UCUENCA

Facultad de Ingeniería

Carrera de Electrónica y Telecomunicaciones

Implementación de un sistema autónomo rastreador de transmisiones no deseadas de bajo coste basado en RTL-SDR

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

Autor:

Geovanny Xavier García Monrroy

CI: 0105979314

Correo electrónico: ggarcia9539@gmail.com

Director:

Ing. Alcides Fabián Araujo Pacheco

CI: 0102358504

Cuenca - Ecuador

8 – Septiembre - 2022

Resumen

Este trabajo de titulación comprende el desarrollo de un algoritmo que permite el escaneo de ondas radioeléctricas ubicadas en los servicios de radio FM (88 MHz -108 MHz) y TV (UHF, VHF) e identificar transmisiones no deseadas, mediante radio definida por software, empleado el dispositivo RTL-SDR Blog v3. Los datos son procesados con Python y sus diferentes librerías como Pandas y Numpy, creando la infraestructura que procesa, manipula y analiza los datos, para diferenciar frecuencias autorizadas y no autorizadas o no deseadas, para este proceso se emplea operadores estadísticos como la correlación y comparadores de señal como la raíz del error cuadrático medio, permitiendo tener una medición de la similitud entre la señal escaneada real y una señal referencia. La información es presentada en el navegador mediante una interfaz web que se ha programado con Python, HTML y CSS. Para esta aplicación web el usuario necesita crear una cuenta y seleccionar las opciones de escaneo (FM o TV), al detectarse una señal no deseada se activará una alerta mostrando la frecuencia y potencia de la señal no deseada. La interfaz web se ejecutará en una Raspberry Pi 3 que será el servidor y ejecutará las ordenes de escaneo y procesamiento de datos, además de conectarse con la base de datos Google Cloud Storage y almacenar en la nube todas las alertas de transmisiónes no deseadas.

Palabras clave : RTL-SDR. Pandas. Correlación. Interfaz Web.

Abstract

This graduate degree thesis pertains to the development of an algorithm that scans radio wave spectra within the FM Radio (88 MHz -108 MHz) and TV (UHF, VHF) bandwidths, with the use of software defined radio, specifically using a RTL-SDR Blog V3 device. The data are processed in Python with the use of the Pandas and Numpy Python libraries. These elements together constitute the infrastructure that processes and analyzes the data with the goal of discerning between legal and illegal radio transmissions. Necessary for this process was the use of statistical operators such as correlation as well as signal comparators like the root mean square error, to serve as metrics of the similarity between a scanned signal and a reference signal. All this information is presented through a web browser interface by way of a web interface programmed in Python, HTML and CSS. Within the web interface a user first needs to create an account. Once logged in, the user then selects the scanning bandwidth (FM or TV) in which the system will detect any undesired signals. The web application runs on a Raspberry Pi 3 which acts as the web server. This server then initiates the scanning and data processing commands, as well as connect to a Google Cloud Storage database in order to store any alerts of illegal radio transmissions in the cloud.

Keywords : RTL-SDR. Pandas. Correlation. Web Interface

Índice general

Índice de figuras

Índice de tablas

Cláusula de Propiedad Intelectual

Yo, Geovanny Xavier García Monrroy, autor de la tesis "Implementación de un sistema autónomo rastreador de transmisiones no deseadas de bajo coste basado en RTL-SDR ", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 8 de septiembre de 2022

Geovanny Xavier García Monrroy
010597931-4

UCUENCA

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo, Geovanny Xavier García Monrroy en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Implementación de un sistema autónomo rastreador de transmisiones no deseadas de bajo coste basado en RTL-SDR ", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 8 de septiembre de 2022

Geovanny Xavier García Monrroy
010597931-4

Certifico

Que el presente proyecto de tesis: Implementación de un sistema autónomo rastreador de transmisiones no deseadas de bajo coste basado en RTL-SDR , fue dirigido y revisado por mi persona.

Firmado electrónicamente por:
ALCIDES FABIAN
ARAUJO PACHECO

Ing. Alcides Fabian Araujo Pacheco, MsC
Director

Dedicatoria

Este trabajo de titulación está dedicado a mis padres Raúl García y Florencia Monrroy que, con su dedicación, paciencia, respeto y amor cultivaron dentro de mí la curiosidad intelectual y el amor por las ciencias, dándome grandes lecciones de vida.

A mis hermanos Daniel y Paulo que estuvieron siempre conmigo apoyándome y dándome consejos, estando incondicionalmente en los buenos y malos momentos. Finalmente, dedicar este trabajo de titulación a todos mis amigos que a lo largo de mi vida se han convertido en un apoyo emocional.

Geovanny Xavier García Monrroy

Agradecimientos

Primeramente, agradezco a Dios por abrirme las puertas a la sabiduría y el conocimiento, por ponerme en el momento exacto para elegir una Ingeniería como meta profesional. También un agradecimiento a todos mis maestros secundarios y universitarios que con su pasión por enseñar crearon dentro de mi ese espíritu de nunca parar de aprender.

Finalmente agradezco al Ing. Alcides Araujo, director de este trabajo de titulación, quien, con sus conocimientos y dirección me permitió desarrollar los diferentes algoritmos de este trabajo de titulación.

