DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA Y TELECOMUNICACIONES

TEMA: DESARROLLO DE UNA APLICACIÓN PARA EL PROCESO DE DECODIFICACIÓN DE INFORMACIÓN DE IMÁGENES ENCRIPTADAS UTILIZANDO EL MIDDLEWARE GINGA NCL Y PYTHON CON HARDWARE DE BAJO COSTO (ISDB-Tb).

AUTOR: GUAYASAMÍN AYALA, JHOSELYN ESTEFANÍA

DIRECTOR: ING. ACOSTA BUENAÑO, FREDDY ROBERTO

SANGOLQUÍ

2019

CERTIFICACIÓN

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICACIÓN

Certifico que el trabajo de titulación, "***DESARROLLO DE UNA APLICACIÓN PARA EL PROCESO DE DECODIFICACIÓN DE INFORMACIÓN DE IMÁGENES ENCRIPTADAS UTILIZANDO EL MIDDLEWARE GINGA NCL Y PYTHON CON HARDWARE DE BAJO COSTO (ISDB-Tb)***" realizado por la Srta. **JHOSELYN ESTEFANÍA GUAYASAMÍN AYALA**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo que cumple con los requisitos teóricos, científicos técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, por lo tanto me permito acreditar y autorizar a la Srta. **JHOSELYN ESTEFANÍA GUAYASAMÍN AYALA** para que lo sustente públicamente.

Sangolquí, 17 de julio del 2019

Ing. Freddy Roberto Acosta Buenaño
DIRECTOR

AUTORÍA DE RESPONSABILIDAD

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORÍA DE RESPONSABILIDAD

Yo **JHOSELYN ESTEFANÍA GUAYASAMÍN AYALA**, con cédula de identidad N° 172113899-6, declaro que este trabajo de titulación ***"DESARROLLO DE UNA APLICACIÓN DE DECODIFICACIÓN DE INFORMACIÓN DE IMÁGENES ENCRIPTADAS UTILIZANDO EL MIDDLEWARE GINGA NCL Y PYTHON CON HARDWARE DE BAJO COSTO (ISDB-Tb)"*** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Sangolquí, 17 de julio del 2019

Jhoselyn Estefanía Guayasamín Ayala
CC. 1721138996

AUTORIZACIÓN

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN

Yo, **JHOSELYN ESTEFANÍA GUAYASAMÍN AYALA**, autorizo a la Universidad de las Fuerzas Armadas ESPE, publicar en la biblioteca Virtual de la institución el presente trabajo de titulación ***"DESARROLLO DE UNA APLICACIÓN DE DECODIFICACIÓN DE INFORMACIÓN DE IMÁGENES ENCRIPTADAS UTILIZANDO EL MIDDLEWARE GINGA NCL Y PYTHON CON HARDWARE DE BAJO COSTO (ISDB-Tb)"*** cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Sangolquí, 17 de julio del 2019

Jhoselyn Estefanía Guayasamín Ayala
CC. 1721138996

DEDICATORIA

El presente trabajo investigativo lo dedico primero a Dios, por ser el inspirador y darme fuerza para continuar en este proceso de obtener uno de los anhelos más deseados.

A mis padres César y Rocío, quienes con su amor, paciencia y esfuerzo me han permitido llegar a cumplir hoy un sueño más, gracias por sus consejos, por inculcar en mí el ejemplo de esfuerzo y valentía, de no temer a las adversidades. Ha sido el orgullo y el privilegio ser su hija, son los mejores padres.

A mi hermana Vanessa y mi sobrina Daniela, por su cariño y apoyo incondicional durante todo este proceso, por estar conmigo en todo momento, ya que con sus consejos y palabras de aliento hicieron de mí una mejor persona y de una u otra forma me acompañan en todos mis sueños y metas.

Jhoselyn Estefanía Guayasamín Ayala

AGRADECIMIENTO

Me faltarían paginas para agradecer a las personas que se han involucrado en la realización de este trabajo, sin embargo, merecen un reconocimiento especial mi madre y mi padre que con su esfuerzo y dedicación me ayudaron a culminar mi carrera universitaria y me dieron el apoyo suficiente para no decaer cuanto todo parecía complicado e imposible.

Así mismo agradezco infinitamente a mi hermana y mi sobrina que con sus palabras me hacían sentir orgullosa de lo que soy y de lo que les puedo enseñar. Ojalá algún día yo me convierta en su fuerza para que puedan seguir avanzando en su camino.

Así también quiero agradecer a mi familia abuelitos, tíos, primos por haber sido mi apoyo a lo largo de toda mi carrera universitaria y a lo largo de mi vida.

De igual forma agradezco a mi director de tesis Ing. Freddy Acosta, quien ha sabido guiarme como persona y como docente, gracias a sus consejos y conocimiento que me ha impartido hoy puedo culminar mi carrera. A los profesores que me han visto crecer como persona y estudiante, por sus conocimientos, hoy puedo sentirme dichosa y contenta.

A la Universidad de las Fuerzas Armadas ESPE, por ser la sede de todo el conocimiento adquirido en estos años.

Finalmente quiero agradecer a mis amigas y amigos, por apoyarme cuando más las necesité, por extender su mano en momentos difíciles y por el amor brindado cada día, siempre les llevo en mi corazón.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En el presente trabajo de titulación se realizó la decodificación de imágenes encriptadas utilizando el lenguaje de programación Python implementando la técnica de transformación de color casi reversible (NRCT), la cual permite que la imagen secreta tome un color semejante al de la imagen portadora haciendo uso de la intensidad media por componente de color (R, G y B) y desviaciones estándar. El proceso de decodificación se ejecuta sobre una tarjeta Raspberry Pi 3B, misma en la que se implementó un STB con hardware de bajo costo y mediante el uso de un sintonizador se consiguió reproducir canales de televisión digital en alta definición (HD), definición estándar (SD) y One-seg; en el proceso de decodificación se planteó tres escenarios para recuperación de imágenes secretas en los que se varía el tamaño de bloque en que se subdivide la imagen, estos escenarios de análisis corresponden a bloques con dimensiones de 8x8, 16x16 y 32x32, las imágenes recuperadas para cada escenario propuesto se presentan en la aplicación interactiva desarrollada con Ginga NCL. Los métodos de evaluación para comparar la imagen recuperada con respecto a la imagen original utilizados son el MOS como técnica subjetiva, implementada en un grupo determinado de personas el cual permite obtener el grado de aceptación, como técnicas objetivas se implementó el RMSE que permite obtener la magnitud de error, MSSIM que se basa en la estructura de la imagen para determinar el nivel de similitud y PSNR que establece la relación de ruido que existe entre las imágenes.

Palabras Clave:

- **PYTHON**
- **DECODIFICACIÓN**
- **GINGA NCL**
- **NRCT**
- **RASPBERRY PI**

ABSTRACT

In the present titration work, the decoding of encrypted images was carried out using the Python programming language, implementing the near-reversible color transformation technique (NRCT), which allows the secret image to take on a color similar to that of the carrier image, making use of the average intensity per color component (R, G and B) and standard deviations. The decoding process is executed on a Raspberry Pi 3B card, which implemented a STB with low-cost hardware and through the use of a tuner was able to reproduce digital television channels in high definition (HD), standard definition (SD) and One-seg; in the decoding process three scenarios were proposed for recovery of secret images in which the size of the block in which the image is subdivided varies, these analysis scenarios correspond to blocks with dimensions of 8x8, 16x16 and 32x32, the images recovered for each proposed scenario are presented in the interactive application developed with Ginga NCL. The evaluation methods to compare the recovered image with respect to the original image used are the MOS as a subjective technique, implemented in a certain group of people which allows to obtain the degree of acceptance, as objective techniques was implemented the RMSE that allows to obtain the magnitude of error, MSSIM that is based on the structure of the image to determine the level of similarity and PSNR that establishes the relation of noise that exists between the images.

Keywords:

- **PYTHON**
- **DECODIFICATION**
- **GINGA NCL**
- **NRCT**
- **RASPBERRY PI**

CAPÍTULO I

1 INTRODUCCIÓN

1.1 Antecedentes

La Televisión Digital Terrestre es una nueva forma de transmitir las señales de Televisión abierta o gratuita con ventajas, como mayor calidad de imagen, video, sonido, e interactividad con el televidente (MINTEL, s.f.). En Ecuador se adoptó el estándar ISDB-Tb (*Integrated Services of Digital Broadcasting – Terrestrial*) con adaptaciones brasileñas para Televisión Digital Terrestre a partir del año 2010, con lo que se inició un proceso de transición para el cambio de tecnología y posterior apagón analógico (TDT Ecuador, s.f.).

El apagón analógico es parte del Plan Maestro de Transición a la Televisión Digital Terrestre en el Ecuador, el mismo que para llevar acabo se planteó alcanzar un porcentaje de que al menos el 90% de los hogares tengan un televisor con el sintonizador de estándar ISDB-T Internacional o un decodificador (Set Top Box STB) para televisión digital terrestre (TDT Ecuador, 2017). Recientemente se realizaron modificaciones en este plan en el cual se plateo llevar a cabo el cese de señales de televisión analógica mediante 4 fases que se cumplirán de manera planificada hasta el año 2023 en el que se concluirá con la ejecución de este plan (Ministerio de Telecomunicaciones y de la Sociedad de la Información, 2018).

Actualmente en Ecuador se realiza la transmisión de señal digital en ciudades como: Quito, Guayaquil, Cuenca, Santo Domingo, Manta, Latacunga y Ambato, debido a que, de 577 estaciones

de televisión, 30 de ellas cuentan con una concesión temporal para brindar el servicio de Televisión Digital Terrestre (TDT Ecuador, s.f.).

La Televisión Digital Terrestre presenta un nuevo concepto de televisión, con el que se brinda mayores beneficios a los espectadores como la posibilidad de recibir más canales en el receptor, ya que se aprovecha el espectro radioeléctrico de mejor forma, también permite interactuar debido a que a medida que avanza la tecnología, los usuarios demandan mayor calidad, movilidad, interactividad, portabilidad, que al final el producto viene hacer un multiservicio (Ministerio de Telecomunicaciones y de la Sociedad de la Información, s.f.).

Con la transmisión de señal digital se generan nuevas opciones de desarrollo de programas y aplicaciones como Telegobierno, Telesalud, Teleducación, programación educativa, cultural, con la que el espectador podrá escoger qué desea observar (Ministerio de Telecomunicaciones y de la Sociedad de la Información, s.f.). Según el Plan Maestro de Transición a la Televisión Digital Terrestre en el Ecuador al generar nuevos contenidos se brindará mayor apoyo al desarrollo del Sistema de Alertas de Emergencia, que permitirá a la población recibir mensajes de emergencia en sus televisores, en caso de que ocurran de siniestros como: tsunamis, erupciones volcánicas, inundaciones, entre otros (Ministerio de Telecomunicaciones y de la Sociedad de la Información, 2018). En el país se realizó una prueba de transmisión de sistema de alerta de emergencia EWBS mediante uso de multiplexores y modulares, así como a través de una red de transporte de microondas para televisión digital, llegando los mensajes hacía los televisores dentro de los requisitos de tiempo real (Olmedo, Acosta , Haro, Villamarín, & Benavides, 2019).

Para el desarrollo de aplicaciones interactivas se realiza mediante el middleware Ginga, que cuenta con un entorno declarativo Ginga NCL (Nested Context Languaje), entorno imperativo Ginga J (Java) y Ginga CC (Common Core) (Galabay Toalongo & Vivar Espinoza, 2012).

Ginga es un Middleware Abierto del Sistema Nipo-Brasileño de TV digital desarrollado por TeleMidia Lab de PUC-Rio que se basa en dos principios: provisión de un buen soporte para la inclusión social/digital y el intercambio de conocimientos de forma libre. Ginga NCL está basado en un modelo conceptual de datos llamado NCM (Nested Context Model) y presenta características como interactividad, sincronismo, espacio temporal entre objetos de media, adaptabilidad, soporte a múltiples dispositivos y soporte a la producción de programas interactivos no-lineales (Alulema, 2012).

Adicionalmente, en el desarrollo de aplicaciones se realizan procesos invisibles para los usuarios por lo que también existen varios lenguajes de programación que permiten realizar procesamientos de datos como son Java, C, C++, Python, entre otros. Python es un lenguaje de programación moderno de alto nivel orientado a objetos, ideal para scripting y desarrollo de aplicaciones en diversas áreas. (Van Rossum, 2009)

1.2 Justificación e importancia

La televisión digital permite la generación de aplicaciones interactivas enfocadas a diferentes áreas como son Telegobierno, Telesalud, Teleduación, programación educativa, cultural, entre otras (Ministerio de Telecomunicaciones y de la Sociedad de la Información, s.f.). Una aplicación

desarrollada en Ginga NCL se puede ver en (Valencia Melo, 2013), que se basa en brindar información sobre protocolos de prevención a poblaciones de alto riesgo de fenómenos naturales.

También, mediante el lenguaje de programación Python se han desarrollado aplicaciones dirigidas para Televisión digital, como en (Moreno, García, Valero, Díaz, & Merino, 2011) que realiza un sistema abierto de TDT accesible para personas con deficiencia visual, el cual permite la configuración de la interfaz gráfica de usuario y la síntesis de voz mediante el uso del conversor de texto a voz (TTS, Text To Speech), desarrollando mediante Python la interfaz de usuario y el acceso a la TTS.

Además, con Python se ha realizado procesamiento de imágenes como en (Pavan Kalubandi, Hemanth , Vishnu , & Agilandeeswari , 2017), que realiza un cifrado de imágenes utilizando técnicas de criptografía visual y AES (Advanced Encryption Standard) para protección de la imagen. Otro trabajo se observa en (Rodes Pastor, 2017), en el cual se plantea el desarrollo de un algoritmo de compresión de imágenes y audio, utilizando Python para la implementación de códecs para la aplicación del algoritmo.

Adicionalmente, en (Chicaiza Jami, 2018) se realiza la implementación de un Set Top Box híbrido para TDT con componentes de bajo costo, en el que se hace uso de plataformas de hardware Raspberry Pi y NanoPi K2 en conjunto con módulos sintonizadores para obtener un decodificador. Por otra parte, en (Espinel Rivera, 2016) se hace la evaluación del rendimiento de un decodificador implementado sobre una Raspberry Pi con contenidos para TDT desarrollados en Ginga NCL.

En la actualidad se han realizado aplicaciones de diferentes tipos, haciendo uso de Ginga NCL y Python para llegar a la sociedad, por lo que para el presente trabajo lo que se pretende es aprovechar las propiedades de este lenguaje de programación Python al desarrollar una aplicación en la que se haga uso de operaciones matemáticas para el proceso de decodificación de información de elementos multimedia como imágenes encriptadas que contienen otra información para diferentes actores, como puede ser mensajes de seguridad, mediante Ginga NCL presentar las imágenes obtenidas para Televisión Digital, y finalmente ejecutar las aplicaciones en un Set Top Box implementado con hardware de bajo costo y evaluar el desempeño de las mismas.

1.3 Alcance del proyecto

El presente proyecto plantea la decodificación de imágenes encriptadas mediante el middleware Ginga NCL y lenguaje de programación Python para la obtención de información oculta, la misma que es destinada para diferentes actores.

Para realizar el proceso de decodificación de las imágenes, en la primera etapa mediante la programación en Python se obtendrá la información de la imagen (RGB) y los datos necesarios como son: tamaño, media, desviación estándar, ángulos de rotación, entre otros.

En la segunda etapa se realizará la recuperación de la imagen y la información encriptada basándose en la información que llega hacia la aplicación y haciendo uso de operaciones matemáticas.

Se desarrollará una aplicación en Ginga NCL para poder mostrar los resultados al obtener la imagen original que contenía información encriptada y la imagen oculta recuperada después del procesamiento.

Para la evaluación de resultados se utilizará el método subjetivo MOS (Mean Opinion Score), el cual consiste en evaluar la calidad de diferentes muestras y ponderar resultados para obtener una puntuación media, en una escala de cinco puntos: mala, pobre, regular, buena y excelente, en donde mala quiere decir que se presenta una distorsión muy molesta y excelente no presenta distorsión.

Adicionalmente se realizarán pruebas como RMSE (Root Mean Square Error) que se basa en medir el promedio de los errores al cuadrado, PSNR (Peak Signal to Noise Ratio) el cual analiza la relación entre la máxima energía posible de una señal y el ruido que la afecta y MSSIM (Mean Structural Similarity) que consiste en analizar la luminancia, el contraste y la estructura de las imágenes por separado. (Acosta Buenaño, Mora Jiménez, Olmedo, & Rojo Alvarez, 2018)

1.4 Objetivos

1.4.1 Objetivo General:

Desarrollar una aplicación de televisión digital (Ginga) empleando el lenguaje de programación Python, con la utilización de operaciones matemáticas para el proceso de decodificación de información de imágenes encriptadas.

1.4.2 Objetivos Específicos:

- Realizar el estudio del estado del arte de proyectos desarrollados con el lenguaje de programación Python y Ginga NCL para televisión digital.

- Determinar las especificaciones de funcionamiento de la aplicación a desarrollar.
- Desarrollar el proceso de decodificación de imágenes y la aplicación mediante la ayuda del lenguaje de programación NCL y Python.
- Implementar un Set Top Box con hardware de bajo costo para ejecutar aplicaciones de televisión digital.
- Realizar pruebas de funcionamiento de la aplicación bajo las especificaciones requeridas.
- Evaluar los resultados obtenidos de funcionamiento al desarrollar aplicaciones con lenguaje de programación Python y operaciones matemáticas en el proceso de decodificación de información de imágenes encriptadas.

1.5 Estructura del documento

La investigación se presenta en cinco capítulos:

El primer capítulo contiene la introducción, los antecedentes que se presentan en la Televisión Digital Terrestre, así como el middleware Ginga y lenguaje de programación Python. Además, se plantea el alcance del proyecto, justificación y los objetivos a cumplir.

En el segundo capítulo se presentan los fundamentos teóricos sobre Televisión Digital Terrestre, Middleware Ginga, lenguaje de programación Python, estadística básica, proceso de transformación de color de imágenes y métodos para la evaluación de resultados.

En el capítulo 3 se realiza una explicación del proceso de implementación de un STB con hardware de bajo costo, así como la decodificación de imágenes encriptadas mediante Python y, la implementación de la aplicación en Ginga NCL para mostrar los resultados.

El capítulo 4 describe los resultados obtenidos y presenta la evaluación de estos mediante el método MOS y las técnicas RMSE, PSNR y MSSIM.

En el capítulo 5 se exponen las conclusiones que se han obtenido en la investigación y se realizan recomendaciones para trabajos futuros.

CAPÍTULO II

2 MARCO TEÓRICO

2.1 Televisión Digital Terrestre

La Televisión Digital Terrestre o conocida como "TDT" es una nueva forma de transmitir las señales de Televisión Abierta o gratuita con ventajas, como mayor calidad de vídeo, imagen y sonido. Con la transmisión en formato digital se podrá aprovechar, de mejor manera, el espectro radioeléctrico**.** (TDT Ecuador, s.f.)

Algunas características principales de la TDT se presentan a continuación:

- Mejor aprovechamiento del espectro radioeléctrico, debido a que para transmitir un programa digital se necesita menor ancho de banda, con una relación de cuatro a cinco veces menos que un programa analógico dependiendo de la programación en calidad estándar o alta definición como se puede ver en la Figura 1. Debido a esto se aumenta la disponibilidad de canales televisivos y la posibilidad de transmitir contenidos interactivos.

