que el cálculo del nivel de audio requiere de un tiempo de procesamiento considerable que, de no ejecutarse en un hilo separado, provocaría instantes en que la GUI no respondería.

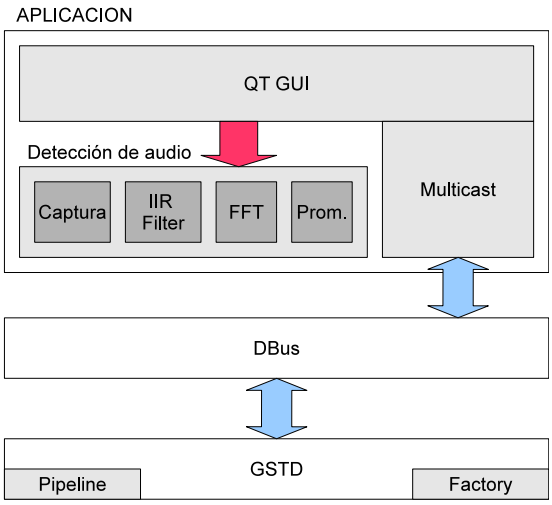


Figura 3.7: Diagrama de bloques del nodo servidor.

En la figura 3.8 se muestra el diagrama de flujo básico del nodo servidor, las flechas de trazos indican ejecuciones que se llevan a cabo en presencia del evento indicado. Como puede observarse, al iniciar la aplicación simplemente se inicializan las variables y se establece el aspecto de la GUI en modo apagado. Al presionar el botón de encendido de la interfaz se verifica si la aplicación ya se encuentra encendida o no, en el último caso se enciende un temporizador (*timer*) interno utilizado para determinar el intervalo de tiempo en que se realizará el análisis del nivel de audio y se cambia el aspecto de la interfaz.

Cada vez que el temporizador se dispara, la señal *TimerTimeOut* se genera y se inicia el análisis del nivel de audio en un hilo de ejecución separado. Si el nivel de audio detectado supera el nivel prefijado se genera la señal *onAudioLevelDetected,* se inician las tuberías de multidifusión de audio y video y la interfaz cambia a modo de alerta, en caso contrario se general la señal *onAudioLevelNotDetected* y se reinicia el proceso.

Si es generada la señal de *onStartButtonClicked* nuevamente, se destruyen las tuberías en caso de haberse disparado la alarma y se retorna el sistema al estado de apagado.

3.5. Diseño e implementación del nodo cliente

En la figura 3.9 se muestra el diagrama de bloques general del nodo cliente, como es posible observar su nivel de complejidad es menor al servidor. Se tiene de igual forma la interfaz gráfica (GUI) realizada en QT^{TM}la cual le permite al usuario encender y apagar la aplicación así como modificar algunos parámetros de la transmisión punto a multipunto (para mayor detalle véase el Apéndice C).

Debido a que el nodo cliente debe recibir la transmisión multimedia y reproducirla, solamente es necesario el uso de la biblioteca de multidifusión para el control de las tuberías de recepción diseñadas.

El diagrama de flujo del nodo cliente se muestra en la figura 3.10, las flechas de trazos indican procesos que se llevan a cabo cuando el evento indicado ocurre. Una vez iniciada la interfaz gráfica, la aplicación queda en espera de algún evento por parte del usuario. Si se da un evento sobre el botón de encendido se verifica si la aplicación se encuentra encendida o no. Si la aplicación se encontraba apagada, se inician las tuberías de audio y video, se cambia el estado de la interfaz a encendido y se enciende el temporizador (timer) interno.

Si el nodo servidor no se encuentra transmitiendo datagramas, las tuberías del nodo cliente generan una señal de *timeout* cada segundo (representada por *SrcTimeOut*), esta señal es utilizada para detectar

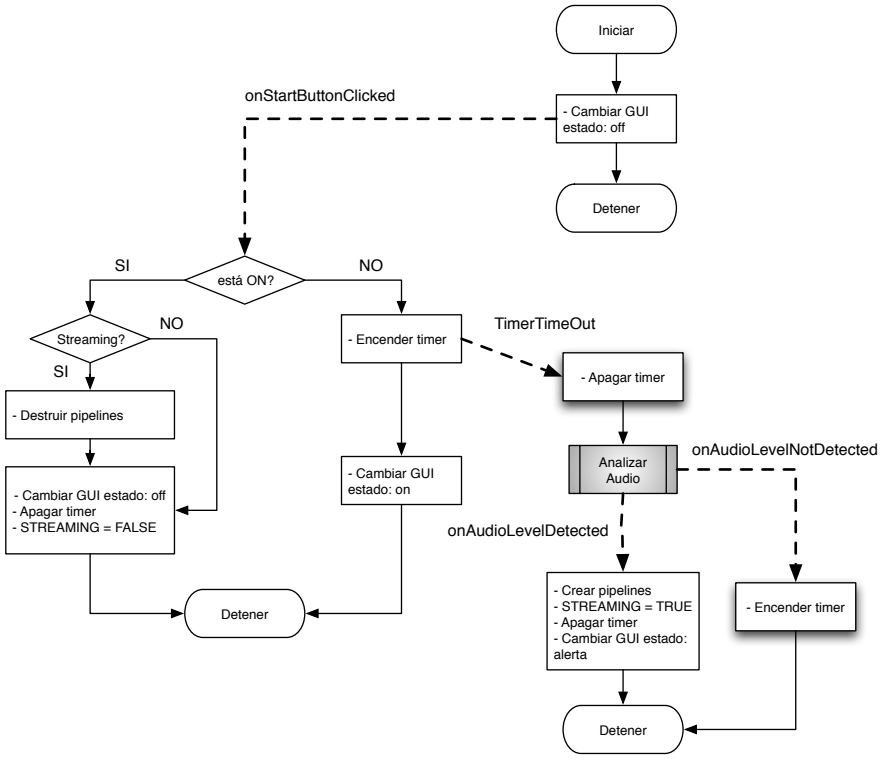


Figura 3.8: Diagrama de flujo del nodo servidor.

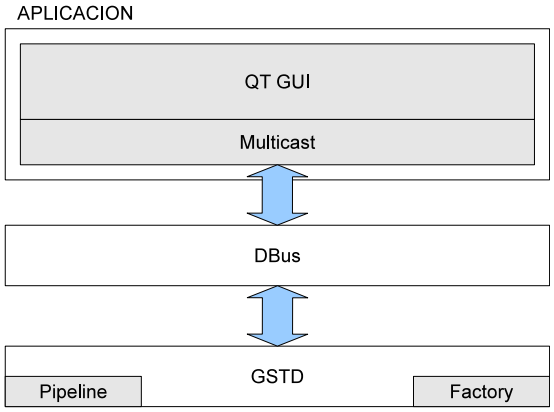


Figura 3.9: Diagrama de bloques del nodo cliente.

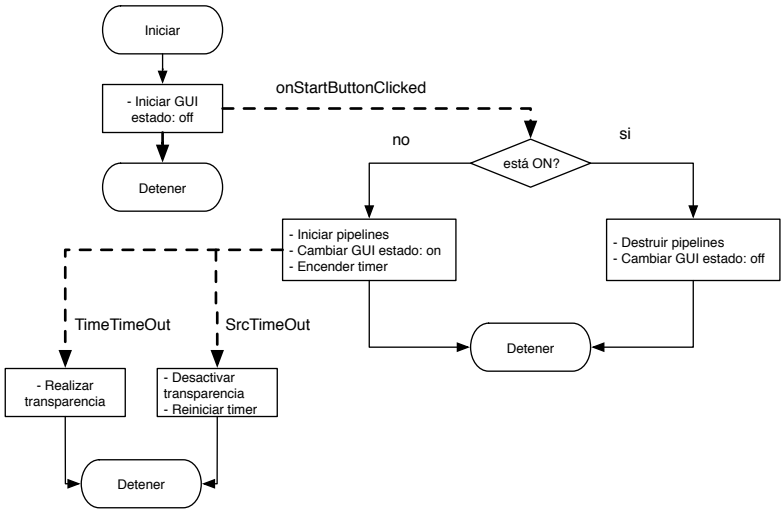


Figura 3.10: Diagrama de flujo del nodo cliente.

una transmisión de datos en la red. Cada vez que ésta señal es disparada se desactiva la transparencia de la zona de video de la GUI y se reinicia el temporizador, dicha transparencia se lleva a cabo mediante el uso de la propiedad de *color key* del VPBE (*Video Processing Back End*) del SoC TMS320DM365.

Si la señal *SrcTimeOut* deja de dispararse debido a una transmisión existente en el grupo de multidifusión inscrito, el temporizador interno dispara una señal de *timeout* después de 5s lo que activa la transparencia y permite mostrar el video recibido.

Si se presiona de nuevo el botón de apagado de la aplicación se procede a destruir las tuberías y se cambia la interfaz a modo de apagado.