EL AUTOR

Abreviaciones y Acrónimos

ADC Analogic Digital Converter. 28–30

CMD command. 32
CPU Central Processing Unit. 30

DAC Digital Analogic Converter. 29, 30
DDC Digital Download Converter. 30
df DataFrame. 17, 46
DSP Digital Signal Processing. 30
DUC Digital Upload Converter. 30
DVB-T Digital Video Broadcasting Terrestrial. 31

ETL Extract Transform Load. 41

FFT Fast Fourier Transform. 59

HDTV High-definition television. 31

IF Intermediate frequency. 29, 30

NoSQL Non SQL. 40

PCB Printed Circuit Board. 32
PPM parts per million. 32
PSD Power Spectral Density. 37, 45, 46, 59, 64–66

RF Radio Frecuencia. 29, 30
RMS Root Mean Square. 23
RMSE Root mean squared error. 17, 38, 40, 47, 48, 63, 69
RTL-SDR Realtek Software defined Radio. 19, 20, 26, 27, 31, 32, 37, 38, 43, 45, 55, 56, 59, 63, 69, 70, 78, 79

SMA SubMiniature version A. 31, 32

TCXO Temperature compensated crystal oscillator. 32

UHF Ultra High Frequency. 16, 85

Capítulo 1

Introducción

En este capítulo se expone la identificación del problema, justificación, estado del arte y estructura del proyecto.

1.1. Identificación del problema

El monitoreo del espectro radioeléctrico emplea equipos tecnológicos que facilitan el control y la regularización de la utilización del mismo. En el Ecuador el ente regulador de las telecomunicaciones es la , institución que dispone de equipos capaces de identificar señales radioeléctricas no deseadas a partir del análisis del espectro radioeléctrico en cada región del país. Sin embargo, la adquisición de estos equipos representa altos costos de inversión, por lo que la búsqueda de alternativas con mecanismos tecnológicos más recientes, podría introducir mejoras en las tareas técnicas de fiscalización del espacio radioeléctrico llevadas a cabo por los entes competentes.

El problema surge con la necesidad de identificar señales ilegales o desconocidas de forma masiva, pues hace que los recursos empleados para el control del espectro radioeléctrico sean insuficientes y los costos sean aún mayores. Por lo que este trabajo de titulación expone una solución tecnológica de bajo coste para la identificación de señales no deseadas a partir de Radio Definida por Software o y software libre.

Al emplearse la tecnología los costes de inversión en tareas técnicas de monitoreo y control se reducirán, además de permitir la fácil lectura del espectro radioeléctrico. Sumado a lo anterior, el lenguaje de programación Python permite que los datos obtenidos con el sean procesados de manera eficiente y de forma sencilla debido a que dispone de varias librerías que permiten un fácil manejo de los datos. En adición, otras tecnologías como la Raspberry Pi y lenguajes como y nos permiten crear una visualización e identificación del espectro mucho más simple y detallada. Por este motivo es necesario que en este trabajo de titulación sean expuestos los detalles técnicos de todas las tecnologías empleadas tanto en *hardware* como en *software*.

1.2. Justificación

El impacto de la pandemia de 2020 aceleró en gran medida el desarrollo de software e infraestructura para los requerimientos de conectividad masiva que surgieron en el mundo. Esto puso en evidencia la necesidad de maximizar la eficiencia y el uso del espectro radioeléctrico. Uno de los mecanismos que garantiza el uso eficiente del espacio radioeléctrico es la regularización del mismo, actividad que es atendida por las diferentes naciones y organismos internacionales del mundo.

El Estado ecuatoriano define al espacio radioeléctrico como un recurso natural limitado cuya administración para el uso y aprovechamiento técnico se ejercerá a través de la autoridad de telecomunicaciones. Las tareas de monitoreo y control del espacio radioeléctrico son llevadas a cabo por la y se realizan con equipamiento especial que permite detectar señales no deseadas, interferencias u operadores ilegales que puedan causar perjuicios técnicos o económicos a la nación.

De acuerdo a la publicación de ARCOTEL con fecha a abril de 2022 con respecto al número de estaciones concesionadas de radiodifusión sonora FM y TV en el Ecuador se obtienen los siguientes datos. Existen 1187 estaciones concesionadas de Radiodifusión Sonora FM. De esa cantidad las provincias con mayor demanda de concesiones son Loja, Manabí y Azuay. Con respecto al número de estaciones concesionadas de televisión por provincia y por frecuencia (Ultra High Frequency (UHF),) con corte a noviembre de 2021, se establece que existen 428 canales de televisión analógica y digital siendo las provincias con mayor demanda de estos espacios Pichincha, Manabí y Azuay. Con esta información se facilita la elaboración de modelos de gestión para la comprobación técnica del espectro radioeléctrico a nivel nacional. Estos datos permiten una mejor identificación del número de canales que están concesionados legalmente de aquellos que no lo están.

Dada la información anterior y sumado al hecho de que la comprobación técnica del espectro radioléctrico nacional utiliza recursos como equipos y personal que representan mayores costos de inversión, para la realización de esta tarea se establece la necesidad de perfeccionar este tipo de procesos a través de las nuevas tecnologías, es decir métodos que permitan un control más eficiente y automatizado de las posibles interferencias o señales radioeléctricas no deseadas.

