

TRABAJO ESPECIAL DE GRADO

DESARROLLO DE UN CONTROLADOR USB PARA UN SISTEMA DE COMUNICACIONES DE TELEVISIÓN DIGITAL ABIERTA EN GNU/LINUX

Presentado ante la ilustre
Universidad Central de Venezuela
por el Br. Edgar Gómez
para optar al título
de Ingeniero Electricista.

Caracas, de 2017

TRABAJO ESPECIAL DE GRADO

DESARROLLO DE UN CONTROLADOR USB PARA UN SISTEMA DE COMUNICACIONES DE TELEVISIÓN DIGITAL ABIERTA EN GNU/LINUX

TUTOR ACADÉMICO: Ebert Brea

TUTOR INDUSTRIAL: Carlelines Gavidia

Presentado ante la ilustre
Universidad Central de Venezuela
por el Br. Edgar Gómez
para optar al título
de Ingeniero Electricista.

Caracas, de 2017

CONSTANCIA DE APROBACIÓN

Caracas, de de 2017

Los abajo firmantes, miembros del Jurado designado por el Consejo de Escuela de Ingeniería Eléctrica, para evaluar el Trabajo Especial de Grado presentado por el Bachiller Edgar J. Gomez R., titulado:

DESARROLLO DE UN CONTROLADOR USB PARA UN SISTEMA DE COMUNICACIONES DE TELEVISIÓN DIGITAL ABIERTA EN GNU/LINUX

Consideran que el mismo cumple con los requisitos exigidos por el de estudios conducente al Título de Ingeniero Electricista en la mención Electrónica y Control, y sin que ello signifique que se hacen solidarios con las ideas expuestas por el autor, lo declaran APROBADO.

Prof.

Jurado

Prof.

Jurado

Prof.

Prof. Guía

RECONOCIMIENTOS Y AGRADECIMIENTOS

A mis Padres por su apoyo incondicional y palabras de aliento.

A mi hermana por estar siempre a mi lado brindándome su apoyo.

A mis familiares, abuelos, tíos y primos que me ayudaron durante todo el trayecto.

A mis amigos por compartir los momentos y por su ayuda incondicional, aunque no los nombro a cada uno ellos saben quiénes son.

A la Universidad Central de Venezuela por permitirme desarrollarme como persona y profesional ha sido una oportunidad invaluable.

A los profesores de la Facultad de Ingeniería y de la escuela de Ingeniería Eléctrica de la UCV por su labor de formación y su actitud desinteresada a la hora de impartir conocimientos, sin personas tan dedicadas como ellos no hubiera sido posible.

A la fundación CENDIT y a sus trabajadores, por apoyarme y permitir desarrollar este trabajo de grado, en especial a Jarson Manjares por su apoyo.

A los ayudantes anónimos del mundo del software que prestan su ayuda sin ninguna retribución a cambio, los cuales brindan una gran ayuda para quien esté dispuesto a pedirla.

RESUMEN

Gómez R. Edgar J.

DESARROLLO DE UN CONTROLADOR USB PARA UN SISTEMA DE COMUNICACIONES DE TELEVISIÓN DIGITAL ABIERTA EN GNU/LINUX

Tutor Académico: Dr. Ebert Brea.

Tutor Industrial: Ing. Carlelines Gavidia.

Palabras Claves: USB, Televisión Digital Abierta, Controlador, VLC, Linux, Software Libre.

Resumen.- Tanto la Televisión Digital Abierta como la incursión en el software libre son dos banderas del desarrollo tecnológico del estado venezolano con lo cual se busca la inclusión de la población en el acceso a las tecnologías y en el ejercicio del derecho a estar informados.

Un dispositivo de TDA móvil capaz de conectarse a un computador vía USB es un puente que une dos tecnologías enfocadas a la información, usando un sistema operativo libre como GNU/Linux podría accederse al contenido de la TDA a través del computador de forma gratuita y sin restricciones, actualmente la fundación CENDIT ha desarrollado un dispositivo móvil de TDA que mediante un software privativo es posible de visualizar el contenido multimedia de la TDA bajo el sistema operativo Windows 7 usando el programa de reproducción multimedia VLC.

El objetivo de este proyecto es desarrollar una rutina de control USB para el dispositivo móvil de TDA bajo el sistema operativo GNU/Linux capaz de realizar funciones de control sobre el dispositivo, la obtención mediante el bus USB del contenido multimedia y la reproducción utilizando el programa de reproducción multimedia VLC el cual es un programa libre.

ÍNDICE

CONSTANCIA DE APROBACIÓN.....	I
RECONOCIMIENTOS Y AGRADECIMIENTOS.....	II
RESUMEN.....	III
ÍNDICE.....	IV
LISTA DE FIGURAS.....	VI
LISTA DE TABLAS.....	VII
LISTA DE SIGLAS Y ACRÓNIMOS.....	VIII
INTRODUCCIÓN.....	1
CAPÍTULO I.....	3
PLANTEAMIENTO DEL PROBLEMA.....	3
MARCO REFERENCIAL.....	6
1.1 Dongle USB.....	6
1.1.1 Modelo del Hardware.....	7
1.1.1.1 Tuner NIM DNOD22QXV104A.....	8
1.1.1.2 Controlador USB CY7C68013A.....	9
1.1.1.3 Prototipo de Prueba.....	11
1.1.1.4 Prototipo Alpha.....	13
1.1.2 El firmware USB.....	15
1.1.3 El software C# y controladores USB privativos para Windows	19
CAPÍTULO II.....	21
OBJETIVOS.....	21
2.1 Objetivo General.....	21
2.2 Objetivos Específicos.....	21
CAPÍTULO III.....	22
MARCO TEÓRICO.....	22
3.1 Televisión Digital Abierta.....	22
3.1.1 Televisión Digital Terrestre.....	23
3.1.2 La Multiprogramación.....	26
3.1.3 El estándar ISDB-Tb.....	30
3.1.4 El flujo de datos BTS.....	34
4.1 El USB.....	36
4.1.1 Versiones USB.....	37
4.1.1 Beneficios del USB.....	38
4.1.3 Endpoint USB.....	42
4.1.4 Descriptores USB.....	43
4.1.5 Tipos transferencias USB.....	52
4.1.5.1 Transferencias de Tipo Control.....	52

4.1.5.2 Transferencias de Tipo Masivas.....	58
4.1.5.3 Transferencias de Tipo Interrupción.....	59
4.1.5.3 Transferencias de Tipo Isócronas.....	62
5.1 El sistema operativo GNU/Linux.....	62
CAPÍTULO IV.....	67
MARCO METODOLÓGICO.....	67
6.1 Utilidades de Software.....	67
6.1.1 Esquema de Software.....	69
7.1 La API de Samsung.....	70
7.1.1 Funciones del demodulador SemcoTC90527.....	70
7.1.2 Funciones del sintonizador SemcoSTV4100.....	76
7.1.3 Funciones definidas por el desarrollador.....	78
7.1.4 Procesos de inicialización y sintonización.....	78
8.1 GNU/Linux y el Dongel USB de TDA.....	83
9.1 La API Libusb.....	87
9.1.1 Transferencias asincrónicas.....	100
9.1.2 Rutinas USB.....	101
9.1.2.1 Rutinas de Conexión en Caliente.....	101
9.1.2.1 Rutinas de Transferencias de Control.....	105
9.1.2.1 Rutinas de Transferencias de Interrupción.....	111
10.1 La API Libvlc.....	116
10.1.1 Rutinas Multimedia.....	123
11.1 Interfaz Gráfica.....	128
11.1.1 Empaquetamiento gráfico.....	128
11.1.1 Funciones de Enlace entorno Gráfico.....	133
12.1 Instalador del Programa.....	138
12.1.1 Paquetes y software requerido.....	140
12.1.2 Proceso de compilación.....	141
12.1.3 Reglas Udev para el Dongle USB.....	142
12.1.4 Acceso directo.....	143
12.1.5 Script de instalación del programa.....	144
12.1.6 Script de desinstalación del programa.....	146
13.1 Documentación.....	146
CAPÍTULO V.....	152
ANÁLISIS DE RESULTADOS.....	152
14.1 Ejecución del instalador del programa.....	152
14.2 Montaje físico de los equipos.....	159
14.3 Pruebas de la aplicación desarrollada.....	162
14.4 Ejecución del script de desinstalación del programa.....	168
CAPÍTULO VI.....	170
CONCLUSIONES Y RECOMENDACIONES.....	170
REFERENCIAS BIBLIOGRÁFICAS.....	172
ANEXOS.....	175

LISTA DE FIGURAS

Figura 1.1.1: Composición de un sistema USB de TDA.	6
Figura 1.1.2: Diagrama de hardware, comunicaciones y flujo de datos.	8
Figura 1.1.3: Representación del hardware Tuner NIM.	9
Figura 1.1.4: Diagrama de bloques del dispositivo USB CY7C68013A.	10
Figura 1.1.5: Placa de desarrollo del dispositivo USB CY7C68013A.	11
Figura 1.1.6: Acople del módulo de sintonización con la tarjeta de desarrollo. .	11
Figura 1.1.7: Diseño del PCB del módulo de sintonización ISDB-Tb.	12
Figura 1.1.8: Modelado en 3D del módulo de sintonización ISDB-Tb.	13
Figura 1.1.9: Diseño del PCB del Prototipo Alfa.	14
Figura 1.2.1: Modelado 3D del Prototipo Alfa.	15
Figura 1.2.2: Funcionamiento del firmware en el microcontrolador CY7C68013A.	17
Figura 1.2.3: Funciones para comunicarse con el Tuner NIM desde el firmware.	18
Figura 1.2.4: IDE C# desarrollo del software privativo para el Dongle USB de TDA.	19
Figura 1.2.5: Aplicación desarrollada en Windows para el Dongle USB de TDA en funcionamiento.	20
Figura 3.1.1: Ilustración anchura banda de un canal.	27
Figura 3.1.2: Área de cobertura de TDA en Venezuela.	29
Figura 3.1.3: Estándares de TVD-T en los distintos países del mundo.	31
Figura 3.1.4: Bloques funcionales de un transmisor de TVD-T.	32
Figura 3.1.5: Diagrama funcional simplificado de una estación de Televisión Digital.	35
Figura 3.1.6: Estructura de un paquete BTS o TSP.	36
Figura 4.1.1: Estructura de un descriptor USB.	44
Figura 4.1.2: Esquema de un dispositivo USB con dos configuraciones.	48
Figura 5.1.1: Capas de un sistema operativo Linux.	63
Figura 5.1.2: Especificación gráfica de la función de la interfaz de llamadas al sistema, en un sistema operativo GNU/Linux.....	65
Figura 7.1.1: Procedimiento de inicialización del demodulador y sintonizador..	79
Figura 7.1.2: Procedimiento de sintonización de una frecuencia.....	79
Figura 8.1.1: Salida de la ejecución del comando lsusb.....	84
Figura 8.1.2: Salida de la ejecución del comando lusb-v.....	86
Figura 11.1. 1: Distribución de la rejilla, coordenadas de distribución pares de puntos (x, y).....	129
Figura 11.1.2: Ventana del Reproductor.....	129
Figura 11.1.3: Botones de Control Multimedia.....	129
Figura 11.1.4: Botones Indicadores.....	130
Figura 11.1.5: Botones de Programación.....	130
Figura 11.1.6: Ventana de Desplazamiento.....	131

Figura 11.1.7: Interfaz gráfica de la aplicación.....	133
Figura 11.1.8: Widget del botón reproducir y pausar, estado reproducir.....	134
Figura 11.1.9: Widget del botón reproducir y pausar, estado pausar.....	134
Figura 11.2.1: Widget del botón detener.....	134
Figura 11.2.2: Widget del botón ajuste de volumen.....	135
Figura 11.2.3: Widget del botón pantalla completa.....	135
Figura 11.2.4: Widget estado conexión USB, desconectado.....	135
Figura 11.2.5: Widget intensidad de la señal, nivel de intensidad 0.....	136
Figura 11.2.6: Widget intensidad de la señal, nivel de intensidad 1.....	136
Figura 11.2.7: Widget intensidad de la señal, nivel de intensidad 2.....	136
Figura 11.2.8: Widget intensidad de la señal, nivel de intensidad 3.....	136
Figura 11.2.9: Widget intensidad de la señal, nivel de intensidad 4.....	136
Figura 11.2.10: Widget estado conexión USB, conectado.....	135
Figura 11.2.11: Widget intensidad de la señal, nivel de intensidad 5.....	136
Figura 11.2.12: Widget intensidad de la señal, nivel de intensidad 6.....	136
Figura 11.2.13: Widget intensidad de la señal, nivel de intensidad 7.....	136
Figura 11.2.14: Widget intensidad de la señal, nivel de intensidad 8.....	136
Figura 11.2.15: Widget intensidad de la señal, nivel de intensidad 9.....	136
Figura 11.2.16: Widget intensidad de la señal, nivel de intensidad 10.....	137
Figura 12.1.1: Estructura de archivos del instalador.....	139
Figura 13.1.1: Documentación Doxygen.....	148
Figura 13.1.2: Documentación Doxygen, enlaces pseudocódigos.....	148
Figura 13.1.3: Documentación Doxygen, ejemplo pesudocódigo.....	149
Figura 13.1.4: Documentación Doxygen, ejemplo documentación de variables.	149
Figura 13.1.5: Documentación Doxygen, ejemplo documentación de funciones.	150
Figura 13.1.6: Manual de instalación muestra número uno.....	151
Figura 13.1.7: Manual de instalación muestra número dos.....	151
Figura 14.1.1: Ejecución del comando de instalación del programa.....	152
Figura 14.1.2: Importar repositorios de software para VLC en el proceso de instalación.....	153
Figura 14.1.3: Instalación de librerías y software necesario en el proceso de instalación.....	154
Figura 14.1.4: Verificar huella del repositorio importado en el proceso de instalación.....	155
Figura 14.1.5: Creación de directorios, compilación y reglas Udev.....	156
Figura 14.1.6: Estructura de archivos de reglas udev.....	156
Figura 14.1.7: Estructura de archivos del directorio de instalación.....	157
Figura 14.1.8: Lanzador de la aplicación.....	158
Figura 14.1.9: Ejecución de la aplicación.....	158

LISTA DE TABLAS

Tabla 3.1.1: Especificaciones de los programas de la TDA Venezolana.	28
Tabla 4.1.1: Versiones y velocidades de los estándares USB.	38
Tabla 4.1.2: Registro de Configuración de Descriptores.	46
Tabla 4.1.3: Registro de Interfaz de Descriptores.	49
Tabla 4.1.4: Registro de Endpoint de Descriptores.	50
Tabla 4.1.5: Descripción tipo de paquetes etapa de configuración USB.	53
Tabla 4.1.6: Descripción tipo de paquetes etapa de datos USB.	55
Tabla 4.1.7: Descripción tipo de paquetes etapa de estado USB.	56
Tabla 4.1.8: Valores de campos para de Transmisiones de Control USB.	57
Tabla 4.1.9: Especificación de tamaños de paquetes según el tipo de velocidad de transmisión USB e intervalo de transmisión.	61
Tabla 7.1.1: Funciones de la API Samsung, demodulador.	70
Tabla 7.1.2: Funciones de la API Samsung, sintonizador.	76
Tabla 7.1.3: Funciones API Samsung definidos por el desarrollador.	78
Tabla 7.1.4: Funciones desarrolladas en base a la API de Samsung y la librería estándar de C++.....	80
Tabla 9.1.1: Funciones de la API Libusb.	87
Tabla 9.1.2: Funciones desarrolladas en base de la API Libusb para proporcionar el soporte de conexión en caliente.....	102
Tabla 9.1.3: Funciones desarrolladas en base de la API Libusb para las transferencias USB de tipo control pertenecientes al demodulador.....	106
Tabla 9.1.4: Funciones desarrolladas en base de la API Libusb para las transferencias USB de tipo control pertenecientes al sintonizador.....	108
Tabla 9.1.5: Funciones desarrolladas en base de la API Libusb para las transferencias USB de tipo interrupción.....	112
Tabla 9.1.6: Distribución de las posiciones de memoria pertenecientes al buffer de 4,1 Mb.....	116
Tabla 10.1.1: Funciones de la API Libvlc.	117
Tabla 10.1.2: Funciones desarrolladas en base a la API Libvlc.....	124
Tabla 11.1.1: Empaquetamiento de widgets.....	129
Tabla 11.1.2: Empaquetamiento en la rejilla donde el valor de la coordenada X es 0.....	132
Tabla 11.1.3: Empaquetamiento en la rejilla donde el valor de la coordenada X es 1.....	132
Tabla 11.1.4: Widgets, descripción y funciones de enlace.....	134
Tabla 12.1.1: Código de Regla Udev para el dispositivo USB de TDA.....	142
Tabla 12.1.2: Código del script de instalación.....	144
Tabla 12.1.3: Código del script de desinstalación.....	146

LISTA DE SIGLAS Y ACRÓNIMOS

CENDIT Fundación Centro Nacional de Desarrollo e Investigación en Telecomunicaciones.

TDA Televisión Digital Abierta.

TVD-T Televisión Digital Terrestre.

USB Universal Serial Bus.

BTS Broadcast Transport Streaming.

INTRODUCCIÓN

La fundación CENDIT, tiene como objeto propiciar e impulsar el desarrollo de las telecomunicaciones en Venezuela a través de la coordinación, gerencia y ejecución de proyectos tecnológicos, de investigación científica y capacitación de talento humano, uno de estos proyectos es el Diseño y Construcción de un Prototipo Industrializable de Dispositivo Portátil de Sintonización Full-Seg de TDA, compatible con la Norma ISDB-Tb, para Recepción en un Computador a través del puerto USB.

En este proyecto se desea desarrollar un controlador USB, el cual es una rutina o programa que enlaza un dispositivo periférico al sistema operativo para la comunicación entre software decodificador de paquetes de transmisión BTS y Dispositivo USB receptor de TDA bajo el sistema operativo GNU/Linux.

El software decodificador de paquetes BTS, es el VLC media player el cual es un reproductor multimedia y entorno de trabajo libre y de código abierto mantenido y desarrollado por la corporación sin fines de lucro VideoLAN, VLC es un reproductor de audio y video capaz de reproducir muchos códecs y formatos de audio y video, entre ellos el formato TS-MPEG el cual es usado por la norma ISDB-Tb. VLC es un software libre, distribuido bajo la licencia GPL (Licencia Pública General).

Para el desarrollo del controlador y la comunicación del receptor USB de TDA, el cual usa el microcontrolador USB CY7C68013A, el fabricante Cypress ofrece el paquete de desarrollo FX3_SDK_LINUX, el cual permite la comunicación en el espacio Índice de ilustraciones de usuario con el dispositivo sin tener que desarrollar un módulo del núcleo de Linux específico para ese dispositivo, este paquete de desarrollo se basa sobre la

API (Application Programming Interface) llamada LIBUSB, esta biblioteca de software permite realizar una abstracción que permite usar en el espacio de usuario las funciones USB programadas en el núcleo Linux.

Mediante la construcción de una rutina de control USB se busca obtener como resultados: que el dispositivo USB receptor de TDA sea reconocido por el sistema operativo GNU/Linux, el manejo de los buses de entrada y salida de datos, transmisión de datos de control y paquetes BTS por la interfaz USB hacia el computador y la reproducción de la información multimedia con el software VLC como software de decodificación de paquetes BTS, también debe realizarse la documentación de los algoritmos implementados y el desarrollo de un manual de instalación de la rutina de control USB.

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

Actualmente se observa un crecimiento por parte de usuarios y empresas del uso de sistemas operativos basados en GNU/Linux debido principalmente a la reducción de costos en licencias de software y al nivel de seguridad que aporta el uso de sistemas operativos libres. El uso de software libre cada vez toma más fuerza en América Latina, esta alternativa tecnológica contribuye al desarrollo científico y tecnológico de los países y a la democratización del acceso a las tecnologías de la información y comunicación. Venezuela, Uruguay, Ecuador, Brasil, Cuba y Bolivia son los países que han hecho del software libre sus proyectos bandera en el desarrollo científico y tecnológico. Estos países se enfocan en mejorar el acceso a las tecnologías de información y comunicación.

En Venezuela, el 17 de agosto de 2014 entra en vigencia la Ley de Infogobierno, ésta establece las normas, principios y lineamientos aplicados a la Tecnología de la Información, con el fin de mejorar la gestión pública y hacerla transparente sobre como las herramientas de software usadas realizan las tareas, para facilitar el acceso de los ciudadanos a la información, además de promover el desarrollo nacional. En su artículo 34 y 35 se establece:

(Ley 402-14, 2013, art. 34) [17]

“El desarrollo, adquisición, implementación y uso de las tecnologías de información por el Poder Público, tiene como base el conocimiento libre. En las actuaciones que se realicen con el uso de las

tecnologías de información, sólo empleará programas informáticos en software libre y estándares abiertos para garantizar al Poder Público el control sobre las tecnologías de información empleadas y el acceso de las personas a los servicios prestados. Los programas informáticos que se empleen para la gestión de los servicios públicos prestados por el Poder Popular, a través de las tecnologías de información, deben ser en software libre y con estándares abiertos.”

(Ley 402-14, 2013, art. 35) [18]

“Las licencias para programas informáticos utilizados en el Poder Público, deben permitir el acceso al código fuente y a la transferencia del conocimiento asociado para su compresión, su libertad de modificación, libertad de uso en cualquier área, aplicación o propósito y libertad de publicación y distribución del código fuente y sus modificaciones. Únicamente se adoptarán aquellas licencias que garanticen que los trabajos derivados se licencien en los mismos términos que la licencia original. El Poder Popular debe garantizar que las licencias de los programas informáticos empleada en la gestión de los servicios públicos transferidos, cumplan con las condiciones y términos establecidos en el presente artículo.”

Siendo la fundación CENDIT un ente del Poder Público y el encargado de propiciar e impulsar las líneas de investigación para el

desarrollo de las telecomunicaciones contribuyendo con la formación y difusión para la apropiación social del conocimiento en tecnologías de información libres en el país este debe cumplir y velar por que se cumpla la Ley de Infogobierno.

Debido a la prioridad de uso del software libre en Venezuela para el cumplimiento de la Ley de Infogobierno, entre ellos el sistema operativo GNU/Linux, y la difusión de la TDA, surge la necesidad de desarrollar e implementar una rutina o programa que enlace el dispositivo de TDA al sistema operativo GNU/Linux, actualmente no existe dicha rutina de software, lo que imposibilita la reproducción del contenido multimedia recibido y el control del dispositivo de TDA en ese sistema operativo limitando el acceso a la información de los usuarios que poseen un sistema operativo libre como Linux, por lo cual para resolver esta problemática se plantea el desarrollo e implementación de dicha rutina para el control y la reproducción del contenido multimedia del dispositivo de TDA en un ordenador bajo el ambiente operativo GNU/Linux.

MARCO REFERENCIAL

La fundación CENDIT se ha propuesto varios proyectos de desarrollo e investigación relacionados con equipos terminales para la transmisión y recepción de la señal de televisión digital bajo el estándar ISDB-Tb por lo que se planteó un proyecto para el desarrollo de un sintonizador para la señal de TDA con la intención de que la misma pueda ser recibida en un computador a través del puerto USB, lo cual sería una alternativa para aquellos casos en los que no se cuente con un televisor fijo, sino más bien se cuenta con un computador de escritorio o portátil. [1]

1.1 Dongle USB

Dongle es una pequeña pieza de hardware que se conecta a otro dispositivo para proporcionarle una funcionalidad adicional. Un dongle USB proporcionar la capacidad de conexiones USB a otro dispositivo. [1]

Las partes que comprenden el dispositivo portátil de sintonización full-seg de TDA (también conocido como dongle de TDA o TVD-T) se muestran en la **Figura 1.1.1**.

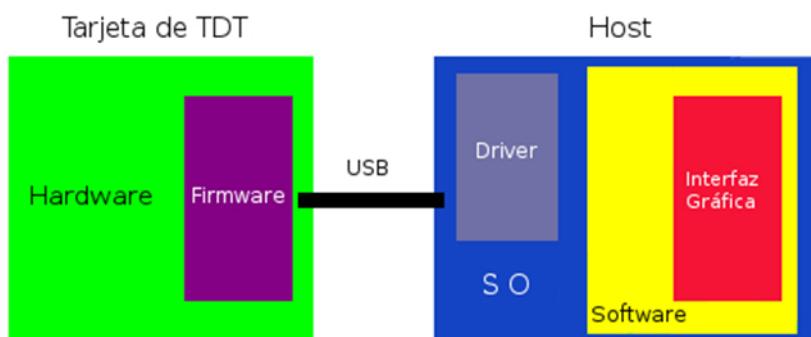


Figura 1.1.1: Composición de un sistema USB de TDA. [1]

La tarjeta de TVD-T o TDA corresponde al periférico principal del sistema y está conformado por una base de hardware (recuadro verde) y un programa que lo controla denominado firmware (recuadro morado). Como medio de comunicación y suministro de energía se utiliza el estándar USB (segmento negro). En el anfitrión se encuentra el sistema operativo (recuadro azul), que a su vez soporta el driver de la tarjeta (recuadro gris) y el software (recuadro amarillo) junto a la interfaz gráfica (recuadro rojo) que es controlada por el usuario. [1]

1.1.1 Modelo del Hardware

El hardware es la interfaz electrónica encargada de capturar la señal trasmisita en el aire, recuperar la información y enviarla de forma digital hacia el computador mediante el puerto USB, este se puede representar en su modelo más básico utilizando un diagrama como se muestra en la **Figura 1.1.2**. Este hardware cuenta con una entrada para un conector tipo F/H (conector para cable coaxial de radiofrecuencia) de 75 ohm, para la conexión de una antena UHF (Ultra High Frequency 300 MHz a 3 GHz), que al conectarla y ubicarla en una zona de cobertura se encarga de captar las señales digitales trasmisitas en el aire, seguidamente se encuentra el Tuner RF, encargado de sintonizar la frecuencia que sea seleccionada entre el rango correspondiente a la TDA (512-698MHz), luego se encuentra el demodulador ISDB-Tb, y se encarga de recuperar la información original que fue trasmisita dentro de la señal digital que le entrega el bloque anterior (tuner RF), colocando como salida, una señal digital del canal asociado conformada por paquetes BTS, y como último bloque, se encuentra el controlador USB que transmite estos datos digitales hacia el computador empleando el protocolo USB, este controlador también tiene la tarea de enviar instrucciones provenientes del software en el

computador para manejar los parámetros de sintonización tanto del Tuner RF como del demodulador ISDB-Tb a través del protocolo I2C. [1]

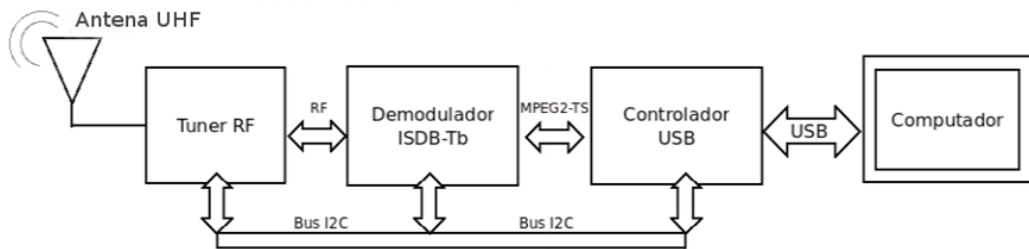


Figura 1.1.2: Diagrama de hardware, comunicaciones y flujo de datos. [1]

1.1.1.1 Tuner NIM DNOD22QXV104A.

El tuner NIM (Network Interface Module) es una solución del bloque Tuner RF y el Demodulador ISDB-Tb contenidos en una misma unidad, su cubierta metálica impide las interferencias electromagnéticas que puedan distorsionar las señales recibidas desde la antena hacia los componentes que lo conforman. Entre sus puertos está: los conectores tipo F para la antena UHF y los pin header macho por donde se encuentran los puertos de comunicación, la entrada de alimentación y el bus de salida de datos para los paquetes BTS. En la **Figura 1.1.3** se puede observar las partes que conforman un tuner NIM. [1]

Para este proyecto se utilizará el **Tuner NIM DNOD22QXH104A** del fabricante Samsung.

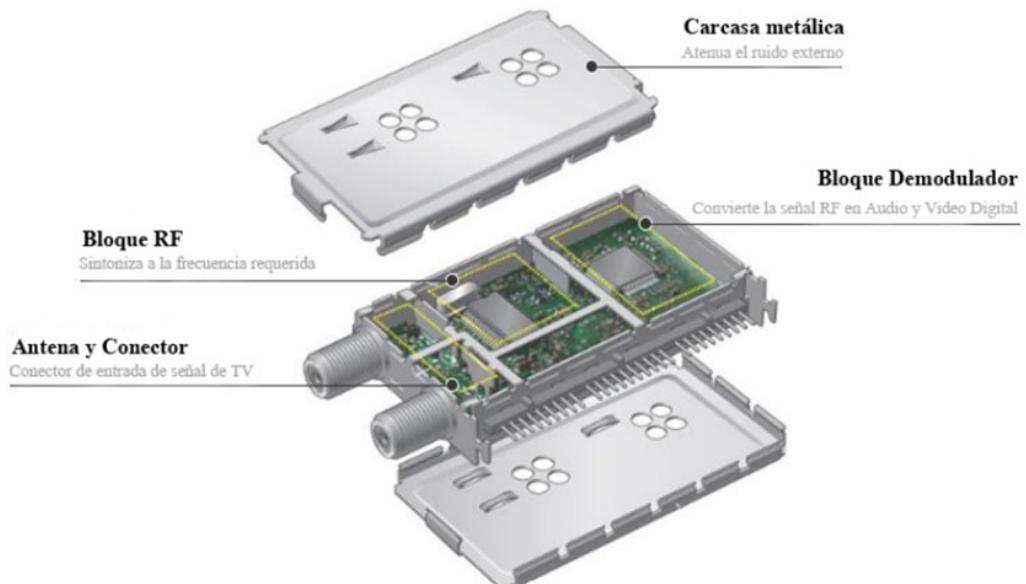


Figura 1.1.3: Representación del hardware Tuner NIM. [1]

1.1.1.2 Controlador USB CY7C68013A.

El controlador USB encargado de comunicarse con el Tuner NIM y trasmisir los paquetes BTS hacia el computador. Su modelo es el EZ-USB serie FX2LP cuyo número de chip es el CY7C68013A, su estructura interna se muestra en la **Figura 1.1.4**. Este es un microcontrolador desarrollado por la empresa Cypress el cual presenta las siguientes características:

- Interfaz USB 2.0 HighSpeed.
- Muy bajo consumo de energía (no más de 85mA).
- Tensión de funcionamiento: 3.3v
- 16Kbytes de RAM.
- Firmware descargado directamente a la RAM vía USB o descargado desde una EEPROM externa hacia la RAM.
- Cuatro interfaces FIFO de data externa de 8 o 16 bit.
- Integrado CPU del estándar 8051 mejorado.

- Opera a 48MHz, 24MHz, o 12MHz.
- Controlador I2C integrado, con frecuencias de operación de 100 o 400Khz.

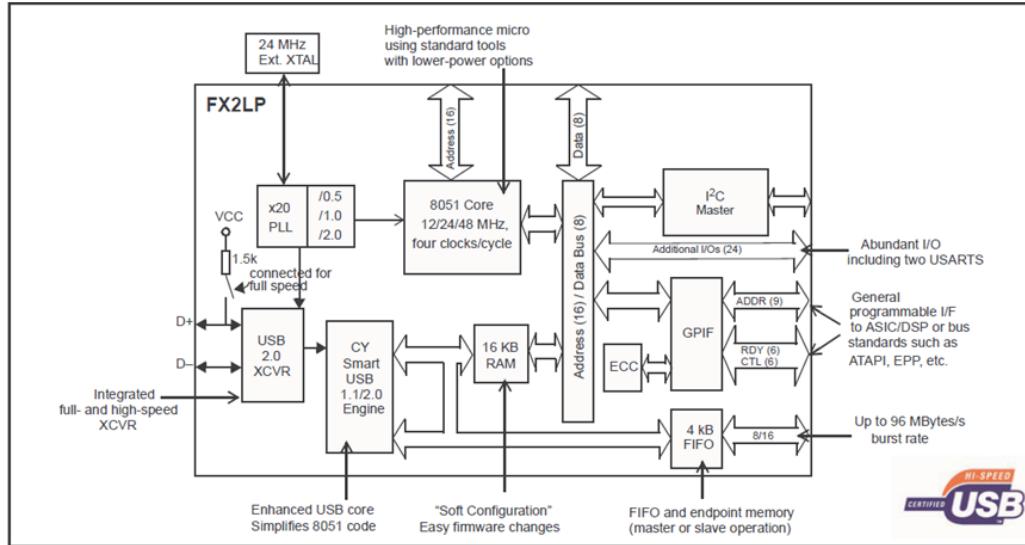


Figura 1.1.4: Diagrama de bloques del dispositivo USB CY7C68013A. [1]

Para la implementación del prototipo de prueba fue necesaria la adquisición de una tarjeta de desarrollo que permita realizar distintas configuraciones de los puertos utilizados y así determinar el método más conveniente para la trasmisión de los datos. Además de incorporar los componentes mínimos para el funcionamiento pleno del microcontrolador. [1]

En este sentido la tarjeta de desarrollo utilizada es la que se muestra en la **Figura 1.1.5**, la cual se accede a sus I/O digitales, interfaces FIFO, etc, a través de sus dos espaldines doble, macho de 20 pines, incorpora una EEPROM de 16KB (modelo 24LC128), LED's de propósito general, botón de reset, un interruptor general y el puerto USB para el suministro de energía y la trasmisión de los datos. [1]

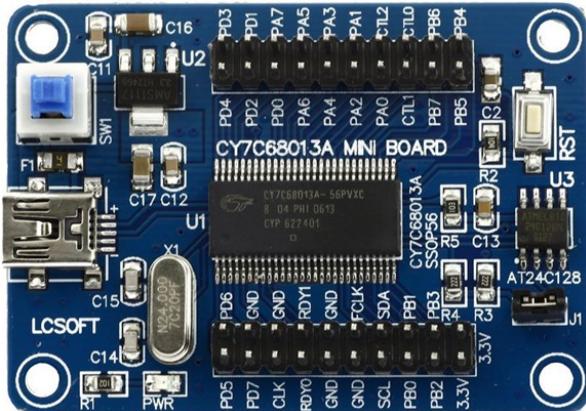


Figura 1.1.5: Placa de desarrollo del dispositivo USB CY7C68013A.[1]

1.1.1.3 Prototipo de Prueba.

El prototipo de prueba está conformado principalmente por dos tarjetas, el módulo de sintonización ISDB-T_b y la tarjeta de desarrollo Cypress, estas se conectan a través de dos pin header de forma apilada, donde es transmitida toda la información necesaria para el funcionamiento del dispositivo. En la **Figura 1.1.6** se puede observar el acople entre estas dos tarjetas. [1]

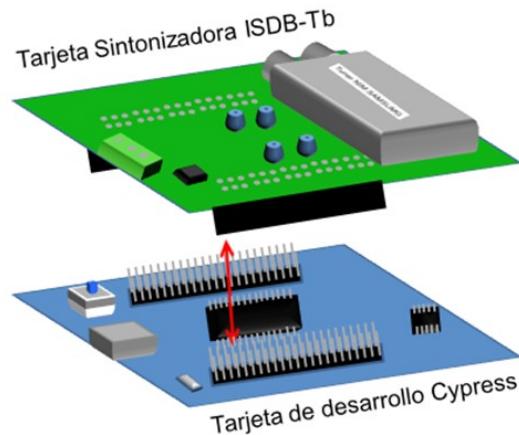


Figura 1.1.6: Acople del módulo de sintonización con la tarjeta de desarrollo. [1]

En la **Figura 1.1.7** se observa el diseño del PCB del módulo de sintonización ISDB-Tb. [1]

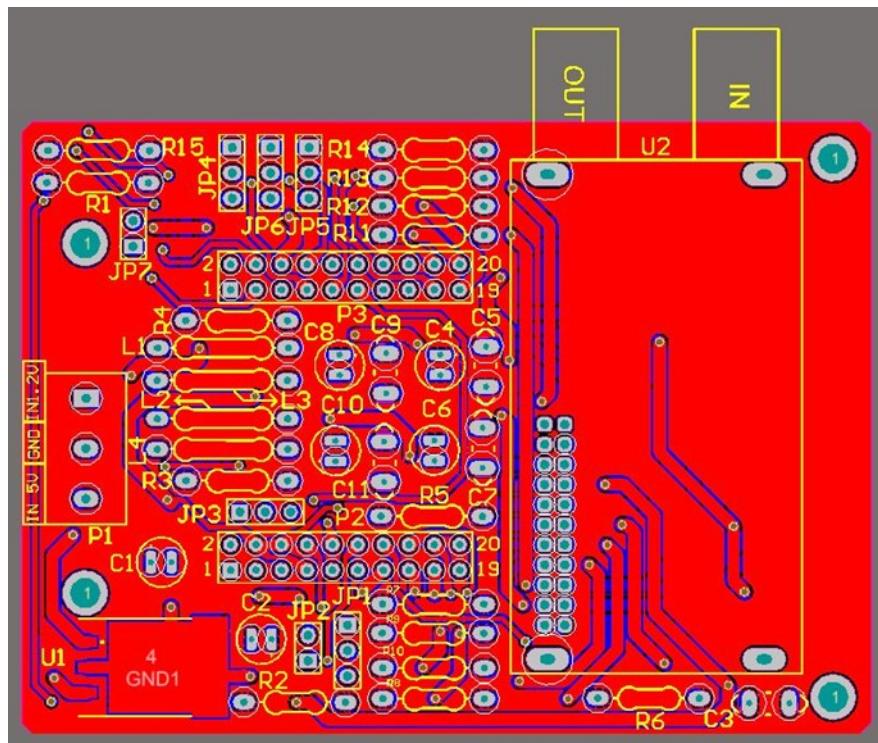


Figura 1.1.7: Diseño del PCB del módulo de sintonización ISDB-Tb. [1]

En la **Figura 1.1.8** se observa un modelado en 3d del módulo de sintonización ISDB-Tb.