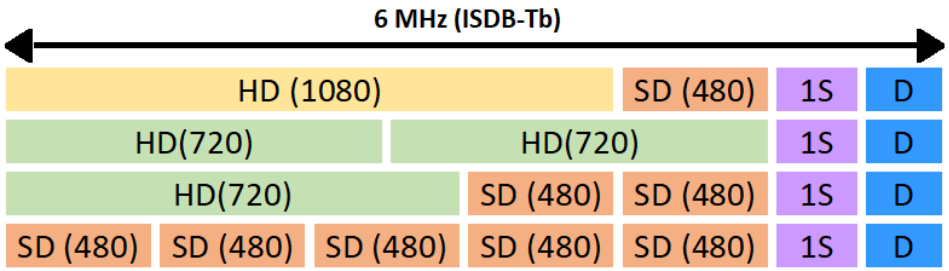

Figura 1. Comparación de ancho de banda para TV digital y analógica.
Adaptado de: (Alulema, 2012)

- La información televisiva que se transmite permite tener mayor definición, aumento de la calidad de imágenes, mejorar el sonido, ya que en las transmisiones digitales es posible utilizar técnicas de corrección de errores. Además, con la televisión digital se permite la implementación de nuevos servicios interactivos.
- La calidad de audio y video en televisión digital permite varios formatos de imagen como son: SD (Standard Definition) con 720x480 pixeles de resolución, HD (High Definition) con 1280x720 pixeles de resolución y Full HD (Full High Definition) con 1920x1080 pixeles de resolución (Grupo de investigación Digitalac (UCAM), s.f.).
- Otra característica, es que la señal transmitida de forma digital es más robusta, por lo que presenta más resistencia ante interferencias en comparación con las transmisiones analógicas, debido a que la recepción de señales análogas es débil mientras que las señales digitales se identifican solo con un "1" o "0". Las interferencias pueden ser condiciones meteorológicas, de tipo eléctricas, entre otras.
- La transmisión de señales digitales en comparación con las señales análogas requiere de menor potencia, aproximadamente con una relación de 1/10.
- La TDT permite tener movilidad, debido a que su señal puede ser receptada por dispositivos móviles y portátiles, permitiéndole al usuario recibir la señal en diferentes condiciones como puede ser caminando o encontrarse dentro de un vehículo en movimiento.
- La televisión digital permite la transmisión de datos, que pueden ser independientes de la programación o parte de ella, como aplicaciones interactivas de Telegobierno, Telesalud, Teleducación, programación educativa, cultural, o sistemas de alerta de emergencia.

- La prestación del servicio de televisión digital terrestre es posible implementar para todos los espectadores, ya que utiliza la misma red de la televisión analógica.

La televisión digital terrestre presenta varios estándares que han sido desarrollados en diferentes partes a nivel mundial. En Europa se creó el estándar llamado Difusión de Video Digital – Terrestres (DVB-T), en Estados Unidos el estándar ATSC por el Comité de Sistemas de Televisión Avanzada, en Japón el estándar Radiodifusión Digital de Servicios Integrados - Terrestre (ISDB-T) y en China se desarrolló Transmisión Multimedia Digital Terrestre (DTMB) (Chie, Zambrano, & Medina, s.f.).

En Ecuador se adoptó el estándar ISDB-Tb, el cual es un estándar que está basado en el japonés ISDB-T que cuenta con adaptaciones brasileñas y se aplica en el país a partir del año 2010.

2.2 ISDB-Tb

El estándar ISDB-T es de origen japonés, desarrollado por el consorcio ARIB (Asociación de Industrias y Negocios de Radiodifusión) junto con otros dos estándares ISDB-S que es satelital e ISDB-C que es por cable. Estos sistemas permitieron brindar mayor flexibilidad, capacidad de expansión y difusión de los servicios de transmisión de multimedia usando estos tipos de redes (DiBEG, 2009).

En Brasil la Agencia Nacional de Telecomunicaciones (ANATEL) realizo algunas modificaciones al estándar ISDB-T para obtener el ISDB-Tb, en donde los principales cambios fueron la exigencia de utilización de la compresión de video en MPEG-4 (H.264), compresión de

audio con HE-AAC y la introducción de GINGA como middleware para el desarrollo de aplicaciones interactivas para televisión digital (Sotelo, Durán, & Joskowicz, 2011).

Las características técnicas de este estándar es que la imagen se presenta con una relación de aspecto de 4:3 y de 16:9, el ancho de canal es de 6 MHz, trabaja en las bandas VHF y UHF del espectro electromagnético y utiliza la modulación OFDM.

ISDB-Tb hace uso de la modulación OFDM utilizando el dominio de la frecuencia y del tiempo. El dominio de la frecuencia se divide en sub-bandas mientras que el dominio del tiempo en intervalos cortos de tiempo.

El ancho de banda que utiliza este estándar es dividido en 14 segmentos, de los cuales 13 de ellos son utilizados para datos y uno para tener una banda de resguardo o separación entre los canales adyacentes. Cada segmento tiene un ancho de banda de 429 KHz. La información a transmitir se organiza en capas jerárquicas llamadas A, B, C, que son destinadas para el servicio de banda angosta “one-seg”, servicio de televisión de alta definición (HDTV) y para el servicio de televisión de definición estándar (SDTV), esta información se organiza en cada banda segmentada para la transmisión como se puede observar en la Figura 2.

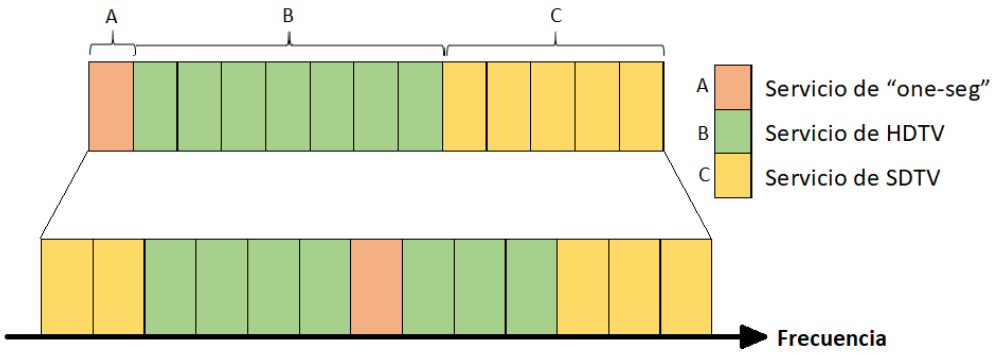

Figura 2. Transmisión en capas jerárquicas
Adaptado de: (Pisciotta, 2010)

El estándar proporciona un servicio de banda angosta llamado “One-seg” que es direccionado para equipos terminales como teléfonos celulares o dispositivos de televisión portátiles. Tiene como característica principal que reduce el consumo de energía del receptor, debido a que se disminuye la velocidad de procesamiento de señal.

2.2.1 Modos de operación

En este estándar se muestran tres modos de operación, que se presentan a continuación:

- *Modo 1 (2k):* En este modo se presentan 2k sub-portadoras, con 1405 sub-portadoras moduladas, de ellas 1248 se utilizan para portar datos y, las demás se utilizan para la transmisión de parámetros adicionales, además, las portadoras OFDM se encuentran espaciadas en 4KHz. Este modo es ideal para realizar transmisiones en áreas reducidas que necesiten menor potencia, así como es el más recomendable para la recepción en equipos terminales móviles.
- *Modo 2 (4k):* Presenta 4k sub-portadoras, con 2809 sub-portadoras moduladas utilizando 2496 para datos y, dejando las demás para transmitir parámetros de modulación y codificación. Presenta portadoras OFDM con un espaciamiento de 2KHz. Este modo de operación se implementó para encontrar un equilibrio entre costo y movilidad.
- *Modo 3 (8k):* Este modo de operación tiene 8k sub-portadoras, con 5617 sub-portadoras moduladas, utilizando para datos 4992 y las restantes para pilotaje y transmisión de otros parámetros. El espaciamiento entre las portadoras OFDM es de 1KHz. El modo de

operación 3 es usado en áreas geográficas extensas con una cobertura 16 veces mayor a la del modo 1.

Para la transmisión de señal digital se realiza mediante conjuntos de símbolos formados por 2, 4 y 6 bits, que son las modulaciones QPSK, 16 QAM y 64 QAM respectivamente, con lo que se pretende evitar la interferencia que existe entre portadoras.

2.3 Ginga NCL

El middleware Ginga NCL (Nested Context Languaje) fue desarrollado en la Pontificia Universidad Católica de Rio de Janeiro por el laboratorio de TeleMidia. NCL es un lenguaje de programación declarativo y de aplicación XML (Extensible Markup Languaje) que permite obtener sincronismo entre espacio y tiempo con los objetos multimedia (video, imagen, sonido, entro otros) (Pillajo , Ochoa, Acosta, & Olmedo, 2016).

NCL permite hacer uso de elementos multimedia en varios formatos, debido a que estos elementos dependen del reproductor que utiliza el intérprete de NCL, y se puede tener (Torres Altamirano, 2010):

- Imagen en GIF, JPEG, PNG, etc.
- Video en MPEG, MOV, etc
- Audio en MP3, WMA, etc
- Texto en TXT, PDF, etc
- Ejecución en Xlet, Lua, etc

2.3.1 Estructura de un documento de tipo NCL

El documento NCL se divide en dos partes, la primera es la cabecera (head) y la segunda el cuerpo del texto (body), como se muestra en la Figura 3.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!-- Generated by NCL Eclipse -->
3  <ncl id="new_ncl_file" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
4      <head>
5
6      </head>
7
8      <body>
9
10     </body>
11 </ncl>
```

Figura 3. Estructura documento NCL

En la cabecera se define la región que es el ¿Dónde se va a mostrar?, descriptor es el ¿Cómo se va a mostrar? y los conectores que identifican el ¿Cuándo se va a mostrar?

Mientras que en el cuerpo del texto se define la puerta que es el lugar por donde ingresa, los elementos multimedia que es lo que se va a mostrar, los enlaces que determinan cuando se va a mostrar y los contextos que permiten la reutilización de código (Acosta & Olmedo, 2014).

2.3.2 Elementos del Lenguaje NCL y atributos

En la sección de cabecera se puede tener los siguientes elementos: <importedDocumentBase>, <ruleBase>, <transitionBase>, <regionBase>, <descriptorBase>, <connectorBase>, <meta> y <metadata>, los cuales se caracterizan en la Tabla 1.

Tabla 1.
Elementos y atributos de cabecera de NCL

ELEMENTO	DESCRIPCIÓN	ATRIBUTOS	CONTENIDO
regionBase	Contiene un conjunto de elementos de <region>, en el cual se especifica las regiones de la pantalla para la presentación de elementos.	Id, device, región Región: id, title, left, right, top, bottom, height, width, zIndex	importBase, region
descriptorBase	Contiene el conjunto de elementos <descriptor>, en donde se determinan los parámetros de tiempo y espacio para la presentación de elementos.	Id Descriptor: id, player, explicitDur, región, freeze, moveLeft, moveRight, moveUp, moveDown, focusIndez, focusBorderColor, focusBorderWidth, focusBorderTransparency, focusSrc, focusSelSrc, selBorderColor, transIn, transOut	importBase, descriptor, descriptorSwitch
connectorBase	Define la activación de enlaces y acciones a ejecutar	Id importBase: alias, documentURI, region	importBase, causalConnector
importedDocument Base	Permite especificar documentos NCL importador de otras bases	id	importNCL
ruleBase	Define las reglas del documento NCL, basándose en la propiedad, operador y valor	Id Rule: id, var, comparator, value compositeRule: id, operator	importBase, rule, compositeRule
transitionBase	Define un conjunto de elementos para efectos de transición	Id Transition: type, subtype, dur, start-Progress, end-Progress, direction, fadeColor	importBase, transition
meta	Presenta información de propiedad/valor sobre los atributos name y content.	Name, content	importBase, transition
metaData	Presenta información relacionada con la metainformación del documento.	empty	RDF tree

Fuente: (Galabay Toalongo & Vivar Espinoza, 2012)

En la sección de cuerpo se pueden presentar los siguientes elementos: <port>, <property>, <media>, <context>, <switch> y <link>. En la Tabla 2 se presenta una descripción de estos elementos.

Tabla 2.
Elementos y atributos de cuerpo del documento NCL

ELEMENTO	DESCRIPCIÓN	ATRIBUTOS	CONTENIDO
media	Define un elemento multimedia determinado.	Id, src, refer, instance, type, descriptor	Area, property
area	Permite obtener opciones temporales para objetos multimedia	Id, coords, begin, end, text, position, first, last, label	
port	Determina la puerta de entrada para la aplicación.	Id, component, interface	
property	Son propiedades de <media>, que se utiliza cuando se requiera.	name, value	
context	Define la estructura de un documento de hipermedia, con los nodos que se presentan.	Id, refer	
switch	Plantea varios nodos que se activas de acuerdo a reglas.	id, refer DefaultComponent: component	defaultComponent, (switchPort, bindRule, media, context)
link	Permite realizar enlaces para interactividad y sincronismo entre multimedia y contextos.	id, xconnectror LinkParam: name, value Bind: role, component, interface, descriptor	LinkParam, Bind

Fuente: (Galabay Toalongo & Vivar Espinoza, 2012)

2.4 Lenguaje de programación Python

Python como lenguaje de programación fue creado por Guido van Rossum a principios de los años 90, llegando a los usuarios en el año 1991 con la primera versión de este lenguaje (Ortiz Ramírez, 2010), con licencia de software libre.

2.4.1 Características:

- Programación orientada a objetos, imperativa y funcional.
- Lenguaje de alto nivel
- Compatible con diferentes plataformas como Windows, Mac OS y Linux
- Ideal para programación de scripts así como desarrollo de aplicaciones de gran tamaño.
- Sintaxis sencilla
- Cuenta con una amplia gama de librerías.

2.4.2 Entornos de desarrollo

Existen varios entornos de desarrollo para Python, entre los principales se tiene: IDLE, PyDev, nbpython (PyAr, s.f.).

- *IDLE (Integrated DeveLopment Environment):* desarrollado por Python, ideal para aplicaciones pequeñas debido a su simplicidad, cuenta con depurador de código. Este entorno se encuentra disponible para sistemas operativos como Windows, Mac OS y Linux.
- *PyDev:* es un plugin de Eclipse, que tiene cuenta con analizador de sintaxis, resaltado de sintaxis, compresión de código, depuración, entre otros. Contiene un sistema de proyectos que ayuda a mejorar la organización de aplicaciones.
- *NbPython:* es una extensión de Netbeans que permite la ejecución de programas desarrollados en Python y tiene soporte para jython (Python implementado en Java), tiene analizador y resaltado de sintaxis, depuración, compresión de código, además de soporte para proyectos.

2.4.3 Librerias

En Python existen varios módulos adicionales para mejorar, reutilizar y organizar la programación, estos módulos se utilizan mediante el comando **import.**

A continuación, se presentan las librerías más comunes utilizadas en Python:

- *Numpy:* Módulo esencial para computación científica, que permite trabajar con matrices, arrays N-dimensionales, álgebra lineal, transformada de Fouries y capacidades de números aleatorios (Numpy desarrolladores, 2019).
- *Scipy:* Es una expanción de Numpy, que cuenta con integración numérica, interpolación, álgebra lineal y estadística (SciPy, 2019).
- *Matplotlib:* Es una librería que permite generar gráficos en 2D, histogramas, espectros de potencia, gráficos de barras, gráficos de error, entre otros (Matplotlib, 2019).

2.5 OpenCV-Python

OpenCV fue creado en 1999 por Gary Bradsky en Intel, posteriormente se unieron Willow Garage y Vadim Pisarevsky para continuar con el desarrollo. En la actualidad OpenCV permite un gran número de algoritmos (Mordvintsev & Abid, 2017).

OpenCV-Python es una aplicación de Python con OpenCV. Esta herramienta es ideal para manipulación de imágenes mediante la complementación con la librería Numpy, debido a las estructuras de matrices, se convierte en matrices Numpy por lo que se pueden combinar las funciones y realizar operaciones con esta librería. Además, se puede hacer uso con otras librerías como SciPy y Matplotlib (Mordvintsev & Abid, 2017).

2.6 Estadística básica

2.6.1 Media

La media o promedio es una medida que identifica un valor central de un conjunto de datos y se obtiene la sumar todos los números correspondientes al conjunto de datos y dividirlo para el número de datos del conjunto (Khan, 2019). Se representa por la ecuación (1):

$$\bar{x} = \frac{1}{n}\sum_{i=i}^{n} X_i \qquad (1)$$

Donde $(X_1, X_2, X_3, \dots, X_n)$ es el conjunto de datos y n es el número de datos.

2.6.2 Desviación estándar

La desviación estándar es una medida que proporciona el grado de dispersión que existe en un conjunto de datos con respecto a la media, se obtiene al hacer la sumatoria de la diferencia de cada muestra con respecto a la media y elevando al cuadrado, esta sumatoria se divide entre el número total de muestras menos 1 y se saca la raíz cuadrada (Suárez Ibujes & Tapia Zambrano, 2013). La ecuación se presenta en (2).

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{x})^2} \qquad (2)$$

Donde $(X_1, X_2, X_3, \dots, X_n)$ es el conjunto de datos y n es el número de datos. Las unidades de la desviación estándar son las mismas del conjunto de datos.

2.7 Imagen Digital

La imagen digital está compuesta por pixeles (*picture element*) dispuestos ordenadamente en un mapa de bits. Los pixeles son conformador por una combinación de color y brillo en una determinada posición.

El número de tonos posibles en una imagen depende del número de bits empleados para su representación, por ejemplo, al utilizar 4 bits se obtendrá 16 tonos. En una imagen a escala de grises se emplean 8 bits, por lo que se tiene 256 tonos, empezando con 0 que es negro y termina en 255 que es blanco, los demás tonos son grises. Las imágenes a color RGB *(red, green, blue)* son formadas por una mezcla de colores rojo, verde y azul, en donde cada color se encuentra representado por 8 bits, variando el nivel de brillo. Para las imágenes RGB los pixeles se representan con 24 bits en total, 8 bits por cada color (Costa Campos & Fernández Bozal, 2005).

2.8 Mosaico de Imágenes

Un mosaico de imágenes o foto-mosaico es la división de una imagen en secciones rectangulares generalmente del mismo tamaño, que luego son reemplazadas con otras imágenes individuales ordenadas tomando en cuenta las características del color original de la imagen, al unir todas las secciones con las nuevas imágenes en un todo se obtiene la imagen original compuesta con diferentes imágenes que se pueden apreciar viendo en detalle (INEGI, 2005).

Figura 4. Ejemplo de Fotomosaico
Fuente: (Fotomosaico, 2019)

2.9 Trasformación de color

2.9.1 NRCT (Nearly Reversible Color Transformations)

La transformación de color con el método NRCT se encargar de transformar las características de color de una imagen por bloques en otra en RGB, para ello se tiene dos imágenes, la primera que es la imagen secrete denominada T y la segunda es la imagen portadora denominada B.

Para la aplicación de este método se divide cada imagen en bloques y es necesario el cálculo de la media y desviación estándar para cada uno de los bloques en cada canal R, G y B de las imágenes T y B respectivamente, haciendo uso de las ecuaciones (1) y (2), por lo que se tiene:

Imagen secreta (T)

$$\mu_c = \frac{1}{n}\sum_{i=i}^{n} c_i\,; \qquad \sigma_c = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(c_i - \mu_i)^2}$$

Imagen portadora (B)

$$\mu_c' = \frac{1}{n}\sum_{i=i}^{n} c_i'\,; \qquad \sigma_c{}' = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(c_i' - \mu_c')^2}$$

En donde c_i y $c_i{}'$ representan los valores de cada canal R, G y B.

A continuación, se realiza el cálculo de los nuevos valores de color para bloque de T, con la ecuación (3):

$$c_i'' = q_c(c_i - \mu_c) + \mu_c' \qquad (3)$$

En donde:

$$q_c = \frac{\sigma_c'}{\sigma_c} \qquad (4)$$

Para comprobar se puede obtener la media y desviación estándar de los nuevos colores y deben ser iguales a los de la imagen portadora B.

En el proceso de recuperación de la imagen secreta, para alcanzar los colores originales se emplea la ecuación (5):

$$c_i = \left(\frac{1}{q_c}\right)(c_i'' - \mu_c') + \mu_c \qquad (5)$$

Se debe tomar en cuenta que q_c debe ser diferente de 0, ya en la ecuación (5), se tiene $\frac{1}{q_c}$ y esto no permitiría la recuperación de los colores originales (Lee & Tsai, 2014).

2.10 Set Top Box para TDT

Un Set Top Box es un decodificador que hace posible la visualización de televisión digital en televisiones análogas que no cuentan con receptor digital. El STB se encarga de recibir la señal digital, sintonizar y demodular, para poder observar el contenido en la televisión.

Un STB está conformado por elementos físicos y lógicos. En los elementos físicos se tiene componentes de hardware y software, entre las cuales están la placa madre, sintonizador o tuner, modulador/ demodulador, demultiplexor, decodificador de audio y video, procesador, memoria e interfaces físicas. Mientras que los elementos lógicos son necesarios para la ejecución de programas y manipulación la información, por lo que se tiene las aplicaciones, middleware, sistema operativo y hardware (Chicaiza Jami, 2018).

2.11 Métodos para la evaluación de resultados

2.11.1 MOS (Mean Opinion Score)

Método de evaluación subjetiva que se realiza a un grupo de usuarios, basándose en la calidad de experiencia. Generalmente se evalúa en ambientes controlados y mediante encuestas, logrando de esta forma obtener también datos sociológicos y psicológicos (Cuéllar, Ortiz, & Arciniegas, 2014).

Para la evaluación de la calidad de diferentes muestras y realizar la ponderación de resultados para obtener una puntuación media, se utiliza una escala de cinco puntos como se muestra en la Tabla 3.

Tabla 3.
Escala MOS

OPINIÓN DEL USUARIO	ESCALA MOS
Excelente	5
Buena	4
Regular	3
Mediocre	2
Mala	1

En donde mala quiere decir que se presenta una distorsión muy molesta y excelente no presenta distorsión (Acosta Buenaño, Mora Jiménez, Olmedo, & Rojo Alvarez, 2018).

2.11.2 MSSIM (Mean Structural Similarity)

MSSIM es un método de evaluación objetivo que consiste en analizar la luminancia, el contraste y la estructura de las imágenes por separado, por lo que se plantea las siguientes ecuaciones:

- Luminancia

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2\mu_y^2 + C_1} \qquad (6)$$

En donde μ_x y μ_y son las intensidades de media de x e y respectivamente.

- Contraste

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2+\sigma_y^2 + C_2} \qquad (7)$$

En donde σ_x y σ_y son las desviaciones estándar de x e y respectivamente.

- Estructura

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \qquad (8)$$

En donde σ_{xy} es el coeficiente de correlación de x e y.

C_1, C_2 y C_3 son constantes que se incluyen para evitar la presencia de inestabilidad.

Con las ecuaciones planteadas $(6), (7)$ y (8), se define la ecuación de similitud estructural SSIM:

$$SSIM(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma} \qquad (9)$$

Para la evaluación de la calidad de la imagen total se realiza mediante la MSSIM, que se define por:

$$MSSIM(X,Y) = \frac{1}{M}\sum_{J=1}^{M} SSIM(x_j, y_j) \qquad (10)$$

En donde X es la imagen de referencia, Y la imagen distorsionada, (x_j, y_j) es el resultado de la ventana local, y M es el número de ventanas en que se divide la imagen (Yang, Gao, & Po, 2007).

2.11.3 RMSE (Root Mean Square Error)

RMSE es una medida objetiva de calidad que mide la magnitud de error, se basa en medir el promedio de los errores al cuadrado, debido a que utiliza la diferencia entre valores reales y valores estimados, como se indica en la ecuación (11).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \qquad (11)$$

Donde y_i son valores reales, $\hat{y}_i$ valores estimados y n es el tamaño de la muestra (Negrón Baez, 2014).

2.11.4 PSNR (Peak Signal to Noise Ratio)

PSNR es la técnica más utilizada para la evaluación de calidad de codecs de compresión con respecto a la reconstrucción de imágenes, el cual analiza la relación entre la máxima energía posible de una señal y el ruido que afecta a la calidad. Su valor es calculado en dB (decibelios) y se representa con la ecuación (12).

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \qquad (12)$$

Donde MSE es el error medio cuadrático y se presenta en la ecuación (13).

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad (13)$$

Se toma 255, debido a que se utiliza datos enteros sin signo de 8 bits. En este caso el PSNR varía entre 30 y 50 dB (Umme, Morium, & Mohammad, 2019).

2.12 Herramientas

2.12.1 Raspberry Pi

Raspberry Pi se diseñó como un mini-ordenador de bajo costo con alto desempeño que permite el uso de lenguajes de alto nivel como Python, C++ y Java. Cuenta con características esenciales que permiten correr un sistema operativo. Utiliza un controlador Broadcom, que es Soc (System on Chip) (Casco, 2014).

Figura 5. Raspberry Pi 3B

Características (Raspberry Pi 3 B)

En la Tabla 4, se presentan las principales características de la Raspberry Pi 3B, referente a hardware y software.

Tabla 4.
Características técnicas Raspberry Pi 3 B

CARACTERÍSTICAS	DESCRIPCIÓN
CPU	1,4 GHz – 4 núcleos
GPU	Broadcom VideoCore IV
RAM	1GB SDRAM
Almacenamiento	Micro SD
Salida de audio	Jack 3,5 mm, HDMI
Salida de video	HDMI
Multimedia	Video decode: MPEG-2 / Video encode: H.264
Ethernet	Ethernet 10/100
Wireless	802.11n / Bluetooth 4.1
Pines I/O digitales	40 pines GPIO
Puertos USB	4 puertos USB 2.0
Sistema operativo	Distribución Linux y Android

Adaptado de: (Chicaiza Jami, 2018)

2.12.2 Ubuntu MATE para Raspberry Pi

Ubuntu MATE es un Sistema Operativo (SO) basado en Ubuntu, estable y con entorno de escritorio intuitivo y atractivo similar a Microsoft Windows o Apple Mac OS, por lo que facilita su utilización. MATE Desktop incluye herramientas como administrador de archivos, editor de texto, calculadora, visor de imágenes y documentos, monitor de sistema y una terminal (Ubuntu MATE, 2019).

Este sistema operativo es compatible con Raspberry Pi Modelo B 2, 3 y 3+, la última versión disponible es Ubuntu MATE 18.04.2. Para la instalación es necesario una tarjeta de memoria de 8GB o más, se debe tomar en cuenta que la partición de raíz se redimensiona automáticamente (Ubuntu MATE , 2019).

Características

- Kernel de Ubuntu
- Expansión automática del sistema de archivos en línea
- Ethernet y WiFi
- Bluetooth
- Salida de audio a través de conector análogo de 3,5 mm o HDMI
- Salida de video a través de HDMI
- Acceso GPIO
- Soporte para Python
- Soporte para el arranque USB

- Aceleración de hardware

2.12.3 PixelView PlayTV SBTVD Full-Seg

Es un módulo sintonizador de televisión que permite la recepción de TV digital en HD, SD y One-seg, es completamente compatible con SBTVD que es el Sistema Brasileño de Televisión Digital y puede funcionar de forma nativa en Linux.

Figura 6. Sintonizador PixelView

Características:

En la Tabla 5 se presentan las principales características del sintonizador PixelView para televisión digital.

Tabla 5.
Características Sintonizador PixelView

CARÁCTERÍSTICAS	DESCRIPCIÓN
Estándar	ISDB-Tb / SBTVD
Rango de frecuencia	54 MHz – 806 MHz
Ancho de banda	6 MHz.
Sensibilidad	-75 dBm
Decodificadores	Video H.264/ MPEG-2 y audio MPEG-4 HE-AAC.
Antena	Antena digital de 75 Ohm
Sistema operativo	Microsoft Windows XP/ Vista/ Win 7 y Linux
Compatibilidad	USB 2.0

Fuente: (Prolink, s.f.)

CAPÍTULO III

3 DISEÑO E IMPLEMENTACIÓN

En este capítulo se presenta el proceso de instalación de las herramientas necesarias para la implementación del Set Top Box con hardware de bajo costo de Televisión digital, así como el proceso de la decodificación de imágenes encriptadas mediante Python y la aplicación en Ginga NCL para la presentación de resultados.

3.1 Instalación del Sistema Operativo Ubuntu MATE para Raspberry Pi 3 B

La instalación del SO Ubuntu MATE se realiza sobre una tarjeta micro SD, es recomendable que la capacidad de memoria sea igual o mayor a 8GB. El proceso de instalación se puede realizar desde cualquier sistema operativo, en este caso se realizó en Windows con ayudar del software *Win32 Disk Imager* para grabar la imagen en la micro SD. El procedimiento de instalación se muestra en la Figura 7.

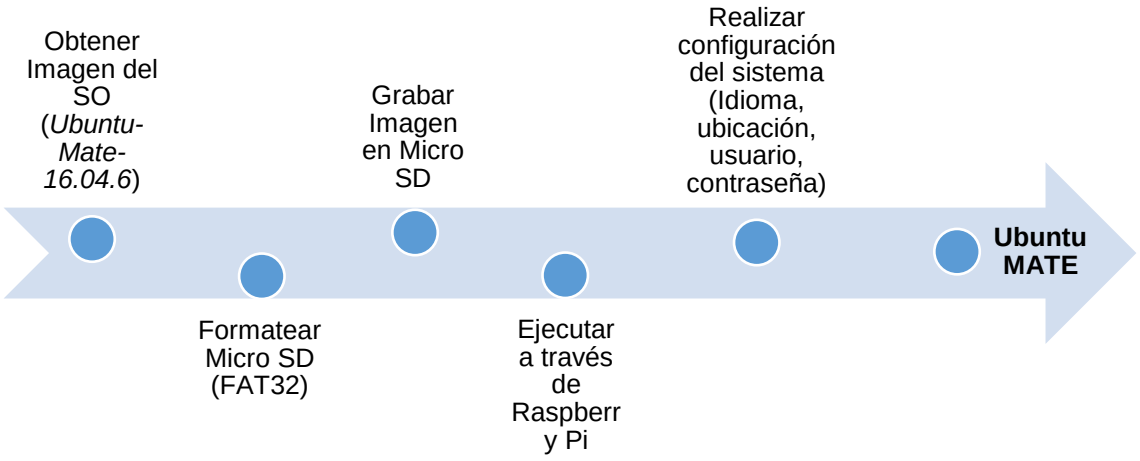

Figura 7. Proceso de instalación Ubuntu MATE

3.2 Implementación del STB con el módulo PixelView PlayTV SBTVD Full-Seg

La implementación del STB se realiza en el SO Ubuntu MATE, con ayuda del sintonizar de televisión digital PixelView, para esto es necesario conectar el módulo sintonizador con la antena digital a la Raspberry Pi, y a una pantalla para poder visualizar, como se indica en la Figura 8.

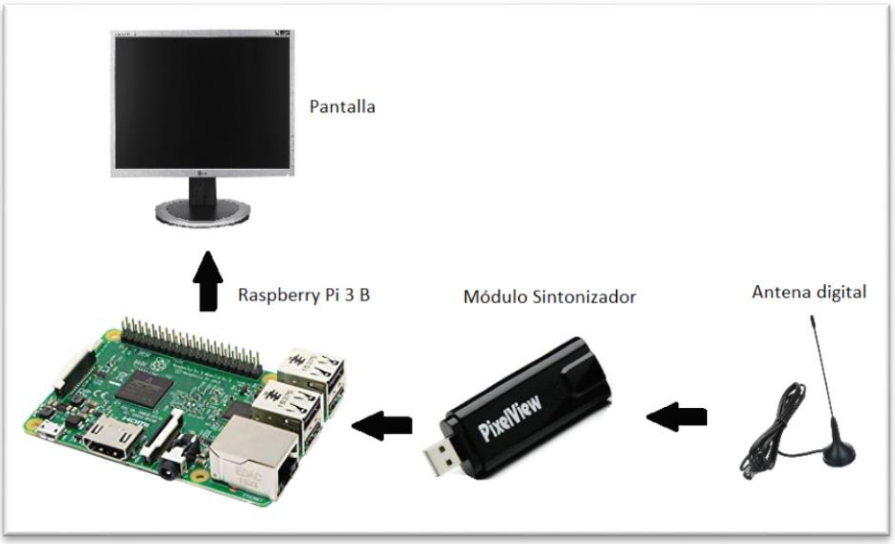

Figura 8. Conexión de STB

Para verificar si el dispositivo está conectado y es reconocido por SO, se emplea el comando "*lsusb"* de Linux, el cual permite desplegar una lista de dispositivos USB activos en la que se detalla el puerto de conexión, fabricante, nombre del dispositivo, entre otros, esto se puede observar en la Figura 9.