1.3. Alcance

Este trabajo de titulación tiene como objetivo crear un algoritmo capaz de detectar trasmisiones no deseadas utilizando un dispositivo de Radio Definida por Software denominado RTL-SDR Blog V3 para el escaneo de las frecuencias en los espectros de radio y televisión y UHF. El lenguaje de programación a emplear para el procesamiento de estas señales, toma de decisiones y diseño del algoritmo principal del proyecto será Python conjuntamente con las librerías más utilizadas en analítica de datos como lo son Pandas, Numpy y Scikit Learn.

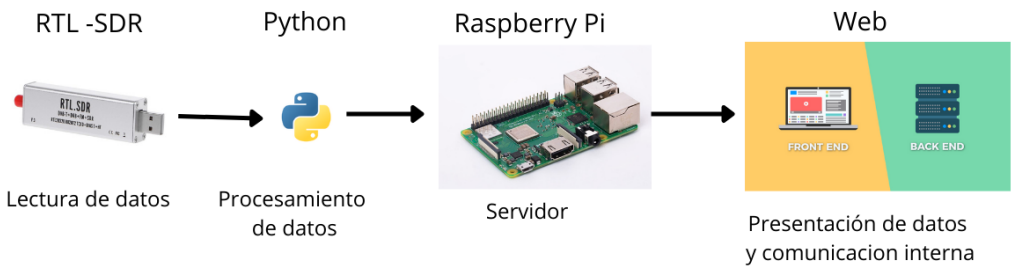

Figura 1.1: Esquema del proyecto

En la figura 1.1 se observa el esquema que sigue el proyecto en donde se aprecia que la parte inicial es la toma de datos del espectro radioeléctrico captado con el dispositivo RTL-SDR. Estos datos pueden ser leídos y procesados una vez que se hayan instalado todos los drivers del en el computador. Siguiendo con el proceso el lenguaje de programación Python junto a la librería Pyrtlsdr permite leer los datos y almacenarlos en una variable de Python para un posterior análisis. Estos datos almacenados en memoria nos permitirán procesarlos con librerías específicas de Python como Pandas, este procesamiento incluye una organización y filtrado de los datos en un DataFrame (df) que es la estructura básica de Pandas. Posteriormente al filtrado y ordenamiento de los datos le seguirá un proceso de toma de decisiones donde operadores estadísticos como la correlación y el Root mean squared error (RMSE) permitirá identificar la transmisión radioeléctrica no deseada. Las librerías usadas para este propósito son Numpy y Scikit Learn, por su fácil manejo de los datos. Para el servidor se empleará una Raspberry Pi 3; esta recibirá todas las peticiones de escaneo de frecuencias y se las procesará conjuntamente con el RTL-SDR. En el caso de la interfaz de usuario se empleará Flask, que es un *framework backend* de Python para la comunicación entre el servidor y el frontend de la interfaz web. Finalmente, la interfaz del usuario estará desarrollada con HTML y CSS que nos permitirán un fácil manejo del algoritmo creado.

El sistema implementado, será comprobado en el laboratorio, enviando una transmisión en las frecuencias de FM o TV en diferentes contextos. Como ejemplo se tomarán diferentes transmisores y se modificará la potencia de transmisión, aumentando o disminuyendo la distancia entre el SDR y el transmisor. Esto con el fin de documentar todas las pruebas realizadas y desarrollar un análisis con todos los resultados positivos del algoritmo, así como sus falsos positivos.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar e implementar un sistema autónomo rastreador de transmisiones no deseadas de bajo coste usando RTL-SDR, Raspberry Pi y software libre.

1.4.2. Objetivos específicos

- Analizar el estado del arte.
- Implementar una arquitectura de hardware de bajo coste capaz de recibir transmisiones de radio.
- Diseñar un algoritmo capaz de procesar las señales receptadas y distinguir transmisiones no deseadas.
- Crear una base de datos con toda la información de las transmisiones que nuestra arquitectura ha receptado.
- Diseñar e implementar un servidor web que muestre la información sobre las señales procesadas con una interfaz amigable para el usuario.
- Implementar un sistema de alertas que notifique al usuario las transmisiones no deseadas.

1.5. Estado del Arte

En el presente apartado se hace una revisión exhaustiva de investigaciones que tratan nociones sobre la detección de transmisiones radioeléctricas mediante el censo del espectro electromagnético. La finalidad de la investigación es dejar en evidencia la ausencia de información relativa a la detección de señales electromagnéticas no deseadas usando *hardware* y *software* libre.

Inicialmente se revisarán bibliográficamente los mejores métodos para la detección de señales electromagnéticas dependiendo del medio. Seguidamente se identificarán los métodos más eficaces para la detección de las señales electromagnéticas en el espacio radioeléctrico. Posteriormente se expondrán los estudios existentes sobre la implementación de *hardware* y *software* libre para el procesamiento de señales electromagnéticas y finalmente se abordará un acercamiento a las nuevas técnicas de procesamiento de señales de radio usando *Deep Learning*.