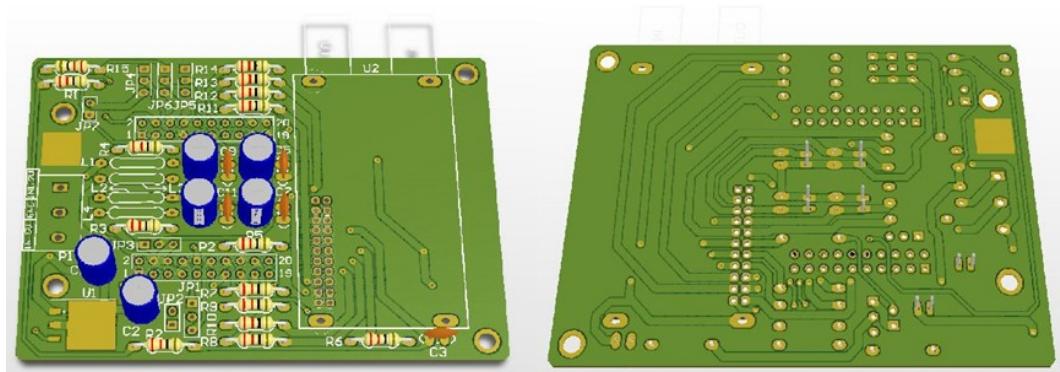


Figura 1.1.8: Modelado en 3D del módulo de sintonización ISDB-Tb. [1]

1.1.1.4 Prototipo Alpha

El PCB del prototipo alfa está orientado al usuario final ya que todo el sistema está integrado en una misma tarjeta. Sus dimensiones reducidas lo hacen un dispositivo lo suficientemente pequeño como para denominarse del tipo dongle, esto es gracias a los componentes con encapsulado SMD, ocupan menos espacio y pueden ubicarse tanto en la capa top como en la capa bottom sin causar interferencia entre las pistas. En la **Figura 1.1.9** se puede observar el diseño del PCB. [1]

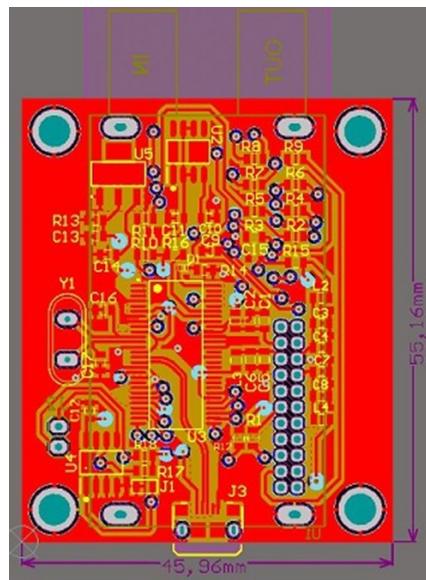


Figura 1.1.9: Diseño del PCB del Prototipo Alfa. [1]

En la **Figura 1.2.1** se puede apreciar mayor detalle sobre este prototipo, donde se destaca el tuner NIM ubicado en la parte inferior de la tarjeta. Un detalle muy importante de este prototipo es que incorpora el puerto USB del cual se toma la alimentación para todo el sistema directamente desde el computador. [1]

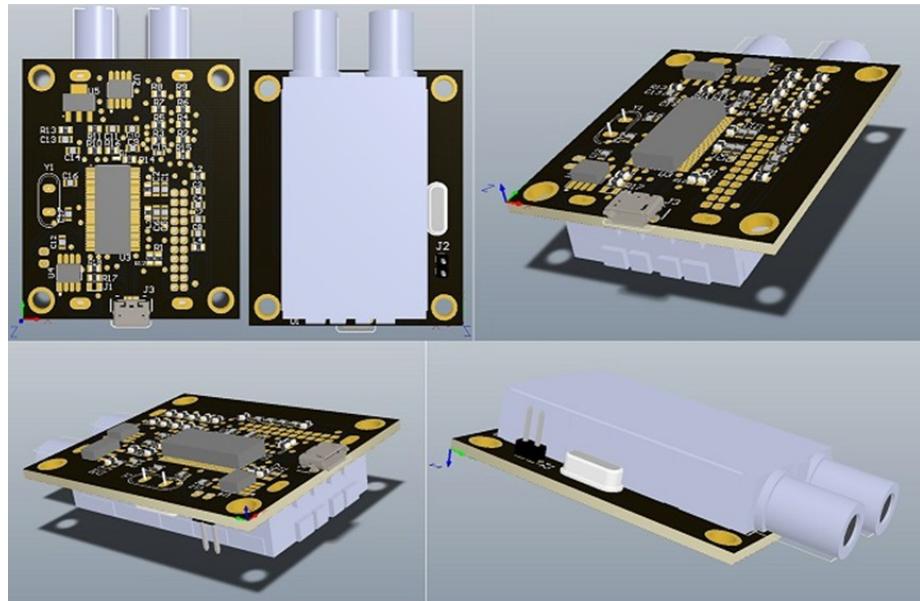


Figura 1.2.1: Modelado 3D del Prototipo Alfa.[1]

1.1.2 El firmware USB

El funcionamiento del firmware está compuesto por distintas etapas las cuales se describen a continuación:

- Inicialización: En esta etapa se carga el programa alojado en la memoria EEPROM externa a la memoria del sistema. Luego se inicializan todas las variables que se utilizaran, junto con las librerías que fueron incluidas. [1]
- Habilitación: Se decide los puertos que se utilizarán en el sistema a través de los registros de cada uno. En este caso se habilita el Puerto USB, I2C y el Buffer FIFO y se configura su comportamiento para el tratamiento de los datos. [1]
- Enumeración: Este proceso se encarga de indicarle al computador que tipo de dispositivo es y determinar que controlador usar, para ello el firmware tiene una sección de código llamada descriptor, donde se

indica quién es su fabricante, la versión USB que soporta, cuantas configuraciones tiene, número de puntos finales, etc. [1]

- Interruptores USB: En este paso se habilitan los tipos de interrupciones que se producirán por el puerto USB. Esto permite atender los requerimientos que haga el software al momento de mandar un paquete a través del puerto USB. [1]
- Reset del Tuner NIM: En este paso se realiza un reset al módulo Tuner NIM a través de un puerto lógico del micro-controlador, llevándolo a un estado lógico bajo, durante 4 mili segundos, y el módulo quedará preparado para ser utilizado. [1]
- Inicialización I2C: En esta etapa se establecen las configuraciones necesarias para la comunicación I2C. [1]
- Bucle Principal: Luego de que el programa pasa por cada una de las etapas anteriores, el programa cae en un bucle infinito. En este caso el microcontrolador no tiene más nada que hacer por su parte, por lo que queda a la espera de solicitudes por parte del software a través del puerto USB. [1]
- Protocolo de comunicación con el Tuner NIM por I2C: Acá se ubican las funciones que se encargan de trasmitir o recibir uno o más datos hacia los registros del tuner RF o el demodulador. Por lo general estas funciones se utilizan cuando el software requiere sintonizar un canal, enviando los datos por el puerto USB y a su vez, el microcontrolador los canaliza hacia estas funciones para que les lleguen hacia el Tuner NIM. Estas funciones se consideran las más importantes del firmware ya que hacen posible las configuraciones del tuner RF y Demodulador ISDB-TB, en la **Figura 1.2.2** se muestra los detalles de estas funciones. [1]
- Buffer FIFO: Cuando se le da la orden al tuner NIM de sintonizar una señal por medio del protocolo I2C, este entrega los paquetes BTS

por un bus paralelo de datos de 8 bits más una señal de sincronización, estos datos son recibidos por el buffer FIFO, almacenando cada dato en cada ciclo de reloj hasta que el buffer llega a su máxima capacidad (1024 byte) enviando automáticamente el paquete por el puerto USB hacia la computadora sin que se pierda ningún dato mientras se realiza esta tarea. En la **Figura 1.2.3** se puede observar las funciones I2C implementados en el firmware para la comunicación entre el dispositivo USB de Cypress y el Tuner NIM. [1]

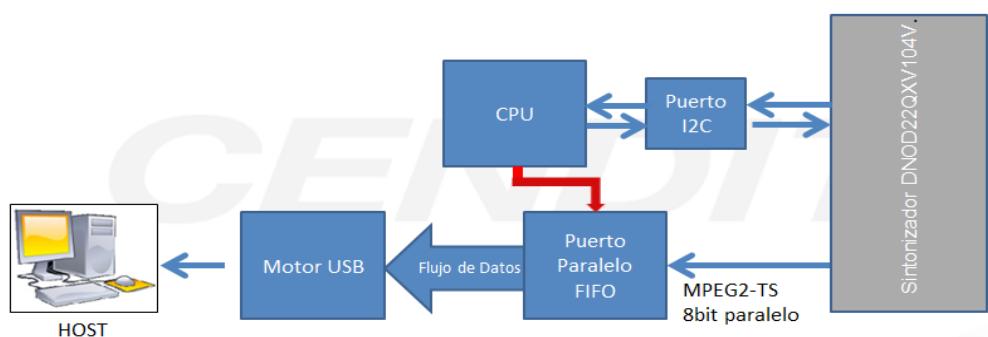


Figura 1.2.2: Funcionamiento del firmware en el microcontrolador CY7C68013A. [1]

- `I2c_Init()`.

Esta función inicializa el bus I²C a 400Khz y desactiva interrupciones no deseadas. Esta se emplea antes de iniciar la transferencia de datos.

- `TC90527_I2cWrite(ChipAddr, Addr, Data)`

Función empleada para transferir un bytes al demodular del tuner NIM, donde “`ChipAddr`” es la dirección de escritura del demodular, “`Addr`” es el registro al cual se desea escribir y “`Data`” es el dato de un Byte que se desea enviar.

- `TC90527_I2cRead (ChipAddr, Addr)`

Esta función se utiliza cuando se desea leer un dato proveniente del demodular devolviendo como resultado el dato de un byte. “`ChipAddr`” es la dirección de lectura del demodulador, “`Addr`” es la dirección del registro que se desea leer.

- `STV4100_I2C_Write(Addr, Data)`

Esta función escribe una palabra (byte) al tuner RF, donde “`Addr`” es la dirección del registro que se desea escribir y “`Data`” es la palabra a enviar.

- `STV4100_I2C_Read(Addr, *lpData)`

Esta función permite realizar una solicitud de lectura de dato del tuner RF, donde “`Addr`” es la dirección del registro que se leerá y “`lpData`” es la variable que contendrá la palabra leída.

Figura 1.2.3: Funciones para comunicarse con el Tuner NIM desde el firmware.
[1]

1.1.3 El software C# y controladores USB privativos para Windows

El software fue realizado en la IDE de Visual Studio 2013 empleando el lenguaje C# para el sistema operativo Windows, como se observa en la **Figura 1.2.4.** [1]

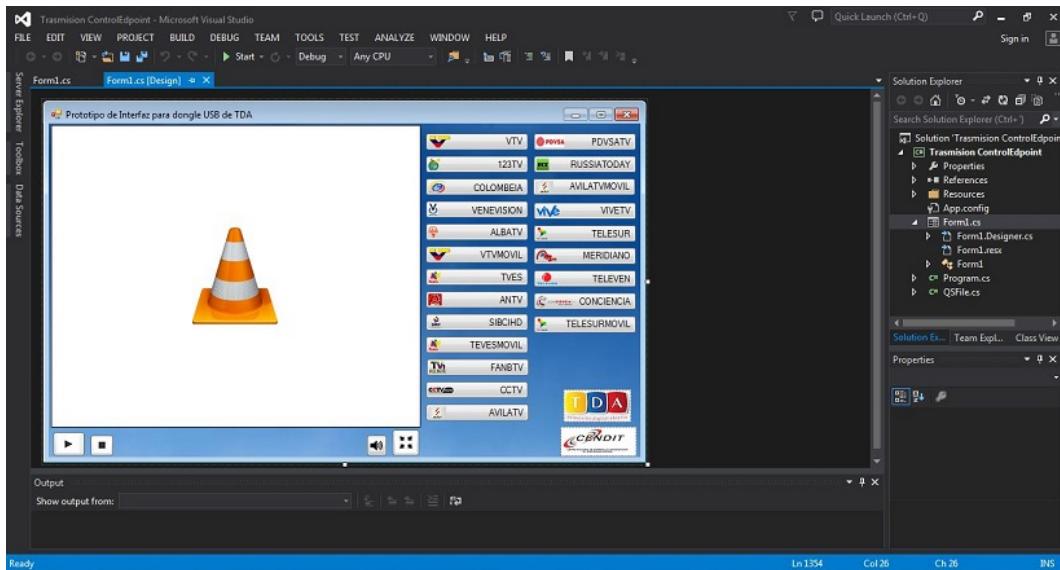


Figura 1.2.4: IDE C# desarrollo del software privativo para el Dongle USB de TDA. [1]

Para la comunicación USB se utiliza la librería CyUSB.dll y el driver cyusb3.sys, proporcionados por Cypress para el sistema operativo Windows, los cuales una vez incluidos en el proyecto, permiten hacer llamados a una serie de funciones para transferir los datos al firmware del microcontrolador, es decir, la librería se conecta con el driver cyusb3.sys y el driver permite la conexión con el firmware. En la **Figura 1.2.5** se muestra en funcionamiento la aplicación desarrollada. [1]



Figura 1.2.5: Aplicación desarrollada en Windows para el Dongle USB de TDA en funcionamiento. [1]

Como se muestra en la **Figura 1.2.5** se comprobó satisfactoriamente que el prototipo de prueba USB receptor de TDA funciona correctamente, ya que se visualizaron todos los contenidos transmitidos de la plataforma TDA (canal 22, 23, 24, 24 de la banda UHF) en el computador con el desarrollo de la programación de sintonización y manejo de paquetes BTS junto a la interfaz gráfica en el sistema operativo Windows 7 64 bits, utilizando el lenguaje C# y la IDE de Visual Studio 2013 en conjunto con el driver USB genérico que proporciona Cypress para micro-controladores de la serie FX2LP y el reproductor VLC para la decodificación de los MPEG2 TS. [1]

CAPÍTULO II

OBJETIVOS

2.1 Objetivo General

Desarrollar un controlador USB para la comunicación entre el software del decodificador de paquetes BTS y el Dispositivo USB receptor de TDA bajo el sistema operativo GNU/Linux.

2.2 Objetivos Específicos

1. Desarrollar la lógica de comunicación entre firmware y software más adecuada para este tipo de aplicación.
2. Desarrollar un controlador funcional para el dispositivo USB receptor de TDA desarrollado en la institución.
3. Realizar pruebas de funcionamiento del controlador.
4. Documentar los algoritmos de programación del controlador desarrollado, así como los métodos que llevaron a la solución del problema.
5. Redactar un manual donde se demuestren los procedimientos para la instalación y utilización del controlador desarrollado, incluyendo pseudocódigos.

CAPÍTULO III

MARCO TEÓRICO

3.1 Televisión Digital Abierta

La televisión en sus distintas versiones, por cable, por satélite o por aire, ha iniciado la actualización de sus servicios y de sus tecnologías incorporando la digitalización en todas sus etapas de funcionamiento, desde la cámara hasta la pantalla del televíidente. Esto le ha permitido mejorar significativamente la calidad de imagen y de sonido, eliminando los problemas de imágenes múltiples, ruidos, intermodulaciones y muchas otras distorsiones propias de los sistemas analógicos, con el agregado de un mejor aprovechamiento del espectro radioeléctrico y la posibilidad de incrementar la cantidad de señales transmitidas, incorporando nuevos servicios tales como la alta definición, múltiples canales de audio, señales para dispositivos móviles, interactividad, etc. [2]

Los desarrollos de la televisión digital terrestre comenzaron en los Estados Unidos de América a mediados de la década del noventa, con la implementación de la norma ATSC (Advanced Television Systems Committee), siguieron luego en Europa con el estándar DVB (Digital Video Broadcasting), más recientemente en Japón con ISDB (Integrated Services of Digital Broadcasting) y ahora en los países asiáticos con la norma DMB (Digital Multimedia Broadcasting). [2]

En Latinoamérica se ha dado un proceso de digitalización acelerado, que tuvo su inicio en Brasil en el año 2004. La mayoría de los países han adoptado una o más normas y en muchos casos ya están utilizando sistemas de transmisión digitales. Por ejemplo, se utiliza la norma

DVB-S y DVB-S2 para la televisión satelital y para el transporte de programación respectivamente, DVB-C o ATSC para la TV por cable e ISDB-Tb para la televisión digital terrestre. El lanzamiento de la TV digital terrestre en la región, ha permitido un gran aporte de la industria de cada país en el diseño, la fabricación y la producción de la mayoría de las partes involucradas en las plantas transmisoras digitales, potenciando el estudio y la capacitación sobre todos los aspectos del sistema, lo que dio como resultado una importante mano de obra especializada que se encuentra enfocada en la puesta en marcha de las nuevas estaciones. En general, puede afirmarse que la TV digital abierta se ha convertido en una política de estado en todos los países latinoamericanos, con el objetivo de ofrecer a toda la población igualdad en las condiciones de acceso a los contenidos audiovisuales (a veces denominado inclusión digital), brindar mayor cantidad de señales y servicios y mejorar la calidad. [2]

3.1.1 Televisión Digital Terrestre

La denominación de TV digital terrestre incluye a todos los servicios de difusión de televisión por aire en los que los flujos de datos son transmitidos mediante sistemas de modulación digital que utilizan el espectro radioeléctrico. Las canalizaciones o anchuras de banda asignadas pueden ser de 6, 7 u 8 MHz y cada país dispone de una canalización específica, en Venezuela es de 8MHz. Las transmisiones son del tipo difusión (broadcast) punto a multipunto y pueden ser de acceso libre y gratuito, o por suscripción. Las emisiones de televisión digital terrestre reemplazarán completamente a las de televisión analógica cuando en cada país se produzca el apagón analógico (switch-off), fecha que la mayoría de los países latinoamericanos ya han definido, en el caso de Venezuela se planificó para el año 2020. [2]

El desarrollo de la TVD-T iguala sus servicios a los de la televisión por satélite y por cable y, por lo tanto, constituye un avance muy importante

para la inclusión social, porque ofrece una diversidad de servicios complementarios unidireccionales y también bidireccionales cuando se establece un canal de retorno adecuado. [2]

La TV digital terrestre presenta numerosas ventajas frente a su contrapartida analógica. Las principales se resumen a continuación:

- a) Exhibe una mejor calidad de sonido e imagen.
- b) Permite contenidos en alta definición (HD).
- c) Posibilita la multiprogramación, al permitir la transmisión de varias señales de la misma anchura de banda asignada a la emisora.
- d) Permite que se integren y se complementen los contenidos con Internet.
- e) Permite nuevos servicios asociados a la interactividad y otros como la "Grilla Electrónica de Programación" (EPG), ejecución de aplicaciones, etc.
- f) Ahorra espectro radioeléctrico, al incorporar mayor cantidad de señales dentro de la misma anchura de banda.
- g) Incrementa la programación ofrecida y con ello se moviliza significativamente la industria de la producción de contenidos audiovisuales y también otras industrias: electrónica, telecomunicaciones, software, etc.
- h) Permite brindar servicios a diversos dispositivos: teléfonos móviles, televisores con decodificador incluido, sintonizadores para computadoras portátiles, GPS con sintonizador, etc., y los integra a la cultura audiovisual.
- i) Permite que un contenido audiovisual pueda verse en diferentes aparatos con diferentes calidades, esto se conoce como producción para multiplataforma.

La introducción de la TVD-T ha permitido que los servicios lleguen al televidente con una mayor calidad de imagen y sonido en comparación con las emisiones analógicas, con la gran ventaja de que la calidad permanece constante dentro de toda el área de cobertura, algo que era imposible garantizar con las transmisiones analógicas. Las emisiones digitales pueden tener distintos formatos en audio y video, pero una vez definido el sistema de transmisión, los parámetros de calidad se mantienen inalterables en todos los puntos de recepción. La mayor calidad de imagen y sonido se relaciona con la alta capacidad de transporte de información, con tasas del orden de los 20 Mbps o mayores. La TVD-T transmite tres tipos de flujos binarios:

- Video y audio correspondiente a la programación, en diversos formatos de resolución y de pantalla, audio en distintos idiomas, etc.
- Datos, que corresponden a una pequeña porción del flujo total transmitido y se utilizan para enviar al receptor información adicional a la programación, tal como interactividad entre la planta transmisora, el receptor o los servidores de datos ubicados en Internet.
- Codificación y sincronización. El primero, destinado a proteger los flujos útiles de las interferencias introducidas en el canal de transmisión y el segundo para que el receptor detecte el esquema de transmisión utilizado y pueda recuperar los datos recibidos. [2]

En cuanto a los formatos de video, las señales se pueden clasificar de acuerdo a su resolución en cantidad de pixels, dando origen a las siguientes denominaciones y velocidad de flujo aproximadas:

- LDTV (Low Definition Television): baja resolución, ejemplo 320x240 pixels. Utilizada en las transmisiones para receptores móviles, requiere una tasa binaria del orden de 450 kbps.

- SDTV (Standart Definition Television): resolución estándar, típica de las transmisiones analógicas de 720x576 pixels. Comúnmente se emplea para multiprogramación, envía varias señales dentro de la anchura de banda del canal. Requiere de una tasa de datos media, situada en el orden de los 3 Mbps.
- EDTV (Enhanced Definition Television): resolución mejorada o intermedia, típicamente en el orden de los 1280x720 pixels. Se obtiene una muy buena calidad de imagen con una tasa no demasiado elevada, en el orden de los 9 Mbps, lo que permite un mejor aprovechamiento del canal de transmisión.
- HDTV (High Definition Television): alta resolución que permite transmitir imágenes de gran calidad, de unos 1920x1080 pixels, que se traducen en tasas binarias situadas en los 13 Mbps. [2]

3.1.2 La Multiprogramación

En este punto es importante visualizar la diferencia entre los términos canal y señal. Se llama canal a la anchura de banda disponible en el espectro para transmitir televisión y señal a la programación que se emite en el canal. En televisión analógica, a cada canal le corresponde una señal, mientras que en televisión digital en cada canal pueden transmitirse varias señales. [2]

Cuando se transmiten varias señales diferentes en un canal se dice que se utiliza "multiprogramación". Los servicios de multiprogramación no están igualmente reglamentados en los distintos países de la región, pues cada administración tiene su propio concepto de multiprogramación, se establecen determinadas jerarquías que definen el grado de robustez de cada señal y por lo tanto el alcance y la calidad con que será vista dentro del área de cobertura.[2]

La **Figura 3.1.1** muestra de una manera conceptual y muy simplificada, algunos ejemplos de utilización de la anchura de banda del

canal de acuerdo a la información que desea transmitirse. Como puede verse, cada servicio que se incluye consume una parte de este espacio y esta afirmación es válida para todas las normas de TVD-T. Tanto en la parte inferior del canal como en la superior, se dejan espacios de protección o bandas de guarda (G). Se transmiten datos de diversos servicios de video y audio (HD, ED, SD y LD) más datos generales (DAT), codificación (COD) y sincronización (SINC). La cantidad y tipo de servicios que se pueden transmitir dependen de la calidad de video y audio requerida y de la robustez que se les asigne para lograr una determinada continuidad del servicio. Por ejemplo, los servicios de HDTV demandan una alta velocidad del flujo binario y habitualmente se utiliza una protección moderada. En cambio, los servicios para receptores móviles y portátiles LDTV requieren de una protección más alta, a los fines de garantizar que la recepción sea buena aún en condiciones muy adversas, habituales en las situaciones de movilidad y portabilidad. [2]

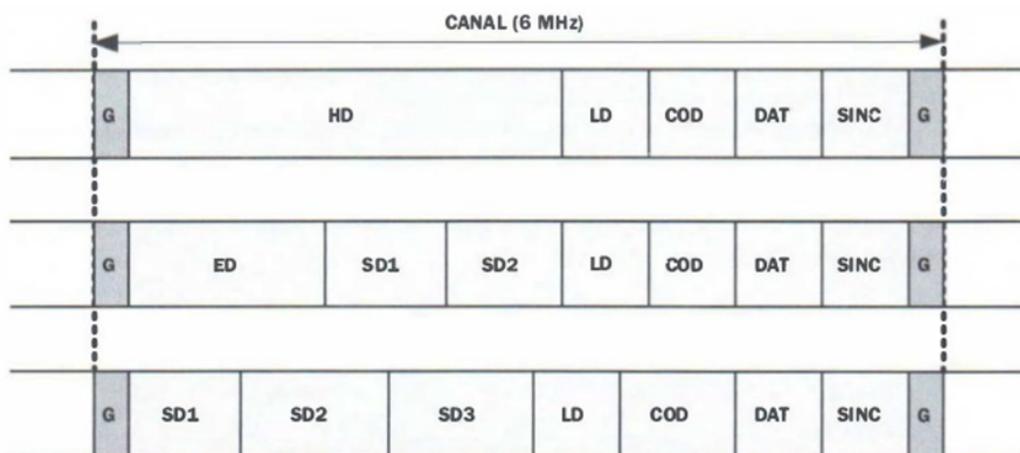


Figura 3.1.1: Ilustración anchura banda de un canal. [2]

A continuación se muestra la **Tabla 3.1.1** especificando los programas de TDA del estado Venezolano.

Tabla 3.1.1: Especificaciones de los programas de la TDA Venezolana. [3]

Canal Frecuencia	y Subcanal	Nombre	Temática	Aspecto y Resolución
22 (521 Mhz)	1	VTV	Informativo	4:3, 480i, SD
	2	123TV	Infantil	4:3, 480i, SD
	3	Colombeia	Educativo	4:3, 480i, SD
	4	Venevisión	Variedades	4:3, 480i, SD
	5	Alba TV	Comunitario	4:3, 480i, SD
	6	VTV Móvil	Informativo	320 x 240 px
23 (527 Mhz)	1	ViVe	General	4:3, 480i, SD
	2	Telesur	Informativo	4:3, 480i, SD
	3	Meridiano Televisión	Deportivo	4:3, 480i, SD
	4	Televen	Variedades	4:3, 480i, SD
	5	TV ConCiencia	Científico	4:3, 480i, SD
	6	Telesur Móvil	Informativo	320 x 240 px
24 (533 Mhz)	1	TVes	Variedades	4:3, 480i, SD
	2	ANTV	Parlamentario	4:3, 480i, SD
	3	SIBCI HD/Corazón Llanero TV	Variedades	16:9, 1080i, HD
	4	TVes Móvil	Variedades	320 x 240 px
25 (539 Mhz)	1	TV FANB	Militar	4:3, 480i, SD
	2	CCTV	Noticias	4:3, 480i, SD
	3	Ávila TV	Juvenil	4:3, 480i, SD
	4	PDVSA TV	Petrolero	16:9, 480i, SD
	5	Russia Today	Noticias	16:9, 480i, SD
	6	Ávila TV Móvil	Juvenil	320 x 240 px

A continuación en la **Figura 3.1.2** se muestra un área de cobertura de la TDA en el estado venezolano.

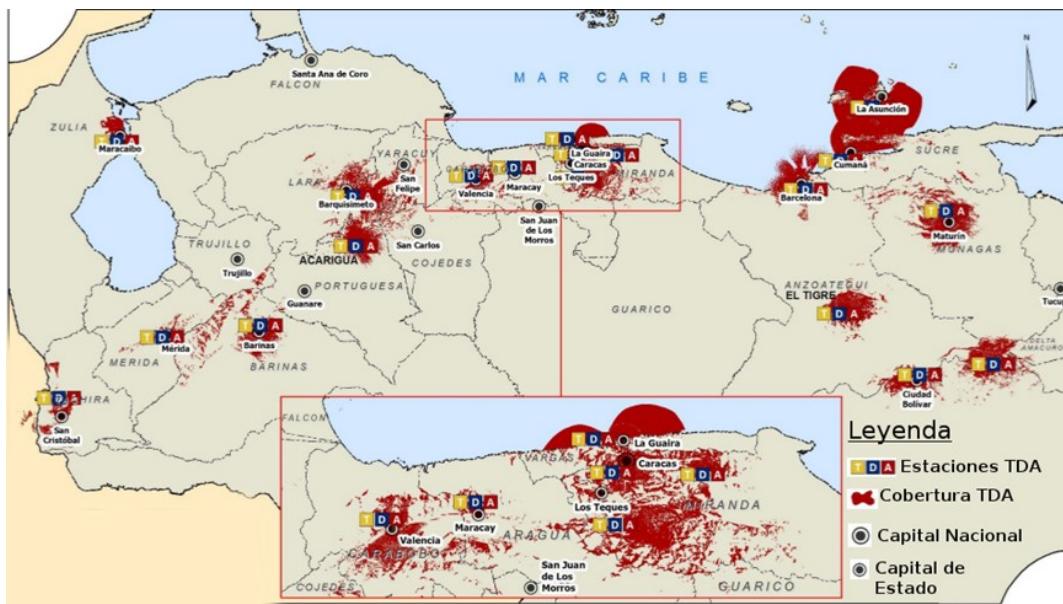


Figura 3.1.2: Área de cobertura de TDA en Venezuela. [3]

Por otra parte la TV móvil es una nueva posibilidad que ofrece la TVD-T. Siguiendo los desarrollos de la telefonía móvil, la TV digital se ha incorporado al teléfono y a muchos otros dispositivos portátiles. La forma en que se incorpora el servicio móvil puede ser mediante un receptor independiente del sistema de telefonía o integrado a la misma red, dependiendo del modelo de implementación adoptado en cada país y, por lo tanto, fuertemente ligado a las normas y a la regulación. [2]

El uso creciente de dispositivos móviles capaces de recibir televisión digital, en aquellos países en donde se está ofreciendo el servicio móvil, por ejemplo Japón, demuestra que hay importantes cambios en el primetime (horario de máxima audiencia), y se registran nuevos máximos de audiencia en los horarios de almuerzo o a la salida de las oficinas. Estas nuevas experiencias de los usuarios llevan al desarrollo de programación específica para los teléfonos móviles, como noticias breves, informes meteorológicos, programación personalizada, integración a las redes sociales, contenidos interactivos, etc. [2]

3.1.3 El estándar ISDB-Tb

Existen varias normas de TVD-T en el mundo, que empezaron a desarrollarse a partir de 1990 aproximadamente y que responden a distintos modelos de migración desde analógico al digital y a criterios de compatibilidad con los sistemas de televisión preexistentes en los países de origen, además de variados intereses tecnológicos, políticos y económicos. Algunos de los conceptos teóricos en los que se basan los distintos estándares se conocen desde hace más de cien años y recién con los avances tecnológicos actuales han podido ser implementados en circuitos electrónicos. Estos avances son cada vez más acelerados y por lo tanto, los distintos centros de desarrollo han pensado sus normas en función de las posibilidades tecnológicas disponibles en un determinado momento, dando origen a cuatro normas y a su vez a distintas versiones de las mismas. [2]

Estas normas son:

- ATSC (Advanced Television Systems Committee), Estados Unidos de América.
- DVB-T (Digital Video Broadcasting -Terrestrial), Europa.
- ISDB-T (Integrated Services of Digital Broadcasting -Terrestrial), Japón.
- ISDB-Tb introduce modificaciones propuestas por Brasil.
- DMB-T (Digital Multimedia Broadcasting - Terrestrial), China. También es conocido como DTMB.

Cada país ha adoptado, o está próximo a hacerlo, alguna de las cuatro propuestas. El mapa mundial de la **Figura 3.1.3** muestra cómo se han distribuido los estándares a nivel mundial.

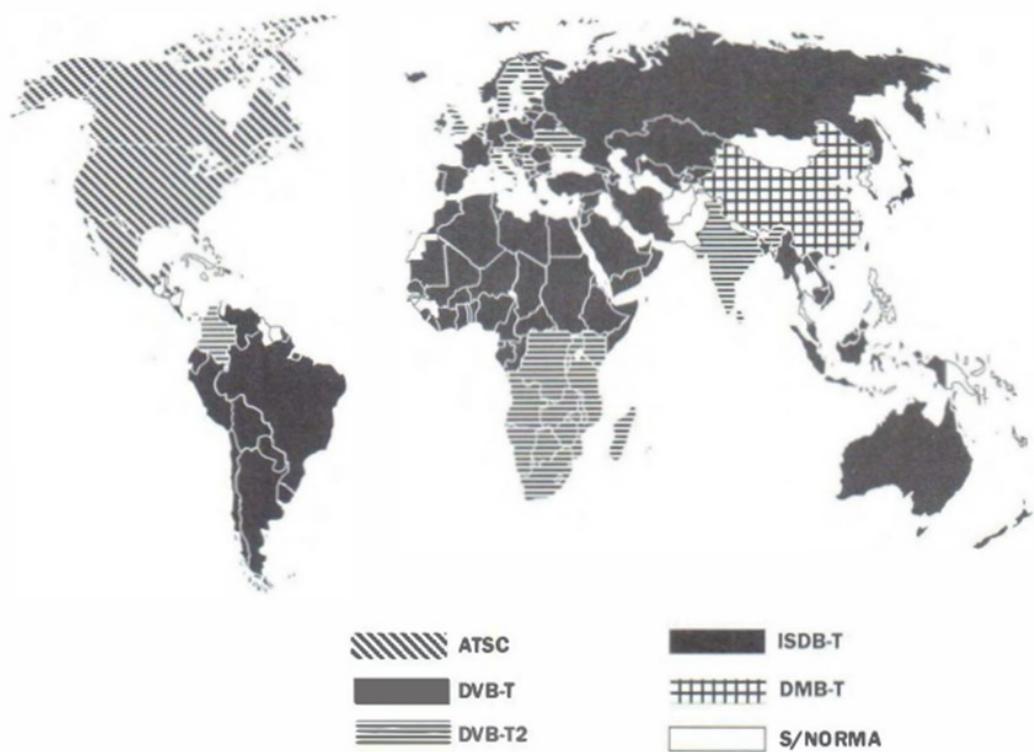


Figura 3.1.3: Estándares de TVD-T en los distintos países del mundo. [2]

Los estándares de TVD-T tienen varias similitudes en su estructura general. En la **Figura 3.1.4** se muestran los bloques funcionales más importantes y sus funciones principales, que a grandes rasgos se explican a continuación.