3.6. Otras características de diseño

La implementación realizada utilizando GStreamer, GSTD y QT como principales herramientas permiten obtener un sistema capáz de ser utilizado en diferentes plataformas tales como Linux Embedded, GNU/Linux, Windows y MacOSX; con un mínimo de cambios para su compilación cruzada. Esto permite obtener un software de referencia para el desarrollo de aplicaciones multidifusión en varios escenarios sin limitar su uso solamente a sistemas empotrados con sistema operativo Linux.

Capítulo 4

Resultados de laboratorio y análisis

En el presente capítulo se muestran y analizan los principales resultados obtenidos de la evaluación del proyecto realizado. En primer lugar se presenta el análisis de la respuesta real obtenida del filtro implementado, seguido se exponen las mediciones de ancho de banda del sistema y consumo de CPU con el fin de compararlas con el comportamiento esperado. Por último se muestran los resultados obtenidos al exponer el sistema a un banco de pruebas de audio y variar los parámetros de detección con el fin de corroborar el adecuado funcionamiento del proyecto.

4.1. Estabilidad de transmisión

Un primer aspecto a considerar fue la estabilidad de la transmisión para una conexión de red cableada y otra inalámbrica. Si se consideran las proyecciones de ancho de banda requeridas mencionadas en la sección 3.3.1 (corroboradas en la sección 4.3) es posible observar que para el caso de una red inalámbrica con el estándar 802.11g, correspondiente a WiFi G (con un ancho de banda promedio efectivo de 25Mbps), no representa un inconveniente la transmisión de dicho contenido multimedia. De igual forma una conexión cableada 10BASE- o 100BASE- con un ancho de banda mínimo de 10Mbps es capaz de transmitir sin ningún inconveniente un flujo multimedia con las características especificadas.

Al realizar las pruebas de funcionamiento en ambos tipos de red se obtuvo un funcionamiento correcto para el caso de red cableada.

Para la prueba de la red inalámbrica se hizo uso de un adaptador BELKIN F5D7050A, sin embargo la transmisión se vio afectada debido a limitantes en el controlador de USB de la tarjeta LeopardBoard DM365 debido a que al elevarse el ancho de banda consumido alrededor de 1Mbps se obtuvo un error del kernel debido a un manejo erróneo del búfer de datos de USB. En la figura 4.1 se muestra un extracto del error reportado, como es posible observar este proviene del archivo *drivers/usb/musb/musb_ host.c* el cual es parte del controlador USB de la tarjeta. Debido a que el problema expuesto se encontraba fuera de los alcances del proyecto, se decidió utilizar conexión cableada para la transmisión de datos, lo cual permitía continuar con el proyecto sin afectar los objetivos del mismo (obtener una aplicación de referencia para transmisiones multicast).

```
1.  ------------[ cut here ]------------
2.  WARNING: at drivers/usb/musb/musb_host.c:126 musb_h_tx_flush_fifo+0x94/0xcc()
3.  Could not flush host TX2 fifo: csr: 2103
4.  Modules linked in: dm365mmap irqk edmak cmemk p54usb p54common
5.  Backtrace:
6.  [<c0031700>] (dump_backtrace+0x0/0x110) from [<c0031844>] (dump_stack+0x18/0x1c)
7.   r6:c02002a8 r5:c03eeb80 r4:0000007e
8.  [<c003182c>] (dump_stack+0x0/0x1c) from [<c00423ac>] (warn_slowpath_common+0x50/0x68)
9.  [<c004235c>] (warn_slowpath_common+0x0/0x68) from [<c0042410>] (warn_slowpath_fmt+0x30/0x38)
10.  r7:ffffffff r6:fec64520 r5:00002103 r4:00002103
11. [<c00423e0>] (warn_slowpath_fmt+0x0/0x38) from [<c02002a8>] (musb_h_tx_flush_fifo+0x94/0xcc)
12.  r3:00000002 r2:c03eeba0
```

Figura 4.1: Error en el controlador USB obtenido para una transmisión inalámbrica.

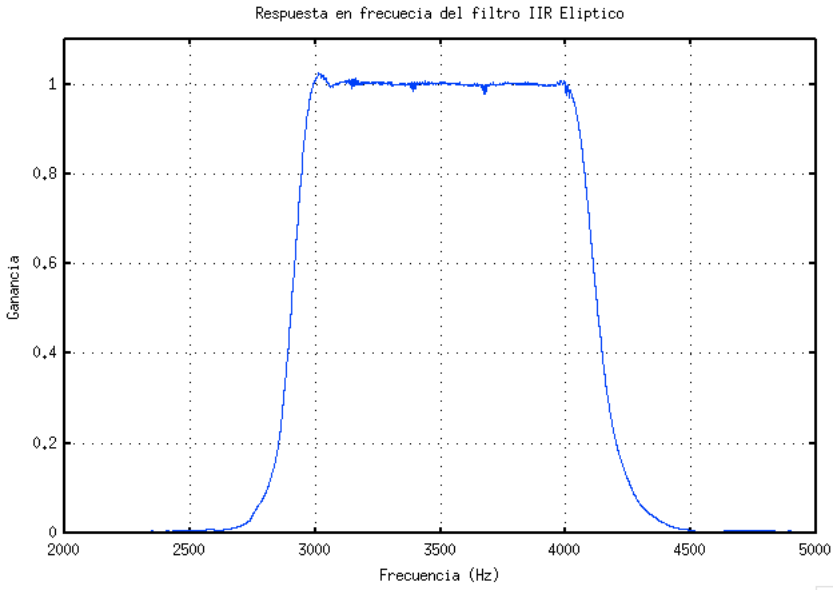


Figura 4.2: Respuesta en frecuencia del filtro digital implementado.

4.2. Respuesta en frecuencia del sistema de detección de audio

Como una primera aproximación del funcionamiento del sistema se requirió verificar la selección de frecuencias en el rango de 3 a 4kHz. Para dicha medición se utilizó la biblioteca *STK* para generar un barrido de señal de entrada al filtro digital implementado. Dicho barrido se conformó de señales sinusoidales entre 2kHz y 5kHz con un espaciamiento de 1Hz. En la figura 4.2 se muestra la respuesta en frecuencia obtenida del filtro digital implementado en la tarjeta LeopardBoard DM365. Como puede observarse, se obtuvo una respuesta en frecuencia con un factor de forma aproximado de 1.64, un ancho de banda de 1.16kHz a -3dB, una ganancia unitaria en la banda de 3-4kHz y una pendiente ligeramente pronunciada que se extiende en una banda de transición de 500Hz, tal y como fue diseñada.

Por otra parte, en las figuras 4.3(a) y 4.3(b) se muestran los efectos del filtro sobre una señal de amplitud unitaria. Tanto en el caso de una señal de frecuencia permitida como de una fuera del rango deseado, se presenta un breve transitorio al inicio de la señal de salida, éste transitorio es típico de los filtros IIR tal y como se mencionó en el Capítulo 2. A pesar de que dicho transitorio modifica la composición de la señal en el dominio de la frecuencia, su efecto es mínimo debido los niveles de atenuación obtenidos por parte del filtro digital.

4.3. Estudio de ancho de banda y rendimiento del sistema

Un segundo aspecto a considerar en el funcionamiento del sistema es el requerimiento de ancho de banda y de consumo de recursos. Debido a que la codificación de video es un factor dependiente de las características del mismo (nivel y velocidad de cambios en el escenario), el uso de ancho de banda y CPU del sistema también son variables. Con el fin de establecer un margen de variación de estas características se optó por la medición de dos situaciones límite: la transmisión de un video estático y la de un video con movimiento, los resultados obtenidos se muestran a continuación.

4.3.1. Transmisión de un video estático

En la figura 4.4 se muestra la medición de ancho de banda de la transmisión de video realizada mediante el programa NetMeter^{TM}[16], como es posible observar el hecho de tratarse de un video sin movimiento conlleva a poca variación del ancho de banda requerido puesto que la información a