A continuación se desarrollará una breve explicación de los estudios realizados dentro del campo de estudio de detección de señales en el espectro electromagnético:

- **Raspberry Pi and RTL-SDR for Spectrum Sensing based on FM Real Signals** [1]: El estudio muestra técnicas de medición del espectro radioeléctrico empleando software libre, definiendo que la detección por energía es el método más eficiente debido a que compara un umbral (energía del ruido en el canal); cuando la medición de energía en la señal detectada excede el umbral, la señal ha sido emitida por un usuario, caso contrario no ha sido emitida. Aquí se compara con el método de *Matched Filter Detection* en donde se necesita cierta información de la señal enviada, como la frecuencia de la portadora, la forma de los pulsos y el tipo de modulación empleada, siendo estos requerimientos inaccesibles cuando se requiera detectar señales no deseadas. También se realiza una comparación con el método *Cyclostationarity based detection*, donde para la detección de las señales se necesitan un convertidor analógico – digital de alta precisión , debido a que este método auto correlaciona la señal para detectar periodicidades escondidas en el espectro. Por lo que el método de *Energy Detection* es el óptimo para realizar una detección de las señales. El

desempeño del *Energy Detector* puede ser caracterizado usando dos probabilidades, la probabilidad de detección y la probabilidad de una falsa alarma en la detección. Finalmente, el investigador concluye que el método de *Energy Detection* es el método más apropiado para el censo del espectro electromagnético debido a la complejidad baja de implementación y que no requiere información de las propiedades de la señal enviada.

De acuerdo a la investigación ejecutada [2] existen diferentes técnicas usadas para el censo del espectro electromagnético, estas se dividen en: *Energy Detection* (ED), *Matched Filter Detection* (MFD) y *Cyclostationarity based detection* (CSD)

- **Energy Detection Spectrum Sensing on RTL-SDR based IoT Platform** [3]: Estudio que muestra las aplicaciones del pueden emplear tecnología de para dar un acceso más dinámico a los nodos IoT al espectro radioeléctrico. Para medir la energía del espectro radioeléctrico el autor emplea un nodo implementado en una Raspberry Pi y un Realtek Software defined Radio (RTL-SDR), esta información permite comparar el espectro radioeléctrico medido en el nodo con el espectro calculado con la teoría. Las mediciones del espectro, empleando plataformas como Raspberry Pi, RTL-SDR y software libre, permiten el desarrollo de aplicaciones de monitoreo del espectro para identificar bandas de frecuencia libre.

  En el estudio [4] se definen metodologías para el procesamiento de grandes cantidades de datos del espectro radioeléctrico, empleando una arquitectura digital basada en *Deep Learning* permite presentar al autor un nuevo enfoque para el procesamiento de datos, creando modelos de procesamiento de datos basados en *Deep Learning*.

  En la tecnología al emplear el espectro radioeléctrico es necesario definir las bandas de frecuencia libres y ocupadas para evitar una interferencia de señales, por ende, es necesario definir técnicas de y procesamiento de grandes cantidades de datos, para optimizar el uso del espectro.

- **Big Data Processing Architecture for Radio Signals Empowered by Deep Learning: Concept, Experiment, Applications and Challenges** [5]: El *Big Data* para señales de radio no solamente necesita de la señal a estudiar, si no también necesita del manejo y uso de los datos históricos para extraer valor de la gran cantidad de datos. Además, con el continuo crecimiento de los emisores de radio y el incremento de la densidad de cobertura del espectro, es necesario un incremento de regulaciones en el uso del espectro. Para realizar regulaciones más eficaces, los departamentos de regulación necesitan identificar fuentes de interferencia, estaciones ilegales y radio bases maliciosas para que a través del monitoreo del espectro electromagnético se detecte comportamientos anormales en el espectro, existiendo 3 tipos de comportamientos anormales:

  - Usuarios que por fallas en el equipo o usuarios que por maximizar el desempeño de sus comunicaciones utilizan el espectro de manera incorrecta a las regulaciones estipuladas en cada país, como transmitir a una potencia más elevada de los niveles de operación permitidos, siendo este comportamiento anormal detectado mediante la estimación de los parámetros de la señal recibida.
  - Usuarios ilegales que usan formas de ondas diferentes a las permitidas como por ejemplo que maliciosamente corrompen las comunicaciones.

- Usuarios ilegales que imitan sistemas de usuario autorizado, como por ejemplo estaciones de radio ilegales.

Según el estudio [6] en los años recientes, los nuevos métodos de comunicación inteligente como las radios cognitivas requieren receptores de radio que posean capacidades de censo del espectro electromagnético de banda ancha, debido a esto se necesita cierto grado de capacidades de procesamiento de grandes cantidades de datos,campo de estudio conocido como *Big Data*, para mejorar el entendimiento del ambiente que rodea al espectro electromagnético, optimizando los sistemas de comunicación, los recursos de red y sobretodo mejorando el desempeño de las comunicaciones inalámbricas.

De acuerdo al estudio [7] el *Deep Learning* ha tenido remarcables progresos en los recientes años y sus aplicaciones están en todas las industrias y los campos de investigación. En el campo del procesamiento de señales electromagnéticas, la aplicación del *Deep Learning* está recién comenzando y recientes aplicaciones incluyen clasificaciones de señales de radio (clasificación de modulación), RF *fingerprinting* y estimaciones del canal.