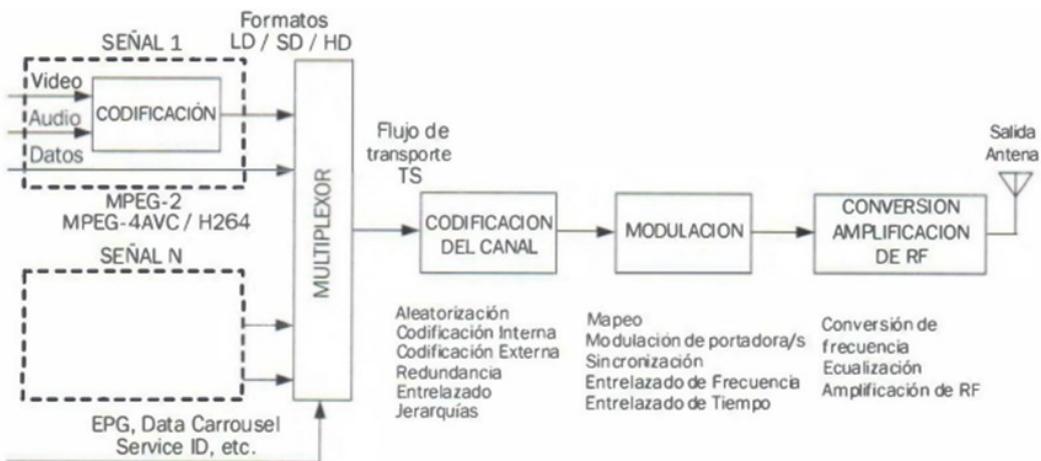


Figura 3.1.4: Bloques funcionales de un transmisor de TVD-T. [2]

- **Codificación:** El video y el audio digitalizados son codificados para reducir la velocidad de transferencia, lo que implica reducir la tasa binaria de cada flujo con la menor pérdida de calidad posible.
- **Multiplexor:** Conforma el flujo de transporte de paquetes de bits correspondientes a cada señal de video, audio y datos de cada servicio, entrelazando las secuencias de los distintos servicios, agregando la información adicional correspondiente a la planta transmisora. Si bien cada estándar tiene sus particularidades, todos utilizan el modelo de transporte correspondiente a MPEG-2 TS (ISO/IEC 13818-1).
- **Codificación del Canal:** Para proteger los datos a transmitir frente a las características propias e indeseadas del canal de propagación, tales como el ruido, las interferencias por múltiples trayectos y otras señales indeseadas.
- **Modulación:** En estos bloques se define la forma de transportar los datos binarios, aplicando esquemas de modulación digital a una portadora o a miles de portadoras de acuerdo al estándar. También se agrega la información de sincronización necesaria para que el

receptor se ajuste al patrón de transmisión y pueda realizar el proceso inverso, recuperando los datos originales.

- **Conversión-amplificación de RF:** En estas etapas se conforma la anchura de banda de emisión mediante filtros que eliminan o atenúan las emisiones no deseadas sobre los canales adyacentes. En los pasos finales se realiza una conversión que lleva la señal desde una frecuencia intermedia hasta la frecuencia de emisión del canal. Por último se amplifica potencia hasta obtener el nivel necesario para conseguir el área de cobertura buscada para el servicio considerado.
[2]

El gobierno Brasileño definió los lineamientos para la TVD-T en noviembre de 2003, y puso en marcha el Sistema Brasileño de Televisión Digital Terrestre (SBTVD-T). Bajo este marco se estableció una alianza con el gobierno japonés que permitió introducir algunas modificaciones al estándar ISDB-T, para adecuarlo a ciertas necesidades planteadas por Brasil. En virtud de dichas modificaciones, surgió el estándar ISDB-Tb también conocido como ISDB-T Internacional, que fue adoptado por Brasil en junio de 2006. Uno de los acuerdos celebrados entre ambos países estableció la necesidad de difundir el estándar en toda la región, con el objetivo de formar un conglomerado que permitiera disminuir los costos de implementación y generara nuevas industrias relacionadas con la televisión.

[2]

El 6 de octubre de 2009 Venezuela adoptó oficialmente el estándar ISDB-Tb como su estándar de TDA. El mecanismo planteado por el Estado venezolano para transmitir las señales de televisión digital, implica el uso exclusivo de instalaciones de la empresa estatal de telecomunicaciones CANTV que es la empresa encargada de difundir a través de una sola antena todas las señales disponibles a un área de cobertura determinada. El equipo de transmisión consiste en cabecera, telepuerto y transmisores:

- Cabecera: Es la instalación que recibe todos los flujos de señales televisivas a ser digitalizadas, sin importar su resolución.
- Telepuerto: Despues de la digitalización, las señales son subidas por esta instalación, mediante una antena parabólica al Satélite Simón Bolívar. Algunas de estas señales estarán destinadas al servicio pago prestado por CANTV llamado CANTV Satelital.
- Transmisor: El equipo transmisor, colocado en la parte más alta de la zona donde se provee la señal digital de televisión, posee otra antena parabólica que recibe del satélite mencionado el conjunto de señales. Dentro de esta instalación, los flujos de señales son organizados y controlados constantemente y es bajada la frecuencia de transmisión hasta llegar al rango que pueden manejar los receptores fijos y móviles.[3]

3.1.4 El flujo de datos BTS

Para el transporte de la información de audio y video el estándar ISDB-Tb usa un flujo especial de datos denominado BTS (Broadcast Transport Stream), que utiliza el formato TS MPEG-2. [2]

El flujo TS MPEG-2 es el estándar adoptado de manera universal para el transporte de la información digitalizada de audio, video y datos, en forma multiplexada. Este flujo, esencialmente asincrónico y concebido originalmente para ser compatible con ATM (Asynchronous Transfer Mode), está formado por una sucesión de paquetes de longitud fija, de 188 bytes cada uno. Sin embargo, pese a su gran versatilidad, este flujo no está preparado para realizar transmisiones en donde la información responda a un orden jerárquico y, menos aún, para permitir la recepción parcial. El objetivo central de la norma ISDB-Tb es que los receptores puedan separar la información transmitida antes de que el flujo de datos llegue a las etapas de decodificación MPEG, que se encuentran al final de la cadena de

procesamiento. Para lograr dicho objetivo, es necesario introducir tres modificaciones en el flujo TS MPEG-2 mediante un proceso denominado "remultiplexación". En la **Figura 3.1.5** se observa el diagrama funcional de una estación de Televisión Digital donde se realiza el proceso de remultiplexación. [2]



Figura 3.1.5: Diagrama funcional simplificado de una estación de Televisión Digital. [2]

El dispositivo que realiza esta tarea, denominado remultiplexor, combina los paquetes de 188 bytes (TS) de las tramas de transporte de entrada (flujos multiplexados), para entregar a su salida un flujo binario único (remultiplexado) llamado Broadcast Transport Stream (BTS).

El remultiplexor básicamente realiza las siguientes funciones:

- Agrega 16 bytes a los paquetes TS, que luego se completan con información específica.
- Forma nuevos paquetes de 204 bytes de longitud (188 +16), llamados TSP (Transport Stream Processed Packet).
- Entrega a su salida un flujo sincrónico cuya tasa binaria es constante e igual a 32,5079 Mbps.
- Posiciona y dispone los paquetes TSP para posibilitar la transmisión jerárquica y la recepción parcial.
- Inserta una determinada cantidad de TSP nulos con el objetivo de mantener la tasa del BTS constante. La cantidad de paquetes nulos depende de la configuración adoptada para cada capa jerárquica. [2]

En la **Figura 3.1.6** se observa la conformación de un paquete BTS o TSP.

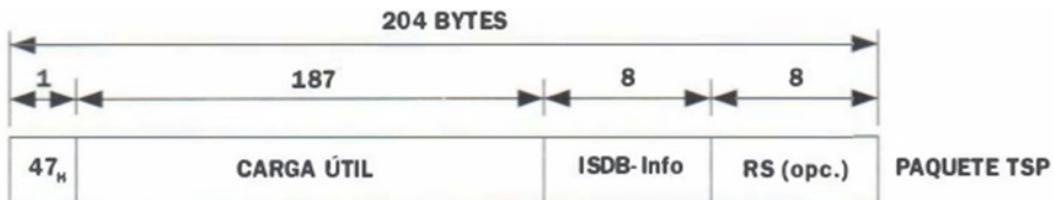


Figura 3.1.6: Estructura de un paquete BTS o TSP. [2]

Con un byte de inicio de valor 47 en base hexadecimal, 187 bytes de carga útil, 8 bytes de ISB-Info que proveen la siguiente información: indicador de la capa jerárquica, contador de TSP, señalización del TSP cabecera de cuadro e información destinada a los canales auxiliares, entre otros. Los 8 bytes restantes permiten incorporar, de manera opcional, un bloque de paridad que permite la corrección de hasta 4 bytes erróneos en cada TSP. [2]

4.1 El USB

Es la interfaz de computadora personal más exitosa de la historia. PC, tabletas, teléfonos y otros dispositivos tienen puertos USB que pueden conectarse a todo, desde teclados, ratones y controladores de juegos a cámaras, impresoras, unidades, dispositivos de audio y vídeo y más. El USB es versátil, confiable, rápido, conservador de energía, económico y soportado por sistemas operativos para ordenadores grandes y pequeños. [4]

Con continuas mejoras y una entrega de energía más flexible, es probable que USB continúe dominando como la interfaz de elección para una gama cada vez mayor de dispositivos. [4]

USB es una solución probable cada vez que desee utilizar una computadora para comunicarse con un dispositivo. La computadora puede

ser una PC convencional o un dispositivo con un procesador incorporado. La interfaz USB es adecuada para dispositivos de consumo producidos en serie, así como productos especializados de pequeño volumen y proyectos únicos. [4]

4.1.1 Versiones USB

El estándar USB ha cambiado nuestra forma de trabajar con el PC. Gracias a él, podemos tener un único cable para conectar toda clase de dispositivos como discos duros externos, cámaras, escáneres o impresoras. [4]

Como no podía ser de otra forma el USB ha experimentado una gran evolución desde sus comienzos llevando al surgimiento de varias versiones:

USB 1.0. El primero, pensado para funcionar con teclados, ratones y dispositivos que requieran de un ancho de banda muy pequeño. Permite trabajar a una velocidad aproximada de 1.5 Megabits por segundo (LowSpeed).

- USB 1.1. Como no podría ser de otra forma y gracias a su éxito no tarda mucho en diseñarse otro estándar que supera al anterior. En este caso su velocidad se multiplica por ocho hasta los 12 Megabits por segundo (FullSpeed).
- USB 2.0. Con este tenemos un salto mayúsculo. Se multiplica la velocidad por 40 veces para llegar a los 480 Megabits por segundo (HighSpeed).
- USB 3.0. Aparece en 2008. Multiplica la velocidad hasta 5 Gigabits por segundo, es decir casi 10 veces más rápido que el USB 2.0 (SuperSpeed).
- USB 3.1. Con tasas de transferencia de 10 Gigabits por segundo, no sólo es mejor en la transferencia de archivos a través del cable, sino

que requiere de menos energía que la versión 3.0 (SuperSpeedPlus). [4]

En la **Tabla 4.1.1** se muestran las especificaciones de velocidades cada versión del USB soporta.

<i>Tabla 4.1.1: Versiones y velocidades de los estándares USB. [4]</i>				
Versión USB	Velocidad Máxima	Megabits por segundo	Megabytes por segundo	Velocidades Soportadas
USB 1.0	LowSpeed	1.5 Mbps	187.5 KB/s	Low Speed
USB 1.1	FullSpeed	12 Mbps	1.5 MB/s	LowSpeed FullSpeed HighSpeed
USB 2.0	HighSpeed	480 Mbps	60 MB/s	LowSpeed FullSpeed HighSpeed
USB 3.0	SuperSpeed	5000 Mbps	625 MB/s	LowSpeed FullSpeed HighSpeed, SuperSpeed
USB 3.1	SuperSpeedPlus	10000 Mbps	1250 MB/s	LowSpeed FullSpeed HighSpeed SuperSpeed SuperSpeedPlus

4.1.1 Beneficios del USB

Los beneficios de USB son la facilidad de uso, transferencia de datos rápida y confiable, bajo costo y ahorro de energía, beneficios:

- **Una interfaz para muchos dispositivos:** USB es lo suficientemente versátil para casi cualquier función periférica estándar, así como dispositivos con funciones especializadas. En lugar de tener un conector diferente y tipo de cable para cada función periférica, una interfaz sirve a muchos.

- **Muchos puertos:** Un PC típico tiene múltiples puertos USB, y los concentradores permiten agregar más puertos.
- **Conexión en caliente:** Los usuarios pueden conectar y desconectar un dispositivo USB cuando lo deseen, independientemente de si el sistema y el dispositivo están alimentados, sin dañar la PC o el dispositivo. El sistema operativo detecta cuando se conecta un dispositivo.
- **Configuración automática:** Cuando un usuario conecta un dispositivo USB a un PC, el sistema operativo detecta el dispositivo y carga el controlador de software apropiado. La primera vez que el dispositivo se conecta, el sistema operativo puede pedir al usuario que identifique un controlador, pero aparte de eso, la instalación es automática. Los usuarios no necesitan reiniciar antes de usar un nuevo dispositivo.
- **No hay configuración de usuario:** Los dispositivos USB no tienen configuraciones seleccionables por el usuario, por lo que los usuarios no tienen puentes para configurar o ejecutar utilidades de configuración.
- **No se requiere alimentación:** La interfaz USB incluye líneas de alimentación y de tierra que proporcionan un valor nominal + 5V desde la PC. Un dispositivo que requiere hasta 500 mA (USB 2.0) o 900 mA (USB 3.1) puede sacar toda su energía del bus en lugar de usar una fuente dedicada.
- **Cables convenientes:** Los conectores USB son pequeños y compactos en comparación con los conectores utilizados por otras interfaces como RS-232. Para garantizar un funcionamiento fiable, la especificación USB define los requisitos eléctricos de los cables y conectores. Un segmento de cable puede ser de hasta 5 m dependiendo de la velocidad del bus y del tipo de conector.

Dependiendo de la velocidad del bus y del tipo de conector, un dispositivo puede estar a 30 m de su computadora anfitrión.

- **De confianza:** La fiabilidad de USB se debe tanto a su hardware como a sus protocolos. Las especificaciones de hardware para controladores USB, receptores y cables aseguran una interfaz eléctricamente silenciosa que reduce el ruido que podría resultar en errores de datos. Los protocolos USB permiten detectar errores en los datos recibidos y notificar al remitente para que pueda retransmitirlo. El hardware realiza la detección, notificación y retransmisión sin el soporte del software o del usuario.
- **Bajo costo:** Debido a que la computadora anfitrión proporciona la mayor parte de la inteligencia para controlar la interfaz, los componentes para dispositivos USB son baratos. Es probable que un dispositivo con una interfaz USB cueste el mismo o menos que un dispositivo equivalente con una interfaz diferente.
- **Ahorro de energía:** Los circuitos y protocolos de ahorro de energía reducen el consumo de energía de un dispositivo mientras mantienen el dispositivo listo para comunicarse cuando sea necesario. Reducir el consumo de energía ahorra dinero, ayuda al medio ambiente, y para los dispositivos de batería. [4]

Muchas de las ventajas de usuario descritas anteriormente también facilitan las cosas para los desarrolladores. Por ejemplo, los estándares de cable USB y la comprobación de errores significan que los desarrolladores no tienen que especificar las características del cable o desarrollar protocolos de comprobación de errores. [4]

Otras ventajas ayudan a los diseñadores de hardware que seleccionan componentes y diseñan los circuitos en los dispositivos y a los

programadores que escriben el firmware incorporado en los dispositivos y el software para comunicarse con los dispositivos. [4]

El USB es versátil posee cuatro tipos de transferencia y cinco velocidades que hacen la interfaz factible para muchos tipos de periféricos. USB tiene tipos de transferencia adecuados para intercambiar grandes y pequeños bloques de datos con y sin restricciones de tiempo. Para los datos que no pueden tolerar retrasos, el ordenador anfitrión puede garantizar el ancho de banda. Estas habilidades son especialmente bienvenidas bajo Windows y otros sistemas operativos de escritorio donde acceder a periféricos en tiempo real es a menudo un desafío. Aunque el sistema operativo, los controladores de dispositivos y el software de aplicación pueden introducir retrasos inevitables, USB facilita la transferencia de transferencias en tiempo real, incluso en sistemas de escritorio. [4]

Soporte de varios sistemas operativos como Windows para PC, Linux, Mac OS y Android. Algunos kernels en tiempo real también admiten USB. En el nivel más básico, un sistema operativo (SO) que admite USB debe hacer tres cosas:

- Detectar cuándo los dispositivos se conectan y se eliminan del sistema.
- Comuníquese con dispositivos recién conectados para averiguar cómo intercambiar datos con ellos.
- Proporcione un mecanismo que permita a los controladores de software transmitir las comunicaciones entre el hardware USB y las aplicaciones que deseen acceder a periféricos USB. [4]

Por otra parte las clases USB definen protocolos para comunicarse con periféricos comunes como impresoras, teclados y unidades. Los desarrolladores pueden programar un dispositivo para que se ajuste a una especificación de clase en lugar de tener que reinventar todo desde la base. [4]

4.1.3 Endpoint USB

Para enviar o recibir datos, el anfitrión USB inicia una transferencia USB. Cada transferencia utiliza un formato definido para enviar datos, una dirección, bits de detección de errores e información de estado y control. El formato varía según el tipo de transferencia y la dirección. Cada comunicación USB se desarrolla entre un anfitrión y un dispositivo. El anfitrión gestiona el tráfico en el bus y el dispositivo responde a las comunicaciones desde el anfitrión. [4]

El USB utiliza un buffer de dispositivo el cual es llamado endpoint que almacena los datos recibidos o datos para transmitir. Cada dirección de endpoint tiene un número el cual varía entre un valor de 0 y 15, una dirección la cual es definida desde la perspectiva del anfitrión, un endpoint IN proporciona datos para enviar al anfitrión y un endpoint OUT almacena los datos recibidos del anfitrión, un número máximo de bytes de datos que el endpoint puede enviar o recibir en una transacción. Cada dispositivo debe tener un endpoint 0 configurado como endpoint de control, por el cual se pueden obtener los datos de configuración del dispositivo, un endpoint configurado para transferencias de control debe transferir datos en ambas direcciones de modo que un endpoint de control consiste en un par de direcciones de endpoint IN y OUT vistas desde la perspectiva del anfitrión que comparten un número de endpoint, generalmente el número 0. [4]

Cada transferencia USB consiste en una o más transacciones que pueden transportar datos hacia o desde un endpoint. Una transacción USB comienza cuando el anfitrión envía un paquete del tipo token (paquete simbólico) en el bus. El paquete token contiene el número y la dirección del endpoint objetivo. [4]

Un paquete token IN solicita un paquete de datos desde el endpoint. Un paquete token OUT precede a un paquete de datos del anfitrión. Además de los datos, cada paquete de datos contiene bits de

comprobación de errores y un Paquete ID (PID) con un valor de secuencia de datos. Muchas transacciones también tienen un paquete de tipo handshake (paquete apretón de manos, ACK) donde el receptor de los datos informa el éxito o el fracaso de la transacción. [4]

Los endpoints pueden ser de diferentes tipos; de control como se especificó anteriormente, interrupción (interrupt en inglés), masivos (bulk en inglés), isócronos (isochronous en inglés). Los diferentes tipos de transferencias USB están relacionados con el tipo de endpoint usado, para transferencias de control se usan endpoints de control, para transferencias de tipo interrupción se usan endpoints de tipo interrupción, para transferencias de tipo masivas usan endpoints de tipo masivo y para transferencias de tipo isócronas se usan endpoints isócronos. [4]

4.1.4 Descriptores USB

Antes de que las aplicaciones puedan comunicarse con un dispositivo, el equipo anfitrión debe conocer el dispositivo y asignar un controlador, la enumeración es el intercambio de información que realiza estas tareas. El proceso incluye asignar una dirección al dispositivo, leer descriptores desde el dispositivo, asignar y cargar un controlador y seleccionar una configuración que especifique los requisitos de alimentación del dispositivo y las interfaces. El dispositivo está entonces listo para transferir datos. [4]

El anfitrión utiliza transferencias de control para solicitar una serie de estructuras de datos denominadas descriptores del dispositivo, mediante el endpoint 0. Los descriptores proporcionan información sobre las capacidades del dispositivo y ayudan al anfitrión a decidir qué controlador asignar al dispositivo. [4]

Los dispositivos USB sólo pueden tener un descriptor de dispositivo. El descriptor de dispositivo incluye información como la versión USB a la que cumple el dispositivo, los ID de producto (idProduct) y de

proveedor (idVendor) utilizados para cargar los controladores apropiados y el número de configuraciones posibles que el dispositivo puede tener. El número de configuraciones indica cuántas ramas de descriptores de configuración están disponibles, dentro de las ramas de configuración de descriptores se encuentran las interfaces de descriptor que podrían verse como un encabezado o agrupación de endpoints en un grupo funcional que realiza una única característica del dispositivo. La **Figura 4.1.1** muestra la estructura de un descriptor USB. [6]

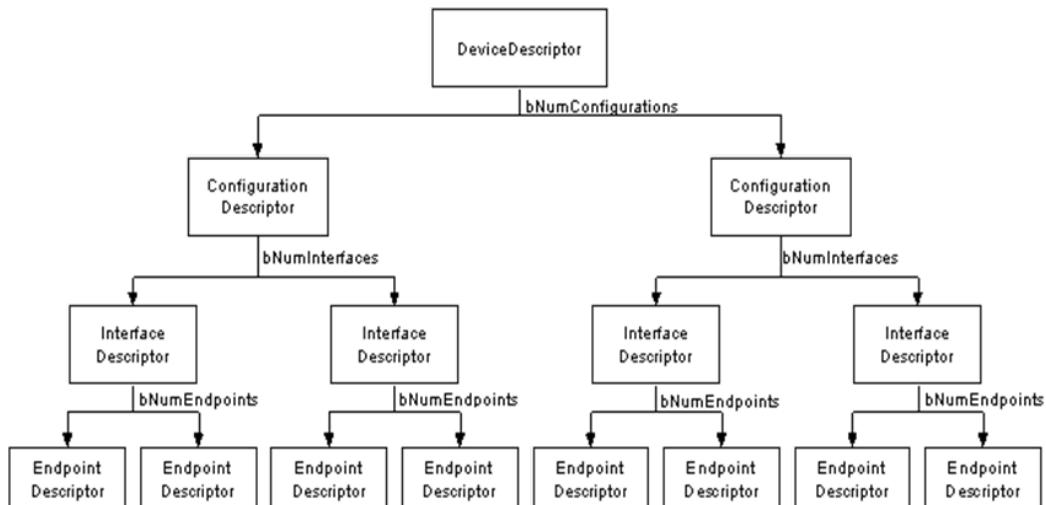


Figura 4.1.1: Estructura de un descriptor USB. [6]

Sin embargo, debe tenerse en cuenta que el cambio de la configuración requiere que se detenga toda la actividad en cada endpoint. Aunque USB ofrece esta flexibilidad, muy pocos dispositivos tienen más de una configuración. [6]

Configuración de Descriptores, un dispositivo USB puede tener varias configuraciones diferentes, aunque la mayoría de los dispositivos son simples y solo tienen uno. El descriptor de configuración especifica cómo se alimenta el dispositivo, cuál es el consumo máximo de energía, el número de

interfaces que tiene. Por lo tanto, es posible tener dos configuraciones, una para cuando el dispositivo está alimentado por bus y otra cuando está alimentado por la red. Como se trata de una "cabecera" de los descriptores de la interfaz, también es factible tener una configuración que utiliza un modo de transferencia diferente a la de otra configuración. Una vez que todas las configuraciones han sido examinadas por el anfitrión, el anfitrión enviará un comando SetConfiguration con un valor no cero que coincide con el valor bConfigurationValue de una de las configuraciones. Esto se utiliza para seleccionar la configuración deseada. En la **Tabla 4.1.2** se muestran los campos del registro de una Configuración de Descriptores. [6]

Tabla 4.1.2: Registro de Configuración de Descriptores. [6]

Número de Campo	Campo	Tamaño en Bytes	Valor	Descripción
0	bLength	1	Numérico	Tamaño del descriptor en bytes.
1	bDescriptorType	1	Constante	Configuración Descriptor (0X02)
2	wTotalLength	2	Numérico	Longitud total en bytes de datos devueltos.
4	bNumInterfaces	1	Numérico	Número de interfaces.
5	bConfigurationValue	1	Numérico	Valor a utilizar como argumento para seleccionar esta configuración.
6	iConfiguration	1	Índice	Índice del descriptor, texto que describe esta configuración.
7	bmAttributes	1	Mapas de Bits	D7 Reservado, establecido en 1. (USB 1.0 Bus Powered) D6 Auto alimentado. D5 Despertador a distancia. D4 Reservado, puesto a 0.
8	bMaxPower	1	mA	Consumo máximo de energía en unidades de 2 mA.

Cuando se lee el descriptor de configuración, devuelve toda la jerarquía de configuración que incluye todos los descriptores de interfaz y de endpoints relacionados.

El campo **wTotalLength** refleja el número de bytes en la jerarquía.

El campo **bNumInterfaces** especifica el número de interfaces presentes para esta configuración.

El campo **bConfigurationValue** es utilizado por la petición SetConfiguration para seleccionar esta configuración.

El campo **iConfiguration** es un índice de un descriptor de cadena que describe la configuración en forma legible por humanos.

El campo **bmAttributes** especifica parámetros de potencia para la configuración. Si un dispositivo es autoalimentado, establece D6 en uno. El bit D7 se usó en USB 1.0 para indicar un dispositivo alimentado por bus, pero esta función en versiones posteriores se realiza especificando el campo bMaxPower. Si un dispositivo utiliza cualquier energía del bus, ya sea como un dispositivo alimentado por bus o como un dispositivo autoalimentado, debe informar de su consumo de energía en bMaxPower. Los dispositivos también pueden admitir la activación remota que permite al dispositivo activar el anfitrión cuando este está en suspensión.

El campo **bMaxPower** define la potencia máxima que el dispositivo drenará del bus. Se trata de unidades de 2mA, por lo que se puede especificar un máximo de aproximadamente 500mA. La especificación permite que un dispositivo alimentado por bus de alta potencia no drene más de 500mA. Si un dispositivo pierde energía externa, entonces no debe drenar más de lo indicado en bMaxPower. Debe fallar cualquier operación que no pueda realizar sin poder externo. En la **Figura 4.1.2** se observa un esquema de un dispositivo USB con dos configuraciones de descriptores.

[6]

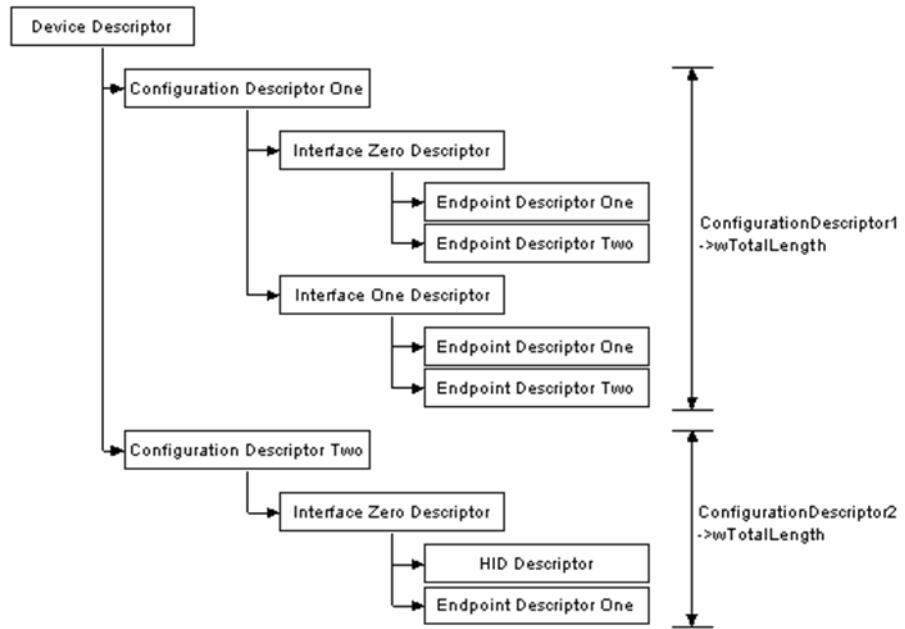


Figura 4.1.2: Esquema de un dispositivo USB con dos configuraciones.
[6]

La **Interfaz de Descriptores**, el descriptor de interfaz podría verse como un encabezado o agrupación de los endpoints en un grupo funcional que realiza una única característica del dispositivo. El registro del descriptor de interfaz cumple con el siguiente formato mostrado en la **Tabla 4.1.3**.

Tabla 4.1.3: Registro de Interfaz de Descriptores. [6]				
Número de Campo	Campo	Tamaño en Bytes	Valor	Descripción
0	bLength	1	Numérico	Tamaño del descriptor en bytes. (9 Bytes).
1	bDescriptorType	1	Constante	Configuración Descriptor (0X04).
2	bInterfaceNumber	1	Numérico	Número de la interfaz.
3	bAlternateSetting	1	Numérico	Valor utilizado para seleccionar un ajuste alternativo.
4	bNumEndpoints	1	Numérico	Número de endpoints utilizados para esta interfaz.
5	bInterfaceClass	1	Clase	Código de clase (asignado por USB Org).
6	bInterfaceSubClass	1	Subclase	Código de subclase (asignado por USB Org).
7	bInterfaceProtocol	1	Protocolo	Código de protocolo (asignado por USB Org).
8	iInterface	1	Índice	Índice del descriptor, texto que describe esta configuración.

El campo **bInterfaceNumber** indica el índice del descriptor de interfaz. Esto debe ser basado en cero, e incrementado una vez para cada nuevo descriptor de interfaz.

El campo **bAlternativeSetting** se puede utilizar para especificar interfaces alternativas. Estas interfaces alternativas se pueden seleccionar con la petición SetInterface.

El campo **bNumEndpoints** indica el número de endpoints utilizados por la interfaz. Este valor debe excluir el endpoint 0 y se utiliza para indicar el número de descriptores de endpoints.

El campo **bInterfaceClass**, **bInterfaceSubClass** y **bInterfaceProtocol** se pueden utilizar para especificar clases soportadas (por ejemplo, HID, comunicaciones, almacenamiento masivo, etc.) Esto permite que muchos dispositivos utilicen controladores de clase que evitan la necesidad de escribir controladores específicos para su dispositivo. El campo **iInterface** permite una descripción de texto de la interfaz.

Los **Descriptores de Endpoint**, los descriptores de endpoint se usan para describir endpoints distintos del endpoint cero. Siempre se supone que el endpoint cero es un endpoint de control y se configura antes de que se soliciten incluso los descriptores. El anfitrión utilizará la información devuelta de estos descriptores para determinar los requisitos de ancho de banda del bus, la **Tabla 4.1.4** muestra la conformación de un registro para un descriptor de un endpoint. [6]

<i>Tabla 4.1.4: Registro de Endpoint de Descriptores. [6]</i>				
Número de Campo	Campo	Tamaño en Bytes	Tipo de Valor	Descripción
0	bLength	1	Número	Tamaño del descriptor en bytes (7 bytes).
1	bDescriptorType	1	Constante	Configuración Descriptor (0X05)
2	bEndpointAddress	2	Endpoint	Dirección del Endpoint.
3	bmAttributes	1	Mapa de bits	Bits 0..1 Tipo de Transferencia 00 = Control 01 = Isócronas 10 = Masiva 11 = Interrupción

				<p>Bits 2..7 están reservados. Si endpoint isócrono.</p> <p>Bits 3..2 = Tipo de sincronización(Modo Isócrono)</p> <table border="0"> <tr><td>00</td><td>=</td><td>No sincronización</td></tr> <tr><td>01</td><td>=</td><td>Asincrónico</td></tr> <tr><td>10</td><td>=</td><td>Adaptado</td></tr> <tr><td>11</td><td>=</td><td>Sincrónico</td></tr> </table> <p>Bits 5..4 = Tipo de uso (Modo Isócrono)</p> <table border="0"> <tr><td>00</td><td>=</td><td>Data Endpoint</td></tr> <tr><td>01</td><td>=</td><td>Respuesta Endpoint</td></tr> <tr><td>10</td><td>=</td><td>Datos de respuesta explícita Endpoint.</td></tr> <tr><td>11</td><td>=</td><td>Reservado.</td></tr> </table>	00	=	No sincronización	01	=	Asincrónico	10	=	Adaptado	11	=	Sincrónico	00	=	Data Endpoint	01	=	Respuesta Endpoint	10	=	Datos de respuesta explícita Endpoint.	11	=	Reservado.
00	=	No sincronización																										
01	=	Asincrónico																										
10	=	Adaptado																										
11	=	Sincrónico																										
00	=	Data Endpoint																										
01	=	Respuesta Endpoint																										
10	=	Datos de respuesta explícita Endpoint.																										
11	=	Reservado.																										
4	wMaxPacketSize	2	Número	Tamaño máximo de paquete que este endpoint puede enviar o recibir																								
6	blInterval	1	Número	Intervalo para las transferencias de datos de endpoint. Valor se cuenta en fotogramas. Se ignora para endpoints Masivos y de Control. Para endpoints Isócronos el valor debe ser 1 y el campo puede variar de 1 a 255 para endpoints de Interrupción.																								

El campo **bEndpointAddress** indica que endpoint describe este descriptor.

El campo **bmAttributes** especifica el tipo de transferencia. Esto puede ser Control, Interrupción, Isocrónicas o Masivas. Si se especifica un punto final isócrono, se pueden seleccionar atributos adicionales como los tipos Sincronización y uso.

El campo **wMaxPacketSize** indica el tamaño máximo de la carga útil para este endpoint.