```
root@jhos-desktop:/home/jhos/Escritorio# lsusb
Bus 001 Device 052: ID 1554:5010 Prolink Microsystems Corp. PV-D231U(RN)-F [Pixe
lView PlayTV SBTVD Full-Seg]
Bus 001 Device 006: ID 125f:c08a A-DATA Technology Co., Ltd. C008 Flash Drive
Bus 001 Device 005: ID 0458:0152 KYE Systems Corp. (Mouse Systems)
Bus 001 Device 004: ID 413c:2107 Dell Computer Corp.
```

Figura 9. Reconocimiento Módulo sintonizador

En donde se observa que el sintonizador es reconocido con *ID 1554:5010*, fabricante *Prolink Microsystems Corp.* y su nombre *PixelView PlayTV SBTVD Full-Seg.*

Para el funcionamiento del sintonizador es necesario la instalación de las cabeceras de Linux (Linux-headers), ya que permiten la compilación de drivers y su construcción en el núcleo.

3.2.1 Instalación de Video4Linux

Linux cuenta con el módulo Video4Linux (V4L), el cual permite controlar dispositivos de medios y se puede obtener desde el repositorio de *GitHub.* Para la compilación de este módulo primero se debe instalar algunos paquetes que son requisitos para el funcionamiento, y luego se procede a la construcción.

Los paquetes necesarios se enlistan en la Figura 10.

Figura 10. Lista de paquetes para V4L
Fuente: (GitHub-Gregor Jansy, s.f.)

Para realizar la construcción de este módulo se deben ejecutar los comandos presentados en la Figura 11.

Figura 11. Comandos para construcción de V4L

Al terminar la construcción del módulo V4L, se procede a verificar si el sintonizador se conectó e inicializó, mediante el comando *dmesg*, el cual permite observar si el dispositivo si fue detectado y si se le asigno los controladores necesarios para su funcionamiento. Esto se puede ver en la Figura 12.