Después de realizar una amplia revisión sobre el tema de interés para este proyecto de titulación es posible concluir que no existe información clara sobre la detección de señales no deseadas usando herramientas de fuente libre dado que los estudios principalmente se han centrado en el monitoreo del espectro electromagnético en base a regulaciones internacionales. Adicionalmente, no se conocen herramientas que permitan la detección de señales de radio no deseadas usando Raspberry Pi ni RTL-SDR.

1.6. Estructura de tesis

El siguiente trabajo de titulación está estructurado de la siguiente manera: El capítulo 1 aborda los antecedentes que permiten plantear el problema, la justificación y el alcance de la investigación. Se incluyen los objetivos a desarrollar, así como la metodología sobre la cual se va a fundamentar el trabajo. Seguidamente en el capítulo 2 se abordarán todos los conceptos básicos sobre la radio definida por software, información sobre el hardware utilizado, así como el software necesario para un correcto funcionamiento en cualquier computador. Cabe mencionar que se hará énfasis en los conceptos básicos del y las características técnicas del RTL-SDR blog v3, nociones que permitirán un mejor entendimiento de la tecnología sobre la cual se sustenta este estudio.

A continuación, en el capítulo 3 se expondrá el algoritmo desarrollado en Python que permitirá la detección de una señal radioeléctrica no deseada. Conjuntamente se darán a conocer conceptos claves como los operadores estadísticos usados y la lógica de programación empleada para la construcción del algoritmo. También se desarrollarán los contenidos en cuanto a la programación para el *Backend* y *Frontend* de la aplicación. En el capítulo 4 se expondrán las pruebas de laboratorio realizadas para corroborar el funcionamiento del algoritmo. Para el propósito de esta investigación se analizarán los resultados de las pruebas con distintas configuraciones del algoritmo que permitirán identificar el mejor escenario en donde el algoritmo sea capaz de detectar sin problemas una transmisión ilegal. Finalmente, en el capítulo 5 se presentan las conclusiones, recomendaciones del trabajo de titulación.

CAPÍTULO 2

Fundamentos y técnicas de medición del espectro radioeléctrico

El espectro radio eléctrico es el conjunto de ondas electromagnéticas por debajo de los 3000 GHz que se propagan por el espacio sin necesidad de una guía artificial. La Ley Orgánica de Comunicación del Ecuador [8] define al espectro radioeléctrico como un bien de dominio público del Estado y un recurso natural limitado. Se constituye un medio intangible sujeto a la regularización y control por parte del Estado [9]. Los rangos de frecuencias dependen de la longitud de onda de las señales y sus características de propagación. A continuación, se muestra en la tabla 2.1 la clasificación de bandas especificada en :

Tabla 2.1: Clasificación de bandas frecuencias de la IEEE

Rango de Frecuencias	Longitud de onda	Banda IEEE
300 KHz – 3 MHz	1 km a 100 metros	MF
3 MHz – 30 MHz	100 metros a 10 metros	HF
30 MHz – 300 MHz	10 metros a 1 metros	VHF
300 MHz – 3GHz	1 metros a 10 cm	UHF
1 GHz - 2 GHz	30 cm a 15 cm	Banda L
2 GHz – 4 GHz	15 cm a 5 cm	Banda S
4 GHz - 8 GHz	5 cm a 3.75 cm	Banda C
8 GHz – 12 GHz	3.75 cm a 2.5 cm	Banda X
12 GHz – 18 GHz	2.5 cm a 1.6 cm	Banda Ku
18 GHz – 26 GHz	1.6 cm a 1.2 cm	Banda K
26 GHz – 40 GHz	1.2 cm a 750 mm	Banda Ka
40 GHz – 75 GHz	750 mm a 40 mm	Banda V
75 GHz – 111 GHz	40 mm a 28 mm	Banda W

El rango de frecuencias mediante las cuales se transmite la información es denominado ancho de banda y depende directamente de la cantidad de información a transmitirse. Las telecomunicaciones son la tecnología que usa el espectro radioeléctrico por defecto, debido a esto la asignación de un rango de frecuencias por donde la información será transmitida es de vital importancia para su adecuada gestión. En la figura 2.1 se observa el rango de frecuencias asignado para cada banda y en qué servicio de telecomunicaciones son empleados:

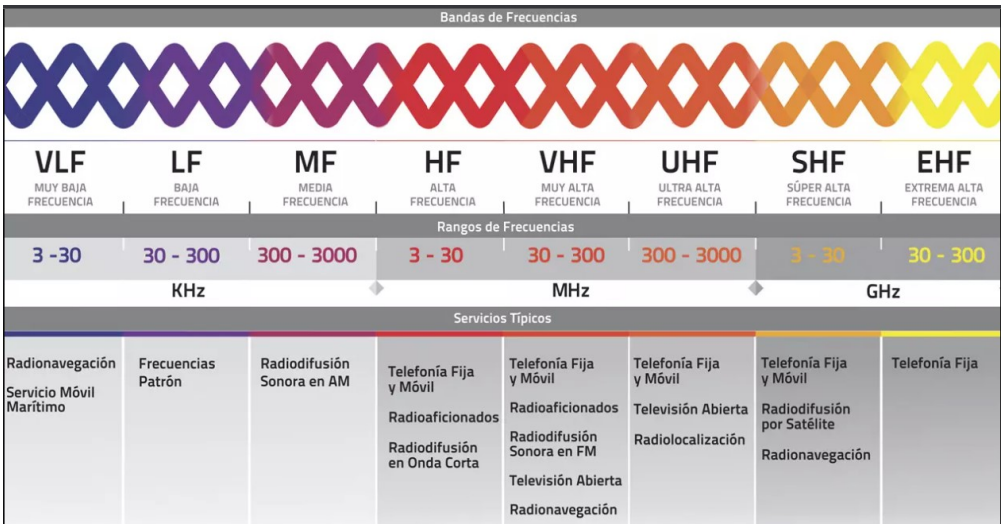

Figura 2.1: Clasificación y usos del espectro radioeléctrico [10]