El campo **bInterval** se utiliza para especificar el intervalo de sondeo de ciertas transferencias. Las unidades se expresan en marcos, por lo que equivale a 1 mili segundo para dispositivos de velocidad baja y 125 micro segundos para dispositivos de alta velocidad. [6]

4.1.5 Tipos transferencias USB

Las transferencias de control permiten al anfitrión leer información acerca del dispositivo asignar una dirección a un dispositivo, etc y posee tres etapas, la etapa de Configuración, la etapa de Datos y y finalmente la etapa denominada Estado. [4]

Los otros tipos de transferencia no tienen etapas definidas. En su lugar, el software de nivel superior define cómo interpretar los datos sin procesar. Las transferencias de tipo masivo son las más rápidas, pero no tienen un tiempo garantizado, las transferencias de interrupción tienen garantizada una latencia máxima o el tiempo entre los intentos de transacción. Las transferencias isócronas garantizan el tiempo pero no corrigen errores. [4]

4.1.5.1 Transferencias de Tipo Control

Las transferencias de control tienen hasta tres etapas: Configuración, Datos (opcional) y Estado. La etapa de configuración contiene la solicitud. Cuando está presente, la etapa Datos contiene datos

del anfitrión o del dispositivo, dependiendo de la solicitud. La etapa Estado contiene información sobre el éxito de la transferencia. En una transferencia de lectura de control, el dispositivo envía datos en la etapa Datos. En una transferencia de escritura de control, el anfitrión envía datos en la etapa Datos o la etapa Datos está ausente. [4]

Etapa de Configuración: La etapa de configuración consiste en una transacción de configuración que identifica la transferencia como una transferencia de control y transmite la solicitud y otra información que el dispositivo necesita para completar la solicitud. Los dispositivos deben devolver un apretón de manos (ACK) para cada transacción de configuración recibida sin error. Un endpoint que está en medio de otra transferencia de control debe abandonar esa transferencia y reconocer la nueva transacción de instalación. En la **Tabla 4.1.5** se especifican los paquetes en una etapa de configuración. [4]

Tabla 4.1.5: Descripción tipo de paquetes etapa de configuración USB. [4]	
Paquete Token	Propósito: identifica al receptor e identifica la transacción como una transacción de configuración. Enviado por: el anfitrión. PID: SETUP. Contenidos adicionales: direcciones de dispositivos y endpoints.
Paquete de datos	Propósito: transmite la solicitud y la información relacionada. Enviado por: el anfitrión. PID: DATA0. Contenidos adicionales: ocho bytes en cinco campos: bmRequestType especifica la dirección del flujo de datos, el tipo de solicitud y el destinatario. bRequest identifica la solicitud, lectura o escritura. wValue puede pasar información específica de la solicitud al dispositivo. Cada petición puede definir el significado de estos dos bytes a su manera. Por ejemplo, en una solicitud SetAddres (donde el anfitrión especifica una dirección para usar en futuras comunicaciones con el dispositivo.), wValue contiene la dirección del dispositivo.

wIndex puede pasar información específica de la solicitud al dispositivo. Un uso típico es pasar un índice u offset tal como un número de interfaz o de endpoint, pero cada petición puede definir el significado de estos dos bytes de cualquier manera.

wLength son dos bytes que contienen el número de bytes de datos. Para una transferencia de anfitrión a dispositivo, wLength es el número exacto de bytes que el anfitrión tiene intención de transferir. Para una transferencia de dispositivo a servidor, wLength es el número máximo de bytes a transferir, y el dispositivo puede devolver este número de bytes o menos. Si el campo es cero, la transferencia no tiene etapa de datos.

Paquete de Apretón de manos

Propósito: transmite el acuse de recibo del dispositivo.

Enviado por: el dispositivo.

PID: ACK (apretón de manos, datos recibidos sin error).

Contenidos adicionales: ninguno. El paquete de apretón de manos consiste en el PID solo.

Comentarios: Si el dispositivo detecta un error en la configuración o paquete de datos recibidos, el dispositivo no devuelve ningún apretón de manos. El hardware del dispositivo se encarga de la comprobación de errores y el envío de la ACK sin soporte de firmware necesario.

Etapa de Datos: La etapa Datos, cuando está presente, consiste en una o más transacciones IN u OUT. Una etapa de datos con transacciones IN envía datos al anfitrión. Un ejemplo es la solicitud GetDescriptor, donde el dispositivo envía un descriptor solicitado al anfitrión. Una etapa de datos con transacciones OUT envía datos al dispositivo. Un ejemplo es la solicitud de SetReport, donde el anfitrión envía un informe a un dispositivo. Si wLength en la transacción de instalación es 0x0000, la transferencia no tiene ninguna etapa de datos. En el descriptor del dispositivo, bMaxPacketSize especifica el número máximo de bytes de datos por paquete. Si todos los datos no pueden encajar en un paquete, la etapa utiliza múltiples transacciones. En la **Tabla 4.1.6** se especifican los paquetes en la etapa de datos. [4]

Tabla 4.1.6: Descripción tipo de paquetes etapa de datos USB. [4]	
Paquete Token	<p>Propósito: Identifica al receptor e identifica la transacción como una transacción IN u OUT.</p> <p>Enviado por: el anfitrión.</p> <p>PID: Si la solicitud requiere que el dispositivo envíe datos al anfitrión, el PID es IN. Si la solicitud requiere que el anfitrión envíe datos al dispositivo, el PID es OUT.</p> <p>Contenidos adicionales: direcciones de dispositivos y endpoints.</p>
Paquete de datos	<p>Propósito: Transfiere todo o una porción de los datos especificados en el campo wLength del paquete de datos de la transacción de configuración.</p> <p>Enviado por: el anfitrión.</p> <p>PID: El primer paquete es DATA1. Cualquier paquete adicional en la etapa Datos se alterna DATA0 / DATA1.</p> <p>Contenidos adicionales: El anfitrión envía datos o ZLP (paquete de longitud cero). El dispositivo puede enviar datos o NAK (endpoint ocupado) o STALL (solicitud no admitida o endpoint detenido).</p>
Paquete de Apretón de manos	<p>Propósito: transmite el acuse de recibo del dispositivo.</p> <p>Enviado por: El receptor del paquete de datos de la etapa de datos. Si el PID del paquete de símbolo es IN, el anfitrión envía el paquete de apretón de manos. Si el PID del paquete es OUT, el dispositivo envía el paquete de apretón de manos.</p> <p>PID: Un dispositivo puede devolver ACK, NAK o. Un dispositivo de highspeed que está recibiendo paquetes de datos múltiples puede devolver NYET para indicar que los datos de la transacción actual fueron aceptados pero el endpoint aún no está listo para otro paquete de datos. Un anfitrión puede devolver sólo ACK.</p> <p>Contenidos adicionales: ninguno. El paquete de apretón de manos consiste en el PID solo.</p> <p>Comentarios: Si el dispositivo detecta un error en la configuración o en el paquete de datos recibidos, el dispositivo no devuelve ningún apretón de manos.</p>

Etapa de Estado: La etapa de estado completa la transferencia. En algunos casos (tal como después de recibir el primer paquete de un descriptor de dispositivo durante la enumeración), el anfitrión puede comenzar la etapa de estado antes de que se complete la etapa de datos y el dispositivo debe detectar el paquete token de la etapa de estado,

abandonar la etapa de datos y completar la etapa de estado. En la tabla

Tabla 4.1.7 se especifican los paquetes en una etapa de estado. [4]

Tabla 4.1.7: Descripción tipo de paquetes etapa de estado USB. [4]	
Paquete Token Propósito: Identifica al receptor e identifica la transacción como una transacción IN u OUT. Enviado por: el anfitrión. PID: Lo opuesto a la dirección del paquete de datos de la transacción anterior. Si el PID de la etapa de datos OUT o si no había ninguna etapa de datos, el PID de la etapa de estado es IN. Si el PID de la etapa de datos estaba en IN, el PID de la etapa de estado es OUT.	
Paquete de datos Propósito: Permite al receptor de los datos de la etapa de datos indicar el estado de la transferencia. Enviado por: El dispositivo si el PID del paquete token de la etapa de estado es IN o por el anfitrión si el PID del paquete token de la etapa de estado es OUT. PID: DATA1. Contenidos adicionales: Para la mayoría de las peticiones estándar, un ZLP del dispositivo indica que el dispositivo ha realizado la acción solicitada.	
Paquete de Apretón de manos Propósito: transmite el acuse de recibo del dispositivo. Enviado por: es enviado por el receptor del paquete de datos de la etapa de datos. Si el PID del paquete token es IN, el anfitrión envía el paquete de apretón de manos. Si el PID del paquete es OUT, el dispositivo envía el paquete de apretón de manos. PID: Un dispositivo puede devolver ACK, NAK o STALL. El anfitrión devuelve ACK en respuesta a un paquete de datos recibido sin error. Contenidos adicionales: ninguno. El paquete de apretón de manos consiste en el PID solo. Comentarios: El paquete de apretón de manos de la etapa estado es la transmisión final en la transferencia. Si el receptor detectó un error en el token o en el paquete de datos, el receptor no devuelve ningún paquete de apretón de manos.	

Algunas de las solicitudes del estándar USB son las siguientes: **Get Status** (Obtener estado), el anfitrión solicita el estado de las características de un dispositivo, interfaz o endpoint. **Clear Feature** (eliminar

característica), el anfitrión solicita deshabilitar una función en un dispositivo, interfaz o endpoint. **Set Feature** (fijar una característica), el anfitrión solicita habilitar una función en un dispositivo, una interfaz o un endpoint. **Set Address** (fijar dirección), el anfitrión especifica una dirección para usar en futuras comunicaciones con el dispositivo. **Get Descriptor** (obtener descriptor), el anfitrión solicita un descriptor específico. **Set Descriptor** (fijar descriptor), el anfitrión agrega un descriptor o actualiza un descriptor existente. **Get Configuration** (obtener configuración), el anfitrión solicita el valor de la configuración del dispositivo actual. **Set Configuration** (fijar configuración), el anfitrión solicita al dispositivo que utilice la configuración especificada. **Get Interface** (obtener interfaz) para las interfaces que tienen configuraciones alternativas mutuamente exclusivas, el anfitrión solicita la configuración de la interfaz activa actualmente. **Set Interface** (fijar interfaz), para las interfaces que tienen configuraciones alternativas, mutuamente exclusivas, el anfitrión solicita al dispositivo que utilice una configuración de interfaz específica. En la **Tabla 4.1.8** se muestran los valores campos de los campos de algunas solicitudes en la etapa de transferencias de control. [6]

Tabla 4.1.8: Valores de campos para de Transmisiones de Control USB.[6]

bmRequestType	bRequest	wValue	wIndex	wLength	Data
(1000 0000)b	GET_STATUS (0x00)h	0	0	2	Estado del dispositivo
(0000 0000)b	CLEAR_FEATURE (0x01)h	Selector de Característica	0	0	-
(0000 0000)b	SET_FEATURE (0x03)h	Selector de Característica	0	0	-
(0000 0000)b	SET_ADDRESS (0x05)h	Dirección de Dispositivo	0	0	-
(1000 0000)b	GET_DESCRIPTOR (0x06)h	Tipo Descriptor Índice	0	o e Caracte res ID	Longitud Descriptor

(0000 0000)b	SET_DESCRIPTOR (0x07)h	Tipo Descriptor Índice	0 eCaracteres ID	oLongitud Descripto r	Descriptor
(1000 0000)b	GET_CONFIGURATION (0x08)h	0	0	1	Valor Configuraci ón
(0000 0000)b	SET_CONFIGURATION (0x09)h	Valor Configuraci ón	0	0	-

4.1.5.2 Transferencias de Tipo Masivas

Las transferencias masivas son útiles para transferir datos cuando el tiempo no es crítico. Una transferencia masiva puede enviar grandes cantidades de datos sin obstruir el bus porque las transferencias se diferencian a los otros tipos de transferencia, esperando hasta que el tiempo esté disponible. Los usos para transferencias masivas incluyen enviar datos a una impresora y leer y escribir en una unidad. En un bus inactivo, las transferencias masivas son el tipo de transferencia más rápida para grandes cantidades de datos. [4]

El controlador del anfitrión garantiza que las transferencias masivas se completarán eventualmente pero no reservan el ancho de banda para ellas. Se garantiza que las transferencias de control tienen un 10% del ancho de banda a LowSpeed y FullSpeed y un 20% a HighSpeed y SuperSpeed. Las transferencias de interrupción e isócronas pueden utilizar el resto. Así que si un bus está muy ocupado, una transferencia masiva puede tomar mucho tiempo. [4]

Una transferencia masiva finaliza satisfactoriamente cuando la cantidad esperada de datos se ha transferido o cuando una transacción contiene menos que el tamaño máximo de paquete del endpoint, incluyendo cero bytes de datos. La especificación USB 2.0 no define un protocolo para indicar el número de bytes de datos en una transferencia masiva. Cuando sea necesario, el dispositivo y el anfitrión pueden utilizar un protocolo

específico de clase o definido por el proveedor para transmitir esta información. Por ejemplo, una transferencia puede comenzar con un encabezado que especifica el número de bytes que se van a transferir o el dispositivo o el anfitrión puede utilizar un protocolo específico de clase o definido por el proveedor para solicitar una cantidad de datos. Los dispositivos con velocidad única de tipo LowSpeed no soportan las transferencias masivas. [4]

4.1.5.3 Transferencias de Tipo Interrupción

Las transferencias de interrupción son útiles cuando los datos tienen que transferirse sin demora. Los dispositivos de velocidad LowSpeed, que sólo admiten transferencias de control y de interrupción, probablemente utilizarán transferencias de interrupción para la transmisión y recepción de datos. A velocidades LowSpeed y FullSpeed, el ancho de banda disponible para los endpoints de interrupción es limitado, pero a velocidades HighSpeed y la SuperSpeed el ancho de banda es mejor y más flexible. Una transferencia de interrupción garantiza una latencia máxima, o tiempo entre intentos de transacción. En otras palabras, no hay una tasa de transferencia garantizada, solo la garantía de que el anfitrión hará que el ancho de banda esté disponible para un intento de transacción en cada período de latencia máxima. [4]

Las transferencias de interrupción son parecidas a las interrupciones utilizadas en los microcontroladores porque garantizan una respuesta rápida del anfitrión. Para los endpoints masivos y de interrupción, el firmware utiliza típicamente interrupciones para detectar nuevos datos recibidos. En un bus USB 2.0, los endpoints masivos y de interrupción deben esperar a que el anfitrión solicite datos antes de enviar los datos, todas las velocidades permiten transferencias de interrupción. [4]

Una transferencia de interrupción USB 2.0 consiste en una o más transacciones IN u OUT. La transferencia de datos en ambas direcciones

requiere una transferencia y una tubería (pipe) de transmisión separadas para cada dirección. En el bus, las transacciones de interrupción son idénticas a las transacciones masivas con la diferencia de que las transacciones de interrupción han garantizado una latencia máxima y, por lo tanto, un tiempo diferente para recibirlas o enviarlas por parte del anfitrión. [4]

Una transferencia de interrupción finaliza satisfactoriamente cuando la cantidad esperada de datos se ha transferido o cuando una transacción contiene menos que el tamaño máximo de paquete del punto final, incluyendo cero bytes de datos. La especificación USB no define un protocolo para especificar la cantidad de datos en una transferencia de interrupción. Cuando sea necesario, el dispositivo y el anfitrión pueden utilizar un protocolo específico de clase o definido por el proveedor para transmitir esta información. [4]

Los bytes de datos máximos permitidos en el paquete de datos de una transacción de interrupción varían con la velocidad del bus y el número de paquetes por microframe (alta velocidad) o el número de paquetes por intervalo de bus y el valor bMaxBurst (Enhanced SuperSpeed), la **Tabla 4.1.9** muestra los tamaños de paquetes que soporta cada velocidad de transmisión USB y el intervalo de transmisión. Donde un frame representa a 1 mili segundo, un microframe equivale a 125 micro segundos, intervalo bus depende del campo bMaxBurst. [4]

Tabla 4.1.9: Especificación de tamaños de paquetes según el tipo de velocidad de transmisión USB e intervalo de transmisión. [4]

Velocidad del Bus	Tamaño máximo del paquete de datos. (Bytes)	Paquetes máximos garantizados / intervalo
SpeedLow	1-8	1 / 10 frame
SpeedFull	1-64	1 / frame
SpeedHigh	1-1024	1 / microframe
	513-1024	2 / microframe
	683-1024	3 / microframe
SuperSpeed/SuperSpeed Plus	1-1024 y bMaxBurst = 0 1024 y bMaxBurst > 0	1 / intervalo bus 3 / intervalo bus

Un endpoint de interrupción de un dispositivo LowSpeed sólo puede solicitar hasta 8 bytes cada 10 mili segundos. Los dispositivos con endpoints que necesitan transferir más de 800 bytes/s no deben usar una interfaz LowSpeed. Un endpoint de velocidad FullSpeed puede solicitar hasta 64 bytes por fotograma, o 64 kB/s. Un endpoint de velocidad HighSpeed puede solicitar hasta tres paquetes de 1024 bytes por microframe para un rendimiento máximo de 24.576 MB/s. Un endpoint de velocidad HighSpeed que solicita más de 1024 bytes por microframe se llama endpoint de alto ancho de banda. Para los anfitriones que no soportan extremos de interrupción de alto ancho de banda, el rendimiento máximo es 8.192 MB/s. Si el controlador del anfitrión no soporta interfaces alternativas, el máximo es el 64 kB/s permitido para la interfaz por defecto. Un endpoint de velocidad SuperSpeed puede solicitar una ráfaga de hasta tres paquetes de 1024 bytes por intervalo de bus para un rendimiento de datos máximo de 24.576 MB/s, igual que para el de HighSpeed. [4]

Si un dispositivo no devuelve un paquete de apretón de manos esperado, los controladores del anfitrión vuelven a intentar hasta un máximo de dos veces más. Al recibir NAK, un anfitrión USB 2.0 puede volver a intentar sin límites de intentos. [4]

4.1.5.3 Transferencias de Tipo Isócronas

Las transferencias isócronas son transferencias en tiempo real que son útiles cuando los datos deben llegar a una velocidad constante o dentro de un límite de tiempo específico y los errores ocasionales son tolerables. A FullSpeed y EnhancedSuperSpeed, las transferencias isócronas pueden transferir más datos por intervalo de trama o de bus comparado con las transferencias de interrupción, pero el tipo de transferencia no admite el reenvío automático de los datos recibidos con errores. [4]

Los datos que eventualmente se consumen a una velocidad constante no requieren siempre una transferencia isócrona. Por ejemplo, un anfitrión puede usar una transferencia masiva o de interrupción para enviar un archivo de música a un dispositivo. Al comenzar o después de recibir el archivo, el dispositivo puede transmitir la música a petición. [4]

Una transferencia isócrona es una manera de asegurar que cualquier bloque de datos tiene ancho de banda reservado en un bus ocupado. A diferencia de las transferencias masivas, un anfitrión garantiza que el ancho de banda isócrono requerido de una configuración estará disponible, por lo que el tiempo de finalización es predecible. [4]

5.1 El sistema operativo GNU/Linux

GNU/Linux, es el término empleado para referirse a la combinación del sistema operativo GNU, desarrollado por la FSF (Free Software Foundation), y el núcleo (kernel) Linux, desarrollado por Linus Torvalds y la Linux Foundation. Su desarrollo es uno de los ejemplos más prominentes de software libre; todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL (Licencia Pública General de GNU) y otra serie de licencias libres. [9]

El sistema operativo GNU/Linux es, en sí, una interfaz de software que nos permite a nosotros, como humanos, interactuar con los dispositivos de hardware. [8]

Esta interfaz está dividida en varias capas, cada una de las cuales tendrá una funcionalidad específica, como se observa en la **Figura 5.1.1**.

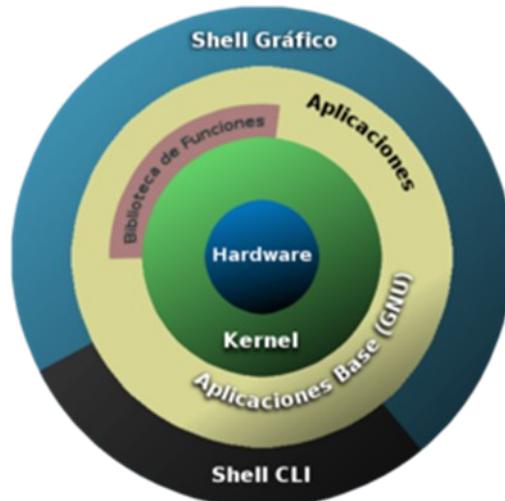


Figura 5.1.1: Capas de un sistema operativo Linux. [8]

El hardware, como se ve, se encuentra en el centro de la figura, y tenemos varias “capas” de software que lo envuelven, siendo el operador externo quien interactúa, desde afuera, con todo el esquema. [8]

Algunas de las partes más importantes que componen a un sistema operativo GNU/Linux se listan a continuación:

1. El núcleo Linux

El núcleo del sistema es una capa de software que recubre al hardware, y que contiene muchas utilidades incorporadas. Primero y principal, el kernel se comunica con el hardware por medio de controladores de dispositivos. Estos controladores son programados por los propios

fabricantes de los dispositivos, o, en su defecto, por programadores de Linux que realizan ingeniería inversa sobre controladores privativos.

El núcleo también brinda otras utilidades, que podrían, englobarse en los siguientes áreas:

- Administración de memoria virtual.
- Administración de procesos.
- Administración de Entrada/Salida.
- Administración de red.

Con estas características, nuestro kernel podrá planificar procesos en el procesador, gestionar recursos de almacenamiento principal, como ser memoria RAM, y espacio de intercambio (swap), y gestionar las peticiones de entrada/salida con cualquier dispositivo e interfaz de red. [8]

2. El sistema operativo GNU

Una capa externa al núcleo es la del propio sistema operativo, y sus aplicaciones. El sistema operativo se compone, entre otras cosas, de utilidades como editores de texto (vi/vim, Emacs, nano, etc), compiladores, intérprete de scripts (Bash, sh, rsh, python, etc), entre otras.

El sistema operativo es el encargado de brindar soporte a aplicaciones de usuario, mediante intérpretes de órdenes, como ser una ventana de línea de comandos, o un entorno gráfico, ya sea un entorno de escritorio completo, o simples administradores de ventanas. [8]

3. La shell de usuario

Se unen, bajo el concepto de shell de usuario, las aplicaciones de usuario comunes, como paquetes de ofimática, navegadores web, etc.

Aquí tenemos dos tipos de aplicaciones, aquellas que corren por línea de comandos, como los comandos comunes de linux, y la mayor parte de los servicios, y aplicaciones gráficas, que corren sobre terminales

gráficas montadas en un servidor gráfico llamado Servidor X. Así, toda aplicación que estemos ejecutando, correrá en una terminal, ya sea de texto, o gráfica. [8]

4. La interfaz de llamadas al sistema

Existe un elemento intermedio entre el núcleo Linux y el sistema GNU, que no se muestra en la Figura 6.1.1, y que se denomina Interfaz de llamadas al sistema, o “syscall”. Esta interfaz, provee al sistema operativo un API, o interfaz de programación de aplicaciones, que no es otra cosa que una serie de funciones del núcleo que pueden ser accedidas desde el sistema operativo mediante llamadas. Así es como el sistema operativo puede comunicarse con el núcleo, enviarle instrucciones, y recibir resultados, en la **Figura 5.1.2** se muestra de manera gráfica la función de la interfaz de llamadas del sistema. [8]

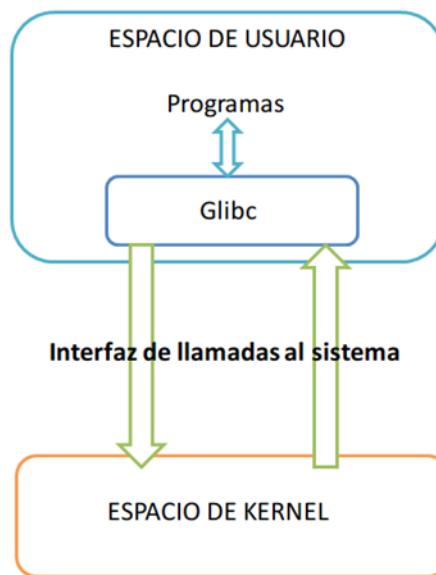


Figura 5.1.2: Especificación gráfica de la función de la interfaz de llamadas al sistema, en un sistema operativo GNU/Linux.

Espacio del núcleo (kernel space). Toda subrutina que forma parte del núcleo tales como los módulos o drivers se consideran que están en el espacio del núcleo. [8]

Espacio de usuario (user space). Los programas que utiliza el usuario final, tales como las “shell” u otras aplicaciones con ventanas residen en el espacio de usuario. Como es lógico estas aplicaciones necesitan interaccionar con el hardware del sistema, pero no lo hacen directamente, sino a través de las funciones que soporta el núcleo. [8]

CAPÍTULO IV

MARCO METODOLÓGICO

6.1 Utilidades de Software.

Las rutinas de software desarrolladas en este trabajo de grado tienen como finalidad otorgar la capacidad reconocer al dispositivo USB de TDA cuando este se conecte o desconecte desde un computador (De escritorio o laptop) bajo un sistema operativo GNU/Linux, realizar acciones de control sobre el dispositivo USB, controlar los buses de datos, reproducir el contenido multimedia obtenido a través de los buses de datos del dispositivo y por último deben documentarse las rutinas desarrolladas y crear un manual de instalación del software desarrollado.

Se realizó un proceso de investigación sobre las herramientas existentes útiles con la finalidad de seleccionar aquellas que permitieran alcanzar los objetivos planteados.

En cuanto a la interacción USB anfitrión-dispositivo se optó por una librería de nombre Libusb que ofrece diversas soluciones en el campo de la comunicación USB, la librería escrita en lenguaje C posee un código muy sencillo de usar e interpretar, además el fabricante Cypress hace uso de esta librería para manejar sus dispositivos USB en un ambiente GNU/Linux.

Cabe destacar que Samsung el fabricante del dispositivo Tuner NIM proporciona una API para el control del dispositivo, la cual fue empleada en este proyecto.

Para el proceso de control multimedia ya el paquete de software VLC proporciona una API llamada Libvlc para utilizar el núcleo de este

reproductor multimedia en software desarrollados por terceros, esta fue la librería usada para interactuar con el reproductor VLC.

En cuanto a la interfaz gráfica se utilizaron las bibliotecas de software de GTK debido al gran soporte que posee y a su abstracción a la hora de programar. Para la elaboración de la documentación se utilizó el software Doxygen, una poderosa herramienta destinada solo a la documentación de software con muchas funcionalidades para este fin, la creación del manual de instalación se hizo utilizando LibreOffice un popular software libre de ofimática.

Como el proyecto de software fue desarrollado bajo un ambiente GNU/Linux se uso la distribución de nombre Fedora en su versión 24, ya que Fedora es excelente como sistema operativo de desarrollo de software debido a que maneja los repositorios de software más actualizados de las versiones de los programas y librerías usadas para el desarrollo de software.

El lenguaje de programación usado fue C++ debido a que las librerías Libusb, Libvlc, la Api de Samsung y el entorno gráfico GTK están escritos en C. C++ logra un nivel de abstracción superior a C facilitando el desarrollo y disminuyendo el tiempo del mismo sin perder la compatibilidad con rutinas escritas en C.

Como C++ fue seleccionado como lenguaje de programación para desarrollar por completo todo el proyecto se escogió como IDE de desarrollo el software CodeBlocks que permite desarrollar proyectos escritos en C++ haciendo uso del compilador g++, siendo este software muy popular en los entornos de desarrolladores en GNU/Linux.

Todas estas herramientas fueron usadas para alcanzar los objetivos planteados en este trabajo de grado y se describe su implementación en los puntos posteriores.

6.1.1 Esquema de Software.

El esquema de software empleado se basa en un modelo piramidal siendo las rutinas USB la base de la pirámide seguidas de las rutinas de la API de Samsung y luego las rutinas multimedia de VLC y por último el empaquetado gráfico. Como se observa en la **Figura 6.1.1** la base de la pirámide son las rutinas USB ya que sin ellas ninguna de las rutinas posteriores pueden llevarse a cabo.

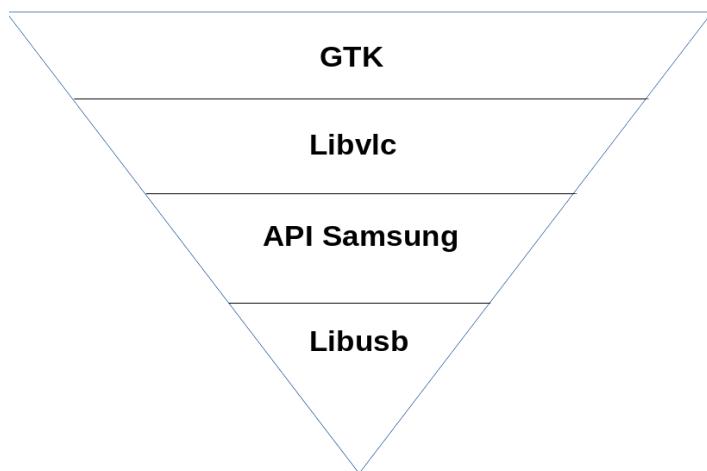


Figura 6.1.1: Pirámide de Dependencia.

Fue necesaria la utilización de hilos de programación, un hilo es simplemente una tarea que puede ser ejecutada al mismo tiempo que otra tarea en paralelo. Esto debido a que si el software estuviera solamente enfocado en un solo proceso como por ejemplo la obtención constante de paquetes BTS del dispositivo TDA este no podría manejar el reproductor multimedia para reproducir estos paquetes lo que hace necesario que sean empleado un hilo diferente para cada proceso haciendo posible que ambos procesos se ejecuten en paralelo.

El software desarrollado emplea tres grandes hilos de procesos que se describen a continuación.

- **Hilo principal:** Este representa al hilo principal de ejecución del programa, en este hilo se ejecutan las tareas de las funciones de la API de Samsung, se ejecutan las actividades del reproductor multimedia VLC y los procesos de la interfaz gráfica.
- **Hilo de Conexión en Caliente:** Este hilo se encarga del monitoreo constantemente de la conexión en caliente del dispositivo USB.
- **Hilo de adquisición de paquetes BTS:** Este hilo se encarga del proceso de obtención de los paquetes BTS desde el dispositivo USB.

7.1 La API de Samsung

Samsung ofrece una API para el manejo del dispositivo Tuner NIM DNOD22QXV104A definiendo funciones para el manejo del demodulador y sintonizador.

7.1.1 Funciones del demodulador SemcoTC90527.

En la **Tabla 7.1.1** se muestran el nombre de las funciones pertenecientes a la API Samsung para el manejo del demodulador y su descripción.

Tabla 7.1.1: Funciones de la API Samsung, demodulador. [10]	
SemcoTC90527Init()	Descripción: Esta función inicializa el demodulador. Parámetros: Ninguno. Retorna: <ul style="list-style-type: none">• 0 En caso exitoso.

	<ul style="list-style-type: none"> • Otro valor <p>En caso de error.</p>
SemcoTC90527SoftReset()	<p>Descripción: Esta función reinicia el demodulador.</p> <p>Parámetros: Ninguno.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • 0 <p>En caso exitoso.</p> <ul style="list-style-type: none"> • Otro valor <p>En caso de error.</p>
SemcoTC90527RegReset()	<p>Descripción: Esta función reiniciar los registros del demodulador.</p> <p>Parámetros: Ninguno.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • 0 <p>En caso exitoso.</p> <ul style="list-style-type: none"> • Otro valor <p>En caso de error.</p>
SemcoTC90527SleepOn()	<p>Descripción: Esta función coloca al demodulador en modo suspensión.</p> <p>Parámetros: Ninguno.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • 0 <p>En caso exitoso.</p> <ul style="list-style-type: none"> • Otro valor <p>En caso de error.</p>

SemcoTC90527SleepOff()	<p>Descripción: Esta función saca del modo de suspensión al demodulador.</p> <p>Parámetros: Ninguno.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • 0 <p>En caso exitoso.</p> <ul style="list-style-type: none"> • Otro valor <p>En caso de error.</p>
SemcoTC90527MasterLock()	<p>Descripción: Esta función retorna el estado de bloqueo del demodulador.</p> <p>Parámetros: Ninguno.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • 1 <p>Dispositivo bloqueado.</p> <ul style="list-style-type: none"> • 0 <p>Dispositivo desbloqueado.</p>
SemcoTC90527_ISDBT_SNRResultCheck()	<p>Descripción: Esta función retorna la relación señal/ruido.</p> <p>Parámetros: Ninguno.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • Punto flotante <p>Relación señal/ruido.</p>
SemcoTC90527_GetSignalQuality()	<p>Descripción: Esta función retorna escala de calidad de la señal.</p>

	<p>Parámetros:</p> <ul style="list-style-type: none"> • Estructura <p>Esta estructura aloja los siguientes valores:</p> <p>bLock, estado de bloqueo.</p> <p>SNR, relación señal/ruido.</p> <p>pre_BER, (Bit Error Rate) tasa de error binario, de la señal entrante.</p> <p>post_BER, tasa de error binario, de la señal corregida.</p> <p>mode_90527, modo de demodulación 0:DQPSK, 1:QPSK, 2:16QAM, 3:64QAM.</p> <p>code_rate, 0:1/2, 1:2/3, 2:3/4, 3:5/6, 4:7/8.</p> <p>hab_error, cantidad errores detectados en la señal corregida.</p>
SemcoTC90527_Set_TS_Output()	<p>Retorna:</p> <ul style="list-style-type: none"> • entero <p>Con el valor de 1, 2, 3, 4, 5, 6, 7, 8, 9 o 10. Mayor escala mejor calidad de la señal.</p> <p>Descripción: Esta función determinar el tipo la salida de los paquetes BTS;</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • Entero <p>El valor 0 designa serial, otro valor diferente designa paralelo.</p>

	<p>Retorna:</p> <ul style="list-style-type: none"> • 0 <p>En caso exitoso.</p> <ul style="list-style-type: none"> • Otro valor <p>En caso de error.</p>
SemcoTC90527_GetDatas()	<p>Descripción: Esta función retorna el estado de bloqueo, la relación señal/ruido, tasa de error binario, tipo de demodulación y errores a través del llenado de una estructura.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • Estructura <p>Esta estructura aloja los siguientes valores:</p> <p>bLock, estado de bloqueo.</p> <p>SNR, relación señal/ruido.</p> <p>pre_BER, (Bit Error Rate) tasa de error binario, de la señal entrante.</p> <p>post_BER, tasa de error binario, de la señal corregida.</p> <p>mode_90527, modo de demodulacion 0:DQPSK, 1:QPSK, 2:16QAM, 3:64QAM.</p> <p>code_rate, 0:1/2, 1:2/3, 2:3/4, 3:5/6, 4:7/8.</p> <p>hab_error, cantidad errores detectados en la señal corregida.</p>

	<p>Retorna:</p> <ul style="list-style-type: none"> • 0 <p>En caso exitoso.</p> <ul style="list-style-type: none"> • Otro valor <p>En caso de error.</p>
SemcoTC90527_GetLfAgcLevel()	<p>Descripción: Esta función retorna el nivel de la intensidad de la señal.</p> <p>Parámetros: Ninguno.</p>
	<p>Retorna:</p> <ul style="list-style-type: none"> • Punto flotante <p>Intensidad de la señal.</p>
SemcoTC90527_GetSSI()	<p>Descripción: Esta función retorna la escala de la intensidad de señal.</p> <p>Parámetros: Ninguno.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • entero <p>Con el valor de 1, 2, 3, 4, 5, 6, 7, 8, 9 o 10. Mayor escala mayor intensidad de la señal.</p>

7.1.2 Funciones del sintonizador SemcoSTV4100.

En la **Tabla 7.1.2** se muestran el nombre de las funciones pertenecientes a la API Samsung para el manejo del sintonizador y su descripción.

Tabla 7.1.2: Funciones de la API Samsung, sintonizador. [10]	
SemcoSTV4100_Initialize()	<p>Descripción: Esta función inicializa el sintonizador.</p> <p>Parámetros: Ninguno.</p> <p>Retorna:</p> <ul style="list-style-type: none">• 0 <p>En caso exitoso.</p> <ul style="list-style-type: none">• Otro valor <p>En caso de error.</p>
SemcoSTV4100_SetFrequency()	<p>Descripción: Esta función sintoniza una frecuencia deseada.</p> <p>Parámetros:</p> <ul style="list-style-type: none">• entero <p>Indica el valor de la frecuencia en Hz a sintonizar.</p> <ul style="list-style-type: none">• entero <p>Indica el ancho de bando, por defecto el de Venezuela 6 MHz.</p> <p>Retorna:</p> <ul style="list-style-type: none">• 0 <p>En caso exitoso.</p> <ul style="list-style-type: none">• Otro valor

	En caso de error.
SemcoSTV4100_GetLockStatus()	<p>Descripción: Esta función retorna el estado de bloqueo del sintonizador.</p> <p>Parámetros: Ninguno.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • 1 Dispositivo bloqueado. • 0 Dispositivo desbloqueado.
SemcoSTV4100_On()	<p>Descripción: Esta función coloca al sintonizador en modo de suspensión.</p> <p>Parámetros: Ninguno.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • 0 En caso exitoso. • Otro valor En caso de error.
SemcoSTV4100_StandBy()	<p>Descripción: Esta función saca del modo de suspensión al sintonizador.</p> <p>Parámetros: Ninguno.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • 0 En caso exitoso. • Otro valor En caso de error.