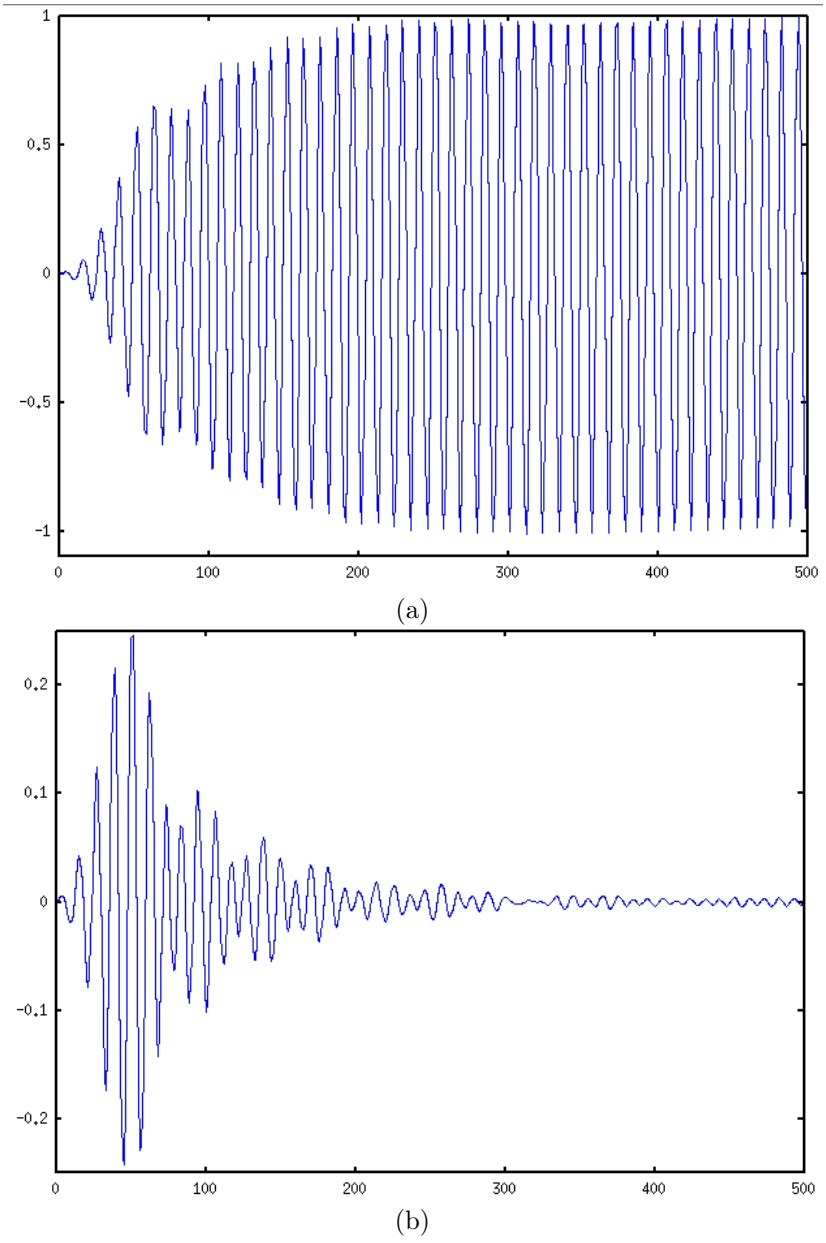


Figura 4.3: Forma de onda de salida del filtro digital para una señal sinusoidal de (a) 4000Hz y (b) 4500Hz.

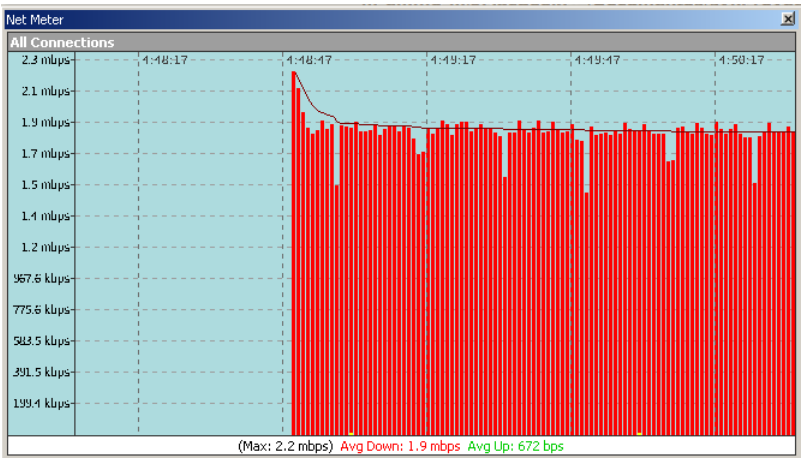


Figura 4.4: Medición de ancho de banda para la transmisión de video sin movimiento.

enviar es esencialmente la misma, esto se traduce de igual forma a un requerimiento de ancho de banda promedio bajo, de aproximadamente 1.9Mbps.

En el cuadro 4.1 se resumen las mediciones de ancho de banda y consumo de CPU para la transmisión de video estático, tanto para el nodo servidor como cliente. Cabe resaltar el hecho que el uso de CPU necesario para la transmisión es superior en casi un 50 % del lado del receptor que del servidor, esto es cierto puesto que si bien el codificador se encarga de la compresión del video de acuerdo a su nivel de variación (algoritmos de predicción), es el cliente quien debe reconstruir los cuadros a partir de dicha información y desplegarlos en la pantalla.

Variable	Valor promedio
Ancho de banda	1.9Mbps
Uso de CPU (servidor)	26 %
Uso de CPU (cliente)	40 %

Cuadro 4.1: Ancho de banda requerido y consumo de CPU para una transmisión de video sin movimiento.

4.3.2. Transmisión de un video en movimiento

Para realizar la prueba de transmisión de video con movimiento se realizó la captura de un video con las características mostradas en el cuadro 4.2. En la figura 4.5 se muestra la medición de ancho de banda realizada, en este caso se obtuvo una variación significativa debido a la constante modificación del video, lo cual altera la codificación del mismo y por ende la carga de los datos a transmitir. Tal y como se mencionó, la adición de movimiento conlleva a una transmisión de mayor cantidad de información debido a una mayor presencia de cuadros I y por ende a un aumento considerable del ancho de banda (con un promedio de 4.6Mbps para el caso mostrado).

Nombre	Enlace	Resolución	Cuadros por segundo
Park_iPod.mp4	http://www.youtube.com/watch?v=F1z8lEBSHUo	320x240 píxeles	30 fps

Cuadro 4.2: Características del video con movimiento utilizado para las pruebas de ancho de banda. (Tomado el 22/12/10)

En el cuadro 4.3 se exponen las mediciones de ancho de banda y consumo de CPU obtenidas. Nuevamente puede observarse la relación en el consumo de CPU entre el nodo servidor y cliente, esto

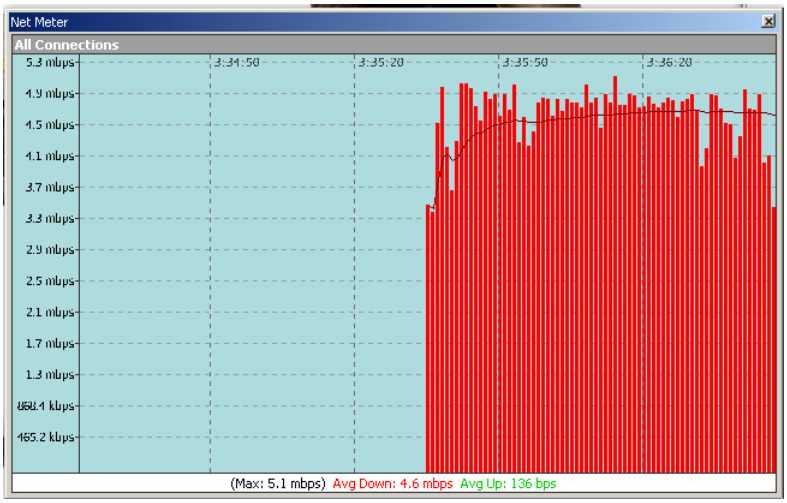


Figura 4.5: Medición de ancho de banda para la transmisión de video con movimiento.

Variable	Valor promedio
Ancho de banda	4.6Mbps
Uso de CPU (servidor)	75 %
Uso de CPU (cliente)	100 %

Cuadro 4.3: Ancho de banda requerido y consumo de CPU para una transmisión de video con movimiento.

debido a la mayor carga de procesamiento por parte del nodo receptor, sin embargo la razón entre ambos términos es menor debido a que una mayor transferencia de cuadros I requiere de un menor procesamiento para la decodificación del video.

Cabe destacar que la medición realizada no representa el límite superior para una transmisión con movimiento ya que el consumo de ancho de banda es variante dependiendo de las características del video y su codificación (algunos algorítmos de codificación presentan funciones de predicción de cuadros extremadamente complejas que impiden realizar un cáculo exacto), los datos obtenidos solamente permiten obtener una proyección de dichos valores para transmisiones de escenarios variantes. Para efectos de la aplicación desarrollada, el video utilizado puede ser considerado como una buena aproximación del ancho de banda máximo debido a que no se espera capturar video con tan alto contenido de movimiento. A pesar de lo anterior es alarmante el resultado obtenido para el uso de CPU, principalmente en el nodo cliente, ya que de exponerse el sistema a un escenario de mayor acción podría sobrepasar los recursos de la tarjeta y provocar un fallo en el sistema, esto debe de ser considerado como una limitante del sistema debido a su capacidad de procesamiento.

Como es posible observar en el cuadro 4.3, es necesario un ancho de banda mínimo promedio de 5Mbps para una transmisión de video con las características descritas, sin embargo si se desea elevar la calidad del video o si éste posee mayor variabilidad, el ancho de banda requerido aumentará.

4.3.3. Transmisión de audio

Las características de la transmisión de audio son independientes del video que se transmite y en general, debido a las características aleatorias de una señal de audio, se puede despreciar el efecto del tipo de señal en los requerimientos de ancho de banda y CPU. En la figura 4.6 se muestra la medición de ancho de banda de la transmisión de audio para una señal de una emisora de radio, como es posible observar el consumo de ancho de banda es relativamente constante y con una magnitud promedio de 48.7Kbps.