2.1. Parámetros técnicos del uso del espectro radioeléctrico

Los parámetros técnicos empleados en el uso del espectro radioeléctrico son los siguientes:

- **Frecuencia:** Indican el número de oscilaciones completas de una función periódica en el tiempo por unidad de tiempo, su unidad es el Hertz (Hz).
- **Propagación:** Habilidad que tiene para propagarse una onda radioeléctrica sin la necesidad de una guía artificial, el alcance de propagación depende principalmente de los parámetros técnicos de potencia y frecuencia, siendo que a menor frecuencia se tiene una menor atenuación y por ende una mayor distancia de cobertura.
- **Capacidad de la banda de frecuencia:** Cada banda de frecuencias tiene su propia capacidad de transmisión, siendo que a frecuencias menores la cantidad de información que puede transmitirse es menor y a frecuencias más altas, la capacidad de transmisión es mayor.
- **Ancho de banda:** Especifica la cantidad de información que puede transmitirse, a una velocidad determinada y calidad requerida en condiciones específicas.
- **Potencia:** Parámetro ligado a la infraestructura que emite señales radioeléctricas dadas las características de propagación
- **Interferencia:** Las transmisiones de ondas radioeléctricas ocasionan interferencia en los receptores, donde es necesario medir cuanto se degrada la calidad o cuanta información se pierde. [11]

2.2. Medición del espectro radioeléctrico

Para el uso del espectro radioeléctrico son necesarios equipos que permitan medir con precisión el conjunto de ondas radioeléctricas que se pretenden emplear; estas mediciones son empleadas para verificar el correcto funcionamiento de la transmisión y recepción de estas ondas radioeléctricas. El equipo de medición electrónico empleado para estos propósitos es el analizador de espectro, que se define como un dispositivo electrónico de frecuencia selectiva que mide el máximo voltaje durante un ciclo de frecuencia (*Peak- Responding Voltmeter*) configurado para mostrar los valores Root Mean Square (RMS) de una onda sinusoidal. Es importe aclarar que el analizador de espectro no es un medidor de potencia, sin embargo, puede mostrar los valores de potencia de las señales, siempre que se conozcan los valores máximos o promedio de las ondas sinusoidales tratadas y también la resistencia con la que se mide este valor; a partir de estas variables se puede configurar el voltímetro como un indicador de potencia. Para la medición de cualquier onda eléctrica la unidad de referencia es el tiempo, pero para propósitos de uso del espectro radioeléctrico es necesario visualizar las ondas radioeléctricas en referencia a su frecuencia, ya que las mediciones en el dominio de la frecuencia indica que tanta energía está presente en cada frecuencia en particular. Para trasformar del dominio temporal al dominio de la frecuencia es necesario emplear la transformada de Fourier, que indica que cualquier fenómeno eléctrico en el dominio del tiempo puede ser generado mediante ondas sinusoidales con la apropiada frecuencia, amplitud y fase. Es importante recordar que el espectro radioeléctrico se define como la colección de ondas sinusoidales que combinadas apropiadamente producen señales en el dominio del tiempo y que con la transformada de Fourier permite transformarlas al dominio de la frecuencia.

La medición del espectro radioeléctrico es importante para toda la industria de las comunicaciones inalámbricas, ya que el éxito de las trasmisiones generadas radica en medir las emisiones fuera de banda y espurias, un ejemplo sería un sistema de radio celular en el que se verifica que armónicos de su señal portadora no generen interferencia en otros sistemas operando en la misma frecuencia de estos armónicos. Existen diferentes tipos de analizadores de señal, según sea la necesidad de medición se tiene:

- **Analizador de espectros:** Mide la magnitud de la señal de entrada versus la frecuencia dentro del rango de frecuencias admisibles del equipo. Su uso principal radica en mostrar y medir la Amplitud versus Frecuencia de señales y microondas.
- **Analizador vectorial de señales:** Mide la magnitud y fase de la señal de entrada de una frecuencia en particular.
- **Analizador de Señales:** Combina las funciones de un analizador de espectro y un analizador vectorial de señales.

Los fundamentos para el funcionamiento de un analizador de espectros clásico se muestran en la figura 2.2 donde se muestra el diagrama de bloques de un analizador de espectro superheterodino. Heterodino significa la mezcla o traslado de frecuencias y súper indica que es apto para frecuencias por encima del rango auditivo.