```
[  561.133934] DVB: registering new adapter (Prolink Pixelview SBTVD)
[  561.391804] usb 1-1.5: DVB: registering adapter 0 frontend 0 (DiBcom 8000 ISD
B-T)...
[  561.598815] DiB0070: successfully identified
[  561.598859] Registered IR keymap rc-dib0700-nec
[  561.599468] input: IR-receiver inside an USB DVB receiver as /devices/platfor
m/soc/3f980000.usb/usb1/1-1/1-1.5/rc/rc0/input522
[  561.599743] rc0: IR-receiver inside an USB DVB receiver as /devices/platform/
soc/3f980000.usb/usb1/1-1/1-1.5/rc/rc0
[  561.599865] dvb-usb: schedule remote query interval to 50 msecs.
[  561.599880] dvb-usb: Prolink Pixelview SBTVD successfully initialized and con
nected.
```

Figura 12. Conexión e inicialización del Sintonizador

Cuando el sintonizador se encuentra listo para utilizar se crean los archivos *demux0, dvr0, frontend0 y net0* en el directorio */dev/dvd/adapter0.*

3.2.2 Escaneo de canales de Televisión

Para la visualización de Televisión digital a través del sintonizador PixelView es necesario realizar un escaneo de canales televisivos por frecuencias, por lo que se debe instalar el paquete *w-scan*, con el comando *sudo apt-get install w-scan.*

Una vez instalado el paquete se ejecuta el comando *w_scan –a0 –cBR* con el que se hace un barrido por todas las frecuencias y se obtiene una lista de canales. En donde –c se utiliza para definir la zona en donde se encuentra ubicado, en este caso se utilizó BR de Brasil y el escaneo se realizó con éxito como se muestra en la Figura 13 y 14.

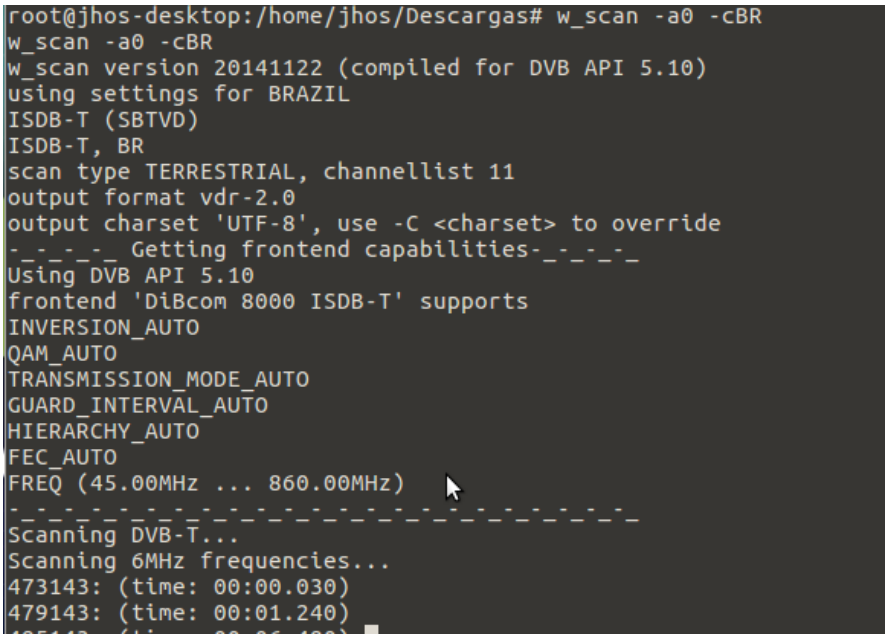

Figura 13. Escaneo de canales de televisión

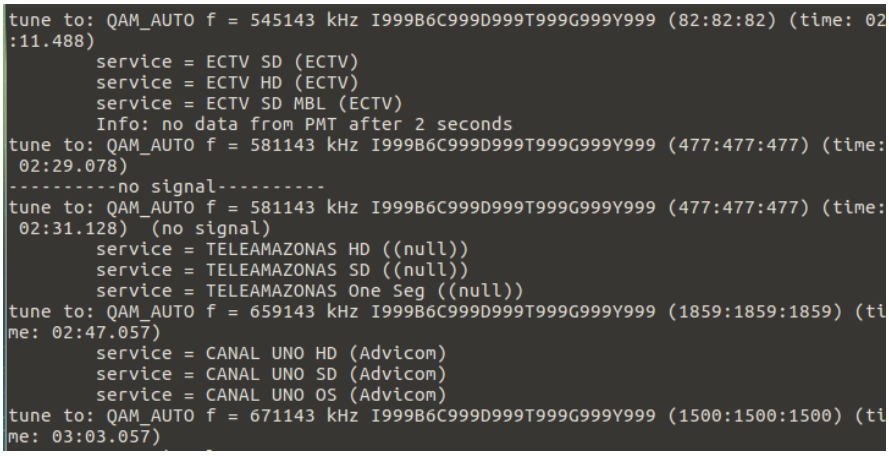

Figura 14. Canales de televisión obtenidos

Mediante el comando *w_scan –a0 –cBR –L lista.xspf* se genera un archivo con la lista de canales y para la reproducción de estos se realiza por medio de VLC Media Player que es un reproductor multimedia de código abierto, instalado por defecto en Ubuntu MATE.

3.3 Instalación de Ginga NCL

El middleware Ginga NCL se encuentra disponible para descargar e instalar en el repositorio de *GitHub* de TeleMidia, al igual que en V4L es necesario la instalación de dependencias para que el middleware funcione adecuadamente, estos paquetes se presentan en la Figura 15.

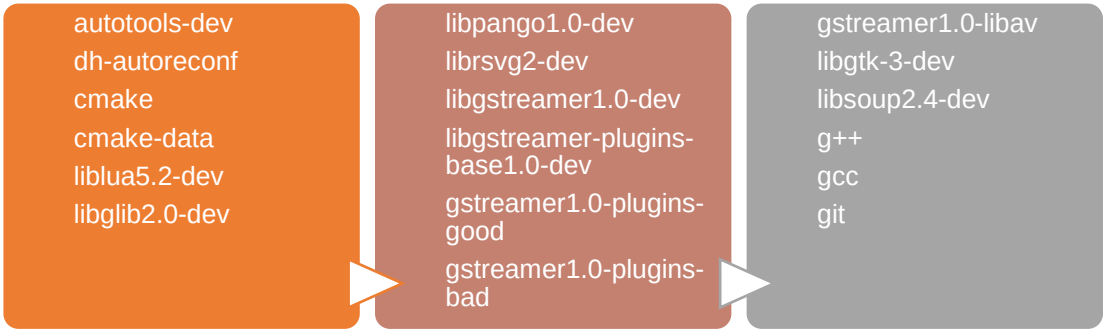

Figura 15. Lista de paquetes para Ginga NCL
Fuente: (TeleMidia, s.f.)

Una vez obtenidas las dependencias necesarias, se debe ejecutar los comandos presentados en la Figura 11, con los que se procederá a la construcción de Ginga NCL.

3.4 Instalación herramientas y librerías Python

En Linux y sus distribuciones, Python viene instalado por defecto en el sistema operativo, por lo que Ubuntu MATE cuenta con Python en la versión 2.7 y 3.5.

La instalación de librerías y herramientas adicionales se realiza por medio de líneas de comando. En este caso se realizó la instalación de las librerías numpy, scipy y matplotlib, así como la herramienta OpenCV que permite realizar manipulación de imágenes. Para esto se utilizó las líneas de comando que se presentan en la Figura 16.

Librerias	sudo apt-get install python-numpy
	sudo apt-get install python-scipy
	sudo apt-get install python-matplotlib
Herramienta	sudo apt-get install python-opencv

Figura 16. Instalación de Librerias y OpenCV Python

3.5 Decodificación de imágenes mediante Python

Actualmente existen varios métodos para el procesamiento de imágenes o elementos multimedia, en este caso se tomó como referencia el algoritmo planteado en (Acosta Buenaño, Mora Jiménez, Olmedo, & Rojo Alvarez, 2018), que utiliza la técnica de transformación de color casi reversible (NRCT) para obtener una imagen mosaico formada por una imagen portadora y una imagen secreta, además, de la formación de un stream de datos para poder realizar el proceso de

recuperación de la imagen secreta. En la decodificación, se realiza el proceso inverso para obtener de una imagen mosaico una imagen secreta recuperada.

Para el proceso de decodificación en Python se debe tomar en cuenta que para la recuperación de la imagen secreta se parte de:

- Valor N (Dimensión para el tamaño de bloques a utilizar)
- Imagen mosaico
- Stream de datos

El proceso de recuperación de la imagen se planta en el diagrama de bloques que se observa en la Figura 17.

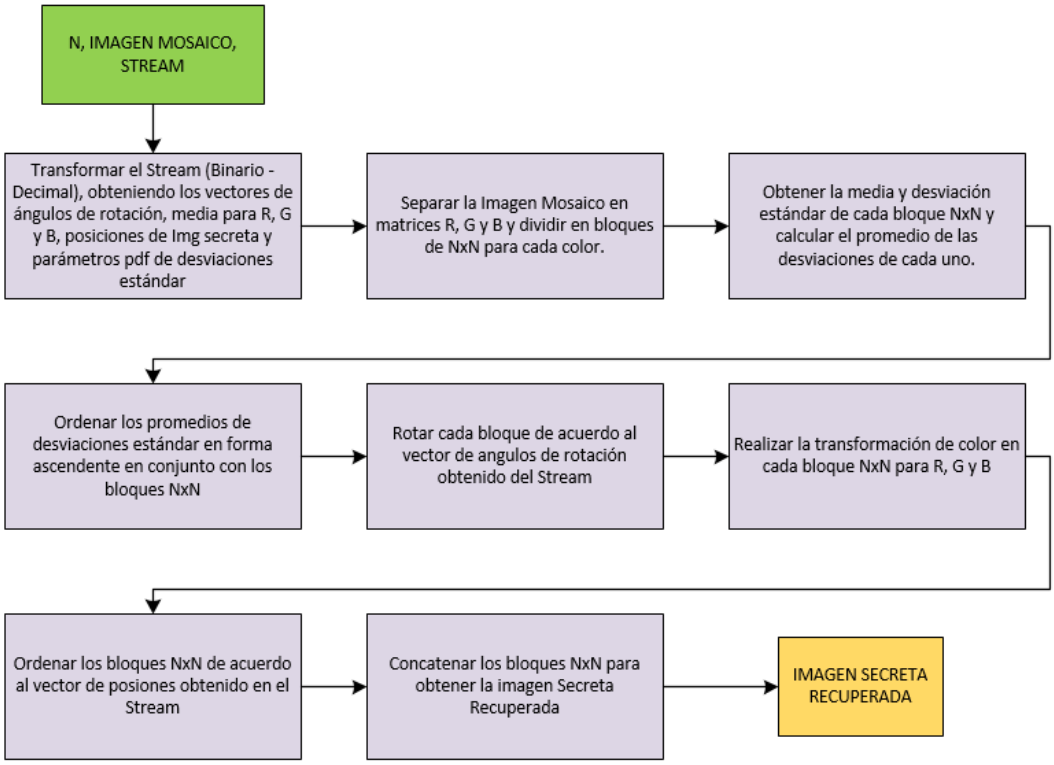

Figura 17. Diagrama de bloques Decodificación

A continuación, se explicará el proceso realizado en cada paso tomando como ejemplo una imagen mosaico con N=128, para que se pueda entender de mejor forma el procedimiento.

Entradas:

- N=128
- Imagen mosaico

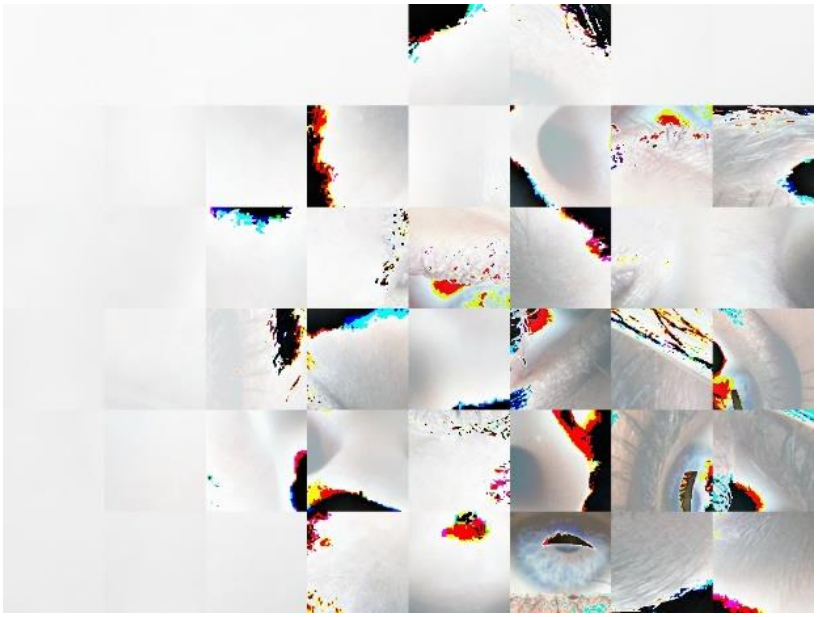

Figura 18. Imagen mosaico con N=128

- Stream de datos

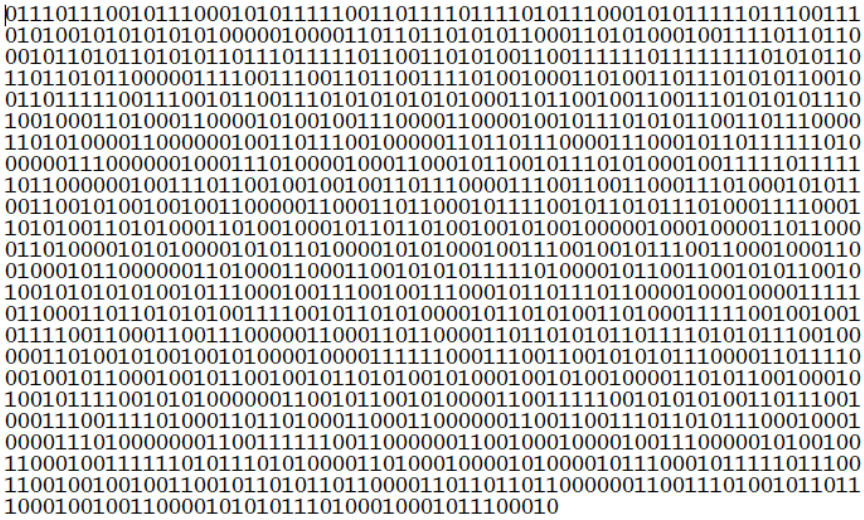

Figura 19. Stream de datos

3.5.1 Información de Stream

Para realizar la conversión del stream de binario a decimal, primero se debe tomar en cuenta como está formado el Stream de datos, por lo que basándose en (Acosta Buenaño, Mora Jiménez, Olmedo, & Rojo Alvarez, 2018) se tiene que el Stream total está determinado por la Ecuación 14:

$$E'_T = \bigcup \{E'_i\}_{i=1}^{b} s_1 s_2 \dots s_p \qquad (14)$$

En donde:

$$b = \frac{v \times w}{N \times N} \qquad (15)$$

$$E'_i = [r_1 r_2]_i \left[m_{sort_1} m_{sort_2} \dots m_{sort_{24}}\right]_i [s_1 s_2 \dots s_n]_i \qquad (16)$$

Teniendo en cuenta que:

- $N \rightarrow$ longitud del lado del bloque
- $v \times w \rightarrow$ tamaño de la imagen
- $[r_1 r_2] \rightarrow$ código para el ángulo de rotación
- $[m_{sort_1} m_{sort_2} \dots m_{sort_{24}}] \rightarrow$ intensidad media por cada componente de color (8 bits por componente)
- $[s_1 s_2 \dots s_n] \rightarrow$ valor del índice de posición
- $s_1 s_2 \dots s_p \rightarrow$ códigos de pdf de distribución lognormal para las desviaciones estándar.

El número de bits que conforman el stream depende del tamaño de bloque en que se subdivide la imagen, mientras más pequeño es el bloque, mayor será el número de bloques que componen la imagen, y por lo tanto mayor será el número de bits necesarios para representar los índices de posición de cada bloque, para hallar el número de bits se toma la ecuación 17.

$$n = [\log_2(b)] \qquad (17)$$

El tamaño de las imágenes utilizadas en este trabajo son de 1024×768 pixeles, entonces para el ejemplo se obtiene el valor de b con la ecuación (15):

$$b = \frac{1024 \times 768}{128 \times 128}$$

$$b = 48$$

Para determinar el número de bits utilizados para representar los índices de posición se utiliza la ecuación (17):

$$n = [\log_2(48)]$$

$$n = 5.58 \rightarrow 6$$

El stream se encuentra formado por:

$$Número\ de\ bits\ por\ bloque\ E_i'$$

$$= 2(códigos\ de\ rotación) + 24(intensidad\ media\ por\ componente)$$

$$+ 6(índice\ de\ posción)$$

$$Número\ de\ bits\ por\ bloque\ E_i' = 32\ bits/bloque$$

$$Número\ de\ bits\ total\ E_T'$$

$$= (48(bloques) * 32(bits/bloque)) + (72(bits\ de\ parámetros\ pdf)))$$

$$Número\ de\ bits\ total\ E_T' = 1608\ bits$$

3.5.2 Desarrollo del programa

Después de saber cómo se encuentra conformado el stream total, mediante programación se procede a realizar el proceso para convertir la información en bits en número decimal y trabajar con estos valores, en la Figura 20 se indica el proceso mediante un diagrama de bloques.

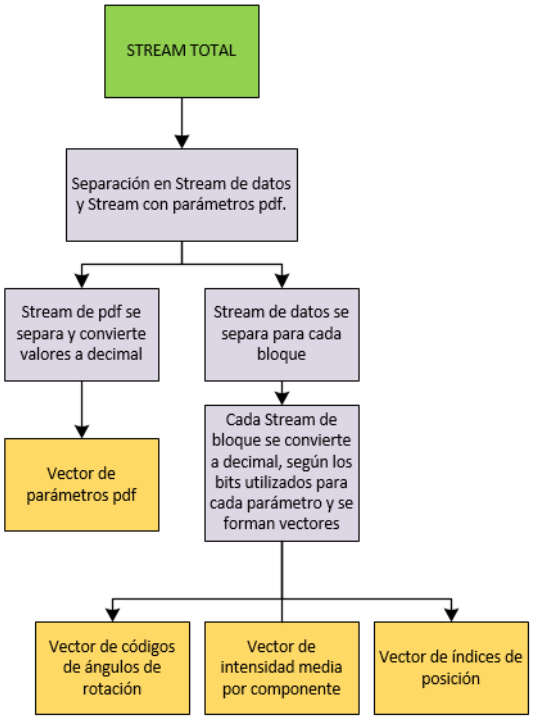

Figura 20. Conversión de Binario a Decimal (Stream)

A continuación, en la Figura 21 se muestra los vectores obtenidos para el ejemplo después de procesar la información del stream.

```
<terminated> Decodificacion3.py [C:\Python27\python.exe]
longitud Stream: 1608
longitud Stream datos: 1536
longitud Stream pdf: 72
('Codigo de rotacion', [1, 3, 1, 2, 2, 3, 3, 1, 2, 2, 0, 0, 2, 0, 1, 1, 2, 2, 1, 2, 1, 3, 0, 0, 2, 3,
('Media_R', [220, 222, 212, 218, 216, 217, 213, 205, 234, 206, 206, 194, 234, 192, 195, 192, 203, 192,
('Media_G', [184, 184, 170, 177, 181, 169, 182, 158, 201, 170, 171, 147, 205, 155, 139, 142, 168, 157,
('Media_B', [175, 175, 160, 168, 171, 159, 176, 145, 190, 163, 164, 134, 195, 144, 126, 132, 159, 146,
('Indice de pisicion', [38, 46, 33, 39, 29, 47, 30, 41, 28, 25, 26, 4, 20, 27, 32, 24, 31, 19, 34, 12,
('Parametros Pdf', [2.7529, 0.9111, 2.8838, 0.7598, 2.9082, 0.7207])
```

Figura 21. Vectores de datos

Una vez obtenida y almacenada la información de la imagen secreta se procede a trabajar con la imagen mosaico, la cual se separa en los componentes R, G y B y estos se dividen en bloques de NxN, para esto se utilizó la función *sub_matrices*, que se presenta en la Figura 22.

```
def sub_matrices(color_c, num):
    matrices = []

    for i in range(int(color_c.shape[0]/num)): # recorre filas/num
        for j in range(int(color_c.shape[1]/num)): #recorre columnas/num
            matrices.append(color_c[i*num:i*num + num, j*num:j*num+num])
    return matrices
```

Figura 22. Función para dividir matrices

El resultado de esta función es una lista con las sub-matrices obtenidas con dimensiones de acuerdo al valor de N ingresado. En el caso del ejemplo se obtienen 48 sub-matrices por cada componente de color.

El siguiente paso es obtener la media y desviación estándar de cada sub-matriz, se realizó mediante los comandos *numpy.mean* y *numpy.std.* Las sub-matrices de deben ordenar de acuerdo al promedio de desviaciones estándar en R, G y B para cada bloque en forma ascendente.

En la Figura 23. se muestra el código para almacenar en una lista cada sub-matriz con la media, desviación estándar y el promedio de desviaciones para luego ser ordenadas de forma ascendente de acuerdo a este último parámetro.

```
matrix_rm=[]
matrix_gm=[]
matrix_bm=[]
for i in range(len(rm_des_std)):
            matrix_rm.append((i, rm_submatrices[i], rm_media[i], rm_des_std[i], promm[i]))
            matrix_rm=sorted(matrix_rm,key=lambda m: m[4] )
            matrix_gm.append((i, gm_submatrices[i], gm_media[i], gm_des_std[i], promm[i]))
            matrix_gm=sorted(matrix_gm,key=lambda m: m[4] )
            matrix_bm.append((i, bm_submatrices[i], bm_media[i], bm_des_std[i], promm[i]))
            matrix_bm=sorted(matrix_bm,key=lambda m: m[4] )
```

Figura 23. Ordenamiento de matrices

Un ejemplo de cómo se encuentran conformados los elementos de las listas se presenta en la figura 24.

```
(0, array([[249, 249, 249, ..., 248, 248, 248],
       [249, 249, 249, ..., 248, 248, 248],
       [249, 249, 249, ..., 248, 248, 248],
       ...,
       [248, 248, 248, ..., 248, 248, 248],
       [249, 249, 248, ..., 248, 248, 248],
       [249, 249, 248, ..., 248, 248, 248]], dtype=uint8), 248.0, 0.5175, 0.5489)
```

Figura 24. Elemento de una lista

A continuación, se deben rotar los bloques en forma inversa al código del ángulo de rotación recibido en el stream, donde: $0,1,2\ y\ 3$ representan $0°, 90°, 180°\ y\ 270°$ respectivamente. Para este paso se planteó la función *rotar_matrices* que se muestra en la Figura 25, la cual ingresa la matriz original y devuelve una matriz rotada en forma inversa.

```
def rotar_matrices(submatrices):
    img_rot=[]
    for i in range(len(submatrices)):
        img_rot.append(np.rot90(submatrices[i][1], 4-ang_dec_list[i]))
    return img_rot
```

Figura 25. Función rotar matrices

Al hacer una comparación en la imagen mosaico vs. la imagen obtenida hasta este paso se puede observar cómo ha sido ordenada de acuerdo a los promedios de desviaciones estándar y las matrices rotadas, esto se observa en la Figura 26.

Figura 26. Imagen Mosaico vs. Imagen procesada

Para continuar con el proceso, se generan valores aleatorios para una distribución Log-normal de acuerdo a los parámetros de pdf obtenidos en el stream para cada componente de color, estos valores se deber ordenar en forma descendente y serán las desviaciones estándar de la imagen secreta.

El siguiente paso en la transformación de color NRCT, para lo cual se hace uso de la ecuación (5), tomando en cuenta que el cociente de las desviaciones estándar debe ser diferente de 0. Se define la función *nuevo_color* como se observa en la Figura 27.

```
def nuevo_color(q,color_mosaico,media_mosaico,media_secreta):
    c_nuevo=[]
    for i in range(len(q)):
        x=np.uint8((1/q[i])*((color_mosaico[i])-media_mosaico[i][2])+(media_secreta[i]))
        c_nuevo.append(x)
    return c_nuevo
```

Figura 27. Función para transformación de color

Después de realizar la transformación de color se obtuvo la Figura 28, en donde se puede distinguir que los colores han cambiado y se deben asimilar a los colores de la imagen secreta a recuperar.

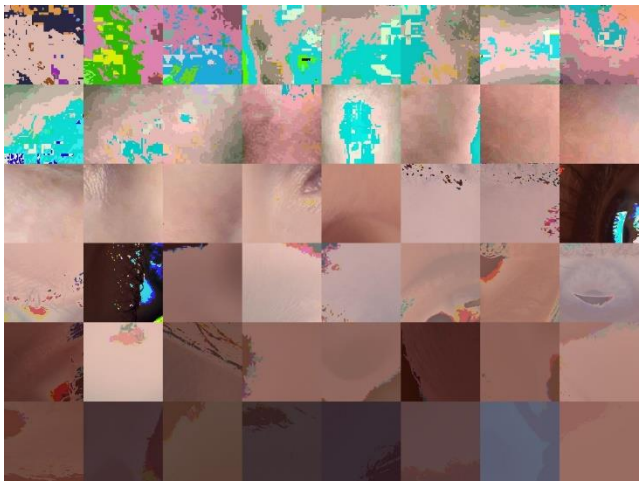

Figura 28. Imagen con transformación de color

Finalmente se deben ordenar los bloques de la imagen de acuerdo a los índices de posiciones de la imagen secreta obtenidos en el stream, y concatenar todos los bloques para obtener la imagen secreta recuperada.

3.6 Aplicación en Ginga-NCL

La aplicación desarrollada en Ginga-NCL tiene como objetivo presentar las imágenes obtenidas luego del proceso de decodificación realizado mediante Python. En la cual se presentará cuatro grupos de imágenes formados por una imagen original y tres imágenes recuperadas con diferente tamaño de bloque.

3.6.1 Características

Para el diseño y desarrollo de la aplicación se plantean las siguientes características:

- El ingreso a la aplicación se realizará al presionar el botón azul del control remoto cuando se presente el ícono en pantalla.
- La aplicación no será invasiva, permitirá la visualización del video o señal de televisión en todo momento.
- En la presentación de imágenes para avanzar o retroceder se realizará mediante la presión del botón derecho o izquierdo respectivamente.
- Se podrá salir de la aplicación en cualquier momento presionando el botón azul.

De acuerdo a las características planteadas se muestra el diagrama de flujo del funcionamiento de la aplicación en la Figura 29.

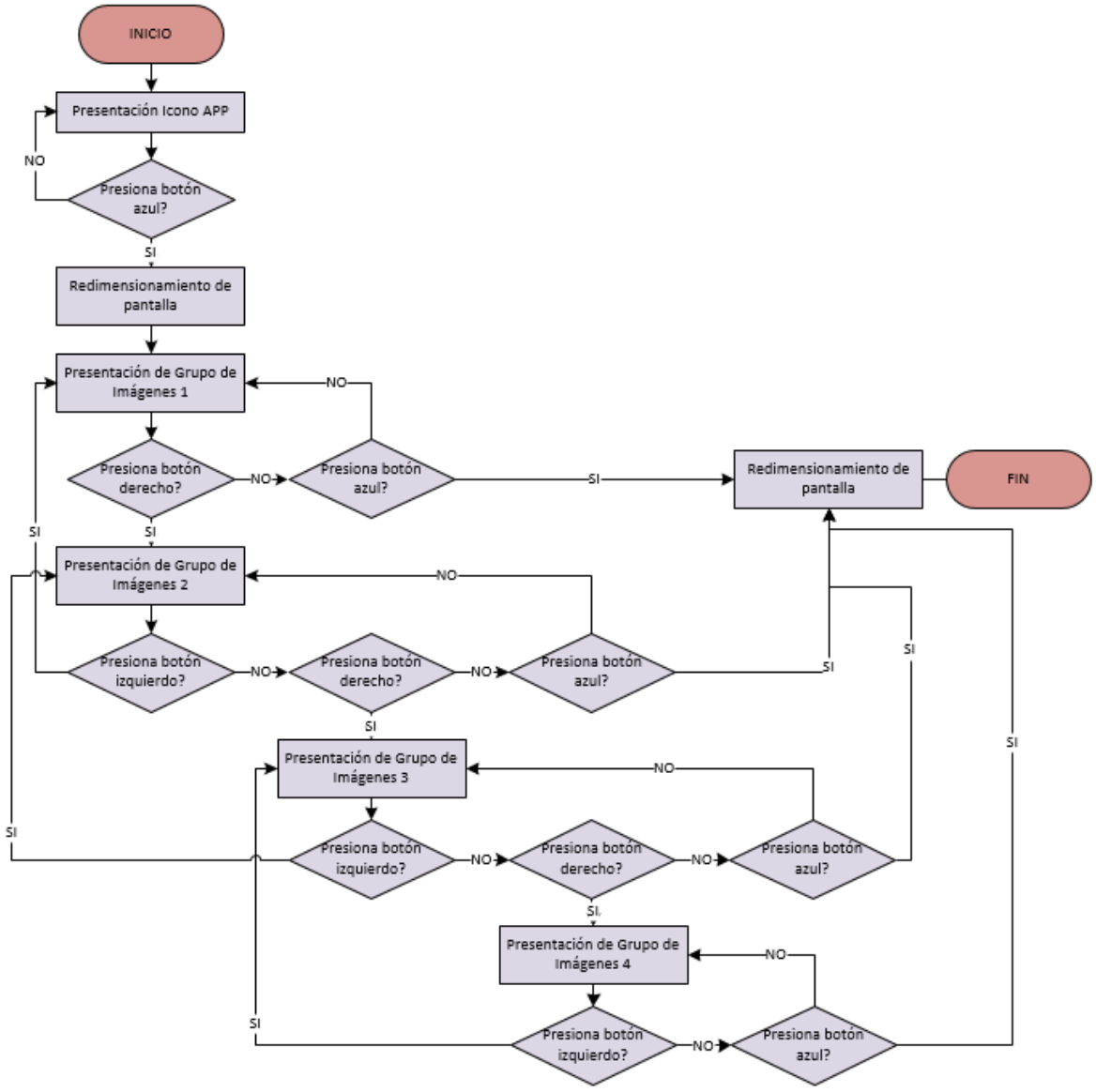

Figura 29. Diagrama de flujo Aplicación NCL

3.6.2 Desarrollo de la aplicación

Para desarrollar la aplicación en la primera parte en el *<Head>*, se realiza la declaración de las regiones, es decir se hace la asignación de espacios en donde se van a mostrar los elementos, como se muestra en la Figura 30.