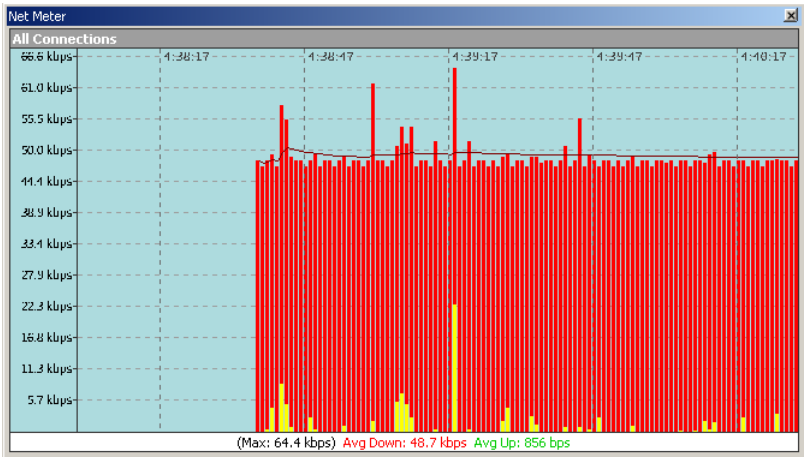


Figura 4.6: Medición de ancho de banda para una transmisión de audio.

En el cuadro 4.4 se muestran los valores de ancho de banda y CPU medidos para el caso de la transmisión de audio. Como es posible observar, el ancho de banda necesario para la transmisión es mucho menor en comparación al video, por lo cual éste último constituye un factor determinante en una transmisión multimedia.

Variable	Valor promedio
Ancho de banda	48.7kbps
Uso de CPU (servidor)	0.7 %
Uso de CPU (cliente)	1 %

Cuadro 4.4: Ancho de banda requerido y consumo de CPU para una transmisión de audio.

4.4. Banco de pruebas general

Con motivo de realizar una prueba general del funcionamiento del sistema se simularon varios escenarios de trabajo mediante el uso de un conjunto de videos cuyo audio se utilizó como entrada. Las pruebas realizadas pueden dividirse en dos conjuntos: un conjunto para la comprobación de la capacidad de detección de llanto del sistema de acuerdo con el ajuste en algunos parámetros de operación, y un segundo conjunto para la comprobación del rechazo del sistema ante varios escenarios comunes donde se espera que el sistema no sea activado.

En el cuadro 4.5 se muestran los enlaces correspondientes a cada uno de los videos utilizados así como el nombre o identificador utilizado a lo largo del documento.

4.4.1. Pruebas de capacidad de detección

El objetivo de estas pruebas es el determinar la capacidad de detección del llanto de un bebé y en general la sensibilidad ante ruidos cercanos a la banda de frecuencia de 3KHz a 4KHz. Para ello se hizo uso de los videos BEBE1, BEBE2 y BEBE3, cada uno se reprodujo 10 veces y su entrada de audio se inyectó a la tarjeta LeopardBoard DM365 a través del puerto de entrada de audio de la misma. El objetivo de lo anterior fue el determinar que tan probable era la detección de cada video.

Cada conjunto de pruebas (30 reproducciones en total) se repitió para cuatro diferentes configuraciones en los parámetros del sistema con el fin de determinar la más adecuada para la aplicación. Los parámetros modificados fueron los siguientes:

- Intervalo de análisis (IA): Este parámetro determina cada cuanto tiempo se realiza una captura de muestras de audio para su análisis.

Identificador	Enlace	Descripción
BEBE1	http://www.youtube.com/ watch?v=qS7nqwGt4-I	Video de un niño llorando.
BEBE2	http://www.youtube.com/ watch?v=laxyoaHOP1c&feature=fvw	Video de gemelos llorando.
BEBE3	http://www.youtube.com/ watch?v=fTh4DCKRVVA&NR=1	Video de un niño llorando.
Video1	http://www.youtube.com/ watch?v=4XpnKHJAok8	Tech Talk: Linus Torvalds on git
Video2	http://www.youtube.com/ watch?v=RmAtVgF9OXs&feature=related	HOUSE M.D SEASON 4 FUNNY MOMENTS
Video3	http://www.youtube.com/ watch?v=XiH7pkCObdA	Small Talk
Video4	http://www.youtube.com/ watch?v=wnZJpPM9YxM	Learn English 13 - Office Talk
Video5	http://www.youtube.com/ watch?v=glrijRGnmc0	How to cook Prime Rib

Cuadro 4.5: Videos de prueba utilizados. (Tomados el 22/12/10)

- Nivel de audio de disparo (ND): Determina el valor promedio de audio mínimo necesario para ser considerado como un posible caso de riesgo e iniciar la transmisión multimedia a los nodos cliente inscritos.

En el cuadro 4.6 se muestran los resultados obtenidos. Como es posible observar, la capacidad de detección del sistema mejoró de forma más notable al disminuir el tiempo de análisis del audio, esto se ve asociado a que entre menor sea dicho intervalo se obtiene mayor detalle de las características del audio del ambiente y por ende, mayor probabilidad de detectar un ruido de corta duración en la banda de frecuencias establecida. En general, el mejor desempeño se obtuvo para un intervalo de análisis de 1s con un nivel de disparo de audio de 1000, por lo cual se eligió dejar esta configuración para las demás pruebas del sistema.

Video/Configuración	IA = 5s ND = 5000	IA = 1s ND = 5000	IA = 5s ND = 1000	IA = 1s ND = 1000
BEBE1	70 %	100 %	70 %	100 %
BEBE2	40 %	70 %	60 %	90 %
BEBE3	70 %	100 %	100 %	100 %

Cuadro 4.6: Porcentaje de aciertos de detección para tres videos de bebés llorando con cuatro configuraciones de sistema.

Utilizando la configuración mencionada se procedió a graficar el nivel promedio de audio para cada uno de los videos en intervalos de 1s, en la figura 4.7 se muestran los gráficos correspondientes a los videos BEBE1, BEBE2 y BEBE3. El intervalo de análisis de 1s permite capturar mayores puntos de audio con valores promedios por encima de 1000, incrementando la probabilidad de detección del video.

4.4.2. Pruebas de respuesta a ambientes comunes

En la sección anterior se determinó y ajustó la sensibilidad del sistema para la detección de niveles de ruido dentro de la banda de frecuencia establecida, el siguiente paso fue corroborar el nivel de inmunidad ante ambientes de ruido comunes donde se desea que el sistema no se dispare. Para lograr esto se hizo uso de los videos VIDEOn los cuales son en términos generales, videos grabados en ambientes de ruido convencionales tales como eventos públicos, oficinas y hogares. El audio de cada video se inyectó en la entrada de audio de la tarjeta para simular la entrada proveniente de un micrófono.

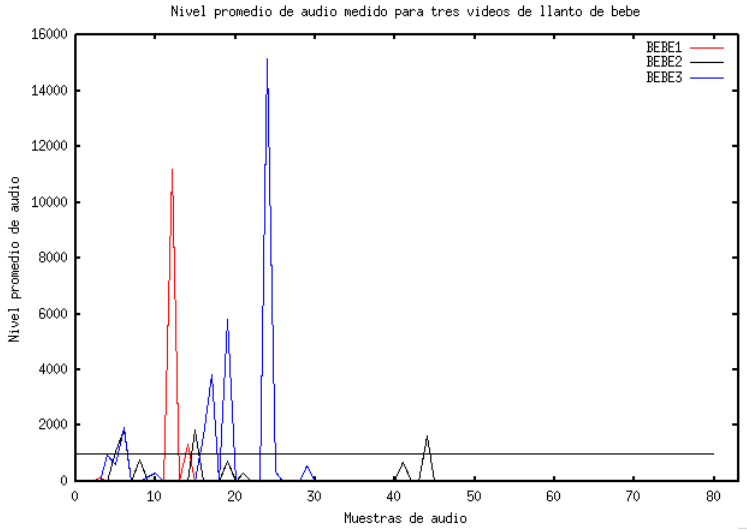


Figura 4.7: Nivel promedio de audio medido para los videos de llanto de bebé. (IA=1s, ND=1000)

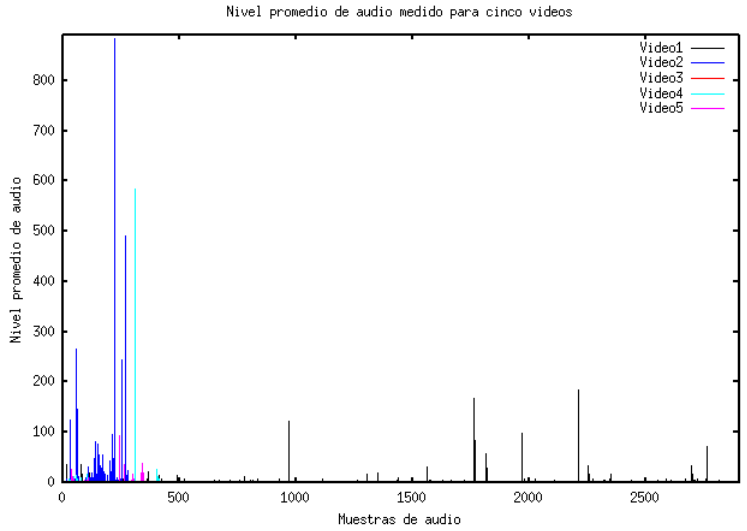


Figura 4.8: Nivel promedio de audio medido para cinco videos diferentes.