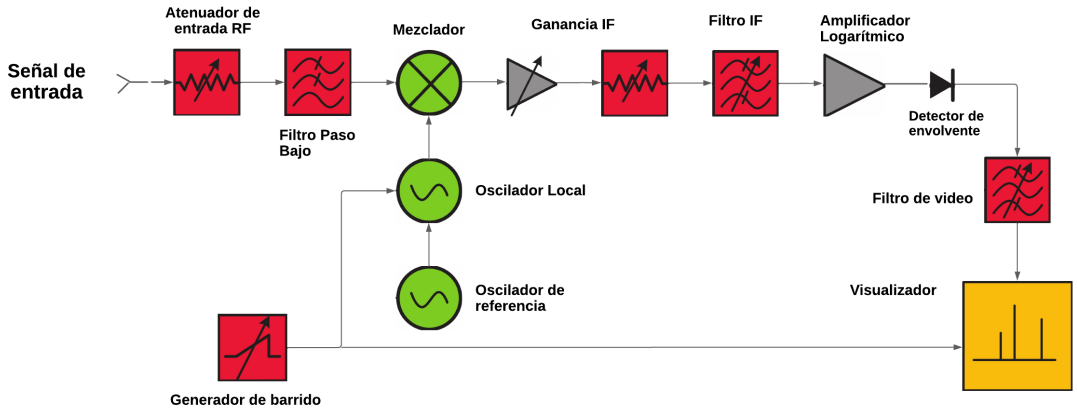

Figura 2.2: Diagrama de bloques de un analizador de espectros superheterodino [12]

En el diagrama se observa una señal de entrada que pasa a través de un atenuador. El propósito del atenuador es asegurar que la señal ingrese al mezclador en un nivel óptimo para prevenir una sobrecarga y distorsión de la señal de entrada, después pasa a través de un filtro paso bajo. Posteriormente va a un mezclador donde se combina con la señal proveniente del oscilador local (LO). Debido a que el mezclador es un dispositivo no lineal su salida no solamente incluye las señales originales sino también sus armónicos y la suma y diferencia de las frecuencias originales y sus armónicos. Si cualquiera de las señales provenientes del mezclador se atenúa con el filtro pasa banda de la frecuencia intermedia, es procesada con un amplificador. En la fase final la señal es rectificada por el detector de envolvente, filtrada a través del filtro paso bajo y finalmente mostrada en pantalla del analizador de espectro.

A partir de los años 80s, uno de los cambios más profundos en el análisis del espectro ha sido la aplicación de tecnología digital para reemplazar los diagramas de bloques anteriormente mostrados, que generalmente eran desarrollados con circuitos analógicos. La aplicación de circuitos digitales ha permitido el desarrollo de analizadores de espectro con más precisión al momento de medir la amplitud y frecuencia de la señal, así como abaratamiento de costes de producción y masificación. [13]

2.3. Técnicas de medición del espectro radioeléctrico

Las técnicas empleadas por defecto en la medición del espectro radioeléctrico:

- **Observación:** La técnica más simple en la medición del espectro radioeléctrico es situarse en las frecuencias deseadas, configurar el analizador de espectro y observar la parte deseada del espectro de radiofrecuencia. Esto permite que se manipule el analizador de espectro para configurarlo con los mejores resultados.
- **Registrar y Examinar:** Es necesario registrar los valores de las frecuencias deseadas, para un análisis posterior en donde el objetivo es examinar el ancho de banda, la forma y el uso del espectro radioeléctrico de la señal.
- **Registrar y Comparar:** Otra técnica de medición efectiva del espectro radioeléctrico es registrar el espectro una vez y comparar ese mismo espectro capturado después de un tiempo. Esta técnica es usada para identificar nuevas señales o para identificar señales que ya no están presentes. [12]

La gestión y administración del espectro radioeléctrico tiene técnicas de medición que incluyen procedimientos de gestión administrativas acorde a la legislación de cada país, por ello para un uso eficiente del espectro radioeléctrico se debe garantizar que los operadores y técnicos que miden y regulan el espectro radioeléctrico posean la información adecuada sobre los procesos administrativos y técnicos empleados en la medición del espectro. Este monitoreo del espectro radioeléctrico es necesario ya que la autorización del uso del espectro no garantiza que este se utilice de manera correcta como se ha previsto por la legislación. [14]

El diagrama de bloques de la figura muestra 2.3 la técnica empleada para una medición correcta, registro y análisis del espectro radioeléctrico; el primer bloque indica un análisis del entorno, es decir, identificar en el entorno donde se va a realizar la medición, emisiones radioeléctricas de las antenas más próximas y de ser posible determinar qué servicio están prestando (telefonía móvil, internet, FM o TV) para identificar el rango de frecuencias en el que estos servicios estarán ubicados. Posteriormente el bloque de configuración del analizador de espectro incluye pasos como la configuración de la ganancia del analizador, *offset*, frecuencia central, polarización y configuración de las antenas receptoras del espectro, esto permitirá una lectura del espectro eficiente por parte del analizador de espectros evitando lecturas erróneas o con emisiones espurias.