```
<regionBase>
    <region id="R1" height="100%" width="100%" zIndex="1"/> <!--video-->
    <region id="R2" right="5%" top="10%" height="12.5%" width="8%" zIndex="2"/> <!--inicio-->
    <region id="R3" left="2%" top="20%" height="27.5%" width="22.5%" zIndex="2" /> <!--Img superior izquierda-->
    <region id="R4" left="26%" top="20%" height="27.5%" width="22.5%" zIndex="2"/> <!--Img superior derecha-->
    <region id="R5" left="2%" top="55%" height="27.5%" width="22.5%" zIndex="2" /> <!--Img inferior izquierda-->
    <region id="R6" left="26%" top="55%" height="27.5%" width="22.5%" zIndex="2"/> <!--Img inferior derecha-->
    <region id="R7" left="62.5%" top="73.5%" height="9%" width="5%" zIndex="2"/> <!--izquierda-->
    <region id="R8" left="82.5%" top="73.5%" height="9%" width="5%" zIndex="2"/> <!--derecha-->
    <region id="R9" left="72.5%" top="73.5%" height="9%" width="5%" zIndex="2"/> <!--salir-->
    <region id="R10" left="7.5%" top="5%" height="12%" width="35%" zIndex="2"/> <!--titulo-->
    <region id="R11" left="7%" top="48%" height="4%" width="12%" zIndex="2"/> <!--rotulo sup izq-->
    <region id="R12" left="34%" top="48%" height="4%" width="6%" zIndex="2"/> <!--rotulo sup der-->
    <region id="R13" left="10%" top="83%" height="4%" width="6%" zIndex="2"/> <!--rotulo inf izq-->
    <region id="R14" left="34%" top="83%" height="4%" width="6%" zIndex="2"/> <!--rotulo inf der-->

</regionBase>
```

Figura 30. Declaración de regiones NCL

En donde se asignan 14 regiones, se debe tomar en cuenta que las regiones se pueden reutilizar. En la Figura 31 se puede observar cómo se encuentra realiza la distribución de las regiones R3 a R14 para mayor comprensión.

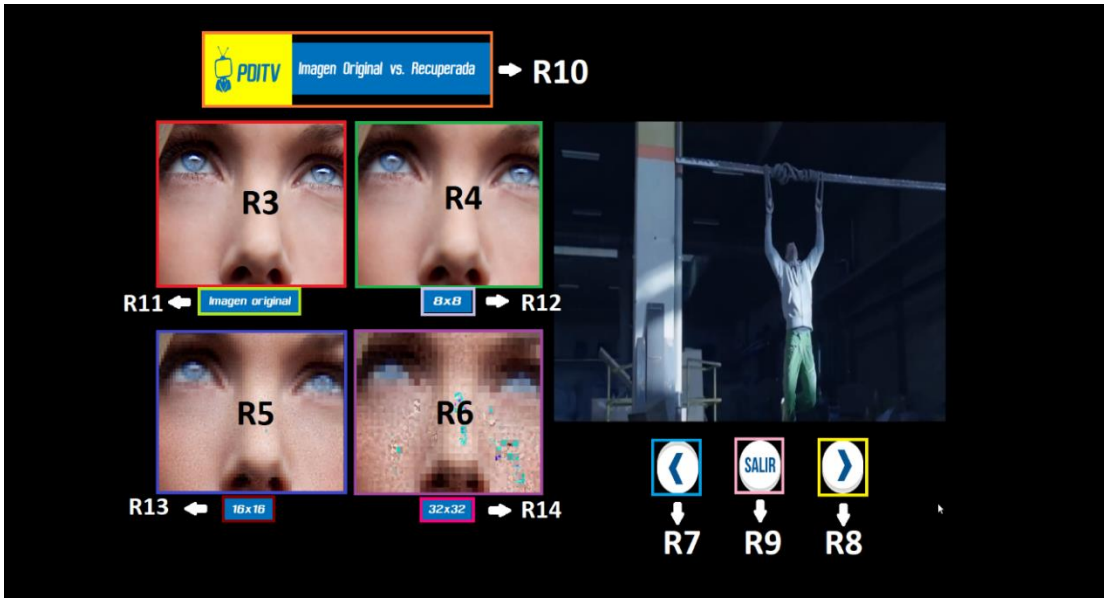

Figura 31. Asignación de regiones NCL

Después de la declaración de regiones se debe realizar la designación de descriptores, cada región debe tener asignado un descriptor, como se indica en la Figura 32.

```
<descriptorBase>
    <descriptor id="D1" region="R1"/>
    <descriptor id="D2" region="R2"/>
    <descriptor id="D3" region="R3"/>
    <descriptor id="D4" region="R4"/>
    <descriptor id="D5" region="R5"/>
    <descriptor id="D6" region="R6"/>
    <descriptor id="D7" region="R7"/>
    <descriptor id="D8" region="R8"/>
    <descriptor id="D9" region="R9"/>
    <descriptor id="D10" region="R10"/>
    <descriptor id="D11" region="R11"/>
    <descriptor id="D12" region="R12"/>
    <descriptor id="D13" region="R13"/>
    <descriptor id="D14" region="R14"/>
</descriptorBase>
```

Figura 32. Descriptores NCL

Para finalizar con la parte del <*Head*>, se importa la base de conectores y si se asigna un "alias" para facilitar el llamado cuando sea necesario, en este caso el alias es "c". Esto se muestra en la Figura 33.

```
<connectorBase>
    <importBase documentURI="ConnectorBase.ncl" alias="c"/>
</connectorBase>
```

Figura 33. Importación de base de conectores

En la parte del cuerpo del programa <*body*>, se inicia realizando la designación de los elementos multimedia y a cada uno de ellos se lo relaciona con un descriptor. Además, se declara una puerta entrada que es por donde va a iniciar la aplicación. En esta parte de deben incluir todos los elementos que se van a presentar en la aplicación como se puede observar en la Figura 34.

```
<port id="P1" component="M1"/>

<media id="M1" src="media/motivacion.mp4" descriptor="D1" >
    <property name="bounds"/>
    <area id="img" begin="5s"/>
</media>
<media id="M2" src="media/logo2.jpeg" descriptor="D2"/>
<media id="M3" src="media/derecha.png" descriptor="D8"/>
<media id="M4" src="media/izquierda.png" descriptor="D7"/>
<media id="M5" src="media/salir.png" descriptor="D9"/>
<media id="M6" src="media/pdiTV1.png" descriptor="D10"/>
<media id="M7" src="SECRETAS_ORIGINALES/AA.jpg" descriptor="D3"/>
<media id="M8" src="SECRETAS_RECUPERADAS/RAK8_MEDIA.jpg" descriptor="D4"/>
<media id="M27" src="SECRETAS_RECUPERADAS/RAK16_MEDIA.jpg" descriptor="D5"/>
<media id="M28" src="SECRETAS_RECUPERADAS/RAK32_MEDIA.jpg" descriptor="D6"/>
<media id="M9" src="SECRETAS_ORIGINALES/BB.jpg" descriptor="D3"/>
<media id="M10" src="SECRETAS_RECUPERADAS/RBK8_MEDIA.jpg" descriptor="D4"/>
<media id="M29" src="SECRETAS_RECUPERADAS/RBK16_MEDIA.jpg" descriptor="D5"/>
<media id="M30" src="SECRETAS_RECUPERADAS/RBK32_MEDIA.jpg" descriptor="D6"/>
<media id="M11" src="SECRETAS_ORIGINALES/CC.jpg" descriptor="D3"/>
<media id="M12" src="SECRETAS_RECUPERADAS/RCK8_MEDIA.jpg" descriptor="D4"/>
<media id="M31" src="SECRETAS_RECUPERADAS/RCK16_MEDIA.jpg" descriptor="D5"/>
<media id="M32" src="SECRETAS_RECUPERADAS/RCK32_MEDIA.jpg" descriptor="D6"/>
<media id="M13" src="SECRETAS_ORIGINALES/DD.jpg" descriptor="D3"/>
<media id="M14" src="SECRETAS_RECUPERADAS/RDK8_MEDIA.jpg" descriptor="D4"/>
<media id="M33" src="SECRETAS_RECUPERADAS/RDK16_MEDIA.jpg" descriptor="D5"/>
<media id="M34" src="SECRETAS_RECUPERADAS/RDK32_MEDIA.jpg" descriptor="D6"/>
<media id="M15" src="media/rotulo1.jpeg" descriptor="D11"/>
<media id="M16" src="media/rotulo2.jpeg" descriptor="D12"/>
<media id="M17" src="media/rotulo3.jpeg" descriptor="D13"/>
<media id="M18" src="media/rotulo4.jpeg" descriptor="D14"/>
```

Figura 34. Designación de elementos multimedia y puerta de entrada

A continuación, se definen las acciones del programa por medio de la utilización de la base de conectores. Como la puerta de entrada del programa es el video se plantea que cuando empieza el video, inicie la imagen del ícono de la aplicación mediante un conector *onBeginStart*, como se muestra en la Figura 35.

```
<link xconnector="c#onBeginStart">
<bind role="onBegin" component="M1" interface="img"/>
<bind role="start" component="M2"/>
</link>
```

Figura 35. Conector para iniciar la aplicación

Una vez iniciado el ícono de la aplicación se debe presionar el botón azul para ingresar en la misma, y al realizar esta acción la pantalla del video se redimensiona y se detiene la presentación del ícono, en la Figura 36 se observa esto mediante el conector *onKeySelectionSetNStopN*.

```
<link xconnector="c#onKeySelectionSetNStopN">
<linkParam name="keyCode" value="BLUE"/>
<bind role="stop" component="M2"/>
<bind role="onSelection" component="M2"/>
<bind role="set" component="M1" interface="bounds">
<!--REDIMENSIONAR PANTALLA-->
<bindParam name="var" value="50%,12.5%,47.5%,65%"/>
</bind>
</link>
```

Figura 36. Conector para redimensionar pantalla

Al ingresar en la aplicación y redimensionar la pantalla se procede a mostrar las imágenes obtenidas por grupos en el proceso de decodificación, así como los indicadores para las acciones que se pueden realizar, es decir ir a la derecha, izquierda o salir. En la Figura 37 se indica como utilizando el conector *onEndStartN* se presenta el primer grupo de imagen tomando como referencia cuando el ícono se ha detenido.

```
<link xconnector="c#onEndStartN">
<bind role="onEnd" component="M2"/>
<bind role="start" component="M3"/>
<!--<bind role="start" component="M4"/>-->
<bind role="start" component="M5"/>
<bind role="start" component="M6"/>
<bind role="start" component="M7"/>
<bind role="start" component="M8"/>
<bind role="start" component="M27"/>
<bind role="start" component="M28"/>
<bind role="start" component="M15"/>
<bind role="start" component="M16"/>
<bind role="start" component="M17"/>
<bind role="start" component="M18"/>
</link>
```

Figura 37. Conector para presentar grupo de Imágenes 1.

Utilizando el conector *onKeySelectionStartNStopN*, se realizan las acciones al presionar los botones: derecha, izquierda o azul para salir, de esta forma de van iniciando nuevos elementos multimedia y deteniendo otros para la presentación, como se muestra en la Figura 38.

```
<!--IMAGEN 1 DERECHA-->
<link xconnector="c#onKeySelectionStartNStopN">
<linkParam name="keyCode" value="CURSOR_RIGHT"/>
<bind role="onSelection" component="M7"/>
<bind role="start" component="M9"/>
<bind role="start" component="M10"/>
<bind role="start" component="M29"/>
<bind role="start" component="M30"/>
<bind role="stop" component="M7"/>
<bind role="stop" component="M8"/>
<bind role="stop" component="M27"/>
<bind role="stop" component="M28"/>
<bind role="start" component="M4"/>
</link>
```

Figura 38. Conector para acciones con botones

Finalmente, a través del conector *onKeySelectionSetNStopN* como se muestra en la Figura 39, se realiza la acción para salir de la aplicación en cualquier momento al presionar el botón azul, deteniendo todos los elementos multimedia y redimensionando la pantalla al valor inicial.