En la figura 4.8 se muestra el gráfico del nivel promedio de audio obtenido para cada video en intervalos de 1s. Como es posible observar, para ninguno de los videos utilizados se obtuvo un valor promedio de audio superior a 1000, con lo cual no se dispara el sistema. De esta forma se logró corroborar la capacidad de rechazo del sistema ante ruidos fuera del rango de frecuencia de 3KHz a 4KHz.

Capítulo 5

Conclusiones y recomendaciones

En el presente capítulo se exponen las principales conclusiones referentes al proyecto realizado y se ofrecen algunas recomendaciones para futuros proyectos.

5.1. Conclusiones

- Si se suministra una conexión de red con un ancho de banda mínimo de 5Mbps el sistema es capaz de transmitir audio y video MPEG-4 (con una resolución de 640x480) en multidifusión.
- La resolución, calidad del video y su variabilidad en el tiempo son factores que inciden negativamente en el ancho de banda necesario para realizar una transmisión de red.
- En una transmisión multimedia el video constituye el factor de mayor exigencia respecto al audio en cuanto a requerimientos de la red se refiere, requiriendo, en el caso específico del proyecto expuesto, anchos de banda en el orden de los 5Mbps mientras que el audio necesita un ancho de banda alrededor de 50Kbps para su transmisión.
- Con un intervalo de análisis de 1s y un nivel de disparo de 1000 el sistema es capaz de detectar con mayor fiabilidad (superior a 97 %) ruidos en el rango de 3-4KHz.
- El filtro IIR Elíptico diseñado presentó un factor de forma aproximado de 1.64 con un orden 10, esto demuestra la eficiencia del este tipo de filtros al obtener valores de factor de forma cercanos a la unidad con órdenes pequeños, lo cual a su vez disminuye los requerimientos de recursos necesarios para su implementación.

5.2. Recomendaciones

- El análisis teórico desarrollado y las mediciones obtenidas demostraron que es factible realizar una transmisión de multidifusión de audio y video (con las características descritas a lo largo del documento) en una red inalámbrica con el estándar 802.11g, sin embargo la comprobación de dicha teoría no fue posible debido a un error en el controlador de USB de la tarjeta LeopardBoard DM365. Se recomienda resolver el problema del controlador y realizar un estudio de transmisión de multidifusión para redes inalámbricas.
- El sistema desarrollado presentó un 99 % de uso del CPU para una transmisión de video en movimiento, se recomienta una mejora en los complementos de GStreamer utilizados y del controlador de red con el fin de reducir dichos requerimientos hasta valores aceptables. Entre las posibles mejoras que pueden efectuarse se encuentran las siguientes:
  - El controlador de red utilizado hace uso de *softirq* o interrupciones por software para el procesamiento de los paquetes entrantes, este método consume mucho CPU de forma dedicada (en un contexto de interrupción), lo cual puede interferir con otros procesos del sistema. Se recomienda cambiar dicha lógica por una de *workqueue* o colas de trabajo, esta mejora

permitiría trasladar el procesamiento de los paquetes al contexto de procesos del kernel y por tanto calanderizar su ejecución.

- Es posible que el decodificador de video utilice un elemento *parser* o analizador, el cual se encarga de analizar el video antes de ser decodificado con el fin de detectar o aproximar la ubicación de cuadros. Esto permite enviar al decodificador una menor cantidad de información para la decodificación y así disminuir el procesamiento de buffers de memoria. Sin embargo sería recomendable analizar la lógica del analizador utilizado y de ser posible mejorarla para disminuir aún más dichos procesamientos, debe quedar claro que dicha mejora debe realizarse considerando también que un algorítmo de detección muy complejo en el analizador requerirá de una mayor cantidad de CPU.
- Pueden cambiarse y/o modificarse los complementos de decodificación y encodificación con el fin de disminuir la complejidad de sus algorítmos de compresión.
- El elemento de salida de video *TIDmaiVideoSink* realiza una copia de los buffers del decodificador a un espacio de memoria contiguo para su despliegue, sin embargo, considerando que los buffers de memoria del decodificador también se encuentran dispuestos de forma contigua, es posible modificar el sumidero de video de forma tal que lea los datos directamente de los buffers del decodificador.

Bibliografía

[1] Cisco. Internetworking Technologies Handbook. http://www.cisco.com/en/US/docs/internetworking/technology/handbook/. Consultada: 3 de mayo de 2011.

[2] Hatem Bettahar. Tutorial on Multicast Video Streaming Techniques. Technical report, Heudiasyc, Universite de Technologie de Compiegne, France., 2005.

[3] Boris Felts Alex MacAulay and Yuval Fisher. IP Streaming of MPEG-4: Native RTP vs MPEG-2 Transport Stream. Technical report, Envivio, 2005.

[4] Wim Taymans et al. *GStreamer Application Development Manual (0.10.30.5)*.

[5] Les Thede. *Practical analog and digital filter design*. Artech House Microwave Library. Artech House Publishers, 2005.

[6] J.G. Proakis and D.G Manolakis. *Digital signal processing: principles, algorithms, and applications*. Prentice-Hall International editions. Prentice Hall, 1996.

[7] Vijay K. Madisetti. *The digital signal processing handbook: Video, speech, and audio signal processing and associated standards*. Electrical engineering handbook series. CRC Press, 2010.

[8] Engineering Productivity Tools Ltd. The fft demystified. http://www.engineeringproductivitytools.com/stuff/T0001/.

[9] Julius Smith. *Introduction to Digital Filters: With Audio Applications*. On Demand Publishing, 2008.

[10] KDE TechBase. Introduction to D-BUS. http://techbase.kde.org/Development/Tutorials/D-Bus/Introduction.

[11] Havoc Pennington et al. D-Bus tutorial. http://dbus.freedesktop.org/doc/dbus-tutorial.html#concepts.

[12] Perry R. Cook and Gary P. Scavone. The synthesis ToolKit in c++ (STK). https://ccrma.stanford.edu/software/stk/. Consultada: 3 de mayo de 2011.

[13] David Huron. Frission - Thrills from Chills. http://www.cogsci.msu.edu/DSS/2008-2009/Huron/. Consultada: 3 de mayo de 2011.

[14] John O'Donnell. Digital filter design software - DISPRO. http://www.digitalfilterdesign.com/. Consultada: 3 de mayo de 2011.

[15] Juan L. Posadas y Gines Benet G. Transformada Rapida de Fourier (FFT) e Interpolacion en Tiempo Real. Technical report, Departamento de Informatica de Sistemas y Computadores, Universidad Politecnica de Valencia. Espana., 1998.

[16] Hoo Technologies. Net meter - LAN net meter, WAN net meter, broadband meter, broadband usage meter. http://www.hootech.com/NetMeter/. Consultada: 3 de mayo de 2011.

[17] SourceForge.net. GStreamer daemon. http://sourceforge.net/apps/mediawiki/gstd/index.php?title=Main_Page. Consultada: 3 de mayo de 2011.

Apéndice A

Pruebas de laboratorio: protocolos de transmisión de multidifusión

A.1. Descripción del banco de pruebas

En esta sección se presenta el resultado de las pruebas de diseño realizadas para diferentes protocolos de comunicación punto a multipunto. En general se realizaron pruebas para los protocolos que se listan a continuación:

- RTP/UDP/IP: en este caso se utiliza RTP nativo para realizar la transferencia de datagramas sin el uso de RTCP para el control de la transmisión.
- RTP & RTCP / UDP/IP: se añade RTCP con el objetivo de proporcionar mayor control sobre la transmisión de datos.
- TS/RTP/UDP/IP: en esta prueba se utiliza la paquetización TS sobre RTP con el objetivo de transferir un solo flujo de datos y obtener una mayor sincronización del audio y video.
- TS/UDP/IP: es posible también utilizar la paquetización TS sobre UDP sin el uso de RTP para la transferencia de multidifusión de audio y video.

La pruebas realizadas se llevaron a cabo mediante el uso de computadores personales con las siguientes características:

- Procesador AMD Semprom 140 2.7GHz
- 2GB de memoria RAM
- GStreamer 0.10.28
- Sistema operativo GNU/Linux Ubuntu 10.04

La elección de computadores personales como banco de pruebas fue con motivo de poder ignorar los efectos de la capacidad de procesamiento del sistema en el desempeño de las mismas, dejando por completo ésta variable como un factor directamente dependiente de los protocolos utilizados.

Por otra parte, todas las pruebas realizadas en este apéndice y los posteriores (a menos que se indique lo contrario) son realizadas utilizando una conexión de red cableada y un router/switch WRT54G de Linksys.

A.2. Pruebas realizadas

Las pruebas realizadas corresponden a transmisiones de multidifusión de dos videos de 640x480 píxeles en los formatos H264 y MPEG-4. Debido a la similitud en los resultados obtenidos será indistinto el formato utilizado en la presentación de las tuberías probados y los resultados obtenidos.