7.1.3 Funciones definidas por el desarrollador

La API no posee funciones para la comunicación USB, estas deben ser creadas por el desarrollador al igual que la función de retardo, estas se especifican en la **Tabla 7.1.3**.

Las funciones de comunicación USB fueron desarrollados con la API Libusb y la función de retardo utiliza una función de la biblioteca estándar del lenguaje C++.

Tabla 7.1.3: Funciones API Samsung definidos por el desarrollador. [10]

Funciones	Descripción
STV4100_I2C_Write()	Se encarga de realizar la transferencia de control en modo de escritura del sintonizador.
STV4100_I2C_Read()	Se encarga de realizar la transferencia de control en modo de lectura del sintonizador.
TC90527_I2cWrite()	Se encarga de realizar la transferencia de control en modo de escritura del demodulador.
TC90527_I2cRead()	Se encarga de realizar la transferencia de control en modo de lectura del demodulador.
procedimiento_retardo_milisegundos()	Realiza retardos en milisegundos.

7.1.4 Procesos de inicialización y sintonización

Para el procedimiento de inicialización del demodulador y sintonizador se realiza el siguiente llamado de funciones de la API como se muestra en la **Figura 7.1.1**.

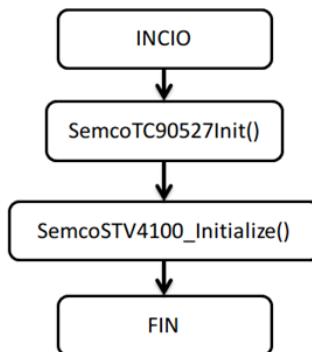


Figura 7.1.1: Procedimiento de inicialización del demodulador y sintonizador.

El procedimiento de sintonización se describe a continuación en la **Figura 7.1.2.**

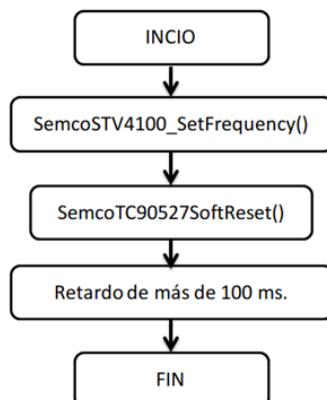


Figura 7.1.2: Procedimiento de sintonización de una frecuencia.

En la **Tabla 7.1.4** se describen las funciones desarrolladas que tiene como base las funciones de la API de Samsung y de la librería estándar de C++.

Tabla 7.1.4: Funciones desarrolladas en base a la API de Samsung y la librería estándar de C++.

procedimiento_retardo_milisegundo_s()	<p>Descripción: Esta función es la encargada de realizar retardos en milisegundos.</p> <p>Variables: Ninguno.</p>
	<p>Funciones de la librería estándar de C++:</p> <ul style="list-style-type: none"> • usleep() <p>tiempo_ms, parámetro entero que representa los milisegundos de retardo.</p>
procedimiento_inicializar_demodulador_sintonizador()	<p>Descripción: Esta función se encarga de inicializar tanto al demodulador como al sintonizador.</p> <p>Variables: Ninguno.</p>
	<p>Funciones de la API Samsung:</p> <ul style="list-style-type: none"> • SemcoTC90527Init() • SemcoSTV4100_Initialize() <p>Funciones de la librería estándar de C++:</p> <ul style="list-style-type: none"> • procedimiento_retardo_milisegundos() <p>tiempo_ms, parámetro entero que representa los milisegundos de retardo.</p>
procedimiento_sintonizar_frecuencia()	<p>Descripción: Esta función realiza la función de sintonizar una frecuencia deseada en el sintonizador. Recibe un entero como parámetro con la frecuencia a sintonizar en Hz.</p> <p>Variables: Ninguno.</p>

	<p>Funciones de la API Samsung:</p> <ul style="list-style-type: none"> • SemcoSTV4100_SetFrequency() <p>nFrequency_KHz, parámetro entero indica el valor de la frecuencia en Hz a sintonizar.</p> <p>BW, parámetro entero indica el ancho de bando, por defecto el de Venezuela 6 MHz.</p> <p>Funciones de la librería estándar de C++:</p> <ul style="list-style-type: none"> • procedimiento_retardo_milisegundos() <p>tiempo_ms, parámetro que representa los milisegundos de retardo.</p>
procedimiento_programa_521000_5 7856() procedimiento_programa_521000_5 7857() procedimiento_programa_521000_5 7858() procedimiento_programa_521000_5 7859() procedimiento_programa_521000_5 7860() procedimiento_programa_521000_5 7880() procedimiento_programa_533000_5 7920() procedimiento_programa_533000_5 7921() procedimiento_programa_533000_5 7922()	<p>Descripción: Este grupo de funciones representan a cada uno de los programas de la TDA venezolana las funciones tienen el nombre de la forma procedimiento_programa_NUMEROx_NUMEROy().</p> <p>NUMEROx representa la frecuencia en Hertz del grupo de programas y NUMEROy representa el ID de un programa en específico.</p> <p>Estas funciones son las encargadas de sintonizar una frecuencia específica, seleccionar el ID del programa a sintonizar y de establecer en alto una bandera que señala que una frecuencia a sido establecida.</p>

<pre> procedimiento_programa_533000_5 7944() procedimiento_programa_539000_5 7952() procedimiento_programa_539000_5 7953() procedimiento_programa_539000_5 7954() procedimiento_programa_539000_5 7955() procedimiento_programa_539000_5 7956(); procedimiento_programa_539000_5 7976() procedimiento_programa_527000_5 7888() procedimiento_programa_527000_5 7889() procedimiento_programa_527000_5 7890() procedimiento_programa_527000_5 7891() procedimiento_programa_527000_5 7892() procedimiento_programa_527000_5 7912() </pre>	<p>Variables:</p> <ul style="list-style-type: none"> • id <p>Esta variable aloja una cadena de caracteres que especifica el ID del programa.</p> <ul style="list-style-type: none"> • estado_frecuencia <p>Esta variable de tipo boolena toma valor de verdadero cuando una frecuencia fue establecida y toma el valor de falso cuando una frecuencia nunca ha sido establecida.</p>
	<p>Funciones de la API Samsung:</p> <ul style="list-style-type: none"> • SemcoTC90527Init() • SemcoSTV4100_Initialize() • SemcoSTV4100_SetFrequency() <p>nFrequency_KHz, parámetro entero indica el valor de la frecuencia en Hz a sintonizar.</p> <p>BW, parámetro entero indica el ancho de banda, por defecto el de Venezuela 6 MHz.</p> <p>Funciones de la librería estándar de C++:</p> <ul style="list-style-type: none"> • procedimiento_retardo_milisegundos() <p>tiempo_ms, parámetro que representa los milisegundos de retardo.</p>

funcion_leer_id()	<p>Descripción: Esta función retorna la cadena de caracteres que representa un ID de programa.</p> <p>Variables:</p> <ul style="list-style-type: none"> • id <p>Esta variable aloja una cadena de caracteres que especifica el ID del programa.</p>
-------------------	--

8.1 GNU/Linux y el Dongel USB de TDA

Al momento de la inserción del dispositivo TDA en el puerto USB del anfitrión o PC, se realiza el proceso de enumeración en el cual se le asigna una dirección al dispositivo y se obtiene la descripción de este dispositivo (configuraciones, interfaces, endpoints, etc). El núcleo de Linux proporciona comandos que nos brindan información sobre los dispositivos USB.

El comando **lsusb** ejecutado en una terminal nos proporciona una lista de los dispositivos USB conectados al anfitrión, en este caso la PC, en la **Figura 8.1.1** se muestra la salida arrojada por el comando.

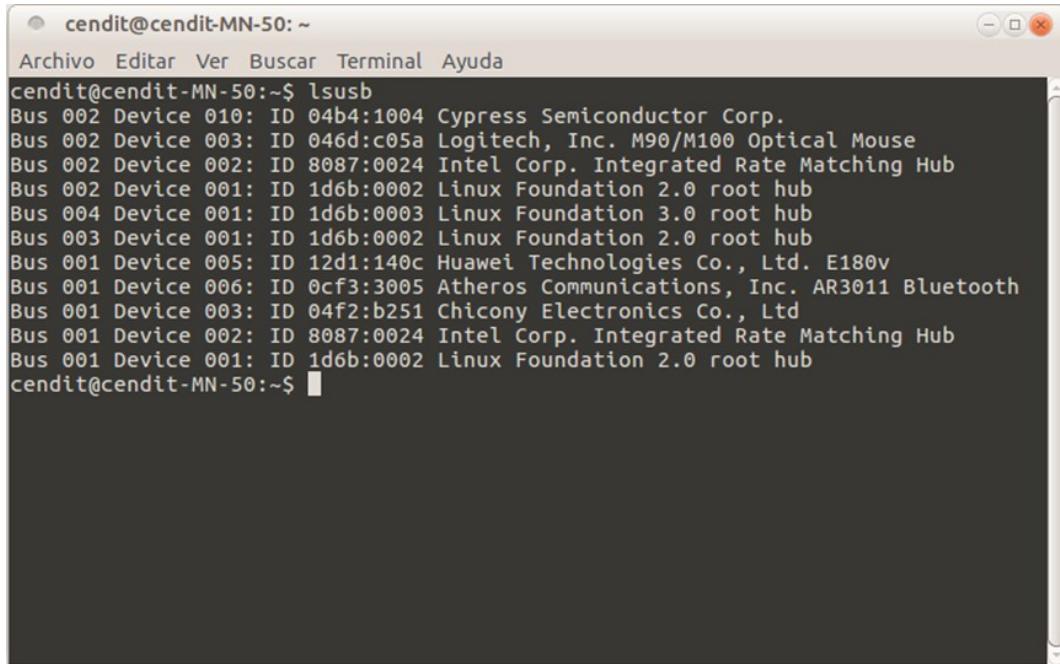
A screenshot of a terminal window titled "cendir@cenedit-MN-50: ~". The window contains the command "lsusb" followed by its output. The output lists various USB devices connected to the system, including a Cypress Semiconductor Corp. device (VendorID 0x4b4, ProductID 0x1004) and several Intel Corp. Integrated Rate Matching Hubs. The terminal window has a standard Linux-style interface with a title bar and a scroll bar.

Figura 8.1.1: Salida de la ejecución del comando **lsusb**.

Como se observa en la **Figura 8.1.1** el dispositivo USB de TDA se identifica como Cypress Semiconductor Corp. Con el VendorID: (0x4b4)h y con el ProductID: (0x1004)h, utiliza el BUS 02 y su número de dispositivo es 010, el número de dispositivo es asignado por el anfitrión.

Para obtener el descriptor del dispositivo USB Cypress. Se utilizó el comando **lsusb -v**, el cual brinda información sobre los descriptores de los dispositivos conectados al equipo. La **Figura 8.1.2** muestra la información obtenida por el sistema operativo del descriptor del dispositivo Cypress, se observa que el dispositivo posee una sola configuración que contiene una sola interfaz, un solo endpoint especificado, ninguna clase en especial asignada, los ID Vendor y Product del dispositivo, su fabricante es señalado como Cypress, dentro de la configuración se observa que la alimentación del dispositivo está definida para que el dispositivo obtenga su alimentación

a través del puerto físico del anfitrión, en la única interfaz identificada como interfaz0 se encuentra especificado el endpoint (0x82)h como endpoint de tipo IN, entrada hacia el anfitrión, tipo de transferencia Interrupción, tipo de sincronización ninguna, el tipo del endpoint se especifica para datos, número máximo de paquetes 1 x 1024 bytes, cada paquete en un microframe, cada 125 micro segundos. El endpoint (0x82)h es el buffer en el dispositivo USB encargado de alojar la información de los paquetes BTS de 1024 bytes de longitud que serán enviados al anfitrión cada 125 micro segundos.

Esta información es de utilidad a la hora de programar un controlador en espacio de usuario usando la API Libusb para definir los valores y parámetros de las funciones que realizan las tareas de control y transferencias de datos ya que estos datos otorgados a través de los comandos **Isusb** y **Isusb-v** definen el comportamiento interno del dispositivo USB de TDA y el modo de interacción con el anfitrión al momento de comunicarse.

```
cendit@cendit-MN-50: ~
cendit@cendit-MN-50:~$ lsusb -v

Bus 002 Device 017: ID 04b4:1004 Cypress Semiconductor Corp.
Device Descriptor:
  bLength          18
  bDescriptorType   1
  bcdUSB         2.00
  bDeviceClass      0 (Defined at Interface level)
  bDeviceSubClass    0
  bDeviceProtocol    0
  bMaxPacketSize0     64
  idVendor        0x04b4 Cypress Semiconductor Corp.
  idProduct        0x1004
  bcdDevice        0.00
  iManufacturer      1 Cypress
  iProduct          2 DMB-TV
  iSerial           0
  bNumConfigurations  1
Configuration Descriptor:
  bLength          9
  bDescriptorType   2
  wTotalLength     25
  bNumInterfaces     1
  bConfigurationValue  1
  iConfiguration      0
  bmAttributes       0x80
    (Bus Powered)
  MaxPower        100mA
Interface Descriptor:
  bLength          9
  bDescriptorType   4
  bInterfaceNumber    0
  bAlternateSetting   0
  bNumEndpoints      1
  bInterfaceClass     255 Vendor Specific Class
  bInterfaceSubClass  0
  bInterfaceProtocol  0
  iInterface          0
Endpoint Descriptor:
  bLength          7
  bDescriptorType   5
  bEndpointAddress  0x82 EP 2 IN
  bmAttributes       3
    Transfer Type      Interrupt
    Sync Type          None
    Usage Type          Data
  wMaxPacketSize     0x0400 1x 1024 bytes
  bInterval          1
Device Qualifier (for other device speed):
  bLength          10
  bDescriptorType   6
  bcdUSB         2.00
  bDeviceClass      0 (Defined at Interface level)
  bDeviceSubClass    0
  bDeviceProtocol    0
  bMaxPacketSize0     64
  bNumConfigurations  1
Device Status: 0x0000
  (Bus Powered)
```

Figura 8.1.2: Salida de la ejecución del comando `lusb-v`.

9.1 La API Libusb

Para el desarrollo del controlador USB, se decidió usar un controlador en espacio de usuario, usando la librería Libusb, que es una API que comunica los procesos USB creados en el espacio de usuario con la interfaz de llamadas al sistema y estas a su vez se comunican con el núcleo GNU/Linux que maneja el hardware de los procesos USB.

Libusb es una biblioteca escrita en lenguaje C que proporciona acceso genérico a dispositivos USB. Está destinada a ser utilizado por los desarrolladores para facilitar la producción de aplicaciones que se comunican con hardware USB.

Es portátil, utilizando una única API multiplataforma, proporciona acceso a dispositivos USB en Linux, OS X, Windows, Android, OpenBSD, etc.

A continuación el **Tabla 9.1.1** se listan las funciones usadas de la librería conjuntamente con una descripción.

Tabla 9.1.1: Funciones de la API Libusb. [11]

libusb_init()	<p>Descripción: Crea una sesión de la librería.</p> <p>Parámetros:</p> <ul style="list-style-type: none">• libusb_context <p>Representa una sesión de la librería.</p> <p>Retorna:</p> <ul style="list-style-type: none">• 0 <p>En caso de éxito o un código</p> <ul style="list-style-type: none">• LIBUSB_ERROR <p>En caso de error.</p>
---------------	---

libusb_exit()	<p>Descripción: Cierra una sesión de la librería.</p>
	<p>Parámetros:</p> <ul style="list-style-type: none"> • libusb_context <p>Representa una sesión de la librería.</p>
libusb_set_debug()	<p>Descripción: Establece el nivel de depuración.</p>
	<p>Parámetros:</p> <ul style="list-style-type: none"> • libusb_context <p>Representa una sesión de la librería.</p> <ul style="list-style-type: none"> • nivelDeDebug <p>Variable de tipo entero que define el nivel de depuración de la librería.</p> <p>LIBUSB_LOG_LEVEL_NONE (nivelDeDebug=0): ningún mensaje impreso por la biblioteca (predeterminado).</p> <p>LIBUSB_LOG_LEVEL_ERROR (nivelDeDebug =1): los mensajes de error se imprimen en stderr (Standard error stream).</p> <p>LIBUSB_LOG_LEVEL_WARNING (nivelDeDebug =2): mensajes de advertencia y de error se imprimen en stderr.</p> <p>LIBUSB_LOG_LEVEL_INFO (nivelDeDebug =3): los mensajes de información se imprimen en stdout (Standard output), los mensajes de advertencia y de error se imprimen en stderr.</p>

	<p>LIBUSB_LOG_LEVEL_DEBUG (nivelDeDebug =4): se imprimen mensajes de depuración e información a stdout, advertencias y errores a stderr.</p> <p>Retorna: Nada.</p>
libusb_open()	<p>Descripción: Establece un manejador a nivel de software de un dispositivo USB detectado por el sistema.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libusb_device <p>Es una variable que representa a un dispositivo USB detectado por el sistema.</p> <ul style="list-style-type: none"> • libusb_device_handle <p>Es una variable que permite el manejo del dispositivo a nivel de software.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • 0 <p>En caso de éxito.</p> <ul style="list-style-type: none"> • LIBUSB_ERROR_NO_MEM <p>En el fallo de asignación de memoria.</p> <ul style="list-style-type: none"> • LIBUSB_ERROR_ACCESS <p>Si el usuario tiene permisos insuficientes.</p> <ul style="list-style-type: none"> • LIBUSB_ERROR_NO_DEVICE

	<p>Si el dispositivo ha sido desconectado.</p> <ul style="list-style-type: none"> • LIBUSB_ERROR <p>En otro error.</p>
libusb_close()	<p>Descripción: Cierra un manejador a nivel de software de un dispositivo USB.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libusb_device_handle <p>Es una variable que permite el manejo del dispositivo a nivel de software.</p> <p>Retorna: Nada.</p>
libusb_hotplug_register_callback ()	<p>Descripción: Función que proporciona una solución a la conexión en caliente (hotplug) para dispositivos USB. Esta función filtra los dispositivos USB para detectar cuando un dispositivo es despachado de el sistema .</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libusb_context <p>Representa una sesión de la librería.</p> <ul style="list-style-type: none"> • libusb_hotplug_event <p>Selecciona que eventos filtrar para un dispositivo USB en el ámbito de conexión en caliente, LIBUSB maneja dos eventos:</p> <p>LIBUSB_HOTPLUG_EVENT_DEVICE_ARRIVED: Un dispositivo ha</p>

	<p>arribado y está listo para usar.</p> <p>LIBUSB_HOTPLUG_EVENT_DEVICE_LEFT: Un dispositivo se ha despachado y no se encuentra disponible.</p> <ul style="list-style-type: none"> • <code>libusb_hotplug_flag</code> <p>Son valores enteros que se usan de bandera para armar la función de llamada:</p> <p>LIBUSB_HOTPLUG_NO_FLAGS: valor por defecto cuando no se utiliza ninguna bandera.</p> <p>LIBUSB_HOTPLUG_ENUMERATE: Arma a la función libusb_hotplug_callback_fn para que se dispare si algún dispositivo USB es detectado con las características especificadas.</p> <ul style="list-style-type: none"> • <code>vendor_id</code> <p>Valor entero para filtrar dispositivos por Vendor ID.</p> <ul style="list-style-type: none"> • <code>product_id</code> <p>Valor entero para filtrar dispositivos por Product ID.</p> <ul style="list-style-type: none"> • <code>dev_class</code> <p>Indica la clase de dispositivo USB a detectar.</p> <ul style="list-style-type: none"> • <code>libusb_hotplug_callback_fn</code> <p>Es una apuntador a una función de llamada que es invocado cuando</p>
--	---

	<p>una coincidencia del dispositivo USB ocurre sea en un evento de arribo o despacho.</p> <ul style="list-style-type: none"> • user_data <p>Es un apuntador a cualquier dato que se desea pasar a la función.</p> <ul style="list-style-type: none"> • libusb_hotplug_callback_handle • <p>Variable que permite manejar el dispositivo USB.</p>
	<p>Retorna:</p> <ul style="list-style-type: none"> • LIBUSB_SUCCESS <p>En caso de éxito o un código</p> <ul style="list-style-type: none"> • LIBUSB_ERROR <p>Para denotar un error.</p>
libusb_hotplug_callback_fn()	<p>Descripción: Es la función de llamada que se ejecuta cuando el dispositivo USB específico es detectado.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libusb_context <p>Comparte la misma información de parámetro con la función libusb_hotplug_register_callback</p> <ul style="list-style-type: none"> • libusb_device <p>Es una variable que representa a un dispositivo USB detectado por el sistema. En este caso el dispositivo detectado.</p>

	<ul style="list-style-type: none"> • libusb_hotplug_event <p>Comparte la misma información de parámetro con la función libusb_hotplug_register_callback</p> <ul style="list-style-type: none"> • user_data <p>Comparte la misma información de parámetro con la función libusb_hotplug_register_callback</p> <p>Retorna: Booleano si esta devolución de llamada ha terminado de procesar eventos. Devolver 1 hará que esta devolución de llamada se cancele.</p>
libusb_control_transfer()	<p>Descripción: Esa función permite realizar y ejecutar transferencia de control de entrada y salida desde el anfitrión hacia el dispositivo USB.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libusb_device_handle <p>Variable LIBUSB para manejar el dispositivo USB.</p> <ul style="list-style-type: none"> • bmRequestType <p>El campo tipo de solicitud para el paquete de configuración de una transferencia de control.</p> <ul style="list-style-type: none"> • bRequest <p>El campo de solicitud para el paquete de configuración de una transferencia de control.</p> <ul style="list-style-type: none"> • wValue <p>El campo de valor para el paquete</p>

	<p>de configuración de una transferencia de control.</p> <ul style="list-style-type: none"> • wIndex <p>El campo de dirección para el paquete de configuración de una transferencia de control.</p> <ul style="list-style-type: none"> • data <p>Apuntador al buffer de datos.</p> <ul style="list-style-type: none"> • wLength <p>Indica la longitud del buffer de datos.</p> <ul style="list-style-type: none"> • timeout <p>Tiempo límite para recibir una respuesta en milisegundos.</p>
Retorna:	<ul style="list-style-type: none"> • En caso de éxito <p>El número de bytes transferidos.</p> <ul style="list-style-type: none"> • LIBUSB_ERROR_TIMEOUT <p>Si la transferencia se ha agotado.</p> <ul style="list-style-type: none"> • LIBUSB_ERROR_PIPE <p>Si la solicitud de control no era compatible con el dispositivo.</p>
libusb_alloc_transfer()	<p>Descripción: Esta función asigna espacio para una transferencia USB de tipo masivo, interrupción o isócrona.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • iso_packets

	<p>Es una variable de tipo entero que representa el número de paquetes isócronos.</p> <p>Retorna:</p> <p>libusb_transfer, es una variable de tipo estructura que contiene los siguientes campos y representa a una transferencia USB:</p> <ul style="list-style-type: none"> • libusb_device_handle <p>Variable Libusb para manejar el dispositivo USB.</p> <ul style="list-style-type: none"> • Flags <p>Son valores enteros que se usan para denotar las acciones a realizar dependiendo del valor de la bandera:</p> <p>LIBUSB_TRANSFER_SHORT_NO_T_OK (0), reporta transferencias incompletas cortas como errores.</p> <p>LIBUSB_TRANSFER_FREE_BUFFER(1), libera automáticamente el buffer de transferencia durante el llamado de la función libusb_free_transfer().</p> <p>LIBUSB_TRANSFER_FREE_TRANSFER (2), automáticamente llama a la función libusb_free_transfer() después que la función de llamada retorna.</p> <p>LIBUSB_TRANSFER_ADD_ZERO_PACKET (3), finaliza las transferencias del endpoint con un paquete de longitud cero adicional.</p>
--	--

	<ul style="list-style-type: none"> • endpoint <p>Dirección del endpoint en el que se enviará esta transferencia.</p> <ul style="list-style-type: none"> • Type <p>Tipo de endpoint, interrupción, masivo o isócrono.</p> <ul style="list-style-type: none"> • Timeout <p>tiempo límite para esta transferencia en milisegundos.</p> <ul style="list-style-type: none"> • libusb_transfer_status <p>son valores enteros que se usan para denotar los estados de las transferencias:</p> <p>LIBUSB_TRANSFER_COMPLETE D (0): La transferencia se completó sin errores.</p> <p>LIBUSB_TRANSFER_ERROR (1): La transferencia falló.</p> <p>LIBUSB_TRANSFER_TIMED_OUT (2): Tiempo de realización de transferencia finalizado.</p> <p>LIBUSB_TRANSFER_CANCELLED (3): La transferencia fue cancelada.</p> <p>LIBUSB_TRANSFER_STALL (4): Condición de halt (condición de parada) detectada en el endpoint.</p> <p>LIBUSB_TRANSFER_NO_DEVICE (5): El dispositivo USB fue desconectado.</p> <p>LIBUSB_TRANSFER_OVERFLOW</p>
--	--

	<p>(6): El dispositivo envió más datos que los solicitados.</p> <ul style="list-style-type: none"> • length <p>Longitud del buffer.</p> <ul style="list-style-type: none"> • actual_length <p>Longitud de los datos que fueron transferidos.</p> <ul style="list-style-type: none"> • libusb_transfer_cb_fn <p>Apuntador a la función de llamada que se ejecuta cuando finaliza una transferencia de datos.</p> <ul style="list-style-type: none"> • user_data <p>Apuntador de datos pasado a la función de llamada.</p> <ul style="list-style-type: none"> • Buffer <p>apuntador al buffer de datos.</p> <ul style="list-style-type: none"> • num_iso_packets <p>Número de paquetes isócronos en caso de transferencias isócronas.</p> <ul style="list-style-type: none"> • libusb_iso_packet_descriptor <p>Estructura para manejar datos isócronos.</p>
libusb_fill_interrupt_transfer()	<p>Descripción: Llena la instancia libusb_transfer con información sobre la transferencia que desea realizar.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libusb_device_handle

	<p>Variable Libusb para manejar el dispositivo USB.</p> <ul style="list-style-type: none"> • endpoint <p>Dirección del endpoint en el que se enviará esta transferencia.</p> <ul style="list-style-type: none"> • buffer <p>Apuntador al buffer de datos.</p> <ul style="list-style-type: none"> • length <p>Longitud del buffer.</p> <ul style="list-style-type: none"> • callback <p>Apuntador a la función de llamada que se ejecuta cuando finaliza una transferencia de datos.</p> <ul style="list-style-type: none"> • user_data <p>Apuntador de datos pasado a la función de llamada.</p> <ul style="list-style-type: none"> • timeout <p>Tiempo límite para esta transferencia en mili segundos.</p>
	<p>Retorna: Nada</p>
libusb_submit_transfer()	<p>Descripción: Esta función ejecuta una transferencia USB de tipo masivo, interrupción o isócrona.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libusb_transfer <p>Representa una transferencia USB.</p> <p>Retorna:</p>

	<ul style="list-style-type: none"> • 0 Si fue exitosa la ejecución. • LIBUSB_ERROR_NO_DEVICE Si el dispositivo ha sido desconectado. • LIBUSB_ERROR_BUSY Si la transferencia ya ha sido enviada. • LIBUSB_ERROR_NOT_SUPPORTED Si los indicadores de transferencia no son compatibles con el sistema operativo. • LIBUSB_ERROR Para denotar otro error.
libusb_free_transfer()	<p>Descripción: Esta función permite desasignar las transferencias USB.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libusb_transfer Representa una transferencia USB. <p>Retorna: Nada</p>
libusb_handle_events()	<p>Descripción: Esta función permite manejar los eventos ocurridos en una transferencia USB.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libusb_context

	Representa una sesión de la librería.
	Retorna:
	<ul style="list-style-type: none"> • 0 <p>En caso de éxito o un código</p> <ul style="list-style-type: none"> • LIBUSB_ERROR
	En caso de error.

9.1.1 Transferencias asincrónicas

Para las transferencias de tipo interrupción que reciben la información de audio y video en el anfitrión se utilizó la interfaz asincrónica para transferencias de interrupción de Libusb, una interfaz asincrónica se refiere a la ocurrencia de eventos de forma independiente de la corriente principal del programa. El proceso para iniciar y realizar una transferencia de interrupción asincrónica contiene los siguientes pasos:

1. **Asignación:** consiste en asignar una variable de tipo **libusb_transfer** para la transferencia, este paso implica asignar memoria para una transferencia USB. En esta etapa, la transferencia está "en blanco" sin detalles sobre qué tipo de Entrada/ Salida se utilizará para la transferencia.

La asignación se realiza con la función **libusb_alloc_transfer(iso_packets)**, se debe utilizar esta función en lugar de asignar las transferencias de manera propia, la función retorna una variable **libusb_transfer** y recibe una variable denominada **iso_packets** de tipo entero con el número de paquetes isócronos, para transferencia de tipo interrupción y masivas la variable debe tener un valor de 0.

2. **Llenado:** Llena la instancia **libusb_transfer** con información sobre la transferencia que desea realizar, este paso es donde se toma una transferencia previamente asignada y se completa con información para determinar el tipo y la dirección del mensaje, el buffer de datos, la función de devolución de llamada, etc.
Puede llenar los campos requeridos utilizando las funciones de ayuda proporcionadas por la biblioteca debido a que la transferencia de contenido multimedia es de tipo interrupción se utilizó la función **libusb_fill_interrupt_transfer()**.
3. **Tratamiento de la terminación:** se examinan los resultados de la transferencia en la estructura:
libusb_transfer->libusb_transfer_status.
4. **Desasignación:** limpiar los recursos, cuando se ha realizado una transferencia (es decir, se ha invocado la función de devolución de llamada), se aconseja liberar la transferencia (a menos que desee volver a enviarla). Las transferencias se desasignan a través de la función **libusb_free_transfer(libusb_transfer)**. [11]

9.1.2 Rutinas USB

A continuación se especifican las rutinas USB creadas que permiten la conexión en caliente del dispositivo el manejo de los buses de entrada y salida y el control del dispositivo.

9.1.2.1 Rutinas de Conexión en Caliente

En la **Tabla 9.1.2** se describen las funciones desarrolladas para la conexión en caliente para el dispositivo USB de TDA y las funciones de la API Libusb que son llamados junto con los parámetros utilizados.

Tabla 9.1.2: Funciones desarrolladas en base de la API Libusb para proporcionar el soporte de conexión en caliente.

procedimiento_hotplug()	<p>Descripción: Esta función se ejecuta en un hilo secundario del programa, es ejecutado al inicio del programa y se mantiene en ejecución dentro de un ciclo infinito hasta que el programa finalice, se encarga de iniciar una sesión de la librería Libusb, libusb_init(), y de finalizarla, libusb_exit(), también maneja el filtrado y registro el dispositivo USB si este arriba o se despacha del puerto USB, libusb_hotplug_register_callback().</p>
	<p>Variables:</p> <ul style="list-style-type: none"> • <code>estado_hotplug</code> <p>Para manejar los estados de conexión en caliente se definió una variable de tipo entero llamado estado_hotplug la cual puede tomar los siguientes valores:</p> <p>Si <code>estado_hotplug</code> igual 0, dispositivo no conectado.</p> <p>Si <code>estado_hotplug</code> igual 1, dispositivo conectado.</p> <p>Si <code>estado_hotplug</code> igual 2, dispositivo desconectado.</p>
	<p>Funciones Libusb:</p> <ul style="list-style-type: none"> • <code>libusb_init()</code> <p>contexto_usb, para indicar la sesión de Libusb usada.</p>

	<ul style="list-style-type: none"> • libusb_exit() <p>contexto_usb, para indicar la sesión de Libusb usada.</p> <ul style="list-style-type: none"> • libusb_hotplug_register_callback() <p>contexto_usb, para indicar la sesión de Libusb usada.</p> <p>(libusb_hotplug_event), banderas que indican los eventos LIBUSB a tomar en cuenta, en este caso arribo y despacho del dispositivo USB.</p> <p>LIBUSB_HOTPLUG_ENUMERATE,arma la función de llamada</p> <p>libusb_hotplug_callback_fn() para que se dispare si algún dispositivo USB es detectado con las características especificadas.</p> <p>1204, IDVendor del dispositivo a detectar, en este caso el del dispositivo USB de Cypress.</p> <p>4100, IDProduct del dispositivo a detectar, en este caso el del dispositivo USB de Cypress.</p> <p>LIBUSB_HOTPLUG_MATCH_ANY, bandera que indica cualquier clase de dispositivo USB a detectar.</p> <p>funcion_de_llamada_hotplug, apuntador a la función de llamada que se ejecuta cuando el dispositivo USB especificado es detectado.</p> <p>NULL, ningún dato es pasado a la función de llamada.</p>
--	---

	manejador_dispositivo_usb_hotplug , variable Libusb para el manejo del dispositivo USB.
funcion_de_llamada_hotplug()	<p>Descripción: Función de llamada que se ejecuta luego de que un dispositivo coincida con las características del dispositivo especificado para la conexión en caliente, se encarga de abrir el dispositivo usando la función libusb_open() y de cerrarlo usando la función libusb_close() y cambiar el valor del variable estado_hotplug.</p> <p>Variables:</p> <ul style="list-style-type: none"> • manejador_dispositivo_usb <p>Esta función es la encargada de crear o eliminar una variable global que define un manejador a nivel de software de un dispositivo USB.</p> <p>Funciones Libusb:</p> <ul style="list-style-type: none"> • libusb_open() <p>dispositivo_usb_hotplug_funcion_llamada, Variable Libusb para manejar el dispositivo USB.</p> <p>manejador_dispositivo_usb, Es una variable que representa a un dispositivo USB detectado por el sistema.</p> <ul style="list-style-type: none"> • libusb_close() <p>manejador_dispositivo_usb, Variable Libusb para manejar el dispositivo USB.</p>

funcion_leer_estado_hotplug()	<p>Descripción: Esta función retorna el valor de la variable estado_hotplug. Para que otras funciones tengan acceso al estado de conexión en caliente del dispositivo USB.</p> <p>Variables:</p> <ul style="list-style-type: none"> • <code>estado_hotplug</code> <p>Esta variable representa el estado de conexión en caliente de un dispositivo USB.</p>
	<p>Funciones Libusb: Ninguna.</p>

9.1.2.1 Rutinas de Transferencias de Control

Estas rutinas son las que especifica la API de Samsung que deben ser creadas por el desarrollador, las cuales se encargan de realizar las transferencias de control USB. Los datos recibidos o enviados a través de estas transferencias de control luego viajan a través del protocolo I2C desde o hacia el chip de USB Cypress, que luego se comunica con el demodulador o el sintonizador del dispositivo Samsung Tuner NIM DNOD22QXV104A lo que permite realizar acciones de control en el dispositivo.

Las funciones de transferencias de control son cuatro, dos pertenecientes al sintonizador y los dos restantes al demodulador, cada dispositivo tienen una función de lectura y uno de escritura desde el anfitrión estas están basadas en la función **libusb_control_transfer()** de la API LIBUSB, las transferencias de control se dirigen en el caso del dispositivo de USB de TDA al endpoint 0. En la **Tabla 9.1.3** se describen las funciones de control USB pertenecientes al demodulador.

Tabla 9.1.3: Funciones desarrolladas en base de la API Libusb para las transferencias USB de tipo control pertenecientes al demodulador.