```
</link>
<!--REDIMESIÓN POR SALIR-->
<link xconnector="c#onKeySelectionSetNStopN">
<linkParam name="keyCode" value="BLUE"/>
<bind role="stop" component="M3"/>
<bind role="stop" component="M4"/>
<bind role="stop" component="M5"/>
<bind role="stop" component="M6"/>
<bind role="stop" component="M7"/>
<bind role="stop" component="M8"/>
<bind role="stop" component="M9"/>
<bind role="stop" component="M10"/>
<bind role="stop" component="M11"/>
<bind role="stop" component="M12"/>
<bind role="stop" component="M13"/>
<bind role="stop" component="M14"/>
<bind role="stop" component="M15"/>
<bind role="stop" component="M16"/>
<bind role="stop" component="M17"/>
<bind role="stop" component="M18"/>
<bind role="stop" component="M27"/>
<bind role="stop" component="M28"/>
<bind role="stop" component="M29"/>
<bind role="stop" component="M30"/>
<bind role="stop" component="M31"/>
<bind role="stop" component="M32"/>
<bind role="stop" component="M33"/>
<bind role="stop" component="M34"/>
<bind role="onSelection" component="M5"/>
<bind role="set" component="M1" interface="bounds">
<!--REDIMENSIONAR PANTALLA-->
<bindParam name="var" value="0%,0%,200%,200%"/> <!--l
</bind>
</link>
```

Figura 39. Conector para salir de la aplicación

CAPÍTULO IV

4 RESULTADOS Y ANÁLISIS

4.1 Resultados de la implementación del Set Top Box

Al implementar el STB sobre la tarjeta Raspberry Pi, se pudo observar la reproducción de canales de televisión a través de VLC Media Player, en definición HD, SD y One-seg, como se presentan en las Figuras 40, 41 y 42 respectivamente.

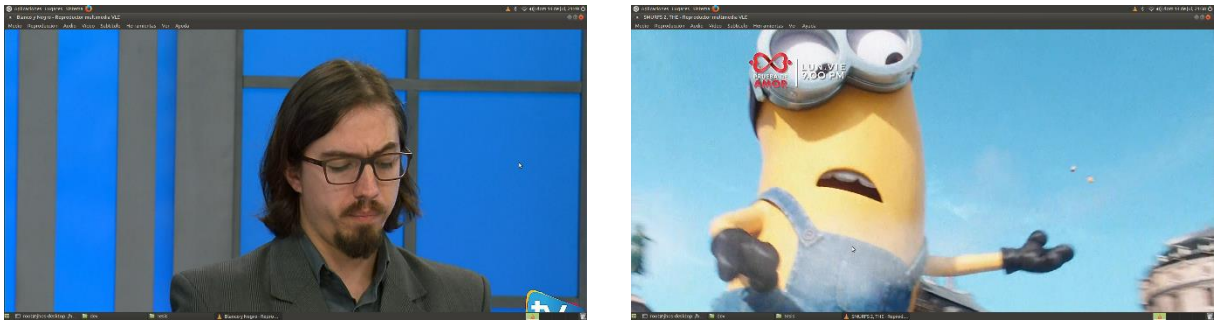

Figura 40. Televisión en HD

Figura 41. Televisión en SD

Figura 42. Televisión en One-seg

Como se puede observar en las Figuras 40, 41 y 42, al reproducir los canales de televisión en HD, SD y One-seg la imagen que se obtiene es nítida, con la diferencia que en One-seg estos canales se reproducen de mejor manera, mostrándose audio y video sin interrupciones, mientras que en SD y HD la calidad de reproducción es pobre, debido a que la reproducción de audio y video se pausan de manera espontánea.

4.2 Resultados de la decodificación de imágenes encriptadas con Python

Para poder apreciar los resultados del proceso de decodificación se presenta la imagen original portadora con respecto a la imagen mosaico que se utiliza para el proceso de recuperación de la imagen secreta, esto se observa en la Figura 43.

Figura 43. Imagen portadora vs. Imagen Mosaico

De igual forma se presentan las imágenes originales secretas a las que se pretende llegar con el proceso de decodificación. Son cuatro imágenes denominadas AA, BB, CC y DD, respectivamente y se muestran en la Figura 44.

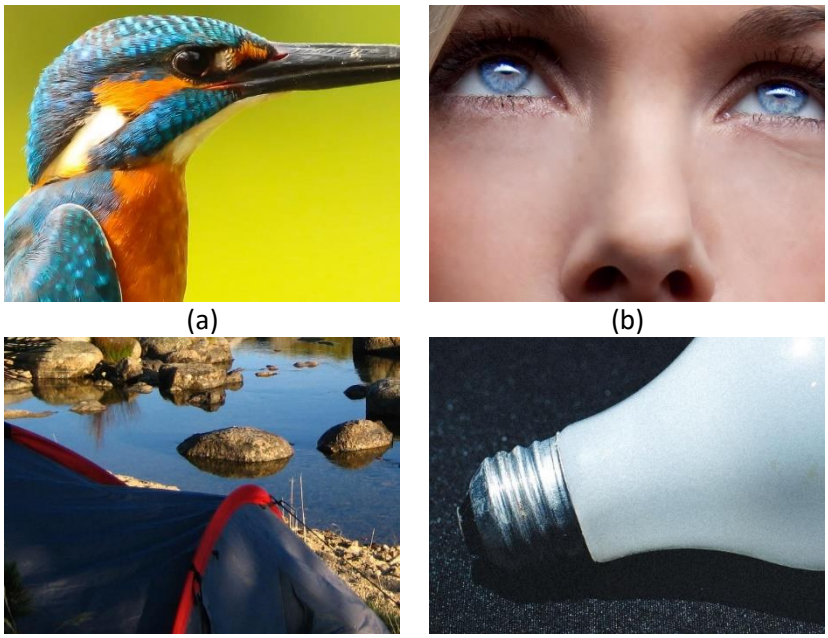

Figura 44. Imágenes secretas originales (a) AA, (b) BB, (c) CC y (d) DD

Para el proceso de decodificación de planteó 3 escenarios, en el primero se utilizó bloques de 8x8, en el segundo bloques de 16x16 y en el tercer escenario con bloques de 32x32. La imagen portadora es la misma en todos los casos, obteniendo las siguientes imágenes secretas recuperadas.

- **Escenario 1**

  Tamaño de bloque: 8x8

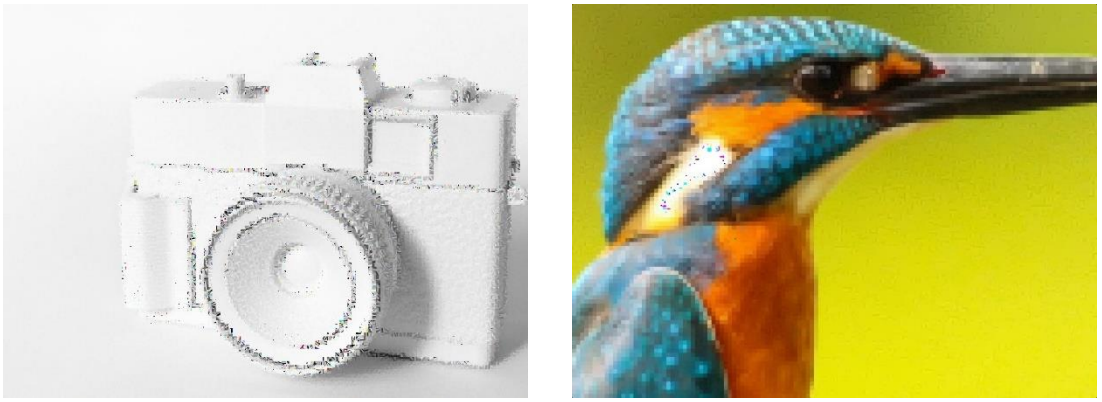

Figura 45. Imagen Mosaico con Imagen Recuperada en 8x8 (AA)

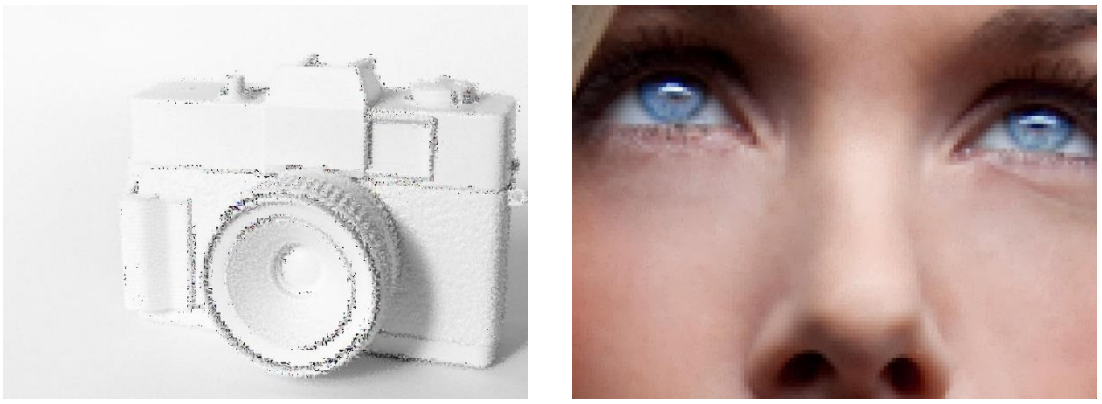

Figura 46. Imagen Mosaico con Imagen Recuperada en 8x8 (BB*)*

Figura 47. Imagen Mosaico con Imagen Recuperada en 8x8 (CC)

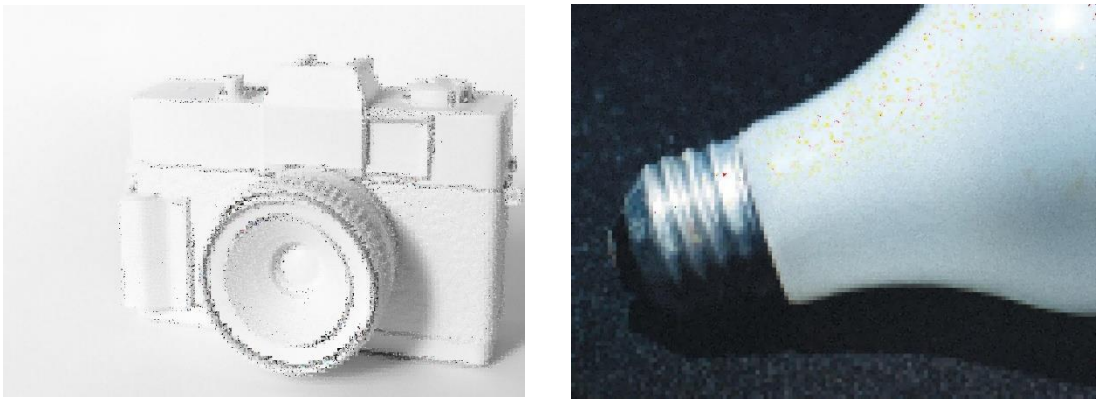

Figura 48. Imagen Mosaico con Imagen Recuperada en 8x8 (DD)

- **Escenario 2**

  Tamaño de bloque: 16x16

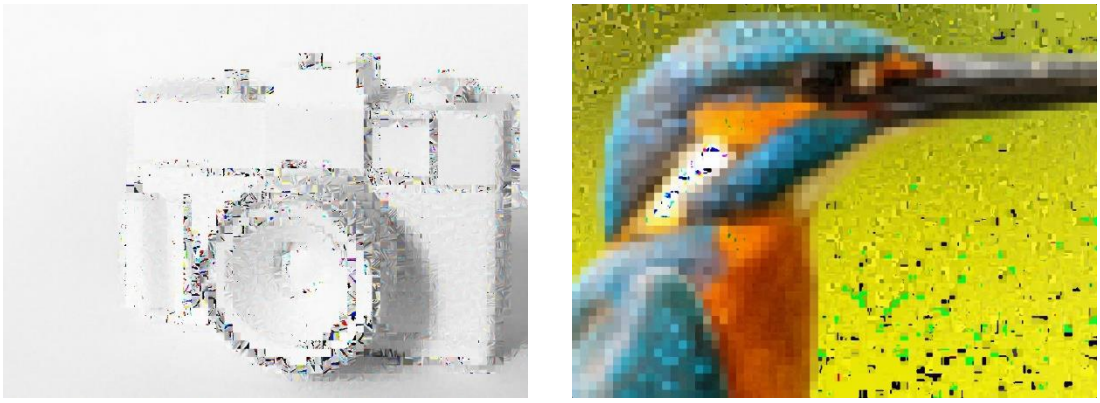

Figura 49. Imagen Mosaico con Imagen Recuperada en 16x16 (AA)

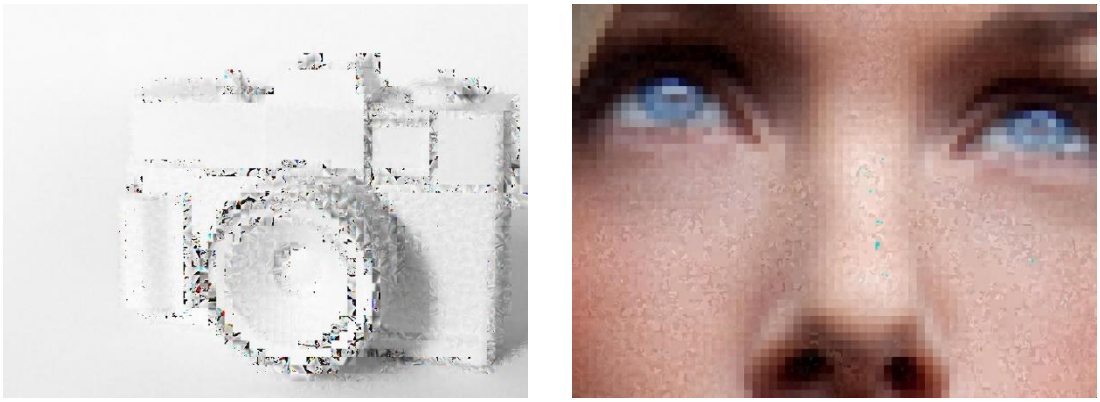

Figura 50. Imagen Mosaico con Imagen Recuperada en 16x16 (BB)

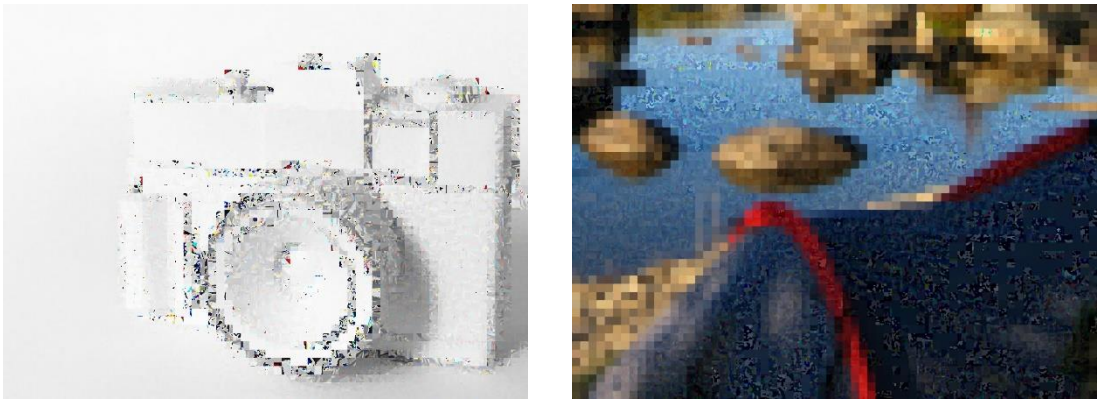

Figura 51. Imagen Mosaico con Imagen Recuperada en 16x16 (CC)

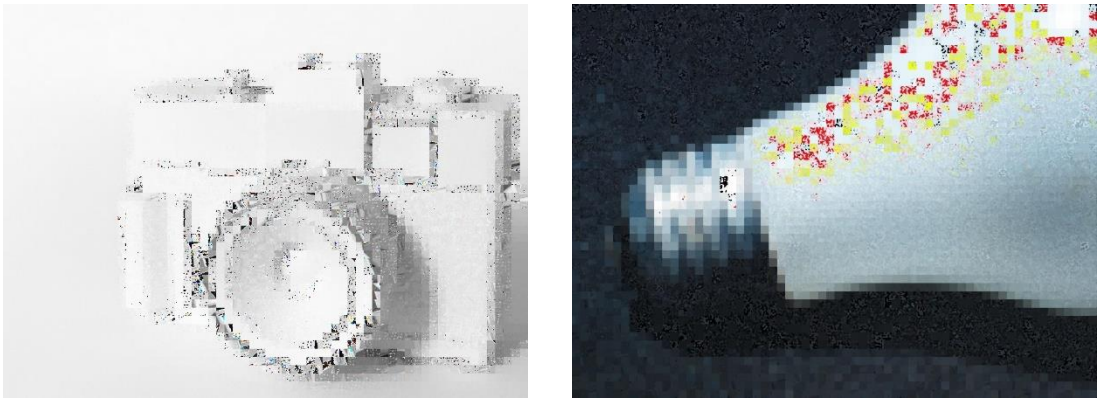

Figura 52. Imagen Mosaico con Imagen Recuperada en 16x16 (DD)

- **Escenario 3**

  Tamaño de bloque: 32x32

Figura 53. Imagen Mosaico con Imagen Recuperada en 32x32 (AA)

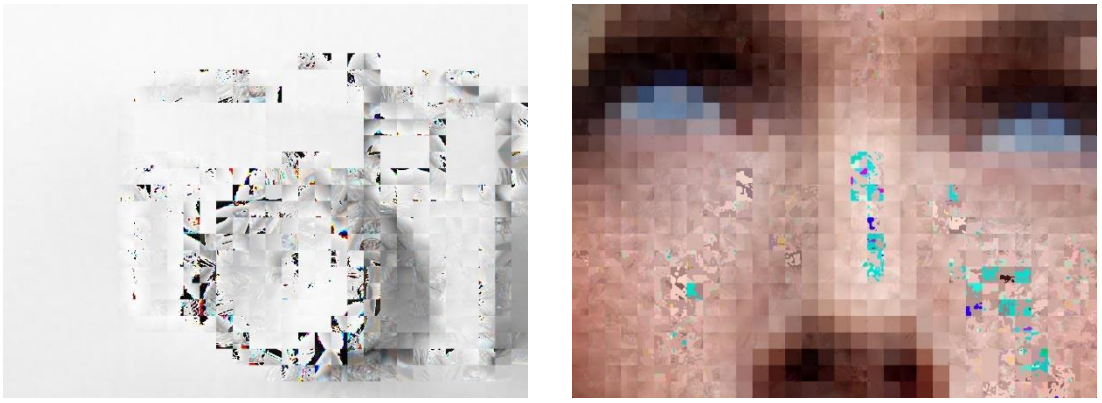

Figura 54. Imagen Mosaico con Imagen Recuperada en 32x32 (BB)

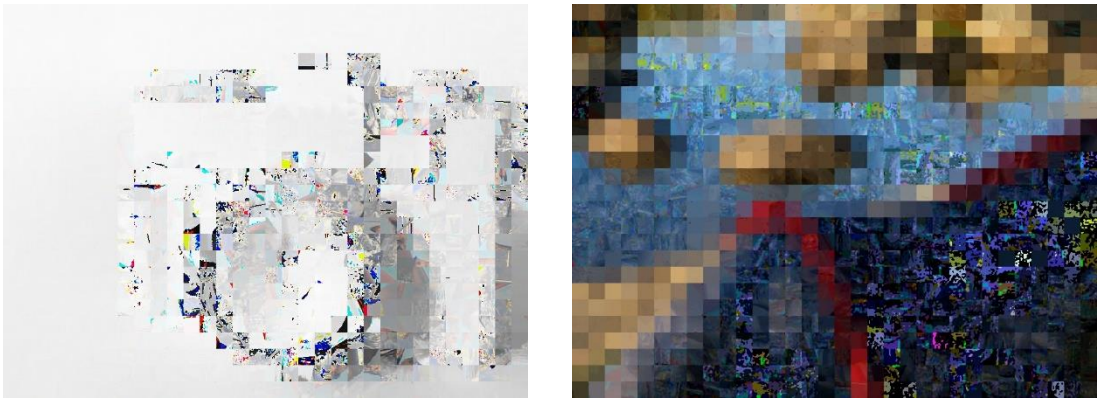

Figura 55. Imagen Mosaico con Imagen Recuperada en 32x32 (CC)

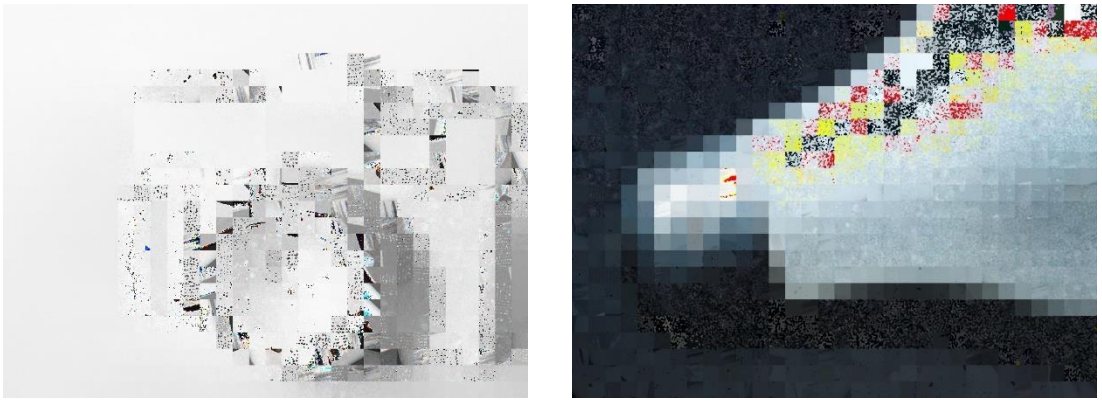

Figura 56. Imagen Mosaico con Imagen Recuperada en 32x32 (DD)

4.3 Aplicación en NCL para presentar las imágenes secretas recuperadas

En la aplicación desarrollada en NCL se muestran las imágenes obtenidas después del proceso de decodificación, de tal forma que se puede hacer una comparación entre la imagen original y las imágenes recuperadas con diferente tamaño de bloque, como se nombró anteriormente.

Esta aplicación fue ejecutada en Ubuntu Mate, sobre la tarjeta Raspberry Pi en la cual se implementó el STB. En la Figura 57 se puede observar la implementación con hardware de bajo costo y la ejecución de la aplicación.

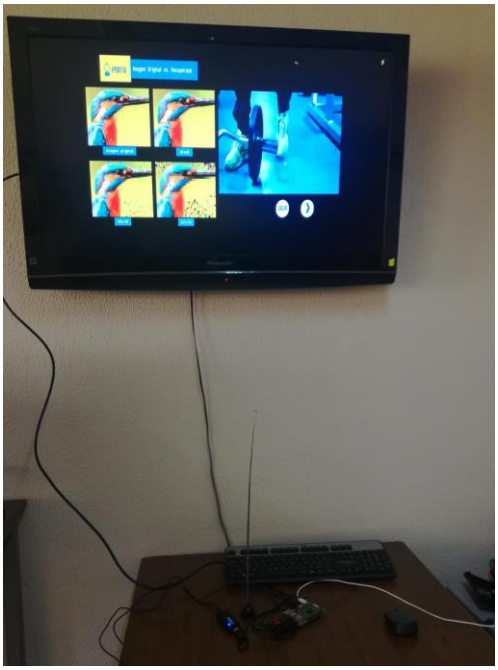

Figura 57. Implementación con Hardware de bajo costo.

La aplicación en NCL inicia con la presentación del ícono en parte superior derecha de la pantalla como se puede ver en la Figura 58, para ingresar en la aplicación se debe presionar el botón azul del control remoto.

Figura 58. Inicio de aplicación NCL

Al ingresar a la aplicación se muestra el primer grupo de imágenes, como en la Figura 59, este grupo está conformado por la imagen original AA y las 3 imágenes recuperadas en los diferentes escenarios, se presenta la opción de avanzar a la derecha o salir de la aplicación.

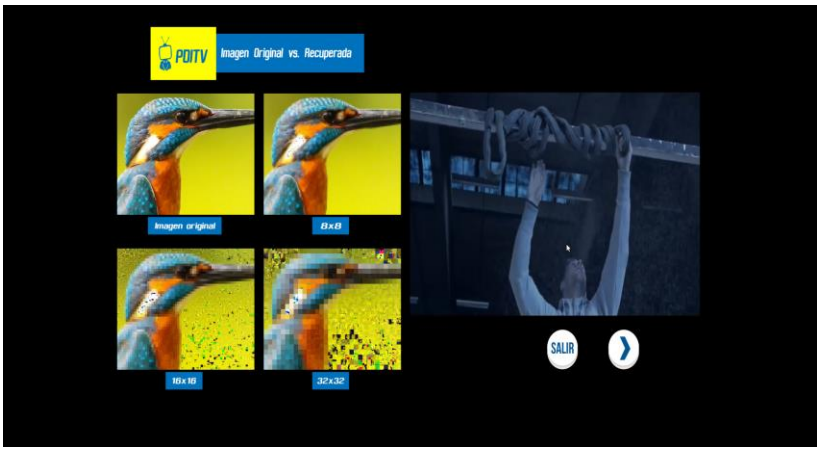

Figura 59. Presentación de Grupo 1 de imágenes

Al avanzar hacia la derecha, presionando el botón del control remoto se irán presentando los grupos de imágenes de BB, CC y DD como se observa en las Figuras 60, 61 y 62 respectivamente, así como también se puede retroceder al presionar el botón izquierdo o salir de la aplicación al presionar el botón azul.

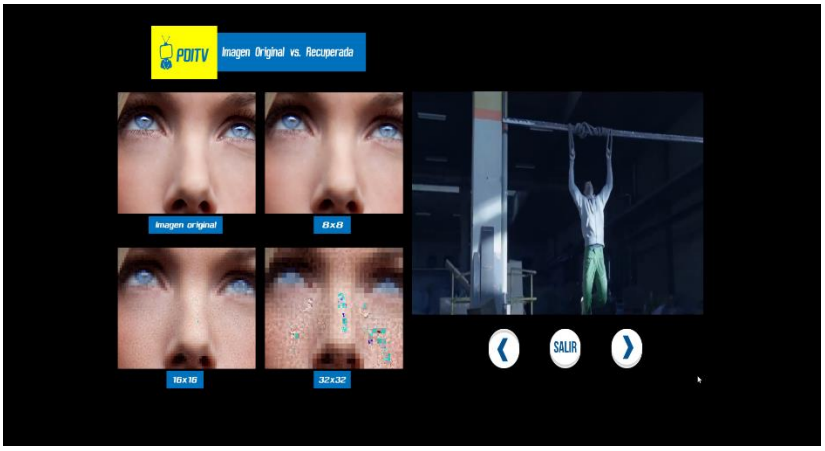

Figura 60. Presentación de Grupo 2 de imágenes

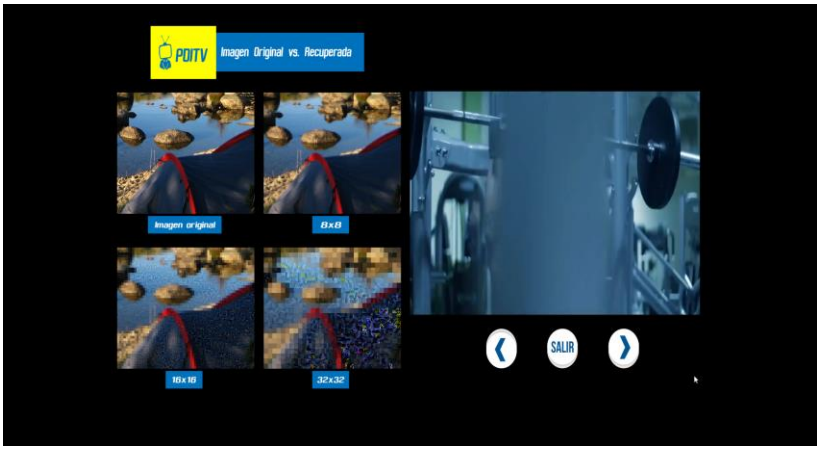

Figura 61. Presentación de Grupo 3 de imágenes

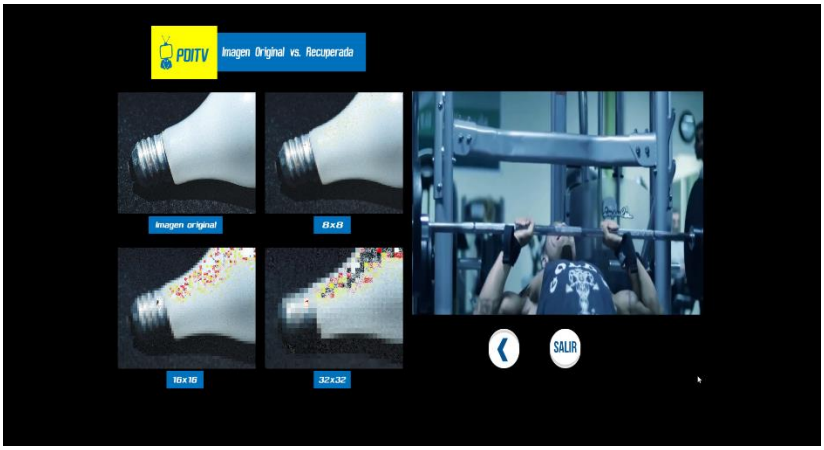

Figura 62. Presentación de Grupo 4 de imágenes

4.4 Análisis de Resultados

Para realizar el análisis de resultados de se obtuvieron datos de número de bits que conforman el stream de acuerdo a cada escenario plantado, el tiempo que se demora en ejecutar la decodificación a través del comando *time* de Python, y para los métodos de evaluación objetivos se realizó un programa mediante Python, en el cual se realiza una comparación entre la imagen original con respecto a la imagen recuperada como se puede observar en la figura 63.