A continuación se describen las tuberías utilizados en cada una de las pruebas realizadas.

RTP/UDP/IP

- Servidor

VIDEO_FILE MULTICAST_GROUP_IP VIDEO_UDP_PORT AUDIO_UDP_PORT gst-launch filesrc location =$VIDEO_FILE ! qtdemux name=dmux dmux. ! queue ! rtpmp4vpay ! udpsink host=$MULTICAST_GROUP_IP port=VIDEO_UDP_PORT auto-multicast=true -v dmux. ! queue ! faad ! audioconvert ! audioresample ! mulawenc ! rtppcmupay ! udpsink host=$MULTICAST_GROUP_IP port=AUDIO_UDP_PORT auto-multicast=true -v

Cuadro A.1: Tubería de servidor para RTP/UDP/IP.

- Cliente

MULTICAST_GROUP_IP VIDEO_UDP_PORT AUDIO_UDP_PORT gst-launch-0.10 udpsrc multicast-group=$MULTICAST_GROUP_IP auto-multicast=true port=$VIDEO_UDP_PORT caps = 'application/x-rtp, media=(string)video, clock-rate=(int)90000, encoding-name=(string)MP4V-ES, profile-level-id=(string)1, config=(string)000001b001000001b589130000010000000012000c48d885dad16843c1463, payload=(int)96, ssrc=(guint)2021670075, clock-base=(guint)3694278979, seqnum-base=(guint)11413' ! rtpmp4vdepay ! queue ! dmaidec_mpeg4 ! TIDmaiVideoSink videoOutput=Component videoStd=720P_60 gst-launch-0.10 udpsrc multicast-group=$MULTICAST_GROUP_IP auto-multicast=true port=$AUDIO_UDP_PORT caps = 'application/x-rtp, media=(string)audio, clock-rate=(int)8000, encoding-name=(string)PCMU, payload=(int)0, ssrc=(guint)821301517, clock-base=(guint)3711788766, seqnum-base=(guint)21991' ! rtppcmudepay ! decodebin ! alsasink

Cuadro A.2: Tubería de cliente para RTP/UDP/IP.

RTP & RTCP / UDP/IP

- Servidor

VIDEO_FILE
MULTICAST_GROUP_IP
VIDEO_UDP_PORT
AUDIO_UDP_PORT

gst-launch gstrtpbin name=rtpbin filesrc location=$VIDEO_FILE ! queue ! qtdemux name=dmux dmux. ! queue ! rtph264pay ! rtpbin.send_rtp_sink_2 rtpbin.send_rtp_src_2 ! udpsink auto-multicast=true host=$MULTICAST_GROUP_IP port=$VIDEO_UDP_PORT rtpbin.send_rtcp_src_2 ! udpsink auto-multicast=true host=$MULTICAST_GROUP_IP port=6003 sync=false async=false

udpsrc multicast-group=$MULTICAST_GROUP_IP auto-multicast=true port=6007 ! rtpbin.recv_rtcp_sink_2 dmux. ! queue ! mad ! audioconvert ! audioresample ! mulawenc ! rtppcmupay ! rtpbin.send_rtp_sink_1 rtpbin.send_rtp_src_1 ! udpsink auto-multicast=true host=$MULTICAST_GROUP_IP port=$AUDIO_UDP_PORT rtpbin.send_rtcp_src_1 ! udpsink auto-multicast=true host=$MULTICAST_GROUP_IP port=5003 sync=false async=false udpsrc multicast-group=$MULTICAST_GROUP_IP auto-multicast=true port=5007 ! rtpbin.recv_rtcp_sink_1 -v

Cuadro A.3: Tubería de servidor para RTP & RTCP/UDP/IP

- Cliente

MULTICAST_GROUP_IP
VIDEO_UDP_PORT
AUDIO_UDP_PORT

gst-launch -v gstrtpbin name=rtpbin udpsrc multicast-group=$MULTICAST_GROUP_IP auto-multicast=true caps='application/x-rtp, media=(string)video, clock-rate=(int)90000, encoding-name=(string)H264, profile-level-id=(string)42c033, sprop-parameter-sets=(string)\"Z0LAM6tAUB7YCIAAAAMAgAAAHkeMGVA\\=\\,aM48gA\\=\\=\", ssrc=(guint)4545098, \ payload=(int)96, clock-base=(guint)2162236596, seqnum-base=(guint)13009' port=$VIDEO_UDP_PORT ! rtpbin.recv_rtp_sink_2 rtpbin. ! rtph264depay ! decodebin ! ffmpegcolorspace ! ximagesink udpsrc port=6003 multicast-group=$MULTICAST_GROUP_IP auto-multicast=true ! rtpbin.recv_rtcp_sink_2 rtpbin.send_rtcp_src_2 ! udpsink host=$MULTICAST_GROUP_IP auto-multicast=true port=6007 sync=false async=false udpsrc multicast-group=$MULTICAST_GROUP_IP auto-multicast=true caps='application/x-rtp, media=(string)audio, clock-rate=(int)8000, encoding-name=(string)PCMU, ssrc=(guint)3613868683, payload=(int)0, clock-base=(guint)2319507683, seqnum-base=(guint)13087' port=$AUDIO_UDP_PORT ! rtpbin.recv_rtp_sink_1 rtpbin. ! rtppcmudepay ! pulsesink udpsrc port=5003 multicast-group=$MULTICAST_GROUP_IP auto-multicast=true ! rtpbin.recv_rtcp_sink_1 rtpbin.send_rtcp_src_1 ! udpsink host=$MULTICAST_GROUP_IP auto-multicast=true port=5007 sync=false async=false

Cuadro A.4: Tubería de cliente para RTP & RTCP/UDP/IP.

TS/RTP/UDP/IP

- Servidor

VIDEO_FILE
MULTICAST_GROUP_IP
UDP_PORT

gst-launch filesrc location=$VIDEO_FILE ! qtdemux name=dem dem. ! queue ! mpegtsmux name=ts ! rtpmp2tpay ! udpsink host=$MULTICAST_GROUP_IP port=$UDP_PORT auto-multicast=true dem. ! queue ! ts. -v

Cuadro A.5: Tubería de servidor para TS/RTP/UDP/IP.

- Cliente

MULTICAST_GROUP_IP
UDP_PORT

gst-launch-0.10 udpsrc multicast-group=$MULTICAST_GROUP_IP auto-multicast=true port=$UDP_PORT ! 'application/x-rtp, media=(string)video,clock-rate=(int)90000, encoding-name=(string)MP2T-ES, payload=(int)33, ssrc=(guint)1773583858, clock-base=(guint)2767248738, seqnum-base=(guint)46945' ! rtpmp2tdepay ! mpegtsdemux name=d d. ! queue ! decodebin ! xvimagesink d. ! queue ! decodebin ! pulsesink -v

Cuadro A.6: Tubería de cliente para TS/RTP/UDP/IP.

TS/UDP/IP

- Servidor

VIDEO_FILE
MULTICAST_GROUP_IP
UDP_PORT

gst-launch filesrc location=$VIDEO_FILE ! qtdemux name=dem dem. ! queue ! mpegtsmux name=ts ! udpsink host=$MULTICAST_GROUP_IP port=$UDP_PORT auto-multicast=true dem. ! queue ! ts. -v

Cuadro A.7: Tubería de servidor para TS/UDP/IP.

- Cliente

MULTICAST_GROUP_IP
UDP_PORT

gst-launch-0.10 udpsrc multicast-group=$MULTICAST_GROUP_IP auto-multicast=true port=$UDP_PORT ! 'application/x-rtp, media=(string)video,clock-rate=(int)90000, encoding-name=(string)MP2T-ES, payload=(int)33, ssrc=(guint)1773583858, clock-base=(guint)2767248738, seqnum-base=(guint)46945' ! mpegtsdemux name=d d. ! queue ! decodebin ! xvimagesink d. ! queue ! decodebin ! pulsesink -v

Cuadro A.8: Tubería de cliente para TS/UDP/IP.

A.3. Resultados obtenidos

Las pruebas se llevaron a cabo con el objetivo de evaluar cuatro aspectos fundamentales del rendimiento de las transmisiones de multidifusión a saber:

- Calidad del video recibido.
- Sincronismo entre el audio y video.
- Sincronismo entre el audio de dos clientes.
- Estabilidad de ejecución.

A continuación se exponen los resultados obtenidos para cada una de las pruebas realizadas.

RTP/UDP/IP

En lo que respecta a la calidad del video recibido no se detectó ningún inconveniente, el video se reprodujo de forma fluida. De igual forma la sincronización entre el audio y el video no presentó ningún problema.

En la figura A.1 se muestra la captura de la forma de onda de la señal de audio recibida por dos clientes de forma simultánea, como es posible observar se obtuvo un ligero desplazamiento de aproximadamente 30ms entre ambas. A pesar que parte de este desplazamiento puede atribuirse a la diferencia de tiempo de ejecución de ambos computadores, es claro que existe una diferencia de tiempo en la llegada de ambas transmisiones a los nodos cliente.