Tipo de operación	Función USB del Demodulador	Parámetros libusb_control_transfer()
Lectura	<p>TC90527_I2cRead():</p> <p>Esta función recibe como parámetro la dirección del registro a leer en el demodulador, retorna el dato leído.</p>	<ul style="list-style-type: none"> libusb_device_handle bmRequestType brequest Este byte es usado para indicar que función de transferencia I2C se ejecutará en el firmware, el valor para lectura desde el demodulador está definido por (0xA3)h. wValue wIndex Recibe el número de endpoint, en este caso 0. data Recibe un apuntador al buffer donde se guarda el resultado de la operación I2C, en este caso el buffer es de dos dimensiones en la primera dimensión se guarda el resultado de la operación;

		<p>0 exitosa, otro valor en caso contrario, y en la segunda dimensión el dato leído.</p> <ul style="list-style-type: none"> • wLength <p>La longitud del buffer en este caso 2.</p> <ul style="list-style-type: none"> • timeout <p>Tiempo de espera a que se realice la operación, 20 ms.</p>
Escritura	TC90527_I2cWrite():	<ul style="list-style-type: none"> • libusb_device_handle <p>Esta función recibe como parámetro la dirección del registro a escribir en el demodulador y la data a ser escrita, retorna 0 otro valor en caso contrario.</p> <p>Recibe una variable Libusb para manejar el dispositivo USB.</p> <ul style="list-style-type: none"> • bmRequestType <p>Es el tipo de operación que se desea realizar, recibe el valor de (0x40)h para operación de escritura indicar que es una operación de haya sido exitosa u escritura.</p> <ul style="list-style-type: none"> • bRequest <p>Este byte es usado para indicar que función de transferencia I2C se ejecutará en el firmware, el valor para escritura hacia el demodulador está definido por (0xA2)h.</p> <ul style="list-style-type: none"> • wValue <p>Es la información a pasar a través de la transferencia de control, en este caso consiste en la combinación de los bytes de dato a escribir y dirección de registro a escribir para formar un dato de dos bytes, esta operación la realiza la función funcion_combinar_bytes() antes de que la información sea mandada al dispositivo.</p> <ul style="list-style-type: none"> • wIndex

	<p>Recibe el número de endpoint, en este caso 0.</p> <ul style="list-style-type: none"> • data <p>Recibe un apuntador al buffer donde se guarda el resultado de la operación I2C, 0 exitosa, otro valor distinto de 0 si no lo fue.</p> <ul style="list-style-type: none"> • wLength <p>La longitud del buffer en este caso 1.</p> <ul style="list-style-type: none"> • timeout <p>Tiempo de espera a que se realice la operación, 20 ms.</p>
--	--

A continuación en la **Tabla 9.1.4** se describen las funciones de control USB pertenecientes al sintonizador.

Tabla 9.1. 4: Funciones desarrolladas en base de la API Libusb para las transferencias USB de tipo control pertenecientes al sintonizador.		
Tipo de operación	Función USB del Sintonizador	Parámetros libusb_control_transfer()
Lectura	<p>STV4100_I2C_Read():</p> <p>Esta función recibe como parámetro la dirección del registro a leer en el sintonizador y un apuntador donde se retorna el dato leído.</p>	<ul style="list-style-type: none"> • libusb_device_handle <p>Recibe una variable Libusb para manejar el dispositivo USB.</p> <ul style="list-style-type: none"> • bmRequestType <p>Es el tipo de operación que se desea realizar, recibe el valor de (0xC0)h para indicar que es una operación de lectura.</p> <ul style="list-style-type: none"> • bRequest

	<p>Este byte es usado para indicar que función de transferencia I2C se ejecutará en el firmware, el valor para lectura desde el sintonizador está definido por (0xA5)h.</p> <ul style="list-style-type: none"> • wValue <p>Es la información a pasar a través de la transferencia de control, en este caso consiste en el byte que representa el registro a leer en el sintonizador.</p> <ul style="list-style-type: none"> • wIndex <p>Recibe el número de endpoint, en este caso 0.</p> <ul style="list-style-type: none"> • data <p>Recibe un apuntador al buffer donde se guarda el resultado de la operación I2C, en este caso el buffer es de dos dimensiones en la primera dimensión se guarda el resultado de la operación; 0 exitosa, otro valor en caso contrario, y en la segunda dimensión el dato leído.</p> <ul style="list-style-type: none"> • wLength <p>La longitud del buffer en este caso 2.</p> <ul style="list-style-type: none"> • timeout <p>Tiempo de espera a que se realice la operación, 20 ms.</p>	
Escritura	<p>STV4100_I2C_Write():</p> <p>Esta función recibe</p>	<ul style="list-style-type: none"> • libusb_device_handle <p>Recibe una variable Libusb para manejar el dispositivo USB.</p>

	<p>como parámetro la dirección del registro a escribir en el sintonizador y la data a ser escrita, retorna 0 en caso de la operación de escritura haya sido exitosa u otro valor en caso contrario.</p>	<ul style="list-style-type: none"> • bmRequestType <p>Es el tipo de operación que se desea realizar, recibe el valor de (0x40)h para indicar que es una operación de escritura.</p> <ul style="list-style-type: none"> • bRequest <p>Este byte es usado para indicar que función de transferencia I2C se ejecutará en el firmware, el valor para escritura hacia el sintonizador está definido por (0xA4)h.</p> <ul style="list-style-type: none"> • wValue <p>Es la información a pasar a través de la transferencia de control, en este caso consiste en la combinación de los bytes de dato a escribir y dirección de registro a escribir para formar un dato de dos bytes, esta operación la realiza la función funcion_combinar_bytes() antes de que la información sea mandada al dispositivo.</p> <ul style="list-style-type: none"> • wlIndex <p>Recibe el número de endpoint, en este caso 0.</p> <ul style="list-style-type: none"> • data <p>Recibe un apuntador al buffer donde se guarda el resultado de la operación I2C, 0 exitosa, otro valor distinto de 0 si no lo fue.</p> <ul style="list-style-type: none"> • wLength <p>La longitud del buffer en este caso 1.</p>
--	---	---

		<ul style="list-style-type: none"> • timeout <p>Tiempo de espera a que se realice la operación, 20 ms.</p>
--	--	---

9.1.2.1 Rutinas de Transferencias de Interrupción

Las transferencias de interrupción son las encargadas de obtener el flujo de audio y video en paquetes BTS, para ello se utilizó la interfaz asincrónica de la API Libusb para este tipo de transferencias, de este modo se puede obtener un flujo constante de video evitando perdidas de paquetes de información entre llamado y llamado de una transferencia de interrupción.

La técnica consiste en colocar una cierta cantidad de transferencias en cola, en este caso se utilizaron 64 transferencias de interrupción, con buffers de información de tamaño considerable, de 240Kb, donde en cada transferencias se espera a llenar en totalidad el buffer con paquetes de datos BTS. [12]

La API Libusb es un puente entre el espacio de usuario y el espacio del núcleo Linux, los procesos en el espacio del núcleo ocurren con mayor rapidez que en el espacio de usuario, es allí en el espacio del núcleo donde el buffer de datos es llenado, entonces para buffers más grandes significa menos viajes hacia el espacio de usuario para realizar una nueva transferencia, una vez que una transferencia finaliza otra transferencia está esperando a ser realizada y la transferencia finalizada es puesta al final de la cola de transferencias para ser llamada nuevamente.

Las funciones encargadas de la obtención de los paquetes BTS se muestran en la **Tabla 9.1.5**.

Tabla 9.1.5: Funciones desarrolladas en base de la API Libusb para las transferencias USB de tipo interrupción.

procedimiento_ciclo_recepcion_BTS() ()	<p>Descripción: Esta función se ejecuta en un hilo secundario del programa. Si el dispositivo se encuentra conectado, se encarga de crear 64 transferencias de interrupción haciendo uso de la función funcion_crear_transferencia_usb() y luego ejecutando cada una de las transferencias con la función libusb_submit_transfer(). Al finalizar la ejecución de las transferencias entra en un ciclo de manejo de eventos de las transferencias que se están realizando y usando la función funcion_leer_estado_hotplug() que chequea el estado de conexión en caliente, si el dispositivo es desconectado de manera abrupta en un proceso de recepción de datos se liberan las transferencias y el hilo termina.</p> <p>Esta función retorna 0 para indicar el fin del proceso ejecutado en el hilo.</p>
	<p>Variables:</p> <ul style="list-style-type: none"> • transferencia_usb <p>Variable Libusb para manejar la transferencia.</p>

	<p>Funciones Libusb:</p> <ul style="list-style-type: none"> • libusb_submit_transfer() <p>transferencia_usb, variable Libusb para manejar la transferencia.</p> <ul style="list-style-type: none"> • libusb_handle_events() <p>contexto_usb, representa una sesión de la librería Libusb.</p> <ul style="list-style-type: none"> • libusb_free_transfer() <p>transferencia_usb, variable Libusb para manejar la transferencia.</p>
funcion_crear_transferencia_usb()	<p>Descripción: Esta función se encarga de alojar, crear y devolver una variable Libusb del tipo libusb_transfer. Se usa la función libusb_alloc_transfer() para alojar la transferencia, para la especificación de los datos de la transferencia se hace uso de la función libusb_fill_interrupt_transfer().</p> <p>Variables:</p> <ul style="list-style-type: none"> • transferencia_usb_2 <p>Variable Libusb para manejar la transferencia.</p>

	<p>Funciones Libusb:</p> <ul style="list-style-type: none"> • libusb_alloc_transfer() <p>0, para alojar la transferencia recibe como parámetro 0 ya que es una transferencia de tipo interrupción.</p> <p>transferencia_usb_2, variable Libusb para manejar la transferencia.</p> <ul style="list-style-type: none"> • libusb_fill_interrupt_transfer() <p>transferencia_usb_2, variable Libusb para manejar la transferencia.</p> <p>manejador_dispositivo_usb, variable Libusb para el manejo del dispositivo USB.</p> <p>0x82, dirección del Endpoint 2.</p> <p>MPEG2_TS, buffer de 240 Kb donde se reciben los paquetes BTS.</p> <p>tamanio_buffer, tamaño del buffer en bytes, en este caso 245760 bytes.</p> <p>procedimiento_de_llamada_leer_transferencia_usb, función de llamada cuando finaliza una transferencia.</p> <p>NULL, ningún dato es pasado a la función de llamada.</p> <p>20000, tiempo de espera para realizar la transferencia en milisegundos.</p>
--	--

procedimiento_de_llamada_leer_transferencia_usb()	Descripción: Esta función se ejecuta de manera asincrónica cada vez que una transferencia finaliza, lee si el estado de la transferencia fue exitosa o no, si fue exitosa se procede a copiar la información del buffer de 240Kb a aun buffer de mayor tamaño utilizado por VLC para la reproducción y luego esa transferencia es puesta en cola nuevamente.
	Funciones Libusb: <ul style="list-style-type: none"> • libusb_submit_transfer() transferencia_usb_2, variable Libusb para manejar la transferencia.

Para la reproducción de audio y video de los datos obtenidos en las transferencias que se encuentran en el buffer de 240Kb, estos datos deben ser copiados en un buffer de mayor tamaño para que el reproductor VLC cuente con la cantidad necesarias de datos para la reproducción, por lo cual se creó un buffer de 4,1 Mb. El procedimiento de copiado usa apuntadores para ir copiando de manera sucesiva los bloques de información que contienen paquetes BTS al buffer utilizado por VLC.

El buffer de 4,1 Mb se define como 20 bloques de posiciones de memoria con una longitud de 240Kb para cada bloque y se representa a continuación en la **Tabla 9.1.6.**

Tabla 9.1.6: Distribución de las posiciones de memoria pertenecientes al buffer de 4,1 Mb.

0	245760	491520	737280	983040
1228800	1474560	1720320	1966080	2211840
2457600	2703360	2949120	3194880	3440640
3686400	3932160	4177920	4423680	4669440

En la primera transferencia recibida el apuntador apunta a la posición cero del buffer utilizado por VLC y copia los 240Kb de datos obtenidos en la transferencia de interrupción, luego el apuntador se incrementa en 245760 posiciones, 240Kb, y realiza el proceso de copia nuevamente en la siguiente posición hasta alcanzar finalmente el bloque número 20, luego el apuntador se inicializa nuevamente en la posición 0 y se realiza el proceso nuevamente.

10.1 La API Libvlc

Libvlc es una librería que puede integrarse en una aplicación para dotarla de capacidades multimedia, debido a que VLC se basa en Libvlc es posible acceder a la mayoría de las características que VLC Media Player posee, las rutinas desarrolladas hacen uso de la versión 3.0.0 de VLC.

En la **Tabla 10.1.1** se listan las funciones usadas de la librería Libvlc conjuntamente con una descripción.

Tabla 10.1.1: Funciones de la API Libvlc. [13]	
libvlc_new()	<p>Descripción: Crea una sesión de la librería.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • argc <p>Argumento tipo entero que denota la cantidad de apuntadores pasados a la función.</p> <ul style="list-style-type: none"> • argv <p>Arreglo de apuntadores pasados a la función.</p>
libvlc_release()	<p>Retorna:</p> <ul style="list-style-type: none"> • libvlc_instance_t <p>Variable que representa una instancia de la librería.</p>
libvlc_media_player_pause()	<p>Descripción: Cierra una sesión de la librería.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libvlc_instance_t <p>Variable que representa una instancia de la librería.</p> <p>Retorna: Nada</p>

	<p>Retorna: Nada.</p>
libvlc_media_player_stop()	<p>Descripción: Esta función detiene la reproducción del reproductor.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libvlc_media_player_t <p>Estructura que representa a un reproductor de la librería.</p>
	<p>Retorna: Nada.</p>
libvlc_media_player_is_playing()	<p>Descripción: Esta función retorna el estado de reproducción del reproductor.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libvlc_media_player_t <p>Estructura que representa a un reproductor de la librería.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • 1 Si el reproductor esta reproduciendo. • 0 Si el reproductor no está reproduciendo.
libvlc_audio_set_volume()	<p>Descripción: Esta función permite fijar el nivel de dB del audio del reproductor.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libvlc_media_player_t <p>Estructura que representa a un reproductor de la librería.</p> <ul style="list-style-type: none"> • entero

	<p>Representa el nivel de dB del audio su valor debe estar entre 0 y 100.</p> <p>Retorna: Nada.</p>
libvlc_media_player_set_media()	<p>Descripción: Esta función permite seleccionar un programa en específico dentro de archivos multimedia que contengan varios programas.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libvlc_media_t <p>Variable que representa al contenido multimedia.</p> <ul style="list-style-type: none"> • psz_options <p>Es una cadena de caracteres que contiene el ID del programa a reproducir.</p> <p>Retorna: Nada.</p>
libvlc_media_player_set_xwindow()	<p>Descripción: Esta función se encarga de embeber una interfaz de video en una interfaz gráfica.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • libvlc_media_player_t <p>Estructura que representa a un reproductor de la librería.</p> <ul style="list-style-type: none"> • entero <p>Representa el ID de la ventana o widget de la interfaz gráfica donde incrustar el reproductor.</p> <p>Retorna: Nada.</p>
libvlc_media_new_callbacks()	<p>Descripción: Este función crea un medio de contenido multimedia con</p>

	<p>devoluciones de funciones llamada personalizadas que leen porciones de datos multimedia.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • <code>libvlc_instance_t</code> <p>Variable que representa una instancia de la librería.</p> <ul style="list-style-type: none"> • <code>libvlc_media_open_cb</code> <p>Apuntador a la función de llamada.</p> <ul style="list-style-type: none"> • <code>libvlc_media_read_cb</code> <p>Apuntador a la función de llamada.</p> <ul style="list-style-type: none"> • <code>libvlc_media_seek_cb</code> <p>Apuntador a la función de llamada.</p> <ul style="list-style-type: none"> • <code>libvlc_media_close_cb</code> <p>Apuntador a la función de llamada.</p> <ul style="list-style-type: none"> • <code>data</code> <p>Apuntador de datos a pasar a la función.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • <code>libvlc_media_t</code> <p>Variable que representa al contenido multimedia.</p>
<code>libvlc_media_open_cb()</code>	<p>Descripción: Esta función se encarga de abrir el medio de datos de entrada. Es una función de llamada de <code>libvlc_media_new_callbacks()</code>.</p>

	<p>Parámetros:</p> <ul style="list-style-type: none"> • opaque <p>Es el apuntador pasado a la función libvlc_media_new_callbacks().</p> <ul style="list-style-type: none"> • datap <p>Define el espacio de almacenamiento para un puntero de datos privado.</p> <ul style="list-style-type: none"> • sizep <p>Longitud del flujo de datos.</p>
	<p>Retorna:</p> <ul style="list-style-type: none"> • 0 <p>Si la operación fue exitosa.</p> <ul style="list-style-type: none"> • Otro valor <p>En caso de error.</p>
libvlc_media_read_cb()	<p>Descripción: Esta función se encarga de leer el medio de datos de entrada. Es una función de llamada de libvlc_media_new_callbacks().</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • opaque <p>Es el apuntador pasado a la función libvlc_media_new_callbacks().</p> <ul style="list-style-type: none"> • buf <p>Dirección de inicio del buffer para leer datos.</p> <ul style="list-style-type: none"> • len

	<p>Longitud del buffer.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • entero <p>Número de bytes leídos.</p> <ul style="list-style-type: none"> • 0 <p>Si llega al final del flujo de datos.</p> <ul style="list-style-type: none"> • -1 <p>Si ocurre un error.</p>
libvlc_media_seek_cb()	<p>Descripción: Esta función se encarga de buscar una posición en el medio de datos de entrada. Es una función de llamada de libvlc_media_new_callbacks().</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • opaque <p>Es el apuntador pasado a la función libvlc_media_new_callbacks().</p> <ul style="list-style-type: none"> • offset <p>Desplazamiento de bytes absolutos para buscar.</p> <p>Retorna:</p> <ul style="list-style-type: none"> • 0 <p>Si la operación fue exitosa.</p> <ul style="list-style-type: none"> • -1 <p>En caso de error.</p>
libvlc_media_close_cb()	<p>Descripción: Esta función se</p>

	<p>encarga de cerrar el medio de datos de entrada. Es una función de llamada de <code>libvlc_media_new_callbacks()</code>.</p> <p>Parámetros:</p> <ul style="list-style-type: none"> • <code>opaque</code> <p>Es el apuntador pasado a la función <code>libvlc_media_new_callbacks()</code>.</p> <p>Retorna: Nada</p>
--	--

10.1.1 Rutinas Multimedia

Para la obtención del contenido multimedia se creó una función que se encarga de obtener la información multimedia del buffer de paquetes BTS que posee 4,1 Mb de longitud, para ello la función se basa en llamadas asincrónicas que obtienen tramos de información del buffer y luego estos datos son pasados como información multimedia al reproductor.

Los tramos de información están divididos en paquetes de 240Kb para un total de 20 tramos, dando un total de 4,1 Mb el tamaño total del buffer, en la primera lectura un apuntador apunta a la posición inicial del buffer y se copian los primeros 240Kb, que equivalen a 245760 posiciones, a la variable de tipo `libvlc_media_t`, luego el apuntador se incrementa 245760 posiciones y procede a realizar la misma operación con el siguiente bloque de datos hasta realizar la última operación correspondiente al bloque 20, después de esta operación el apuntador se inicializa nuevamente en la posición inicial y el proceso se repite nuevamente.

Hay que hacer notar que el buffer de 4,1 Mb es un recurso compartido con la función que se ejecuta en otro hilo del programa encargada de llenarlo con el flujo de paquetes BTS procedente de la transferencias USB de tipo interrupción, por lo cual mientras se obtiene el

contenido multimedia del buffer este a su vez es refrescado con nueva información obtenida a través de transferencias USB desde el dispositivo de TDA.

A continuación en la **Tabla 10.1.2** se especifican las funciones desarrolladas para el control multimedia.

Tabla 10.1.2: Funciones desarrolladas en base a la API Libvlc.

procedimiento_media_callbacks()	<p>Descripción: Esta función se encarga de obtener la información multimedia del buffer de paquetes BTS con la función libvlc_media_new_callbacks(), añadir al reproductor el contenido multimedia a producir usando la función libvlc_media_player_set_media(), seleccionar el programa multimedia a reproducir utilizando la función libvlc_media_add_option() y finalmente comenzar la reproducción con la función libvlc_media_player_play().</p>
	<p>Variables:</p> <ul style="list-style-type: none"> • media <p>Variable que representa al contenido multimedia.</p>
	<p>Funciones Libvlc:</p> <ul style="list-style-type: none"> • libvlc_media_new_callbacks() <p>instancia_vlc, es una variable del tipo libvlc_instance_t, que representa una sesión de la librería.</p> <p>abrir_media, es el apuntador a la</p>

	<p>función de llamada abrir_media().</p> <p>leer_media, es el apuntador a la función de llamada leer_media().</p> <p>posicionar_media, es el apuntador a la función de llamada posicionar_media().</p> <p>NULL, es el apuntador a la función llamada cerrar datos, nulo no se utilizó.</p> <p>apuntador_al_buffer_recepcion_BTS, es un apuntador al buffer de 4,1 Mb que contiene los paquetes BTS.</p> <ul style="list-style-type: none"> • libvlc_media_add_option() <p>media, es una variable del tipo libvlc_media_t, que representa al contenido multimedia.</p> <p>funcion_leer_id(), función que retorna una cadena de caracteres con el ID del programa.</p> <ul style="list-style-type: none"> • libvlc_media_player_set_media() <p>reproductor, es una variable del tipo libvlc_media_player_t, que representa a un reproductor.</p> <p>Media, es una variable del tipo libvlc_media_t, que representa al contenido multimedia.</p> <ul style="list-style-type: none"> • libvlc_media_player_play() <p>reproductor, es una variable del tipo libvlc_media_player_t, que representa a un reproductor.</p>
--	--

abrir_media()	<p>Descripción: Esta función abre el recurso de donde se extrae el contenido multimedia, el buffer de BTS de 4,1 Mb. Esta es una función de llamada de libvlc_media_new_callbacks().</p> <p>Funciones Libusb:</p> <ul style="list-style-type: none"> • libvlc_media_open_cb() <p>opaque, es el apuntador pasado a la función libvlc_media_new_callbacks() del buffer de datos a leer.</p> <p>datap, dirección de datos interna de la función.</p> <p>sizep, longitud del buffer.</p> <p>Retorna 0 como operación exitosa.</p>
leer_media()	<p>Descripción: Esta función lee la información multimedia del buffer de BTS de 4,1 Mb en porciones de 240Kb en cada llamada. Esta es una función de llamada de libvlc_media_new_callbacks().</p> <p>Funciones Libusb:</p> <ul style="list-style-type: none"> • libvlc_media_read_cb() <p>opaque, es el apuntador pasado a la función libvlc_media_new_callbacks() del buffer de datos a leer.</p> <p>buffer_de_llamada, es el buffer interno de la función que guarda los</p>

	<p>datos leídos.</p> <p>longitud_de_llamada, representa la cantidad de datos leídos.</p> <p>Retorna la cantidad de datos leídos.</p>
posicionar_media()	<p>Descripción: Esta función se posiciona en el recurso multimedia para determinar la posición de recolección de datos. Esta es una función de llamada de libvlc_media_new_callbacks().</p> <p>Esta función no es utilizada ya que el apuntador de bloques al buffer de BTS en la función leer_media() realiza la función de offset, pero debe ser implementada para que las llamadas de obtención de datos multimedia de manera asincrónica funcionen.</p> <p>Funciones Libusb:</p> <ul style="list-style-type: none"> • libvlc_media_seek_cb() <p>opaque, es el apuntador pasado a la función libvlc_media_new_callbacks() del buffer de datos a leer.</p> <p>offset, ultima posición de datos leídos.</p> <p>Retorna 0.</p>

11.1 Interfaz Gráfica

Existen muchas soluciones para la realización y manejo de interfaces gráficas en los sistemas operativos basados en GNU/Linux, entre ellas se tomaron en cuenta a la hora de realizar el proyecto QT, GTK y WxWidgets por el soporte y códigos de ejemplo de la API Libvlc sobre estas interfaces, estos dados por la VideoLAN Organization empresa sin fines de lucro responsable por el desarrollo y mantenimiento del reproductor VLC Media Player, resultando seleccionada la utilidad GTK debido a su facilidad a la hora de embeber el reproductor en la interfaz gráfica y la sencillez del código para implementar esta interfaz, además de que GTK es la interfaz con la cual está escrito el escrito GNOME para sistemas Linux y muchas aplicaciones hoy en día usan este tipo de interfaz lo cual augura un soporte a largo tiempo de esta interfaz, la documentación de GTK es extensa y existen muchas comunidades a las cuales pedir soporte sin costo alguno, QT fue dejado de lado en parte por su complejidad y la obsolescencia de la versión de QT usada por VLC, aunque esta interfaz gráfica posee gran soporte y documentación fue descartada, también fue descartada la interfaz WxWidgets por la falta de soporte, documentación y la complejidad del código usado para generar los mismo resultados que la interfaz GTK.

11.1.1 Empaquetamiento gráfico.

La aplicación desarrollada posee una sola ventana, la distribución gráfica se realizó usando una rejilla que es embedida a la ventana de la aplicación, la rejilla es una utilidad que ofrece GTK para distribuir los diferentes layaout, un layout es una cuadrícula imaginaria que divide en espacios o campos la página que se diseña para facilitar la distribución de elementos como textos o gráficos, GTK define diferentes tipos de layout como contenedores verticales y horizontales, los cuales pueden contener widgets, los widgets dan fácil acceso a funciones frecuentemente usadas y

proveen de información visual como botones, imágenes, etiquetas, ventanas, etc.

En la **Figura 11.1.1** se observa la distribución de la rejilla, donde las coordenadas de distribución se muestran como pares de puntos (x, y).

0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	0,10	0,11
1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	1,10	1,11

Figura 11.1. 1: Distribución de la rejilla, coordenadas de distribución pares de puntos (x, y).

A continuación en **Tabla 11.1.1** se muestra el empaquetamiento de los distintos widgets que componen la interfaz gráfica.

Tabla 11.1.1: Empaquetamiento de widgets.	
Widgets	Empaquetamiento
	La ventana del reproductor multimedia está representada por el widget mostrado en la Figura 11.1.2 , este widget se encarga de dibujar un área en la aplicación donde se muestra el contenido del reproductor, el widget está insertado directamente a la rejilla.
	Los widgets que realizan las acciones de reproducción, pausar, detener, control de volumen del contenido multimedia y además modo de reproducción a pantalla completa, se encuentran empaquetados en el layout Botones de Control Multimedia que contiene

	<p>a sus hijos en un formato de caja horizontal, como se muestra en la Figura 11.1.3.</p>
 <p>Figura 11.1.4: Botones Indicadores.</p>	<p>Los widgets que tienen como función entregar información visual acerca del estado de conexión del dispositivo USB así como la intensidad de la señal recibida por el dispositivo están agrupados en un layout de nombre Botones Indicadores, que contiene a sus hijos en un formato de caja horizontal, como se muestra en la Figura 11.1.4.</p>
 <p>Figura 11.1.5: Botones de Programación.</p>	<p>Los widgets de botones que representan a los distintos programas de la TDA venezolana se encuentran agrupados en un layout que contiene a sus hijos en un formato de caja vertical caja vertical el cual se muestra en la Figura 11.1.5.</p>



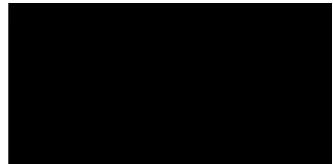
Figura 11.1.6:
Ventana de
Desplazamiento.

El widget de una ventana con una barra de desplazamiento vertical contiene al layout de Caja Vertical Botones de Programación como se muestra en la **Figura 11.1.6**.

El modo de empaquetamiento de los widgets y layouts en la rejilla se realizó de la siguiente manera:

En la **Tabla 11.1.2** se muestran las coordenadas ocupadas por los elementos donde la coordenada X de la rejilla posee un valor fijo de 0.

Tabla 11.1.2: Empaquetamiento en la rejilla donde el valor de la coordenada X es 0.

Coordenadas rejilla X/Y	0..9	10..11
0		 <p>Figura 11.1.6: Ventana de Desplazamiento.</p>

En la **Tabla 11.1.3** se muestran las coordenadas ocupadas por los elementos donde la coordenada X de la rejilla posee un valor fijo de 1.

Coordenadas rejilla X/Y	0..1	9..10
1	 <p>Figura 11.1.3: Botones de Control Multimedia.</p>	 <p>Figura 11.1.4: Botones Indicadores.</p>

Las acciones de empaquetamiento realizado dan como resultado la interfaz gráfica de la **Figura 11.1.7**.

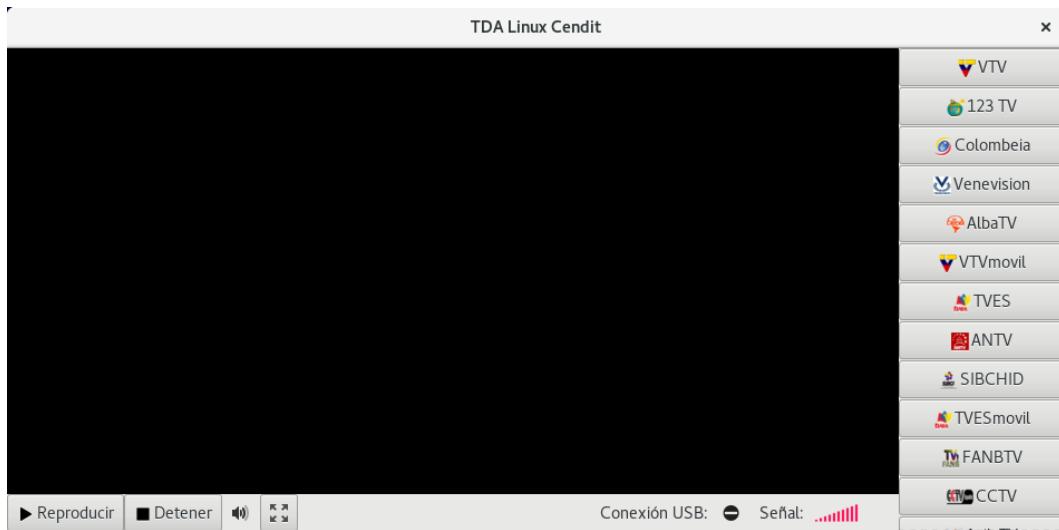


Figura 11.1.7: Interfaz gráfica de la aplicación.

11.1.1 Funciones de Enlace entorno Gráfico.

Las rutinas de enlace del entorno gráfico son las funciones encargados de hacer la conexión entre el front-end y back-end de la aplicación, el front-end es la parte del software que interactúa con el usuario, en este caso la interfaz gráfica desarrollada en GTK, y el back-end es la parte que procesa la entrada desde el front-end, son las funciones encargadas de los procesos USB y la reproducción de video. La separación del sistema en front-ends y back-ends es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas.

En la Tabla se muestran los widgets, el nombre de los métodos que actúan de conexión entre el front-end y el back-end conjuntamente con una descripción de los procesos que ejecutan.

Tabla 11.1.4: Widgets, descripción y funciones de enlace.

	<p>Descripción: Esto widget se encarga de reproducir o pausar el contenido multimedia, si ningún programa ha sido seleccionado anteriormente se sintonizara y reproducirá por defecto el Programa VTV al momento de hacer click sobre el botón reproducir.</p>
	<p>Funciones:</p> <ul style="list-style-type: none"> • procedimiento_de_llamada_reproducir_pausar() • procedimiento_de_llamada_hilo_recepcion()
	<p>Descripción: Este widget al hacer click sobre él detiene el contenido multimedia que se está reproduciendo, sino hay cometido en reproducción no causa efecto alguno.</p>

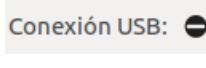
 <p>Figura 11.2.2: Widget del botón ajuste de volumen</p>	<p>Descripción: Este widget permite controlar el nivel del audio en dB del reproductor.</p> <p>Funciones:</p> <ul style="list-style-type: none"> • procedimiento_de_llamada_volumen()
 <p>Figura 11.2.3: Widget del botón pantalla completa.</p>	<p>Descripción: Este widget a hacer click sobre el permite colocar en modo pantalla completa la ventana del reproductor.</p> <p>Funciones:</p> <ul style="list-style-type: none"> • procedimiento_de_llamada_maxima_pantalla()
<p style="text-align: center;">Widgets del Layout Botones Indicadores</p>	
<p>Descripción: Los widgets indicadores de estado muestran cuando el dispositivo USB esta conectado como en la Figura 11.2.5 o desconectado como se muestra en la Figura 11.2.4, los otros widgets muestran la intensidad de la señal recibida entre una escala de 0 y 10 siendo 10 la máxima intensidad de señal.</p>	
<p>Funciones:</p> <ul style="list-style-type: none"> • procedimiento_imagenes_caja_horizontal_indicadores() 	
<p>Widgets:</p>	
 <p>Figura 11.2.4: Widget estado conexión USB, desconectado.</p>	 <p>Figura 11.2.10: Widget estado conexión USB, conectado.</p>



Figura 11.2.5:
Widget intensidad de la señal, nivel de intensidad 0.



Figura 11.2.6:
Widget intensidad de la señal, nivel de intensidad 1.



Figura 11.2.7:
Widget intensidad de la señal, nivel de intensidad 2.

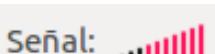


Figura 11.2.8:
Widget intensidad de la señal, nivel de intensidad 3.



Figura 11.2.9:
Widget intensidad de la señal, nivel de intensidad 4.



Figura 11.2.11:
Widget intensidad de la señal, nivel de intensidad 5.



Figura 11.2.12:
Widget intensidad de la señal, nivel de intensidad 6.



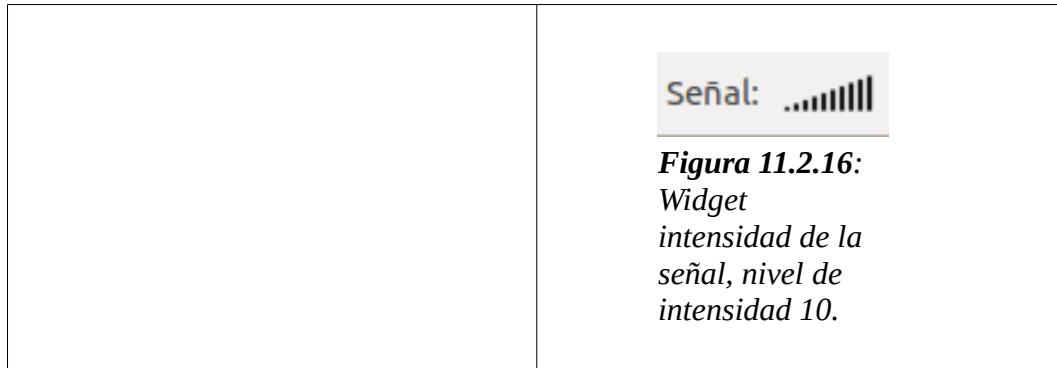
Figura 11.2.13:
Widget intensidad de la señal, nivel de intensidad 7.



Figura 11.2.14:
Widget intensidad de la señal, nivel de intensidad 8.



Figura 11.2.15:
Widget intensidad de la señal, nivel de intensidad 9.



Se añadieron otras características que interactúa con el usuario con la creación de una función de nombre **procedimiento_de_llamada_tecla_presionada()** la cual habilita un grupo de teclas calientes para un cierto número de acciones de control de la interfaz de usuario. A continuación se definen las teclas calientes programadas y su función.

- TECLA_MÁS: Incrementar volumen.
- TECLA_MENOS: Decremento de volumen.
- TECLA_A: Reproducir o pausar contenido multimedia.
- TECLA_a: Reproducir o pausar contenido multimedia.
- TECLA_D: Detener contenido multimedia.
- TECLA_d: Detener contenido multimedia.
- TECLA_S: Entrar en modo pantalla completa.
- TECLA_s: Entrar en modo pantalla completa.
- TECLA_ESCAPE: Salir del modo pantalla completa.

12.1 Instalador del Programa

El sistema operativo para el cual se crearon rutinas de instalación y desinstalación del software desarrollado es el sistema operativo **GNU/Linux Fedora 24** y por lo tanto es en el cual se realizaron las pruebas de funcionamiento de las rutinas.

El programa desarrollado para su compilación depende de ciertas librerías y de la definición de algunas banderas de compilación que enlacen el proceso de generar el archivo binario con las librerías necesarias.

Las distribuciones que usan el núcleo GNU/Linux cuenta con un directorio ubicado en la raíz de archivos **/opt** cuya existencia está definida para la instalación de paquetes de software de aplicación complementaria, en este directorio es donde se ubicará el archivo binario ejecutable de la aplicación y la imagen del ícono de acceso directo.