```
import cv2
import numpy as np
import math
from scipy import signal, ndimage
import skimage

img_recuperada = cv2.imread("/home/jhos/Documentos/PyJhos/Recuperadas/RDK32_MEDIA.jpg")
img_original=cv2.imread("/home/jhos/Documentos/PyJhos/Secretas/DD.jpg")

def rmse(predictions, targets):
    return np.sqrt(((predictions - targets) ** 2).mean())


def psnr(img1, img2):
    mse=np.mean((img1-img2)**2)
    return 10*math.log10(255*2/(mse))


print('D 32x32')
print('PSNR= %f' %(psnr(img_original, img_recuperada)))
print('RMSE= %f' %(round(rmse(img_original, img_recuperada),3)))
print('MSSIM= %f' %(skimage.measure.compare_ssim(img_original, img_recuperada, multichannel=True)))

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figura 63. Programa con métodos de evaluación objetivos

Para realizar el programa se hizo uso de las ecuaciones (11) y (12) para RMSE y PSNR respectivamente, mientras que para determinar el MSSIM se utilizó la librería *Skimage* de Python, que permite obtener directamente el nivel de similitud entre las imágenes, los resultados se muestran como la Figura 64.

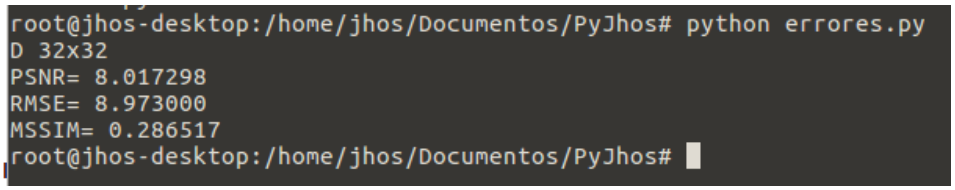

Figura 64. Evaluación de métodos objetivos para imagen DD con bloque 32x32

4.4.1 Número de bits utilizados por Imagen

El número de bits utilizado para la información de cada imagen se determinó con la siguiente ecuación:

$$Número\ de\ bits\ total\ E'_T = (b * Número\ de\ bits\ por\ bloque) + (bits\ de\ parámetros\ pdf)$$

Obteniendo los resultados que se presentan en la Tabla 6, y la representación gráfica en la figura 65.

Tabla 6.
Número de bits de acuerdo al tamaño de bloque

TAMAÑO DE BLOQUE	NÚMERO DE BITS
8x8	491592
16x16	116808
32x32	27720

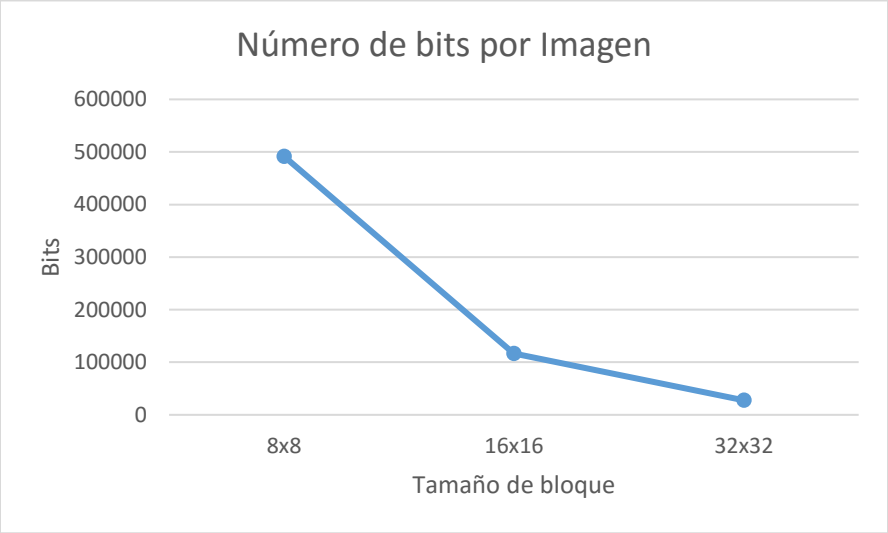

Figura 65. Número de bits de imagen por tamaño de bloque

Un stream está conformado por bits en donde el número de bits depende del tamaño de bloque en que se subdivide la imagen ya que mientras menor sea el tamaño mayor será el número de bloques obtenidos, en cada uno de estos bloques se obtienen datos que lo caracterizan para el proceso de recuperación de la imagen.

Por lo tanto, el número de bits utilizados será menor para un tamaño de bloque de mayor, por ejemplo, en el caso de 32x32 se utiliza aproximadamente 95% menos bits que en 8x8.

4.4.2 Tiempo de ejecución

Al ejecutar el proceso de decodificación de imágenes desarrollado en Python, para los tres escenarios se tomó el tiempo en segundos que se tarda en obtener la imagen recuperada al procesar en la Raspberry Pi y en una PC, las características principales de estos dispositivos se muestran en la Tabla 7.

Tabla 7.
Comparación Raspberry Pi vs. PC

CARACTERÍSTICAS	RASPBERRY PI 3B	PC
CPU	1,4 GHz – 4 núcleos	2.30 GHz Core i5-5300U
RAM	1GB SDRAM	8 GB
Sistema operativo	Ubunto MATE 16.04	Windows 10

El tiempo de ejecución al realizar la decodificación en la Raspberry Pi se muestran en la Tabla 8, y el tiempo de ejecución en la PC de muestra en la Tabla 9.

Tabla 8.
Tiempo de ejecución por tamaño de bloque en Raspberry Pi.

TAMAÑO DE BLOQUE	AA [s]	BB[s]	CC[s]	DD[s]
8x8	862,62	866,345	872,123	869,48
16x16	53,628	53,089	54,218	55,966
32x32	6,354	6,21	7,064	7,706

Tabla 9.
Tiempo de ejecución por tamaño de bloque en PC

TAMAÑO DE BLOQUE	AA [s]	BB[s]	CC[s]	DD[s]
8x8	127,2530	121,874	123,151	115,02
16x16	11,001	10,259	10,528	11,716
32x32	1,758	1,821	2,375	2,362

La representación gráfica de los tiempos obtenidos en segundos se observa en la Figura 66, en donde se muestran los tiempos de ejecución en la Raspberry Pi vs. Tiempo de ejecución en la PC.

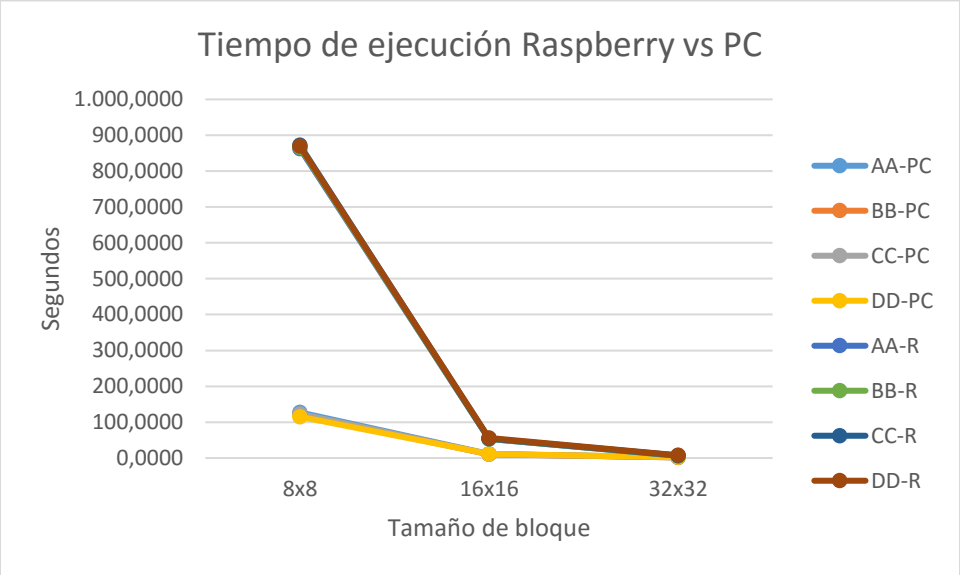

Figura 66. Tiempo de ejecución por tamaño de bloque en Raspberry Pi vs. PC.

Al realizar la decodificación de las diferentes imágenes utilizando el sistema operativo Ubuntu Mate implementado en la Raspberry Pi 3B, se obtuvo que la diferencia de tiempo al variar el tamaño de bloque es significativa al utilizar la opción de 8x8 en comparación a las otras opciones, debido a que el tiempo de ejecución para este tamaño de bloque es aproximadamente el 94% más con respecto a 16x16 y el 99% más en comparación a 32x32, mientras que la diferencia de tiempo para el tamaño 16x16 es aproximadamente el 89% más del tiempo de la opción 32x32.

Mientras que al realizar el proceso de decodificación en la PC el tiempo de ejecución al utilizar bloques de 8x8 es 98% más tiempo que al realizar con bloques de 32x32, además, al hacer la comparación de tiempos entre la ejecución en la Raspberry Pi y la PC, el tiempo de ejecución es mayor en un 85% al realizar sobre las Raspberry.

Concluyendo que mientras menor es el tamaño de bloque en que se subdivide la imagen, mayor será el tiempo de ejecución del programa debido a que se procesa mayor información.

4.4.3 RMSE (Error cuadrático medio)

Para realizar el análisis mediante este método de evaluación, se realizó la comparación entre la imagen original secreta y las imágenes recuperadas en los tres escenarios por lo que obtuvieron los resultados presentados en la Tabla 10, y con estos resultados se realizó la presentación gráfica que se muestra en la Figura 67.

Tabla 10.
Resultados de RMSE

TAMAÑO DE BLOQUE	AA	BB	CC	DD
8x8	5,9800	4,361	6,588	7,38
16x16	8,232	6,62	8,704	8,46
32x32	9,204	8,325	9,579	8,973

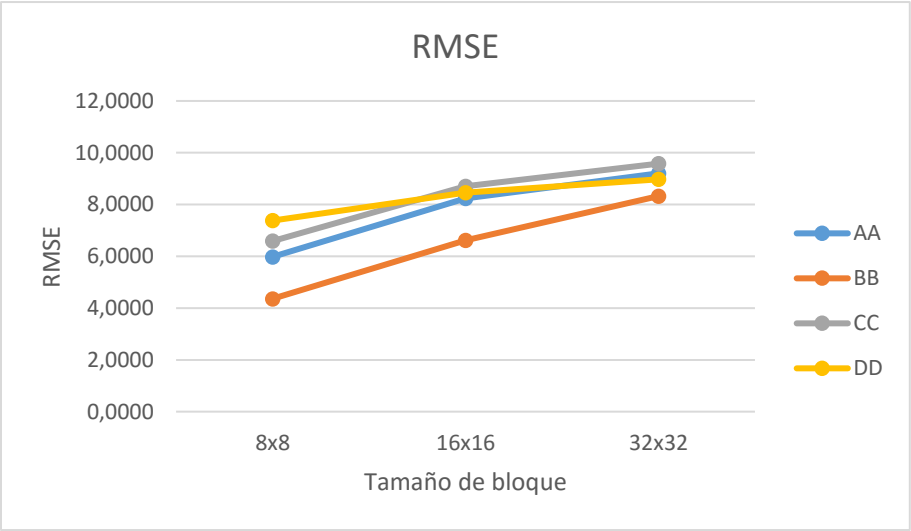

Figura 67. RMSE por tamaño de bloque

El RMSE representa la magnitud de error que existe entre la imagen original y la imagen recuperada, en este caso se puede observar que al utilizar bloques de tamaño más pequeño existe un RMSE menor con respecto a la magnitud de error que existe en las imágenes recuperadas con tamaños de bloque mayores. Tomando en cuenta que los valores de los pixeles varían de 0 a 255 y que el mayor valor de error es 9.579, entonces se puede decir que el error es aceptable. Además, se puede observar que las imágenes recuperadas de BB presentan menor error ante las otras imágenes en los tres escenarios y que la imagen CC recuperada con bloques de 32x32 es la que mayor error presenta, y esto se puede verificar en la visualización de las imágenes.

4.4.4 PSNR (Relación señal a ruido pico)

Con el método de evaluación PSNR se define la relación de señal a ruido que existe entre la imagen original y las imágenes recuperadas, al realizar este método objetivo se obtuvieron los resultados que se indican en la Tabla 11.

Tabla 11.
PSNR por tamaño de bloque.

TAMAÑO DE BLOQUE	AA [dB]	BB[dB]	CC[dB]	DD[dB]
8x8	11,5411	14,2849	10,7002	9,7149
16x16	8,7656	10,6589	8,2809	8,5277
32x32	7,796	8,6676	7,4497	8,0172

Al representar en forma gráfica los resultados se obtuvo la Figura 68.

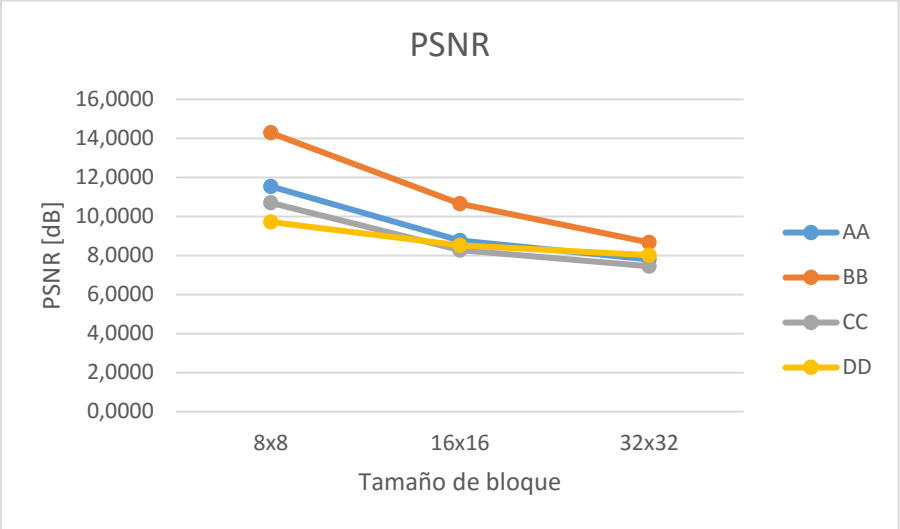

Figura 68. PSNR por Tamaño de bloque

El PSNR para ser aceptable debe estar en un rango de 30 a 50 dB para imágenes representadas con 8 bits como es el caso (Umme, Morium, & Mohammad, 2019), después de realizar el proceso de decodificación se pudo observar que el PSNR obtenido es mayor en las imágenes con bloques de menor dimensión, lo cual es lo más conveniente ya que mientras el valor de PSNR sea mayor mas se asemeja la imagen recuperada a la imagen original.

Para este caso el valor más alto de PSNR fue 14,2849 el cual no se encuentra dentro del rango aceptable para que la distorsión sea imperceptible al ojo humano, por lo cual las imágenes se asemejan, pero se nota una distorsión leve al comparar la imagen recuperada con la imagen original.

Por otro lado, en las imágenes que utilizan un tamaño de bloque mayor el valor obtenido de PSNR fue pequeño, por ejemplo, el peor caso fue en una imagen con bloques de 32x32 en el cual

se obtuvo un PSNR de 7,44 reflejándose estos valores en la distorsión que presenta la imagen recupera la cual es más perceptible por el ojo humano.

4.4.5 MSSIM (Similitud estructural media)

MSSIM es un método de evaluación objetivo que analiza la similitud que existe entre dos imágenes, tomando en cuenta la luminancia, contraste y estructura de las mismas (Yang, Gao, & Po, 2007). Al realizar esta prueba se obtuvieron los resultados que se presentan en la Tabla 12 y la representación gráfica en la Figura 69.

Tabla 12.
MSSIM por tamaño de bloque

TAMAÑO DE BLOQUE	AA	BB	CC	DD
8x8	0,7094	0,8342	0,6422	0,473
16x16	0,45015	0,6797	0,3829	0,3303
32x32	0,39	0,5966	0,301	0,2865

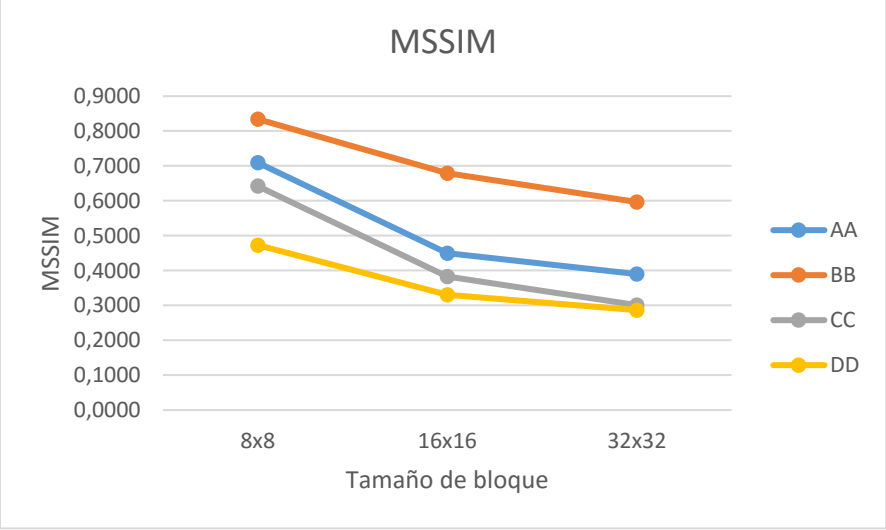

Figura 69. MSSIM por tamaño de bloque

En MSSIM mientras el valor más se acerque a 1, quiere decir que las imágenes son más similares, en este caso podemos observar que de igual forma que en los casos anteriores las imágenes recuperadas con bloques de tamaño 8x8 son las que mejores resultados presentan, ya que el mayor valor es 0.83 de la imagen BB con 8x8, mientras que el valor más bajo es 0.2856 de la imagen DD con 32x32. También, se puede ver que la imagen BB es la que en los tres casos obtiene un valor mayor de MSSIM.