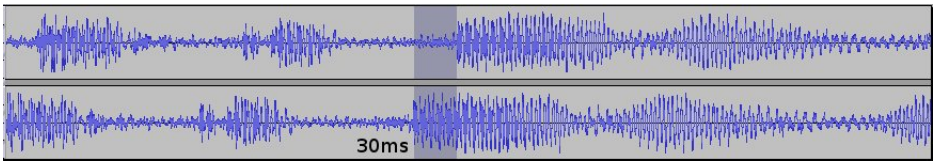


Figura A.1: Señales de audio capturadas desde dos computadores utilizando RTP nativo.

Las tuberías utilizados mostraron ser estables al permitir su reinicio en cualquier condición sin desmejoras en la calidad del medio entregado.

RTP & RTCP /UDP/IP

Al agregar un canal de control de flujo RTCP no se experimentó ningún inconveniente con la calidad del video reproducido ni su sincronía con el audio para un mismo cliente, sin embargo la sincronía entre el audio de dos clientes presentó un desplazamiento considerable de aproximadamente 230ms como se muestra en la figura A.2.

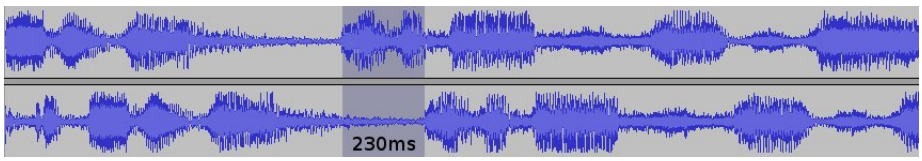


Figura A.2: Señales de audio capturadas desde dos computadores utilizando RTP y RTCP.

En cuanto a estabilidad las tuberías no mostraron ningún inconveniente en su ejecución.

TS/RTP/UDP/IP

La calidad del video y su sincronía con el audio no presentaron inconvenientes para el caso en que se ejecute primero el servidor y después el cliente, en caso contrario se presentaron problemas de sincronización que impedían la reproducción de los medios.

Debido a que para la medición de la sincronía entre el audio de dos clientes era necesario iniciar las tuberías en estos antes que el servidor, no fue posible tomar dicha medición.

Tal y como se comentó antes, la estabilidad de las tuberías se encontró directamente ligada al orden de ejecución de éstos, aún más, presentaron ser inestables cuando se inicia el cliente primero, requisito crítico en muchas aplicaciones de multidifusión donde se requiere que los nodos cliente se encuentren en espera de una transmisión multidifusión.

TS/UDP/IP

Este modelo de transmisión posee una característica particular, en lugar de realizar la transferencia de datos en conjuntos de bytes altos como por ejemplo el caso de RTP, éstos son transmitidos en paquetes de 188 bytes lo cual genera un mayor estrés de transmisión y de procesamiento. La gran cantidad de paquetes transmitidos incrementa también la cantidad de paquetes perdidos en la transmisión y sus efectos sobre la calidad del medio recibido, esto se muestra en la Figura A.3 donde se presenta una captura del video recibido, como es posible observar la pérdida de paquetes tiene efectos mucho más sensibles en este mecanismo en comparación con los expuestos antes.

Figura A.3: Efectos de la pérdida de paquetes en la calidad del video recibido.

Por otro lado se presentaron problemas en la sincronización del audio y el video así como el problema de orden de ejecución descrito en la prueba de TS/RTP/UDP/IP, lo cual impidió nuevamente la medición de la sincronización del audio entre dos clientes.

Esta prueba presentó la peor estabilidad debido a que no fue posible obtener una transmisión de los medios de forma adecuada en ningún caso.

Apéndice B

Formatos multimedia para transmisión de multidifusión

En esta sección se presentan las mediciones realizadas para la selección de los formatos multimedia a utilizar en las transmisiones de multidifusión. En general se consideraron dos aspectos a saber:

- Consumo de CPU de las tuberías tanto para la codificación como para la decodificación del contenido multimedia.
- Requerimientos de ancho de banda para una transmisión de multidifusión.

A continuación se exponen las pruebas realizadas y las principales conclusiones obtenidas.

B.1. Formatos de video

Debido a que para la tarjeta LeopardBoard DM365 se cuenta con plugins de GStreamer para los formatos H264 y MPEG-4 específicos para la plataforma, se decidió realizar las pruebas de video solamente con estos formatos.

En general las pruebas consistieron en una transmisión de multidifusión de un video a través de una red cableada. Se expondrán dos conjuntos de pruebas, el primer conjunto fue realizado entre dos computadores personales con el fin de obtener una idea inicial de los resultados mientras que el segundo conjunto fue realizado en la tarjeta LeopardBoard DM365 con el fin de corroborar la proyección de los datos.

B.1.1. Conjunto de pruebas en computador

A continuación se exponen las pruebas realizadas para los formatos H264 y MPEG-4 para un computador personal. El video utilizado es el video de prueba suministrado por el elemento *videotestsrc* a una resolución de 640x480 píxeles.

- H264

La tubería de servidor utilizado es la siguiente:

```
MULTICAST_GROUP
PORT_NUMBER

gst-launch videotestsrc ! video/x-raw-yuv, width=640, height=480 ! queue ! x264enc cabac=false !
rtph264pay ! udpsink host=$(MULTICAST_GROUP) port=$(PORT_NUMBER)
auto-multicast=true
```

Cuadro B.1: Tubería de servidor para un video de H264 en un computador.

Por otro lado el cliente se encuentra descrito por la siguiente tubería:

```
MULTICAST_GROUP
PORT_NUMBER

gst-launch udpsrc multicast-group=$(MULTICAST_GROUP) auto-multicast=true
port=$(PORT_NUMBER) caps='application/x-rtp, media=(string)video, clock-rate=(int)90000,
encoding-name=(string)H264, sprop-parameter-
sets=(string)\"Z0LAFZJUCg/YCIAAAAMAgAAAHgeLF1A\\=\\,aM48gA\\=\\=\",
payload=(int)96, ssrc=(guint)721886401, clock-base=(guint)3890627210,
seqnum-base=(guint)40574' ! rtph264depay ! queue ! ffdec_h264 ! xvimagesink
```

Cuadro B.2: Tubería de cliente para un video de H264 en un computador.

En el cuadro B.3 se muestran los resultados obtenidos, la medición del ancho de banda fue realizada utilizando el programa NetMeter^{TM}[16].

Parámetro medido	Valor
Consumo de CPU servidor	48 %
Consumo de CPU cliente	15 %
Ancho de banda promedio	2.2 Mbps

Cuadro B.3: Resultados de la prueba en computador con el formato de video H264.

- MPEG-4

La tubería del servidor esta dada por:

```
MULTICAST_GROUP
PORT_NUMBER

gst-launch videotestsrc ! video/x-raw-yuv, width=640, height=480 ! queue ! ffenc_mpeg4 !
rtpmp4vpay ! udpsink host=$(MULTICAST_GROUP) port=$(PORT_NUMBER)
auto-multicast=true
```

Cuadro B.4: Tubería de servidor para un video de MPEG-4 en un computador.

La tubería del cliente utilizado esta dada por:

```
MULTICAST_GROUP
PORT_NUMBER

gst-launch udpsrc multicast-group=$(MULTICAST_GROUP) auto-multicast=true
port=$(PORT_NUMBER) caps='application/x-rtp, media=(string)video, clock-rate=(int)90000,
encoding-name=(string)MP4V-ES, profile-level-id=(string)1, con-
fig=(string)000001b001000001b589130000010000000012000c48d8800f50a041e1463000001b24c6176633
5322e32302e31, payload=(int)96, ssrc=(guint)3673003540, clock-base=(guint)2341200076,
seqnum-base=(guint)12212' ! rtpmp4vdepay ! queue ! ffdec_mpeg4 ! xvimagesink
```

Cuadro B.5: Tubería de cliente para un video de MPEG-4 en un computador.

Parámetro medido	Valor
Consumo de CPU servidor	42 %
Consumo de CPU cliente	10 %
Ancho de banda promedio	1.5 Mbps

Cuadro B.6: Resultados de la prueba en computador con el formato de video MPEG-4.

En el cuadro B.6 se exponen los resultados obtenidos.

B.1.2. Conjunto de pruebas en la tarjeta LeopardBoard DM365

Una vez realizadas las pruebas en computador, se llevaron a cabo de igual forma en la tarjeta LeopardBoard DM365, sustituyendo el elemento *videotestsrc* por *v4l2src* para capturar la imagen de video directamente del sensor de cámara de la tarjeta. Los datos obtenidos corresponden a una grabación de un escenario estático, esto permite obtener los valores mínimos de los parámetros medidos.

- H264

Tubería utilizada para el nodo servidor:

MULTICAST_GROUP PORT_NUMBER gst-launch v4l2src always-copy=FALSE num-buffers=-1 ! video/x-raw-yuv,format=\(fourcc\)NV12, width=640, height=480, framerate=\(fraction\)30/1 ! dmaiaccel ! queue ! dmaienc_h264 ! rtph264pay ! udpsink host=$(MULTICAST_GROUP) port=$(PORT_NUMBER) auto-multicast=true

Cuadro B.7: Tubería de servidor para un video de H264 en la LeopardBoard DM365.