La estructura de archivos del instalador se observa en la **Figura 12.1.1** ejecutando el comando **tree** en la terminal:

```
[cendit@cendit-machine ~]$ tree instalador_fedora
instalador_fedora
├── 40_usb.rules
└── imagenes
    ├── canales
    │   ├── alba_imagen.h
    │   ├── antv_imagen.h
    │   ├── avilatv_imagen.h
    │   ├── cctv_imagen.h
    │   ├── colombeia_imagen.h
    │   ├── conciencia_imagen.h
    │   ├── fanbtv_imagen.h
    │   ├── meridiano_imagen.h
    │   ├── pdvsatv_imagen.h
    │   ├── russiatoday_imagen.h
    │   ├── sibci_imagen.h
    │   ├── telesur_imagen.h
    │   ├── televen_imagen.h
    │   ├── tv123_imagen.h
    │   ├── tves_imagen.h
    │   ├── venevision_imagen.h
    │   ├── vive_imagen.h
    │   └── vtv_imagen.h
    ├── iconos
    │   ├── detener_imagen.h
    │   ├── icon_imagen.h
    │   ├── pantalla_completa_imagen.h
    │   ├── pausar_imagen.h
    │   ├── reproducir_imagen.h
    │   ├── senial_00_imagen.h
    │   ├── senial_01_imagen.h
    │   ├── senial_02_imagen.h
    │   ├── senial_03_imagen.h
    │   ├── senial_04_imagen.h
    │   ├── senial_05_imagen.h
    │   ├── senial_06_imagen.h
    │   ├── senial_07_imagen.h
    │   ├── senial_08_imagen.h
    │   ├── senial_09_imagen.h
    │   ├── senial_10_imagen.h
    │   └── usb_conectado_imagen.h
    └── include
        ├── calls.h
        ├── initialize.h
        ├── samsung.h
        └── thread.h
            └── usb.h
    ├── install
    ├── logo.png
    ├── main.cpp
    └── uninstall
        ├── vlc-3.0.0-0.26snap.20170601git.fc24.x86_64.rpm
        ├── vlc-core-3.0.0-0.26snap.20170601git.fc24.x86_64.rpm
        └── vlc-devel-3.0.0-0.26snap.20170601git.fc24.x86_64.rpm

```

4 directories, 49 files

Figura 12.1.1: Estructura de archivos del instalador.

El instalador se encuentra dividido en dos archivos un archivo que ejecuta la instalación con el nombre **install** y otro que ejecuta la desinstalación llamado **unisntall**.

La carpeta **imagenes** contiene las imágenes usadas por los widgets en formato binario, las cuales se integran al programa como cabeceras de C.

El archivo **40_usb.rules**, es un archivo que añade reglas para los dispositivos USB, en este caso solo para el dispositivo USB de la compañía Cypress, las cuales permiten procesos de lectura y escritura sin necesidad de ser un usuario root del sistema.

El archivo **main.cpp**, es el archivo principal del programa que enlaza a todos los demás archivo, el directorio **include** contiene los archivos cabeceras con las funciones desarrolladas para el programa.

El archivo **logo.png** es el icono utilizado por el lanzador de la aplicación.

Los paquetes **vlc** son los paquetes de instalación del software multimedia VLC.

12.1.1 Paquetes y software requerido.

Los paquetes de software utilizados para el desarrollo y compilación del software fueron:

- libusb-1.0-0, versión: 2:1.0.20-1;
- vlc, versión: 3.0.0.26; vlc-core, versión: 3.0.0.26.
- libgtk-3-0, versión: 3.18.9.
- g++, versión: 4:5.3.1.
- libusb-1.0-0-devel, versión: 2:1.0.20-1.
- libvlc-devel, versión: 3.0.0.
- libgtk-3-devel, versión: 3.18.9.

12.1.2 Proceso de compilación

El proceso de compilación se realiza con el compilador g++, mediante la siguiente línea de comando:

```
g++ main.cpp -std=c++11 -lusb-1.0 -lvlc -pthread -rdynamic  
-lx11 -s `pkg-config --cflags gtk+-3.0` `pkg-config --libs gtk+-3.0` -o  
tda_executable -w
```

El comando especifica el compilador a usar **g++**, el nombre del archivo a compilar en este caso el **main.cpp**, que es archivo principal del programa, el compilador se encarga de enlazar y compilar los otros archivos que estén enlazados dentro del archivo principal para generar el ejecutable binario, las banderas de compilación. Definición de las banderas:

- **-std=c++11**, esta bandera establece la versión del lenguaje c++ 2011.
- **-lusb-1.0**, esta bandera enlaza la librería LIBUSB.
- **-lvlc**, esta bandera enlaza la librería LIBVLC.
- **-pthread**, esta bandera enlaza la librería para poder usar hilos.
- **-rdynamic**, esta bandera permite obtener el rastreo de pila completo con los nombres de función.
- **-lx11**, esta bandera enlaza con la librería Xlib que es usada por VLC.
- **-s**, esta bandera elimina toda la tabla de símbolos y la información de reubicación del archivo ejecutable.
- **`pkg-config --cflags gtk+-3.0`**, **`pkg-config --libs gtk+-3.0`** define la versión y banderas de GTK3.

- **-o** tda_executable, esta bandera establece el nombre del archivo binario compilado resultante.
- **-w** esta bandera es para ignorar las advertencias de compilación.

12.1.3 Reglas Udev para el Dongle USB.

Udev es el gestor de dispositivos que usa el Kernel Linux. Su función es controlar los ficheros de dispositivo en el directorio **/dev**, en **/dev** hay nodos de dispositivo creados para cada dispositivo conocido, también es posible definir reglas para dispositivos individuales gestionados por Udev estas reglas son añadidas como archivos en el directorio **/etc/udev/rules.d/**, la regla definida como **40_usb.rules** para el dispositivo USB de TDA se observa en la **Tabla 12.1.1**.

Tabla 12.1.1: Código de Regla Udev para el dispositivo USB de TDA.

DRIVER=="usb", SUBSYSTEMS=="usb", ATTR{idVendor}=="04b4", ATTR{idProduct}=="1004", GROUP="video", MODE="0666"

Esta regla entrega permisos de lectura y escritura a un dispositivo USB que se conecta en caliente con un IDVendor = (0x04b4)h y un IDProduct = (0x1004)h. Los usuarios pertenecientes al grupo **video** con el uso de esta regla podrán acceder al dispositivo sin necesidad de ser usuarios root del sistema GNU/Linux.

12.1.4 Acceso directo

La creación del acceso directo en GNU/Linux se basa en un archivo de texto plano con extensión **.desktop**, este archivo crea un lanzador de la aplicación, los accesos directos poseen el siguiente formato:

[Desktop Entry]
Version=Version del programa
Name=Nombre del programa
Comment=Comentario sobre el programa
Exec=Dirección del ejecutable
Icon=Dirección del ícono
Terminal=Si es lanzada en la terminal o no la aplicación
Type=Tipo de Aplicación.
Categories=Categoría de la aplicación.

Llenado los campos para el acceso directo de la aplicación desarrollada se obtuvo el siguiente formato de acceso directo.

[DesktopEntry]
Version=0.9
Name=TDA_LINUX
Comment=Televisión Digital Abierta Venezolana
Exec=/opt/TDA_LINUX/tda_executable
Icon=/opt/TDA_LINUX/logo.png
Terminal=false
Type=Application
Categories= Video;AudioVideo;

12.1.5 Script de instalación del programa

El script de instalación del programa son líneas de comandos que se ejecutan en la terminal del sistema operativo y realizan los procesos de obtención e instalación de paquetes necesarios para las rutinas desarrolladas, ejecuta el comando de compilación de la aplicación, crea el acceso directo de la aplicación, crea el directorio de instalación y modifica las reglas Udev para dar los permisos necesarios al dispositivo USB, este script debe ejecutarse con permisos elevados de usuario, utilizando el comando **sudo ./install** para la instalación del software.

A continuación se muestran las operaciones que realiza la rutina de instalación en la **Tabla 12.1.2**.

Tabla 12.1.2: Código del script de instalación.

```
#!/bin/bash

echo "Para la instalación TDA LINUX debe ser super usuario." #Mostrar mensaje

echo "Instalacion de dependencias necesarias...\n"      #Mostrar mensaje

#Añade repositorios de VLC
dnf install https://download1.rpmfusion.org/free/fedora/rpmfusion-free-release-$(rpm -E %fedora).noarch.rpm

#Se instalan las dependencias de paquetes necesarios para la compilación y ejecución de la aplicación.
sudo dnf install gcc-c++ gtkmm30-devel gstreamermm-devel cluttermm-devel webkitgtk3-devel libgdammm-devel vlc vlc-devel libusb-devel

#Se instala la versión de VLC del instalador
rpm -e --nodeps vlc-core vlc vlc-devel
rpm -ivh *.rpm

echo "Creando carpeta en la ruta de instalación...\n"      #Mostrar mensaje

mkdir -p /opt/TDA_LINUX                                #Crea la carpeta
```

TDA_LINUX

```
echo "Compilando...\n"          #Mostrar mensaje

g++ main.cpp -std=c++11 -lusb-1.0 -lvlc -pthread -rdynamic -IX11 -s `pkg-
config --cflags gtk+-3.0` `pkg-config --libs gtk+-3.0` -o tda_executable -w
#Compila el código fuente y crea el ejecutable

mv tda_executable /opt/TDA_LINUX #mueve el ejecutable a la carpeta
TDA_LINUX

cp logo.png /opt/TDA_LINUX      #mueve el ejecutable a la carpeta
TDA_LINUX

echo "Creando acceso directo...\n"      #Mostrar mensaje

touch /opt/TDA_LINUX/TDA.desktop      #crea el archivo del lanzador
en la carpeta TDA_LINUX llamado TDA

shopt -s xpg_echo
echo "[Desktop
Entry]\nVersion=0.1\nName=TDA_LINUX\nComment=Televisión     Digital
Abierta
Venezolana\nExec=/opt/TDA_LINUX/tda_executable\nIcon=/opt/TDA_LIN
UX/logo.png\nTerminal=false\nType=Application\nCategories=Video;Audio
Video;" > /opt/TDA_LINUX/TDA.desktop

chmod 777 -R /opt/TDA_LINUX/      #Permisos de ejecución del
lanzador

mv /opt/TDA_LINUX/TDA.desktop /usr/share/applications/ #Mover a
aplicaciones

echo "Copiando regla udev...\n"

cp -R 40_usb.rules /etc/udev/rules.d/ #Mover las reglas Udev al sistema
output=$(whoami)                      #Obtiene el nombre del usuario

sudo usermod -a -G video $output      #Añade al usuario al grupo de
video

echo "Reiniciando reglas udev...\n"    #Mostrar mensaje
```

```
sudo udevadm control --reload-rules && udevadm trigger #Reiniciar las  
reglas Udev
```

12.1.6 Script de desinstalación del programa

El script de desinstalación del programa debe ejecutarse con permisos elevados de usuario, utilizando el comando **sudo ./uninstall** para la desinstalación del software y ejecuta los procesos de eliminación del directorio y los archivos de la aplicación también se carga de eliminar la regla Udev para el dispositivo USB añadida en el proceso de instalación.

A continuación se muestran las operaciones que realiza la rutina de desinstalación en la **Tabla 12.1.3**.

Tabla 12.1.3: Código del script de desinstalación.

```
#!/bin/bash

rm -R /opt/TDA_LINUX           #eliminar la carpeta TDA_LINUX

rm /etc/udev/rules.d/40_usb.rules #eliminar reglas Udev del dispositivo  
TDA

rm /usr/share/applications/TDA.desktop #eliminar el lanzador

sudo udevadm control --reload-rules && udevadm trigger #Reiniciar las  
reglas Udev
echo "Desisntalación finalizada.\n"                      #Mostrar mensaje
```

13.1 Documentación.

La documentación de las rutinas desarrolladas lo que incluye el pseudocódigo se realizó utilizando el software de documentación **Doxxygen** versión 1.8.13. Es la herramienta estándar de facto para generar documentación a partir de fuentes C++, pero también es compatible con otros lenguajes de programación populares como C, Objective-C, C#, PHP, Java, Python, Fortran, VHDL, Tcl, y hasta cierto punto D.

Puede generar una de documentación en línea para ver desde un navegador (en HTML) y/o un manual de referencia de un conjunto de archivos fuente documentados. También soporta salida en formato RTF (MS-Word), PostScript, PDF con hiperenlaces, HTML comprimido y páginas de manual Unix. La documentación se extrae directamente de las fuentes del código, lo que hace que sea mucho más fácil mantener la documentación coherente con el código fuente. Doxygen también puede visualizar las relaciones entre los distintos elementos mediante gráficos de dependencia de inclusión, diagramas de herencia y diagramas de colaboración, que se generan automáticamente. Doxygen está desarrollado bajo Mac OS X y Linux, pero está configurado para ser altamente portátil. Como resultado, funciona en la mayoría de otros sabores de Unix también. Además, los ejecutables para Windows están disponibles.

Doxygen fue configurado para generar la documentación del código fuente en formato HTML para ser visualizada desde un navegador, esta documentación incluye descripción de todas las funciones desarrolladas, pseudocódigo de todos los archivos del código fuente, diagramas de dependencia y herencia de las funciones.

En las **Figuras 13.1.1, 13.1.2, 13.1.3, 13.1.4, 13.1.5, 13.7.6 y 13.1.7** se muestra la documentación generada para el grupo de archivos del código fuente de la aplicación desarrollada. Esta documentación puede ser encontrada en los anexos del presente trabajo de grado.



Figura 13.1.1: Documentación Doxygen.



Figura 13.1.2: Documentación Doxygen, enlaces pseudocódigos.

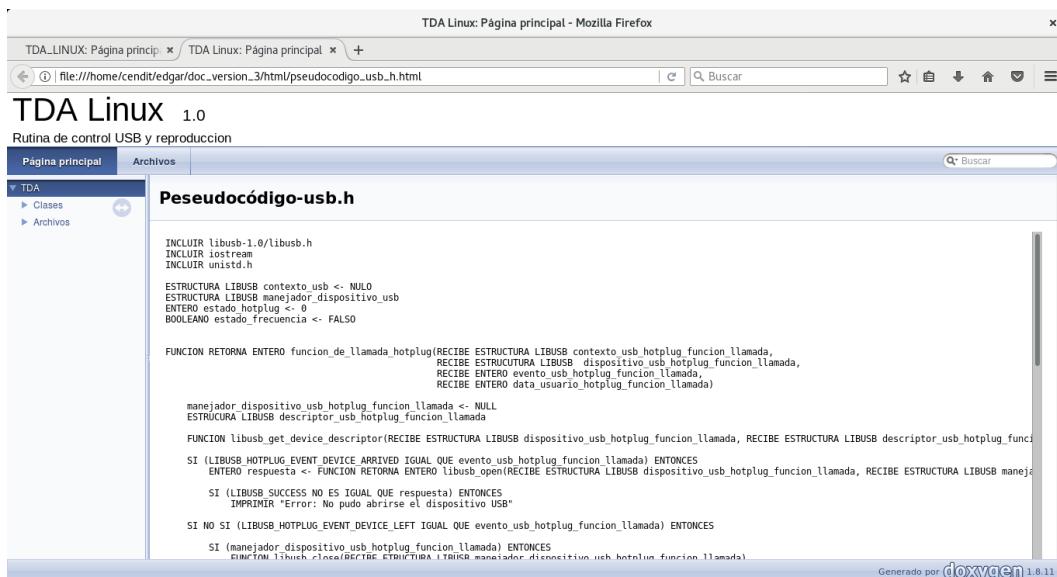


Figura 13.1.3: Documentación Doxygen, ejemplo pesudocódigo.



Figura 13.1.4: Documentación Doxygen, ejemplo documentación de variables.

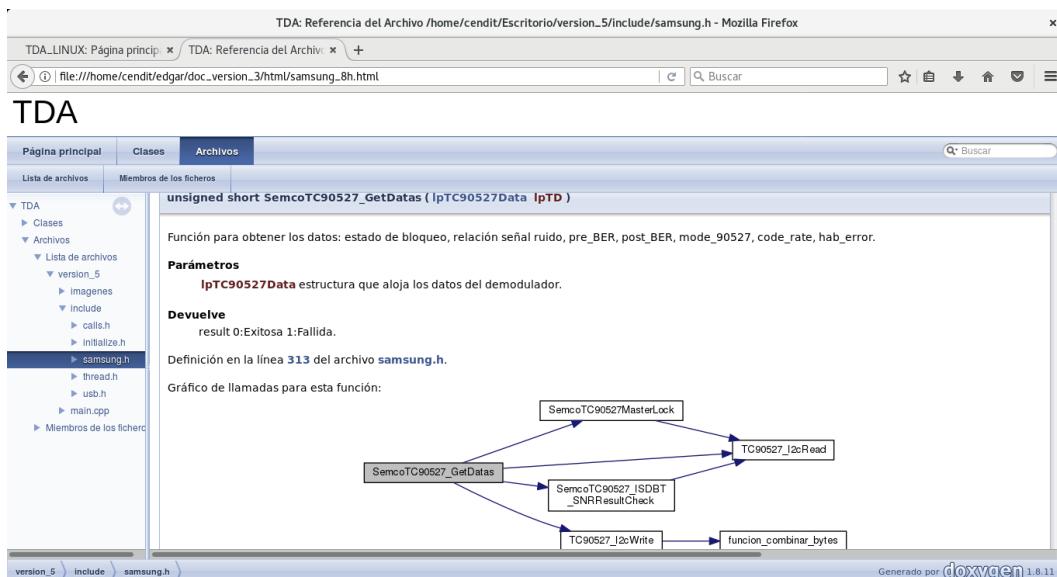


Figura 13.1.5: Documentación Doxygen, ejemplo documentación de funciones.

Para la creación del manual de instalación de la aplicación desarrollada se utilizó el software de ofimática LibreOffice versión 5.3.4.2 el cual es software libre, posee un aplicación llamada LibreOffice Writer la cual es el procesador de texto, esta funciona de manera cada vez más similar a las aplicaciones Microsoft Word y WordPerfect, permite exportar archivos de texto a distintos formatos como pueden ser PDF y HTML sin software adicional, utiliza la funcionalidad WYSIWYG lo que permite escribir un documento viendo directamente el resultado final, también puede utilizarse como un simple editor de textos.

El manual de instalación desarrollado con LibreOffice Writer consta de una serie de pasos detallados los cuales contienen imágenes que pretenden facilitar la realización de las acciones de instalación de la aplicación desarrollada de manera exitosa. El manual se encuentra en formato PDF esto debido a que este formato puede ser visto usando diferentes tipos de software como aplicaciones ofimáticas, visor de documentos o navegadores web. En las **Figura 13.1.6** y **13.1.7** se puede

apreciar muestras del manual de instalación dirigido al usuario, este manual puede ser encontrado en los anexos del presente trabajo de grado.

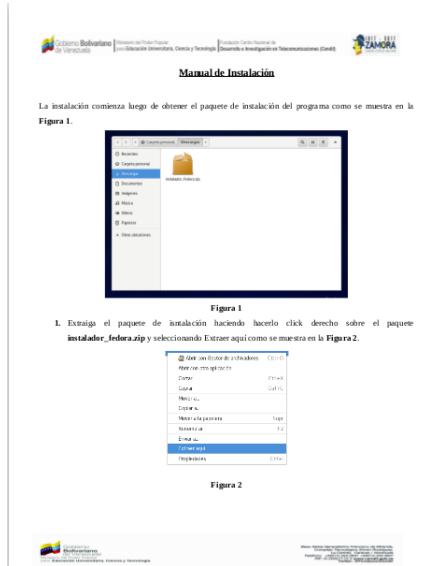


Figura 13.1.6: Manual de instalación muestra número uno.



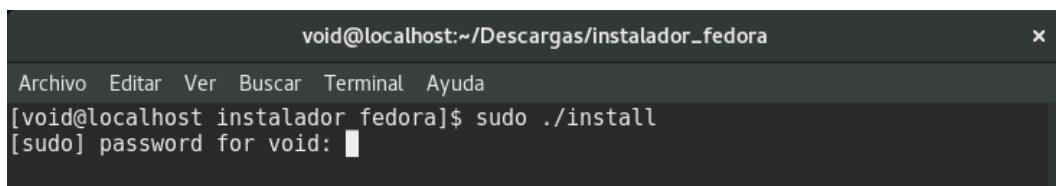
Figura 13.1.7: Manual de instalación muestra número dos.

CAPÍTULO V

ANÁLISIS DE RESULTADOS

14.1 Ejecución del instalador del programa.

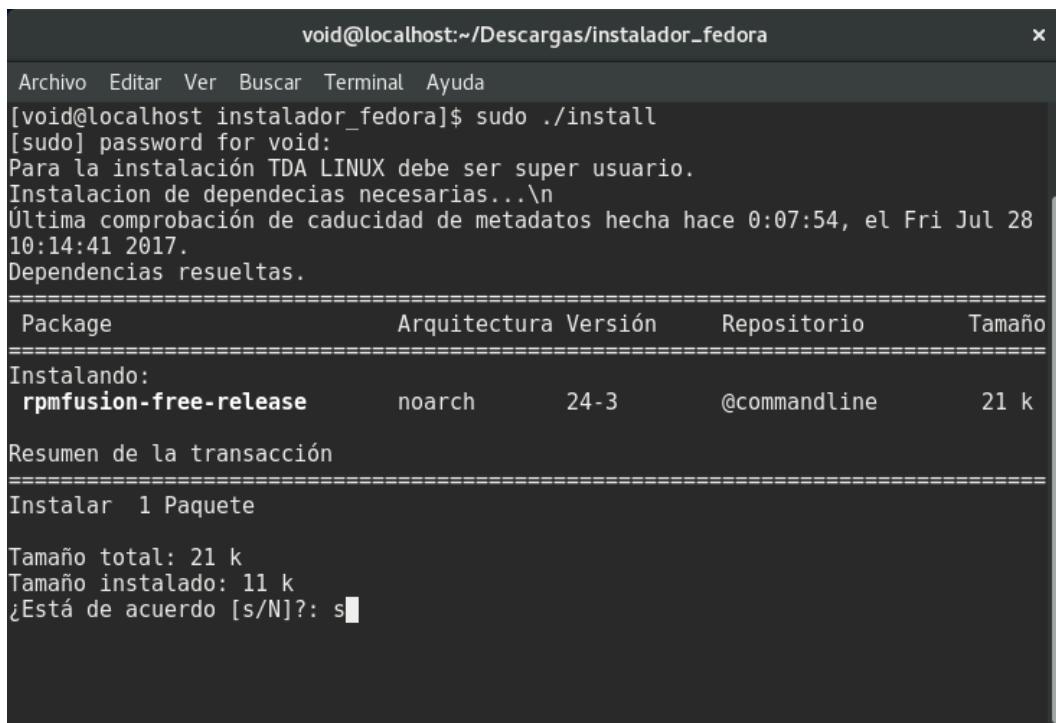
La ejecución del script de instalación se le realizó mediante la terminal usando el comando **sudo ./install** como se muestra en la **Figura 14.1.1** posteriormente se introdujo la contraseña de usuario root y se presionó “ENTER”.



A screenshot of a terminal window titled "void@localhost:~/Descargas/instalador_fedora". The window has a dark theme. The menu bar includes "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The main area shows the command line: [void@localhost instalador_fedora]\$ sudo ./install. Below the command, it says "[sudo] password for void:" followed by a redacted password field.

Figura 14.1.1: Ejecución del comando de instalación del programa.

Para la instalación del reproductor multimedia VLC es necesario importar repositorios de software al sistema, en la **Figura 14.1.2** se muestra el proceso de importación del repositorio **rpmfusion-free-release**. Se introdujo la letra “s” y se presionó “ENTER” para importar el repositorio y continuar con la instalación.



The screenshot shows a terminal window titled "void@localhost:~/Descargas/instalador_fedora". The window contains the following text:

```
Archivo Editar Ver Buscar Terminal Ayuda
[void@localhost instalador_fedora]$ sudo ./install
[sudo] password for void:
Para la instalación TDA LINUX debe ser super usuario.
Instalacion de dependencias necesarias...\n
Última comprobación de caducidad de metadatos hecha hace 0:07:54, el Fri Jul 28
10:14:41 2017.
Dependencias resueltas.
=====
Package           Arquitectura Versión     Repositorio      Tamaño
=====
Instalando:
rpmfusion-free-release    noarch        24-3          @commandline    21 k
Resumen de la transacción
=====
Instalar 1 Paquete
Tamaño total: 21 k
Tamaño instalado: 11 k
¿Está de acuerdo [s/N]?: s
```

Figura 14.1.2: Importar repositorios de software para VLC en el proceso de instalación.

Luego de la importación de los repositorios para VLC se procede a la instalación de las librerías y software necesario para la compilación y posterior ejecución del programa desarrollado, como se muestra en la **Figura 14.1.3** el instalador muestra los paquetes a instalar. Se introdujo la letra “s” y se presionó “ENTER” para la instalación de estos paquetes y para continuar con la instalación.

```

void@localhost:~/Descargas/instalador_fedora

Archivo Editar Ver Buscar Terminal Ayuda
s

x265-libs           x86_64 1.9-1.fc24      rpmfusion-free 574 k
xorg-x11proto-devel noarch 7.7-19.fc24    fedora        556 k
xvidcore            x86_64 1.3.4-2.fc24    rpmfusion-free 287 k
xz-devel             x86_64 5.2.2-2.fc24    fedora        262 k
zlib-devel           x86_64 1.2.8-10.fc24   fedora        60 k
zvbi                 x86_64 0.2.35-1.fc24   fedora        55 k
Actualizando:
cpp                  x86_64 6.3.1-1.fc24   updates       414 k
gcc                  x86_64 6.3.1-1.fc24   updates       9.0 M
gcc-gdb-plugin       x86_64 6.3.1-1.fc24   updates       20 M
libgcc               x86_64 6.3.1-1.fc24   updates       85 k
libgomp              x86_64 6.3.1-1.fc24   updates       89 k
libstdc++             x86_64 6.3.1-1.fc24   updates       191 k
webkitgtk3           x86_64 2.4.11-2.fc24   updates       451 k
Resumen de la transacción
=====
Instalar 149 Paquetes
Actualizar 7 Paquetes

Tamaño total de la descarga: 116 M
¿Está de acuerdo [s/N]?: s

```

Figura 14.1.3: Instalación de librerías y software necesario en el proceso de instalación.

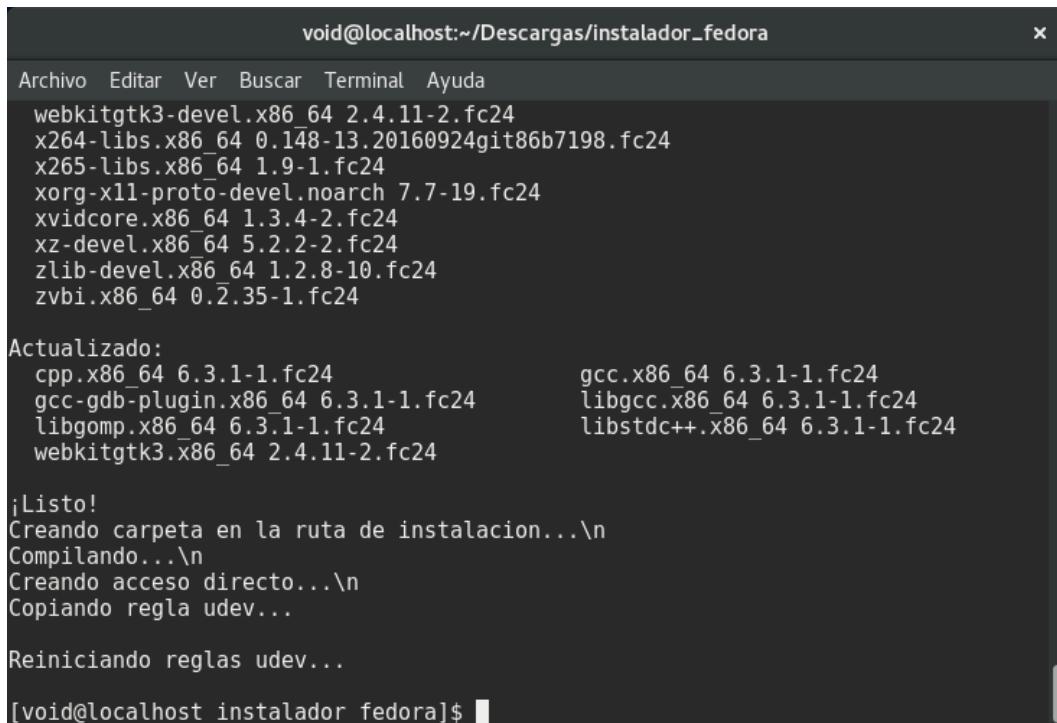
Después de acceder a la instalación de los paquetes necesarios el instalador pide verificar el repositorio importado como método de confianza en el software externo que se instala desde el repositorio externo **rpmfusion-free-release** como se muestra en la **Figura 14.1.4**. Se introdujo la letra “s” y se presionó “ENTER” para verificar el repositorio y continuar con la instalación.

```
void@localhost:~/Descargas/instalador_fedora

Archivo Editar Ver Buscar Terminal Ayuda
(151/156): subunit-devel-1.2.0-4.fc24.x86_64.rpm 51 kB/s | 12 kB    00:00
(152/156): dbus-devel-1.11.2-1.fc24.x86_64.rpm 222 kB/s | 60 kB    00:00
(153/156): liba52-0.7.4-25.fc24.x86_64.rpm 155 kB/s | 44 kB    00:00
(154/156): libicu-devel-56.1-4.fc24.x86_64.rpm 351 kB/s | 750 kB    00:02
(155/156): cpp-6.3.1-1.fc24.x86_64.rpm 325 kB/s | 9.0 MB    00:28
[MIRROR] gcc-6.3.1-1.fc24.x86_64.rpm: Status code: 421 for http://mirror.cedia.org.ec/fedora/updates/24/x86_64/g/gcc-6.3.1-1.fc24.x86_64.rpm
(156/156): gcc-6.3.1-1.fc24.x86_64.rpm 171 kB/s | 20 MB    02:00
[DRPM] libstdc++-6.1.1-2.fc24_6.3.1-1.fc24.x86_64.rpm: hecho
[DRPM] libgomp-6.1.1-2.fc24_6.3.1-1.fc24.x86_64.rpm: hecho
[DRPM] gcc-gdb-plugin-6.1.1-2.fc24_6.3.1-1.fc24.x86_64.rpm: hecho
[DRPM] webkitgtk3-2.4.11-1.fc24_2.4.11-2.fc24.x86_64.rpm: hecho
-----
Total 439 kB/s | 108 MB 04:13
Delta RPMs redujo 115.8 MB de actualizaciones a 108.4 MB (6.1% de ahorro)
advertencia:/var/cache/dnf/rpmfusion-free-updates-1e475027b1818d49/packages/vlc-3.0.0-0.28.git20170622.fc24.x86_64.rpm: EncabezadoV4 RSA/SHA1 Signature, ID de clave b7546f06: NOKEY
Importando llave GPG 0xB7546F06:
ID usuario: "RPM Fusion free repository for Fedora (24) <rpmfusion-buildsys@lists.rpmfusion.org>"
Huella: 55E7 903B 6087 98E4 EC78 64CD 9F63 8721 B754 6F06
Desde: /etc/pki/rpm-gpg/RPM-GPG-KEY-rpmfusion-free-fedora-24
¿Está de acuerdo [s/N]?: s
```

Figura 14.1.4: Verificar huella del repositorio importado en el proceso de instalación.

Finalmente se mostraron los mensajes de las acciones de: Creando carpeta en la ruta de instalación, Compilando, Creando acceso directo, Creando regla udev y Reiniciando regla udev, como se muestra en la **Figura 14.1.5.**



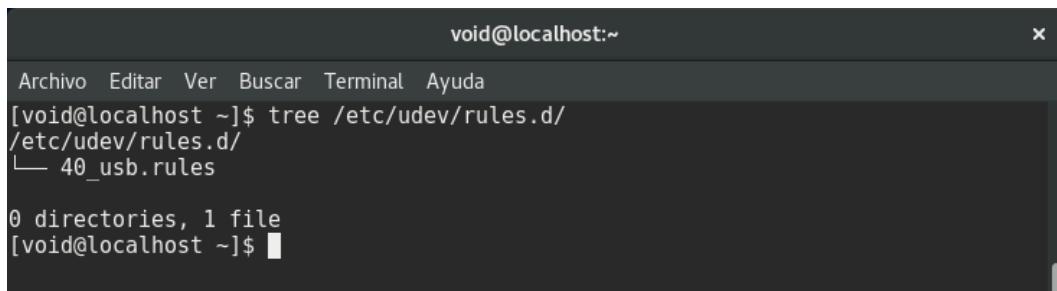
```
void@localhost:~/Descargas/instalador_fedora
Archivo Editar Ver Buscar Terminal Ayuda
webkitgtk3-devel.x86_64 2.4.11-2.fc24
x264-libs.x86_64 0.148-13.20160924git86b7198.fc24
x265-libs.x86_64 1.9-1.fc24
xorg-x11proto-devel.noarch 7.7-19.fc24
xvidcore.x86_64 1.3.4-2.fc24
xz-devel.x86_64 5.2.2-2.fc24
zlib-devel.x86_64 1.2.8-10.fc24
zvbi.x86_64 0.2.35-1.fc24

Actualizado:
cpp.x86_64 6.3.1-1.fc24          gcc.x86_64 6.3.1-1.fc24
gcc-gdb-plugin.x86_64 6.3.1-1.fc24 libgcc.x86_64 6.3.1-1.fc24
libgomp.x86_64 6.3.1-1.fc24      libstdc++.x86_64 6.3.1-1.fc24
webkitgtk3.x86_64 2.4.11-2.fc24

¡Listo!
Creando carpeta en la ruta de instalacion...\n
Compilando...\n
Creando acceso directo...\n
Copiando regla udev...
Reiniciando reglas udev...
[void@localhost instalador_fedora]$
```

Figura 14.1.5: Creación de directorios, compilación y reglas Udev.

En la **Figura 14.1.6** se observa usando el comando **tree /etc/udev/rules.d** que fue añadida la nueva regla llamada **40_usb.rules** para el dispositivo USB de TDA.

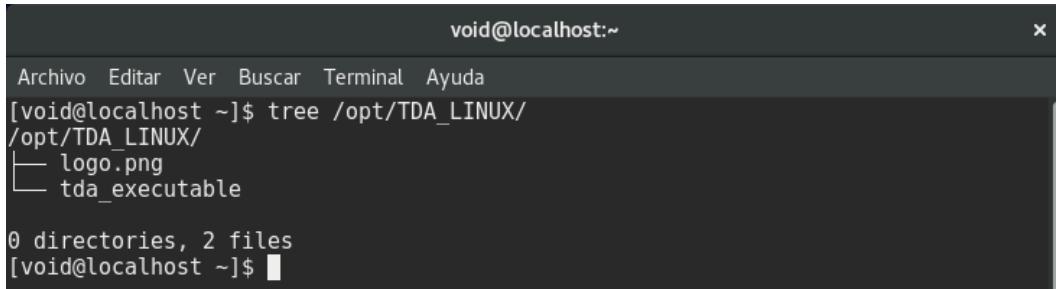


```
void@localhost:~
Archivo Editar Ver Buscar Terminal Ayuda
[void@localhost ~]$ tree /etc/udev/rules.d/
/etc/udev/rules.d/
└── 40_usb.rules

0 directories, 1 file
[void@localhost ~]$
```

Figura 14.1.6: Estructura de archivos de reglas udev.