4.4.6 MOS (Puntuación media de opinión)

El MOS se realizó a 28 personas, en grupos de 4, las cuales fueron ubicadas a una distancia de 2.5 metros con respecto a la pantalla, para la visualización de la aplicación en donde se muestran cuatro grupos de imágenes conformados por la imagen original y tres imágenes recuperadas con diferentes tamaños de bloque como se puede observar en la Figura 71.

Figura 70. Aplicación de MOS

En la Tabla 13 se presenta los valores promedios obtenidos al realizar el MOS para cada una de las imágenes recuperadas, tomando en cuenta que se presenta en una escala del 1 al 5 en donde 1 es malo y 5 es excelente.

Tabla 13.
Análisis MOS

TAMAÑO DE BLOQUE	AA	BB	CC	DD
8x8	4,214/5	4,714/5	4,464/5	4,107/5
16x16	2,607/5	3,107/5	2,393/5	2,134/5
32x32	1,25/5	1,429/5	1,036/5	1,179/5

La Figura 71 muestra los resultados de la Tabla 13.

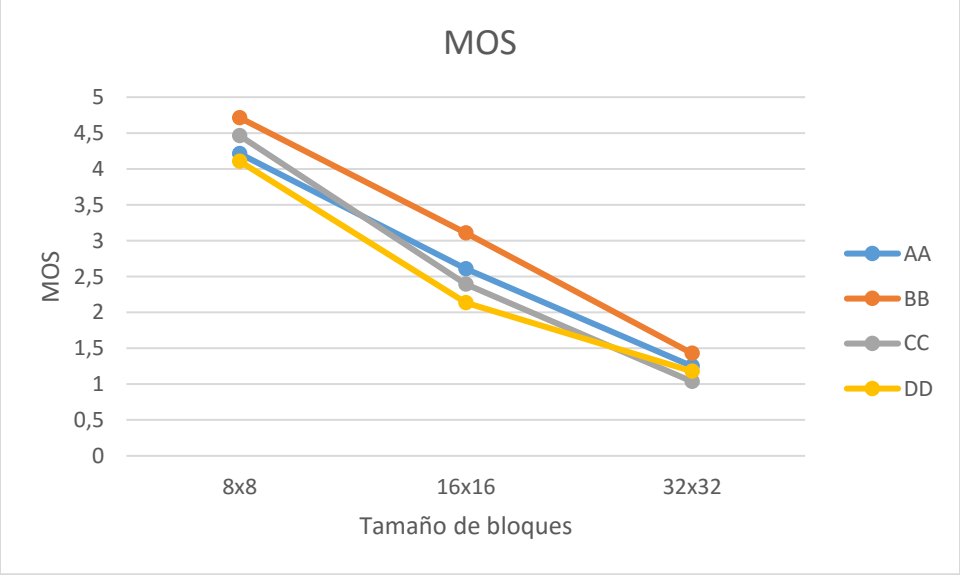

Figura 71. MOS por tamaño de bloque

En esta evaluación subjetiva se observa que las calificaciones para las imágenes recuperadas con bloques de 32x32 tienen una calidad entre mediocre y mala según la escala que se presentó en

la Tabla 3, mientras que en las imágenes recuperadas con tamaño de bloque de 8x8 tienen una aceptación entre buena y excelente, tomando en cuenta que ninguna de las imágenes fue calificada como excelente. Al realiza un análisis por imagen se puede apreciar también que la imagen BB es la que mejor resultado presenta en los tres escenarios.

CAPÍTULO V

5 CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Al implementar un STB con hardware de bajo costo se consiguió reproducir los canales de televisión digital utilizando el sintonizador PixelView, también se logró reproducir las aplicaciones de Ginga NCL, con el inconveniente de que la reproducción es de baja calidad y tiende a volverse lenta en determinados espacios de tiempo.

- Al realizar el proceso de decodificación el tiempo que tarda en ejecutar el programa depende de la información a procesar, ya que se pudo obtener como resultados que al realizar la recuperación de una imagen con bloques de 8x8 consume aproximadamente 94% más de tiempo que tarda en obtener una imagen con bloques de 32x32 y en cuanto a la relación de bits procesados en las imágenes de 8x8 utiliza 95% más bits que la cantidad utilizada en imágenes de 32x32.

- La recuperación de imágenes encriptadas presenta mejores resultados de acuerdo a los análisis objetivos realizados cuando ocupa un tamaño de bloque más pequeño, en este caso se utilizó bloques de 8x8 con lo que se obtuvo un RMSE menor en comparación a la recuperación de imágenes con bloques de 16x16 y 32x32, de igual forma a través del

MSSIM se verificó que se aproxima a 1, lo cual indica mayor similitud entre imágenes y un PSNR mayor que significa que existe menor presencia de ruido.

- Respecto a la decodificación de imágenes ninguna obtuvo una calificación excelente, pero las imágenes decodificadas con un tamaño de bloque de 8x8 tuvieron una mejor aceptación de parte de los participantes del MOS, debido a que la distorsión existente en las imágenes es perceptible para el ojo humano, lo cual se puede verificar mediante las evaluaciones objetivas como es el caso del PSNR en donde el rango aceptable es de 30 a 50 dB, en el presente trabajo ninguna imagen recuperada obtuvo un valor que entre en este rango ya que el valor más alto fue de 14,2849.

- Al realizar el proceso de decodificación se pudo observar las limitaciones que presenta la Raspberry Pi en cuanto al procesador y memoria RAM, ya que, al realizar el mismo proceso en una PC con mejores características, la decodificación es más eficiente ocupando un 85% menos de tiempo en la ejecución.

5.2 Recomendaciones

- Para mejorar la calidad de las imágenes recuperadas en los diferentes escenarios, se puede utilizar herramientas adicionales como filtros de media y filtros de mediana, los cuales permiten una disminución de errores y brindan una mejora en la calidad de la imagen.

- Implementar un método que permita que los valores de intensidad media no pierdan información en los residuos al momento de convertirlos en un número entero ya que esto genera un error mayor en la recuperación de la imagen secreta.

- Tomar en cuenta que las aplicaciones para televisión digital no deben ser invasivas, ya que al espectador le interesa ver la programación televisiva, y si se omite esta característica las aplicaciones poder pasar desapercibidas por los usuarios.

- Al momento de realizar la programación es necesario optimizar el código, debido a que al ejecutar en la Raspberry Pi el procesador no permite obtener un desenvolvimiento óptimo y el procesamiento de datos tiende a volverse lento, por lo tanto, mientras más datos se deban procesar el tiempo de ejecución será aún mayor.

5.3 Trabajos futuros

- Como trabajo futuro se propone realizar el proceso de decodificación obteniendo los datos de entrada de la señal de aire que se recibe para televisión digital terrestre, y de esta forma realizar el proceso en una aplicación en la que funcionen en conjunto Ginga NCL y LUA como lenguaje de programación para el proceso de decodificación, tomando en cuenta las características de hardware del receptor para que no se presenten limitaciones al momento de procesar la información para obtener las imágenes recuperadas.

REFERENCIAS

Acosta Buenaño, F. R., Mora Jiménez, I., Olmedo, G., & Rojo Alvarez, J. L. (12 de Noviembre de 2018). *Data Amount Reduction in Mosaic Image Transmission Techniques for Digital Interactive Television Applications.* Obtenido de IEEE Xplore: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8531605

Acosta, F. R., & Olmedo, G. (2014). GINGA-NCL Televisión Digital Terrestre. Ecuador. Recuperado el 25 de Marzo de 2019, de www.tvd.gub.uy/download.php?m=n&i=31

Alulema, D. (20 de Agosto de 2012). *La Televisión Digital Terrestre en el Ecuador es interactiva.* Obtenido de Revista UTE: https://revistas.ute.edu.ec/index.php/eidos/article/download/89/83

Casco, S. M. (Septiembre de 2014). *Raspberry Pi, Arduino y Beaglebone Black Comparación y Aplicaciones.* Obtenido de Jeuazarru: http://jeuazarru.com/wp-content/uploads/2014/10/MiniPCs.pdf

Chicaiza Jami, W. R. (2018). *Implementación de un Set Top Box híbrido para TDT e IPTV con middleware GINGA basado en componentes de bajo costo.* Recuperado el 17 de Febrero de 2019, de Repositorio ESPE: http://repositorio.espe.edu.ec/bitstream/21000/14102/1/T-ESPE-057658.pdf

Chie, S., Zambrano, M., & Medina, C. (s.f.). *Estándares actuales de televisión digital: Una breve reseña*. Recuperado el 07 de Mayo de 2019, de Revistas Académicas UTP: http://revistas.utp.ac.pa/index.php/prisma/article/view/606/html

Costa Campos, A., & Fernández Bozal, J. (2005). *La imagen digital.* Obtenido de Revista de Ortodoncia: http://www.revistadeortodoncia.com/files/2005_35_3_255-266.pdf

Cuéllar, J. C., Ortiz, J. H., & Arciniegas, J. L. (08 de Mayo de 2014). *Clasificación y Análisis de Métodos para medir Calidad de la Experiencia del Servicio de Televisión sobre Protocolo IP (IPTV).* Obtenido de SciELO: https://scielo.conicyt.cl/scielo.php?pid=S0718-07642014000500017&script=sci_arttext&tlng=e

Dávila Sacoto, M. A. (Enero de 2012). *Diseño de una plataforma de software para televisión digital interactiva de un canal de deportes utilizando Ginga-NCL Lua.* Obtenido de Dspace UPS: http://dspace.ups.edu.ec/handle/123456789/1736

DiBEG. (1 de Septiembre de 2009). *Transmisión de Televisión Digital Terrestre.* Obtenido de https://www.ar.emb-japan.go.jp/Notas/090901TelevisionDigital.pdf

Espinel Rivera, K. V. (2016). *Evaluación del rendimiento y prestaciones de un decodificador de contenidos interactivos basados en Ginga-NCL sobre una Raspberry PI.* Recuperado el Febrero de 2019, de Repositorio ESPE: http://repositorio.espe.edu.ec/xmlui/bitstream/handle/21000/11645/T-ESPE-053069.pdf?sequence=1&isAllowed=y

Fotomosaico. (2019). *Ejemplos de Fotomosaico*. Obtenido de FotoMosaico: https://fotomosaico.com/ejemplos/

Galabay Toalongo, P. T., & Vivar Espinoza, F. R. (Junio de 2012). *Manejo de software Ginga para el desarrollo de aplicaciones interactivas para televisión digital, basado en el estándar Brasileño ISDB-Tb.* Obtenido de Dspace UPS: https://www.lua.org/manual/5.1/es/manual.html

GitHub-Gregor Jansy. (s.f.). *V4L-utils*. Recuperado el 19 de Abril de 2019, de GitHub: https://github.com/gjasny/v4l-utils

Grupo de investigación Digitalac (UCAM). (s.f.). *Guía para el usuario de le televisión en alta definición*. Recuperado el Marzo de 2019, de Televisión Digital España: https://www.televisiondigital.gob.es/ayuda-ciudadano/Documents/GuiaHD.pdf

INEGI. (2005). *Guía para la interpretación de cartografía: fotografía aérea.* Obtenido de INEGI: http://internet.contenidos.inegi.org.mx/contenidos/Productos/prod_serv/contenidos/espanol/bvinegi/productos/historicos/1329/702825231750/702825231750_3.pdf

Khan, S. (2019). *Introducción a la estadística: media, mediana y moda*. Obtenido de Khan Academy: https://es.khanacademy.org/math/probability/data-distributions-a1/summarizing-center-distributions/v/statistics-intro-mean-median-and-mode

Lee, Y. L., & Tsai, W. H. (Abril de 2014). *A New Secure Image Transmission Technique via Secret-Fragment-Visible Mosaic Images by Nearly Reversible Color Transformations.* Obtenido de IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/6609044

Matplotlib. (18 de Mayo de 2019). Obtenido de Matplotlib: https://matplotlib.org/

Ministerio de Telecomunicaciones y de la Sociedad de la Información. (2018). *MINTEL presentó el "Plan Maestro de Transición a la Televisión Digital Terrestre (TDT) 2018-2021"*. Obtenido de Ministerio de Telecomunicaciones y de la Sociedad de la Información: https://www.telecomunicaciones.gob.ec/mintel-presento-plan-maestro-transicion-la-television-digital-terrestre-tdt-2018-2021/

Ministerio de Telecomunicaciones y de la Sociedad de la Información. (s.f.). *Bienvenida señal de la TV Digital*. Recuperado el 5 de Febrero de 2019, de Ministerio de Telecomunicaciones y de la Sociedad de la Información: https://www.telecomunicaciones.gob.ec/bienvenida-senal-de-la-tv-digital/

Ministerio de Telecomunicaciones y de la Sociedad de la Información. (s.f.). *La Televisión Digital Terrestre cada día avanza en el Ecuador*. Recuperado el 17 de Diciembre de 2018, de Ministerio de Telecomunicaciones y de la Sociedad de la Información: https://www.telecomunicaciones.gob.ec/television-digital-terrestre/#

MINTEL. (s.f.). *¿Qué es la TDT?* Recuperado el 17 de Diciembre de 2018, de Televisión Digital Terrestre Ecuador: https://tdtecuador.mintel.gob.ec/que-es-la-tdt/

Mordvintsev, A., & Abid, K. (05 de Noviembre de 2017). *OpenCV-Python Tutorials Documentation.* Obtenido de Readthedocs: https://buildmedia.readthedocs.org/media/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf

Moreno, E., García, C., Valero, M. A., Díaz, C., & Merino, V. (Junio de 2011). *Sistema abierto de Televisión Digital Terrestre (TDT) accesible para personas.* Obtenido de UPM: http://oa.upm.es/12438/1/INVE_MEM_2011_104823.pdf

Negrón Baez, P. A. (Diciembre de 2014). *Redes neuronales sigmoidal con algoritmo LM para pronostico de tendencia del precio de las acciones del IPSA.* Obtenido de PUCV: http://opac.pucv.cl/pucv_txt/txt-5500/UCE5728_01.pdf

Numpy desarrolladores. (2019). *Numpy*. Obtenido de https://www.numpy.org/

Olmedo, G., Acosta , F., Haro, R., Villamarín, D., & Benavides, N. (05 de Julio de 2019). *Broadcast Testing of Emergency Alert System for Digital Terrestrial Television EWBS in Ecuador.* Obtenido de Springer: https://link.springer.com/chapter/10.1007%2F978-3-030-23862-9_13

Ortiz Ramírez, A. (30 de Junio de 2010). *Python como primer lenguaje de programación.* Obtenido de Tecnológico de Monterrey: http://34.212.143.74/publicaciones/primer_lenguaje_30_jun_2010.pdf

Pavan Kalubandi, V. K., Hemanth , V., Vishnu , R., & Agilandeeswari , L. (16 de Marzo de 2017). *A novel image encryption algorithm using AES and visual cryptography*. Obtenido de IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/7877521

Pillajo , C. A., Ochoa, J. S., Acosta, F. R., & Olmedo, G. F. (10 de Noviembre de 2016). *Herramienta multiplataforma para generación automatica de aplicaciones interactivas*

Ginga-NCL basado en plantillas. Obtenido de Scielo: http://scielo.senescyt.gob.ec/pdf/maskay/v6n1/1390-6712-maskay-6-01-00008.pdf

Pisciotta, N. O. (Septiembre de 2010). *Bases de OFDM e ISDB.* Obtenido de UBP: http://www2.elo.utfsm.cl/~elo341/SistemaISDB_Tb.pdf

Prolink. (s.f.). *TV-SBTVD-1SEG*. Recuperado el 27 de Mayo de 2019, de Prolink: https://www.prolink-usa.com/productDetail.php?product_name=TV-SBTVD-1SEG

PyAr. (s.f.). *Comparación de entornos de desarrollo*. Recuperado el 27 de Mayo de 2019, de PyAr: https://www.python.org.ar/wiki/IDEs

Rodes Pastor, E. (2017). *Extensión del algoritmo de codificación por saltos logarítmicos, estudio de su impacto en la codificación Huffman y aplicaciónde éste para el desarrollo de códecs de video y audio.* Obtenido de UPM: http://oa.upm.es/48620/1/PFC_EDUARDO_RODES_PASTOR_2017.pdf

SciPy. (2019). *SciPy.org*. Obtenido de https://www.scipy.org/scipylib/index.html

Sotelo, R., Durán, D., & Joskowicz, J. (Enero de 2011). *Sistema de transmisión ISDB-T.* Obtenido de ResearchGate: https://www.researchgate.net/publication/277269466_Sistema_de_transmision_ISDB-T

Suárez Ibujes, M. O., & Tapia Zambrano, F. A. (07 de Noviembre de 2013). *Interaprendizaje de Estadística Básica.* Obtenido de Repositorio UTN: http://repositorio.utn.edu.ec/handle/123456789/2341

Takahashi, Y. (Junio de 2007). *Antecedentes técnicos de la recepción parcial de 1 segmento (One-seg).* Obtenido de DiBEG: https://www.dibeg.org/news/previous_doc/0706_3Argentina_ISDB-T_seminar/Argentina_ISDB-T_seminar_4_one_seg(Spanish)rev1.pdf

TDT Ecuador. (5 de enero de 2017). *Cese de las Emisiones de Señales Analógicas.* Obtenido de TDT - Televisión Digital Terrestre: https://tdtecuador.mintel.gob.ec/wp-content/uploads/2017/06/Resoluci%C3%B3n-No-CITDT-2017-01-062-Cese-de-las-Emisiones-de-Se%C3%B1ales-Anal%C3%B3gicas.pdf

TDT Ecuador. (s.f.). *Ecuador prepara el escenario para la llegada de la TDT*. Recuperado el 16 de Diciembre de 2018, de TDT - Televisión Digital Terrestre Ecuador: https://tdtecuador.mintel.gob.ec/antecedentes-tdt/#

TDT Ecuador. (s.f.). *La TDT lleva la tecnología a un nuevo nivel*. Recuperado el 16 de Diciembre de 2018, de TDT - Televisión Digital Terrestre Ecuador: https://tdtecuador.mintel.gob.ec/normativas-para-concesionarios-de-senal-abierta/

TeleMidia. (s.f.). *Ginga*. Recuperado el 19 de Abril de 2019, de GitHub: https://github.com/TeleMidia/ginga

Torres Altamirano, J. E. (2010). *Diseño y desarrollo de una aplicación de acontenidos inteactivos para TV Digital basad en el middleware Ginga del sistema Brasileño.* Obtenido de Repositorio ESPE: https://repositorio.espe.edu.ec/bitstream/21000/2647/1/T-ESPE-029809.pdf

Ubuntu MATE . (2019). *Ubuntu MATE*. Obtenido de https://ubuntu-mate.org/raspberry-pi/

Ubuntu MATE. (2019). *Ubuntu MATE*. Obtenido de https://ubuntu-mate.org/what-is-ubuntu-mate/

Umme, S., Morium, A., & Mohammad, S. U. (04 de Marzo de 2019). *Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study*. Obtenido de Scientific Research: https://file.scirp.org/Html/2-1730990_90911.htm

Valencia Melo, J. L. (29 de Abril de 2013). *Diseño y Desarrollo de Aplicaciones Interactivas para el Middleware GINGA de Televisión Digital de la Norma ISDB-Tb para brindar Información de los Protocolos de Prevención a la Población en Lugares de Alto Riesgo de Erupciones Volcánicas, Sismos y Tsunam.* Obtenido de Dspace EPN: http://bibdigital.epn.edu.ec/handle/15000/6218

Van Rossum, G. (Septiembre de 2009). *Turorial de Python.* Obtenido de Python: http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf.

Yang, C. L., Gao, W. R., & Po, L. M. (20 de Febrero de 2007). *Gradient-Based Structural Similarity for Image Quality Assessment.* Obtenido de IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/4107183