La tubería del nodo cliente es la siguiente:

MULTICAST_GROUP PORT_NUMBER gst-launch udpsrc port=$(port_number) multicast-group=$(multicast_group) auto-multicast=true timeout=$(time) ! application/x-rtp, media=(string)video, clock-rate=(int)90000, encoding-name=(string)MP4V-ES, profile-level-id=(string)5, con- fig=(string)000001b005000001b50ecf0000010000001200086e0002ea6600b7c52c999cba98514043c1463, payload=(int)96, ssrc=(guint)1226930637, clock-base=(guint)344275943, seqnum-base=(guint)15648 ! rtpmp4vdepay ! queue ! dmaidec_mpeg4 ! TIDmaiVideoSink videoOutput=component sync=false accelFrameCopy=true videoStd=720P_60 noCopy=true

Cuadro B.8: Tubería de cliente para un video de H264 en la LeopardBoard DM365.

En el cuadro B.9 se muestran los resultados obtenidos.

- MPEG-4

La tubería del servidor esta dada por:

Parámetro medido	Valor
Consumo de CPU servidor	61 %
Consumo de CPU cliente	65 %

Cuadro B.9: Resultados de la prueba en la tarjeta con el formato de video H264.

```
MULTICAST_GROUP
PORT_NUMBER

gst-launch v4l2src always-copy=FALSE ! video/x-raw-yuv,format=(fourcc)NV12, width=640,
height=480, framerate=(fraction)30/1 ! dmaiaccel ! queue ! dmaienc_mpeg4 ! rtpmp4vpay !
udpsink host=$(MULTICAST_GROUP) port=$(PORT_NUMBER) auto-multicast=true
```

Cuadro B.10: Tubería de servidor para un video de MPEG-4 en la LeopardBoard DM365.

Y la tubería del nodo cliente esta dada por:

```
MULTICAST_GROUP
PORT_NUMBER

gst-launch udpsrc port=3000 multicast-group=224.1.1.1 auto-multicast=true
caps="application/x-rtp, media=(string)video, clock-rate=(int)90000,
encoding-name=(string)H264, profile-level-id=(string)640028, sprop-parameter-
sets=(string)\\"J2QAKK2ECSZuIzSQgSTNxGaSECSZuIzSQgSTNxGaSECSZuIzSQgSTNxGaSEF
WuvX1+T+vyfXrrVQgq116+vyf1+T69daq0BQHsg\\=\\,KO48sA\\=\\=\\=\\", payload=(int)96,
ssrc=(guint)3346865707, clock-base=(guint)998465529, seqnum-base=(guint)29866" !
rtph264depay ! queue ! dmaidec_h264 numOutputBufs=6 ! TIDmaiVideoSink
videoOutput=component sync=false accelFrameCopy=true videoStd=720P_60 noCopy=true
```

Cuadro B.11: Tubería de cliente para un video de MPEG-4 en la LeopardBoard DM365..

En el cuadro B.12 se muestran los resultados obtenidos de las pruebas.

Parámetro medido	Valor
Consumo de CPU servidor	26 %
Consumo de CPU cliente	40 %

Cuadro B.12: Resultados de la prueba en la tarjeta con el formato de video MPEG-4.

B.2. Formatos de audio

El formato de audio considerado para la transmisión de multidifusión fue *mu-law*. A continuación se presentan las tuberías de audio utilizados y los resultados obtenidos.

La tubería servidor utilizada es la siguiente:

```
MULTICAST_GROUP
PORT_NUMBER

gst-launch audiotestsrc ! queue ! mulawenc ! rtppcmupay ! udpsink
host=$(MULTICAST_GROUP) port=$(PORT_NUMBER) auto-multicast=true
```

Cuadro B.13: Tubería de servidor para audio en formato mu-law.

Por otro lado la tubería de recepción esta dada por:

MULTICAST_GROUP PORT_NUMBER gst-launch udpsrc multicast-group=$(MULTICAST_GROUP) auto-multicast=true port=$(PORT_NUMBER) caps='application/x-rtp, media=(string)audio, clock-rate=(int)8000, encoding-name=(string)PCMU, payload=(int)0, ssrc=(guint)187937749, clock-base=(guint)3965193198, seqnum-base=(guint)24532' ! rtppcmudepay ! queue ! mulawdec ! pulsesink

Cuadro B.14: Tubería de cliente para audio en formato mu-law.

En el cuadro B.15 se muestran los resultados obtenidos con las tuberías mostradas.

Parámetro medido	Valor
Consumo de CPU servidor	0.7 %
Consumo de CPU cliente	1 %
Ancho de banda promedio	80.1kbps

Cuadro B.15: Resultados de la prueba en la tarjeta con el formato de video MPEG-4.

B.3. Discusión de los resultados

Como ha sido posible observar, en lo que respecta a video MPEG-4 proporcionó el mejor rendimiento al requerir de menos CPU y ancho de banda que H264, además es posible concluir que para poder realizar una transmisión de multidifusión con audio y video estático (en formato MPEG-4) simultáneo es necesario como mínimo poseer un ancho de banda de aproximadamente 2Mbps.

Apéndice C

Manual de uso de las interfaces gráficas implementadas

En esta sección se presenta una guía rápida de uso de la aplicación desarrollada.

C.1. Manual de usuario del nodo servidor

En la figura C.1 se muestra una captura de la interfaz de usuario (GUI) del nodo servidor. En el cuadro C.1 se describen las diferentes secciones de la interfaz, enumeradas en la figura C.1.

Debe considerarse que los parámetros de configuración de la aplicación son accesibles únicamente cuando ésta se encuentra apagada.

NOTA: No debe de seleccionarse el mismo número de puerto para la transmisión de video y audio, esto provocaría un conflicto y una ejecución errónea de la aplicación.

C.2. Manual de usuario del nodo cliente

En la figura C.3 se muestra una captura de la interfaz gráfica del nodo cliente, como es posible observar es similar a la presentada para el nodo servidor. En el Cuadro C.2 se detallan las características de cada uno de los elementos que componen la interfaz.

Un aspecto que debe considerarse es que los parámetros de configuración son modificables únicamente cuando la aplicación se encuentra apagada.

Una vez sea detectada una transmisión de multidifusión en el grupo y puertos especificados se realizará una transparencia en la zona de la imagen para permitir mostrar el vídeo recibido.

NOTA: No debe de seleccionarse el mismo número de puerto para la transmisión de video y audio, esto provocaría un conflicto y una ejecución errónea de la aplicación.

Figura C.1: Interfaz de usuario del nodo servidor.

Número	Nombre	Descripción
1	Acerca de.	Esta pestaña contiene información acerca de la autoría del proyecto.
2	Configuración	Corresponde a la pestaña de configuración, en ésta podrá encontrarse una serie de opciones para la transferencia de multidifusión del audio y video.
3	Encendido/Apagado	El botón de encendido/apagado permite encender o apagar la aplicación, esto es, detiene el algoritmo de detección de niveles acústicos o lo reinicia.
4	Grupo de multidifusión	Esta lista desplegable permite elegir el grupo de multidifusión al cual se desea realizar la trasmisión del audio y video.
5	Puerto de audio	Esta lista desplegable permite seleccionar un puerto de transmisión del contenido de audio.
6	Puerto de video	Esta lista desplegable permite seleccionar un puerto de transmisión del contenido de video.
7	Imagen de estado	Esta imagen muestra el estado actual de la aplicación según sea: Apagada, Encendida, Transmitiendo. En la figura C.2 se muestran las diferentes imágenes relacionadas con cada estado.

Cuadro C.1: Descripción de las opciones de la GUI del nodo servidor.

(a) Apagado (b) Encendido (c) Transmitiendo

Figura C.2: Imágenes de estado del nodo servidor.

Figura C.3: Interfaz de usuario del nodo cliente.

Número	Nombre	Descripción
1	Acerca de.	Esta pestaña contiene información acerca de la autoría del proyecto.
2	Configuración	Corresponde a la pestaña de configuración, en ésta podrá encontrarse una serie de opciones para la recepción de multidifusión del audio y video.
3	Encendido/Apagado	El botón de encendido/apagado permite encender o apagar la aplicación esto es, detiene el algoritmo de detección de niveles acústicos o lo reinicia.
4	Grupo de multidifusión	Esta lista desplegable permite elegir el grupo de multidifusión al cual se desea realizar la trasmisión del audio y video.
5	Puerto de audio	Esta lista desplegable permite seleccionar un puerto de transmisión del contenido de audio.
6	Puerto de video	Esta lista desplegable permite seleccionar un puerto de transmisión del contenido de video.
7	Ajuste de volumen	Este ajuste permite variar el volumen del audio reproducido en una escala lineal de 0 a 10 donde 10 corresponde al máximo volumen.
8	Etiqueta de volumen	Este campo muestra el volumen actual al que se ha ajustado la reproducción.
9	Imagen de estado	Esta imagen muestra el estado actual de la aplicación según sea: Apagada o Encendida de acuerdo con las imágenes mostradas en la figura C.2.

Cuadro C.2: Descripción de las opciones de la GUI del nodo cliente.