En la **Figura 14.1.7** se observa usando el comando **tree** /opt/TDA_LINUX que fue creado el directorio que contiene el ejecutable de la aplicación desarrollada y el logo del lanzador de la aplicación.



```
void@localhost:~$ Archivo Editar Ver Buscar Terminal Ayuda [void@localhost ~]$ tree /opt/TDA_LINUX/ /opt/TDA_LINUX/ └── logo.png └── tda_executable 0 directories, 2 files [void@localhost ~]$
```

A terminal window titled "void@localhost:~". The window shows the output of the "tree" command run on the directory "/opt/TDA_LINUX". The output shows a single directory named "TDA_LINUX" containing two files: "logo.png" and "tda_executable". The status line at the bottom indicates there are 0 directories and 2 files.

Figura 14.1.7: Estructura de archivos del directorio de instalación.

Ya finalizado el proceso de instalación fue creado un lanzador a la aplicación como se muestra en la **Figura 14.1.8** con el nombre **TDA_LINUX**.

Al lanzar la aplicación se abrió la aplicación desarrollada como se muestra en la Figura **14.1.9**.

El proceso de instalación engloba a todas las rutinas desarrolladas en este trabajo de grado menos la rutina de desinstalación, el script de instalación compila a las rutinas del software para crear el ejecutable de la aplicación, crea el lanzador de la aplicación y añade la regla USB para el dispositivo de TDA, durante este proceso de compilación no se evidenciaron errores por lo cual el proceso de compilación fue exitoso también pudo comprobarse la inserción de la nueva regla udev, la creación del lanzador y la correcta ejecución del software.

En las imágenes mostradas se observa que se logró satisfactoriamente el proceso de ejecución e instalación del software controlador USB y reproductor multimedia desarrollado bajo el sistema operativo **GNU/Linux Fedora 24**.

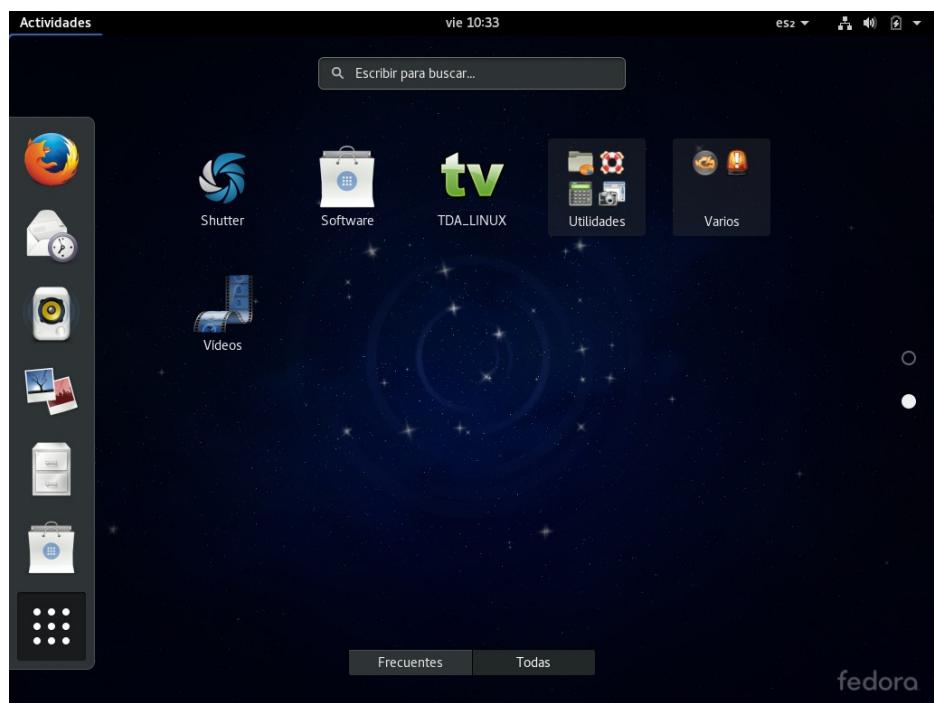


Figura 14.1.8: Lanzador de la aplicación.

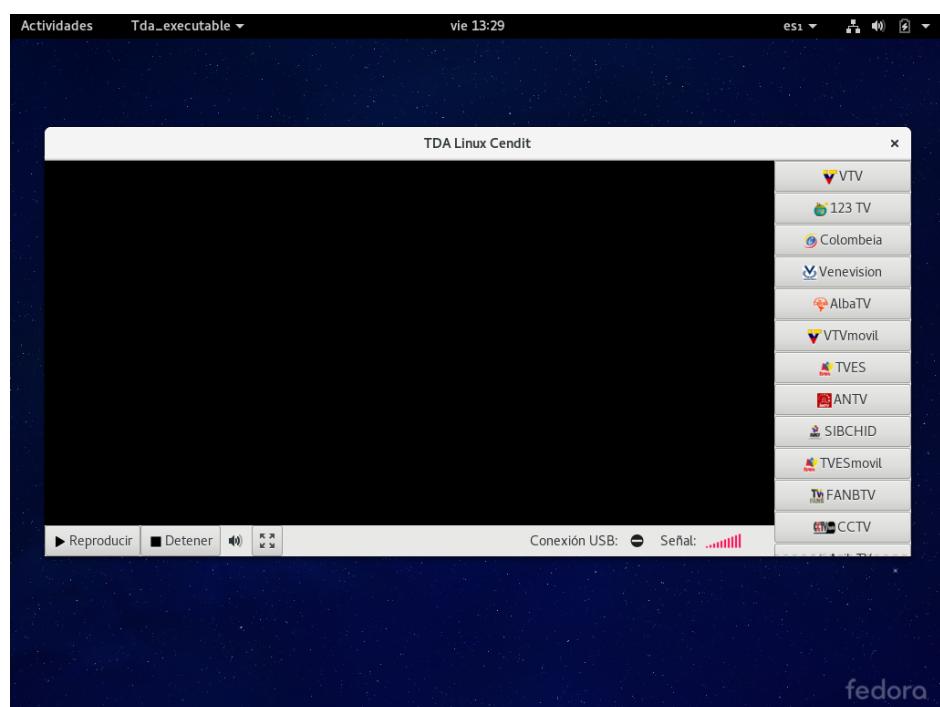


Figura 14.1.9: Ejecución de la aplicación.

14.2 Montaje físico de los equipos.

Para las pruebas del software desarrollado se utilizó el prototipo de pruebas de USB de TDA [6], el siguiente diagrama de conexiones mostrado en la **Figura 14.2.1** muestra las conexiones y equipos utilizados para la realización de las pruebas.

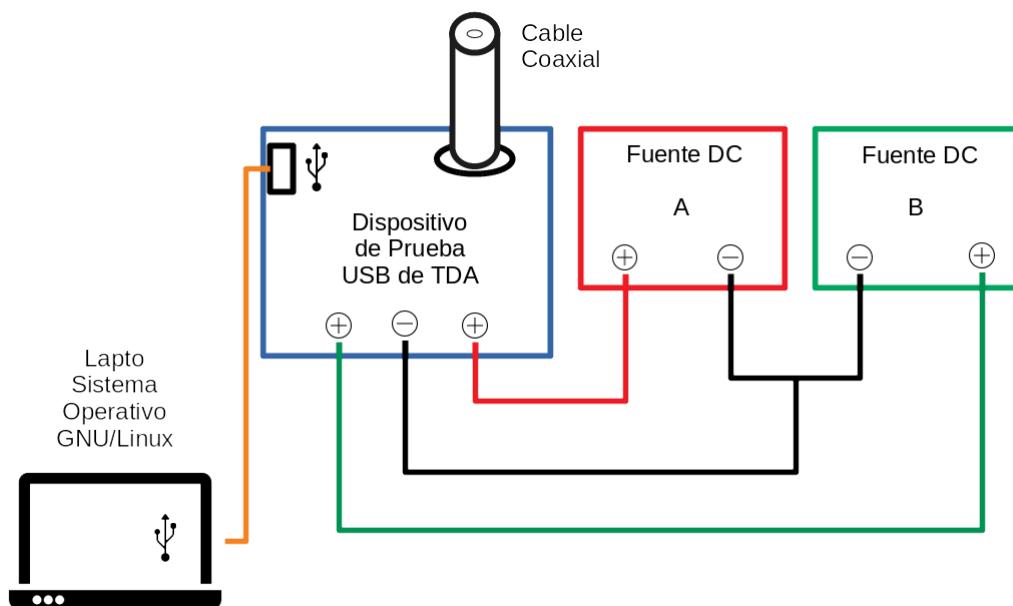


Figura 14.2.1: Diagrama de conexiones montaje físico.

La fuente DC A fue calibrada a (5.1 ± 0.1) V y la fuente DC B fue calibrada a (1.2 ± 0.1) V, estas fuentes entregan el voltaje necesario al dispositivo de prueba para su funcionamiento.

El dispositivo de prueba recibe la señal de TDA a través de un cable coaxial que es conectado al terminal del dispositivo Tuner NIM de Samsung y el puerto USB de la placa de desarrollo es conectado a la PC.

A continuación se muestra el montaje en físico realizado en la **Figura 14.2.2**.

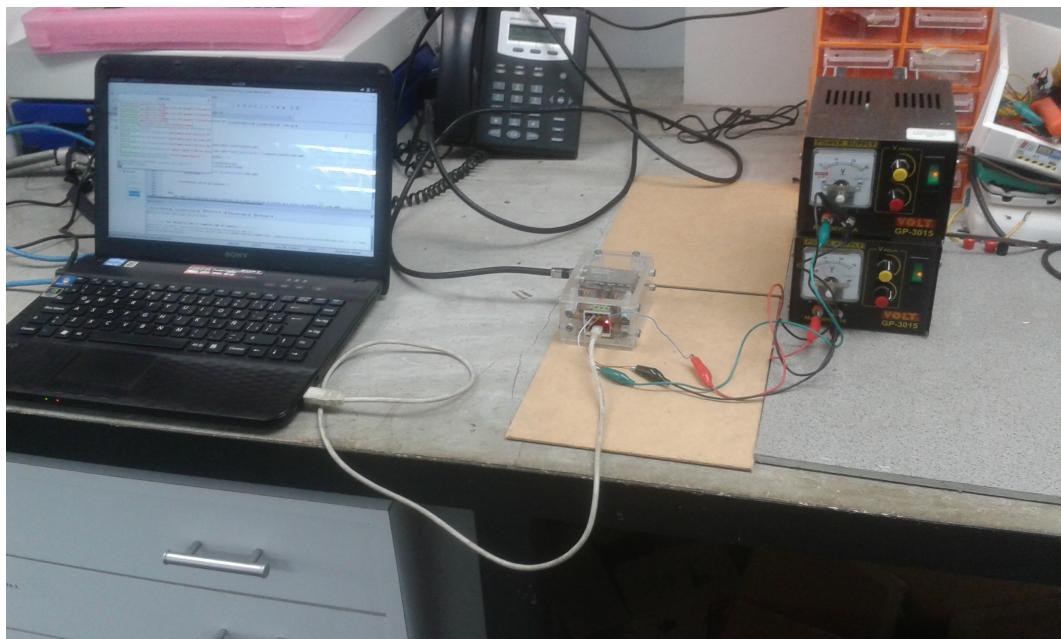


Figura 14.2.2: Montaje en físico de las pruebas.

En la **Figura 14.2.3** se muestra detalladamente la conexión física del dispositivo de prueba USB de TDA.

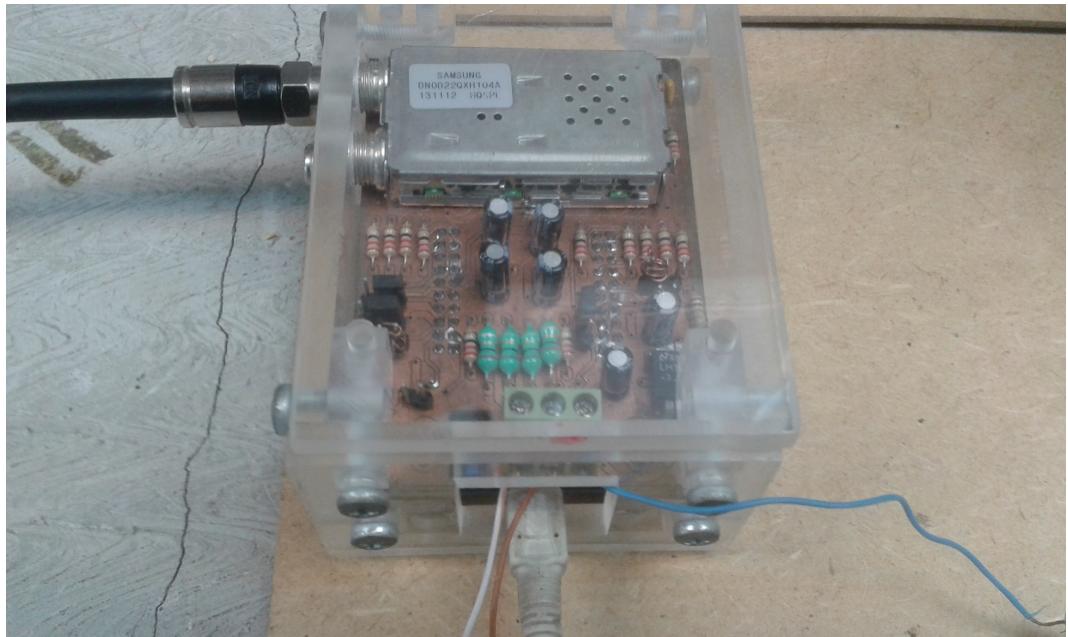


Figura 14.2.3: Conexión en físico del dispositivo USB de TDA.

Lista de equipos utilizados para la realización de las pruebas:

- Lpto Siragon Modelo MN-50.
- Lpto Sony Vaio Modelo VPCCA.
- 2 Fuentes DC Modelo GP-3015.
- 3 cables tipo 16 con conectores caimanes en ambos extremos.
- Cable Coaxial de 1 metro de largo conectores hembras de tipo f.
- Cable de conexión microusb.
- Multímetro Fluke Modelo 179, para calibrar la salida deseada de las fuentes DC.

14.3 Pruebas de la aplicación desarrollada.

Al ejecutar la aplicación esta inició el proceso de detección en caliente para detectar al dispositivo USB de Cypress al ser conectado o desconectado del puerto USB de la PC para luego continuar con las operaciones de recepción y reproducción del contenido multimedia.

Luego de haber insertado el dispositivo de USB a la PC el ícono en la barra de indicadores con la etiqueta “Conexión USB” cambio para mostrar un signo de más indicando que el dispositivo fue reconocido y esta listo para ser usado, como se muestra en la **Figura 14.3.1**.



Figura 14.3.1: Dispositivo USB de TDA conectado.

Luego de conectar el dispositivo se procedió a dar click en el los programas **123 TV**, **SIBCHID** y **AvilaTV Móvil** para su sintonización y reproducción como se muestra en las **Figuras 14.3.1, 14.3.2 y 14.3.3**.

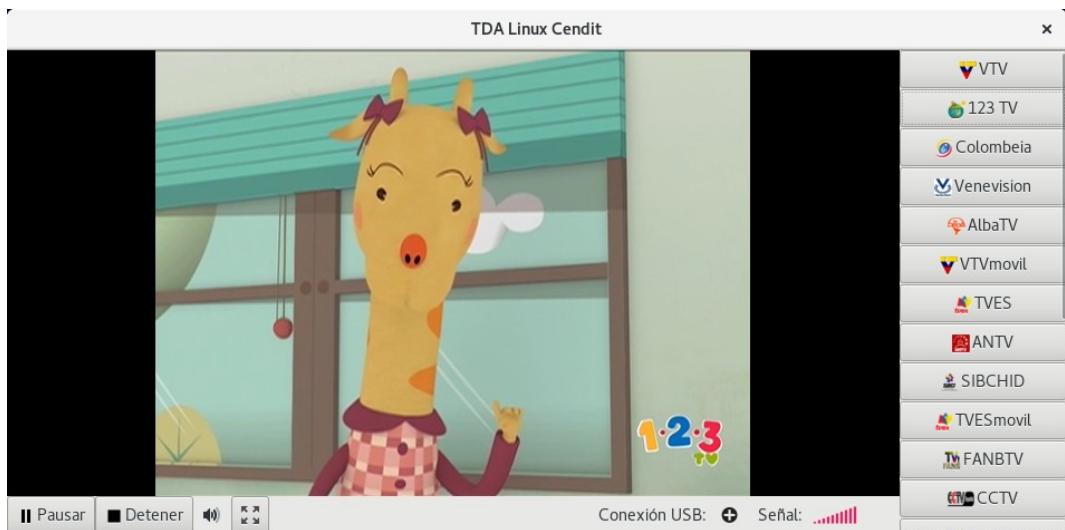


Figura 14.3.2: Prueba de sintonización y reproducción del programa 123 TV.

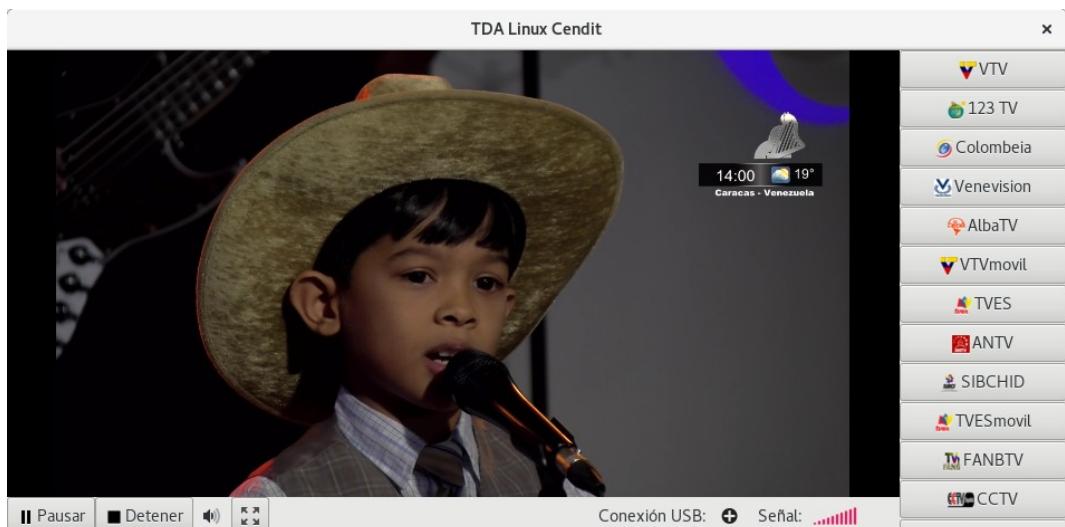


Figura 14.3.3: Prueba de sintonización y reproducción del programa SIBCHID.



Figura 14.3.4: Prueba de sintonización y reproducción del programa Avila TV Móvil.

A continuación se muestran imágenes de la aplicación sintonizando y reproduciendo el programa de TDA Meridiano en el entorno de escritorio, como se muestra en la **Figura 14.3.5** luego la aplicación maximizada en la **Figura 14.3.6** y la aplicación en pantalla completa en la **Figura 14.3.7**.

En la **Figura 14.3.8** se observa la ejecución del control de volumen de la al momento de sintonizar el programa CCTV.

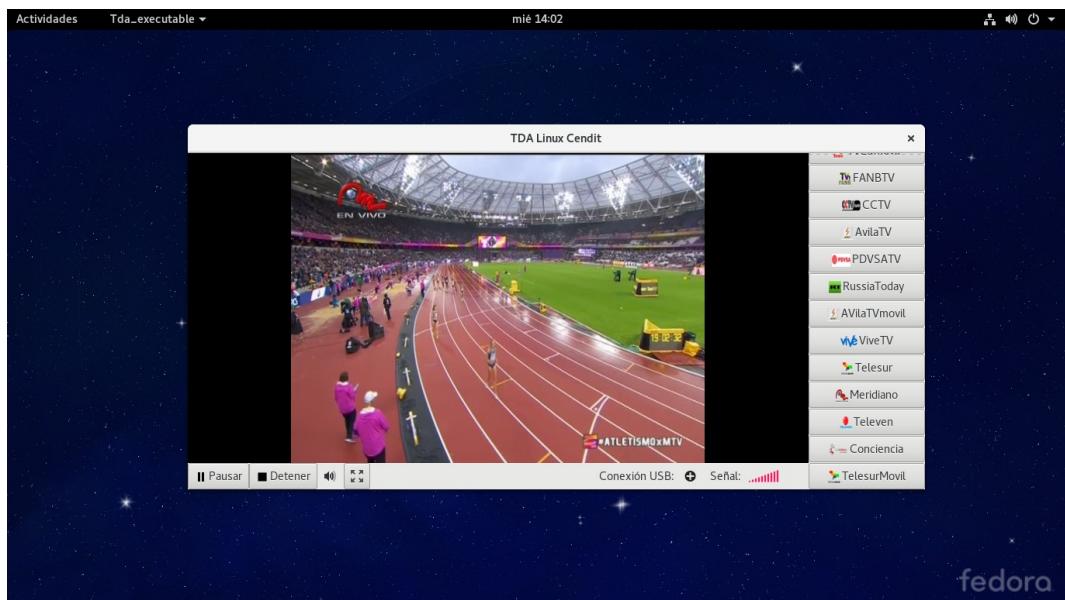


Figura 14.3.5: Prueba de sintonización y reproducción del programa Meridiano, vista del escritorio.



Figura 14.3.6: Prueba de sintonización y reproducción del programa Meridiano, aplicación maximizada.



Figura 14.3.7: Prueba de sintonización y reproducción del programa Meridiano, modo pantalla completa.



Figura 14.3.8: Prueba de sintonización y reproducción del programa CCTV, control de volumen.

En la imagen de la **Figura 14.3.9** se muestra el montaje en físico conjuntamente con la aplicación desarrollada ejecutándose y reproduciendo el programa 123 TV de TDA.

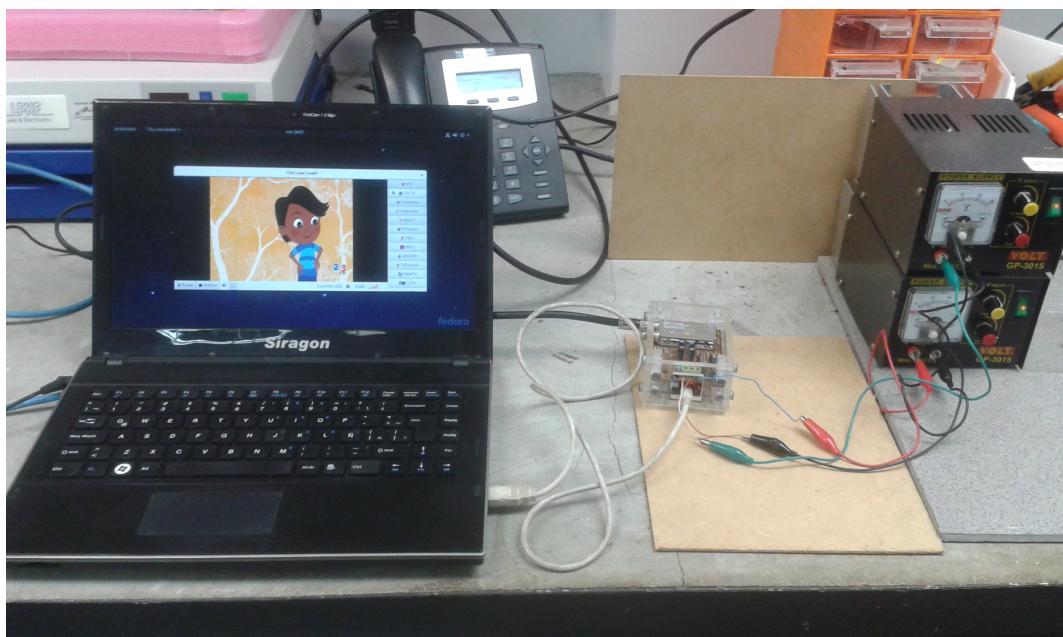


Figura 14.3.9: Montaje físico prueba de sintonización y reproducción del programa 123 TV.

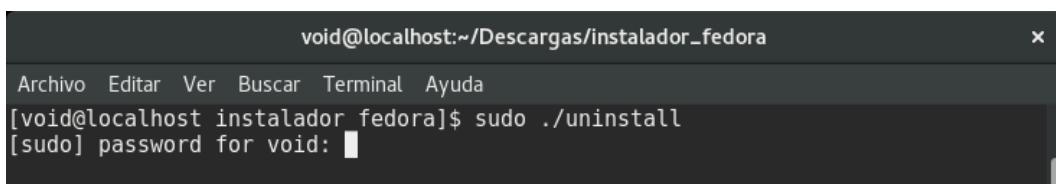
En las imágenes mostradas se evidencia que las rutinas desarrolladas fueron capaces de realizar la detección en caliente del dispositivo a través del puerto USB, realizar acciones de control sobre el dispositivo USB de TDA a la hora de inicializar el demodulador y sintonizador del dispositivo Samsung, ejecutar procesos de sintonización de las frecuencias deseadas de grupos de programas, el manejo y control de los buses de datos al realizar la transferencia de datos del contenido multimedia y transferencias de control, la reproducción y selección de contenido multimedia específico pertenecientes a los paquetes de

frecuencias de la TDA venezolana además de acciones de control sobre el contenido multimedia.

Un inconveniente fue la obtención de la intensidad de la señal la cual no pudo obtenerse utilizando la función de la API de Samsung diseñada para tal proceso, aun así se decidió dejar la imagen de la intensidad de la señal en la aplicación desarrollada para su uso cuando el defecto con la aplicación de la API de Samsung se ha resuelto.

14.4 Ejecución del script de desinstalación del programa.

La ejecución del script de desinstalación se le realizó mediante la terminal usando el comando **sudo ./uninstall** como se muestra en la **Figura 14.4.1** posteriormente se introdujo la contraseña de usuario root y se presionó “ENTER”.

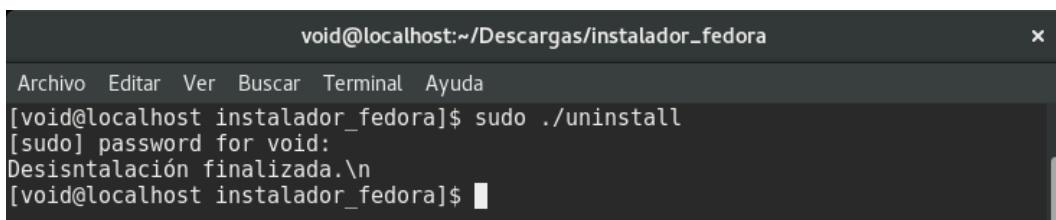


```
void@localhost:~/Descargas/instalador_fedora
Archivo Editar Ver Buscar Terminal Ayuda
[void@localhost instalador_fedora]$ sudo ./uninstall
[sudo] password for void:
```

A screenshot of a terminal window titled "void@localhost:~/Descargas/instalador_fedora". The window has a dark background with white text. It shows a menu bar with options: Archivo, Editar, Ver, Buscar, Terminal, Ayuda. Below the menu is a command line: [void@localhost instalador_fedora]\$ sudo ./uninstall. A password prompt follows: [sudo] password for void: with a cursor at the end of the line.

Figura 14.4.1: Ejecución del comando de desinstalación del programa.

Luego de finalizada la desinstalación se mostró el siguiente mensaje de la **Figura 14.4.2**.

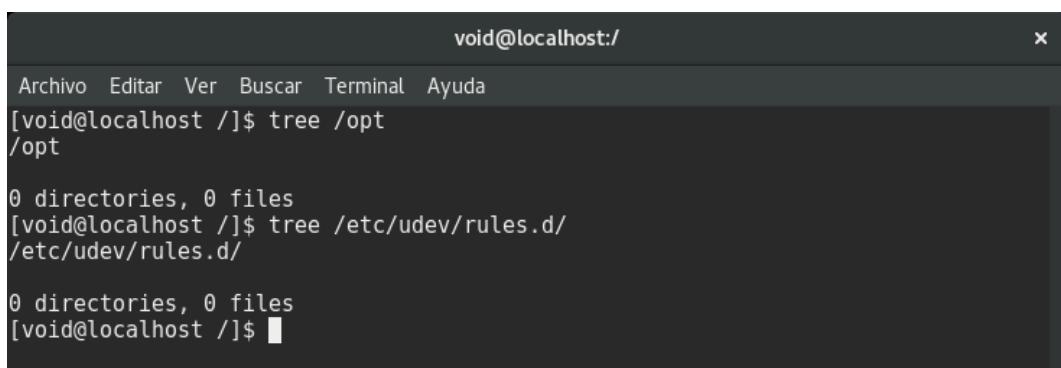


```
void@localhost:~/Descargas/instalador_fedora
Archivo Editar Ver Buscar Terminal Ayuda
[void@localhost instalador_fedora]$ sudo ./uninstall
[sudo] password for void:
Desisntalación finalizada.\n
[void@localhost instalador_fedora]$
```

A screenshot of a terminal window titled "void@localhost:~/Descargas/instalador_fedora". The window has a dark background with white text. It shows a menu bar with options: Archivo, Editar, Ver, Buscar, Terminal, Ayuda. Below the menu is a command line: [void@localhost instalador_fedora]\$ sudo ./uninstall. The output of the command is displayed below: Desisntalación finalizada.\n with a cursor at the end of the line.

Figura 14.4.2: Mensaje de desinstalación del programa.

La desinstalación del programa se verificó la eliminación del directorio de programa mediante el comando **tree /opt** y mediante el comando **tree /etc/udev/rules.d/** la eliminación de la regla udev para el dispositivo USB de TDA como se muestra en la **Figura 14.4.3.**

A screenshot of a terminal window titled "void@localhost:/". The window has a dark background and light-colored text. At the top, there's a menu bar with options: Archivo, Editar, Ver, Buscar, Terminal, Ayuda. Below the menu, the command prompt is "[void@localhost /]\$. The user runs two "tree" commands. The first command, "tree /opt", shows the directory structure of "/opt" which is empty, indicated by "0 directories, 0 files". The second command, "tree /etc/udev/rules.d/", also shows an empty directory structure with "0 directories, 0 files".

```
void@localhost:/
Archivo Editar Ver Buscar Terminal Ayuda
[void@localhost /]$ tree /opt
/opt

0 directories, 0 files
[void@localhost /]$ tree /etc/udev/rules.d/
/etc/udev/rules.d/

0 directories, 0 files
[void@localhost /]$
```

Figura 14.4.3: Verificación de desinstalación.

Al finalizar el script de desinstalación se comprobó la completa eliminación del software desarrollado del sistema operativo GNU/Linux Fedora 24 en el cual fue instalado como se muestra en la **Figura 14.3.3.**

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

GNU/Linux es un sistema operativo que cuenta con él apoya de una cantidad enorme de desarrolladores, el sistema operativo cuenta con miles de repositorios de programas y librerías que facilitan las tareas de desarrollo de software sin verse el desarrollador en la necesidad de empezar desde de cero, el apoyo de la comunidad alrededor de este sistema operativo es muy bueno siempre que surja una duda o dificultad puedes dirigirte directamente a la comunidad preguntar o pedir soporte sobre un tema, es software libre de costos y es accesible para todo aquel que quiera descargarlo. Uno de los repositorios usados para el manejo de dispositivos USB fue la librería Libusb que permite una abstracción a nivel de software para el manejo de dispositivos USB lo cual fue de gran ayuda para realizar un controlador USB en el espacio de usuario para dispositivo de TDA, también la librería Libvlc permitió la creación de rutinas para el manejo de la información multimedia usando el nucleo del reproductor VLC y la interfaz gráfica fue creada en base a las librerías de GTK, todas las herramientas usadas fueron creadas y son mantenidas por un gran número de usuarios que aportan su tiempo y conocimiento en estos proyectos, en este trabajo de grado se juntaron piezas de software de distintos proyectos con la finalidad de hacerlas trabajar en conjunto para lograr los objetivos planteados.

El software desarrollado en este trabajo de grado busca brindar una alternativa al software privativo y dar acceso libre a las tecnologías de la información basándose en la Ley de Infogobierno del Estado Venezolano, la Fundación CENDIT siendo un ente público e impulsor de este proyecto debe velar porque la ley de Infogobierno se cumpla y siempre optar en lo

que sea posible a soluciones de software libre por lo cual este trabajo de grado se llevó a la realización.

En cuanto a las recomendaciones, existen varias recomendaciones que hacer sobre el software desarrollado, las cuales se listan a continuación:

- En cuanto a la obtención de la intensidad de la señal debería contarse con soporte de parte de Samsung sobre el manejo de la API del Tuner NIM ya que hay una falta de información sobre a cuales registros apunta la función a la hora de obtener datos.
- Implementar una rutina de instalación más amistosa para el usuario, podría hacerse mediante una aplicación con interfaz gráfica o un empaquetado de software como RPM p DEB.
- Una rutina de sintonización automática de programas de TDA que logre listar los programas de TDA disponibles para su visualización sin necesidad de modificar el software cada vez que un nuevo programa es agregado.
- Otras funciones extras como capacidad de grabar el contenido en reproducción, adelantar y retroceder el contenido multimedia son funciones que el usuario puede encontrar provechosas.

REFERENCIAS BIBLIOGRÁFICAS

- [1] J. Manjares (2015), *Diseño y construcción de prototipo industrializable de dispositivo portátil de sintonización full-seg de TDA, compatible con la norma ISDB-Tb, para recepción en un computador a través del puerto USB*, CENDIT, Caracas.
- [2] N. Pisciotta, C. Liendo, R. Lauro, *TRANSMISIÓN DE TELEVISIÓN DIGITAL TERRESTRE EN LA NO RMA ISDB-Tb*, 2007.
- [3] *Televisión digital terrestre en Venezuela* (2017). En Wikipedia. Recuperado el 05 de Mayo de 2017 de https://es.wikipedia.org/wiki/Televisi%C3%B3n_digital_terrestre_en_Venezuela
- [4] Axelson, *USB Complete The Developer Guide*, 2015.
- [5] J. Axelson, *Serial Port Complete COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems*, 2007.
- [6] *USB in a NutShell, Making sense of the USB standard* (2010). Recuperado el 05 de Mayo de 2017 de <http://www.beyondlogic.org/usbnutshell/usb1.shtml>
- [7] J. Corbet, A. Rubini, G. Kroah-Hartman, *Linux Device Drivers*, 2007.

[8] D. Córdoba (2014). GNU/Linux: Arquitectura básica del sistema. España. Recuperado de <https://juncotic.com/gnulinux-arquitectura-basica-del-sistema/>

[9] *GNU/Linux* (2017). En Wikipedia. Recuperado el 15 de Mayo de 2017 de <https://es.wikipedia.org/wiki/GNU/Linux>

[10] KJK (2011). *S/W Porting Guide, SDB-T NIM* : *DNOD22QXV104A*, 2011.

[11] *libusb-1.0 API Reference* (2015). Recuperado el 07 de Febrero de 2017 de <http://libusb.sourceforge.net/api-1.0/>

[12] A. Ott, (2014). USB and the Real World. Estados Unidos.: Signal11. Recuperado de <http://www.signal11.us/>

[13] *VLC 3.0.0-git* (2014). Recuperado el 11 de Marzo de 2017 de https://www.videolan.org/developers/vlc/doc/doxygen/html/group__libvlc.html

[14] K .Opasiak, (2015). Understand USB (in Linux). Polonia: linuxfoundation. Recuperado de <http://events.linuxfoundation.org/>

[15] B. Kernighan, D. Ritchie, *The C programming Language*, 1988.

[16] Doxygen(2017) Recuperado el 15 de Marzo de 2017 de <http://www.stack.nl/~dimitri/doxygen/>

[17] Asamblea Nacional de la República Bolivariana de Venezuela. (10 de Octubre de 2013). Artículo 34 [Título I]. *Ley de Infogobierno*. [Ley 402-14 de 2013]. Gaceta: 40.217

[18] Asamblea Nacional de la República Bolivariana de Venezuela. (10 de Octubre de 2013). Artículo 35 [Título I]. *Ley de Infogobierno*. [Ley 402-14 de 2013]. Gaceta: 40.217

ANEXOS

Repositorio del código fuente
Repositorio de la documentación
Repositorio del manual de uso