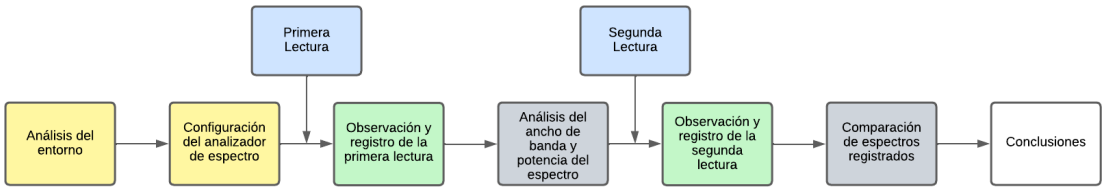

Figura 2.3: Diagrama de bloques de una estructura de técnicas de medición del espectro

Analizado el entorno y configurado el analizador de espectro se procede a realizar una primera lectura del espectro radioeléctrico. En esta etapa, la observación objetiva del operador es necesaria para identificar patrones en el espectro radioeléctrico de la frecuencia asignada, consecuentemente el operador debe identificar las características de este espectro radioeléctrico como el ancho de banda y la potencia máxima del espectro, para en un siguiente paso registrar todos estos datos y realizar un primer análisis en base a la información obtenida. Este análisis puede incluir una comparación de las características registradas con las pautas que se han asignado en la regulación. Una segunda lectura del mismo espectro radioeléctrico después de un tiempo puede asegurar una correcta medición del espectro, esta segunda lectura permite la comparación de las características registradas de la lectura anterior, permitiéndo identificar nuevas señales o señales que ya no estén presentes.

El diagrama de bloques de la figura 2.3 muestra la etapa de prelectura (análisis del entorno y configuración del analizador de espectro) en los bloques de color amarillo, la etapa de lectura en color azul, así como las etapas de observación y análisis en colores vede y gris respectivamente. Siguiendo esta técnica de medición del espectro radioeléctrico se puede obtener mediciones más precisas y conclusiones mucho más acertadas.

Como se describe, la medición correcta del espectro radioeléctrico necesita varias lecturas por parte del equipo receptor, ya que el espectro tiende a variar en el tiempo complicando el procesamiento de los datos. Debido a esto se necesita funciones que permitan realizar varias lecturas en el tiempo y seleccionar entre ellas las mejores. En los analizadores de espectro radioeléctrico están implementadas funciones como *Maximum Hold* o *Average Hold*, que permite realizar varias lecturas del mismo espectro en el tiempo y escoger los valores

máximos o el promedio de valores dentro del conjunto de datos del espectro radioeléctrico leído. Estas funciones permiten que el operador pueda distinguir fácilmente patrones existentes en el espectro leído obviando señales radioeléctricas que puedan interferir en una medición correcta.

De esta manera para el algoritmo de detección de transmisiones no deseadas, en este trabajo de titulación se ha desarrollado la función *max hold* que permite realizar varias lecturas del mismo espectro radioeléctrico y seleccionar el conjunto de datos que tenga el mayor valor de potencia. El número de veces que se realiza la lectura del espectro por parte del RTL-SDR está definido en el código con una variable configurable; mediante la función de Python max se elige el conjunto de datos de mayor valor. Esto permitirá definir un procesamiento de datos más eficiente al momento de ejecutar las funciones creadas en el algoritmo.

Definido el procedimiento de lectura y medición del espectro, la figura 2.4 muestra a manera de ejemplo el espectro radioeléctrico del servicio de radio FM para las frecuencias entre 92 MHz y 97 MHz, en donde se observa el espectro radioeléctrico que posee cada canal FM.

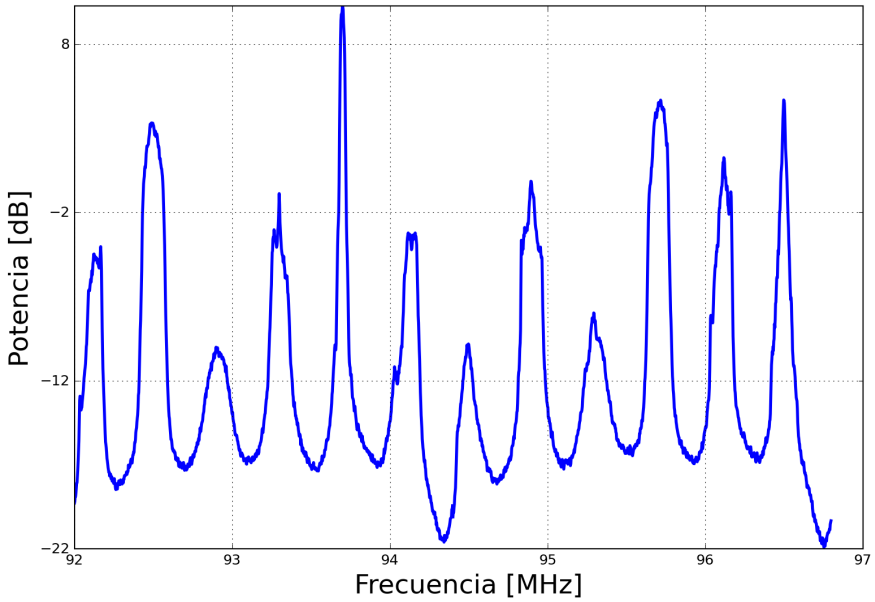

Figura 2.4: Espectro radioeléctrico para las frecuencias entre 92 MHz y 97 MHz

CAPÍTULO 3

Marco teórico

En este capítulo se desarrollará los fundamentos teóricos de las herramientas tanto de hardware como de software empleadas en este trabajo de titulación, definiendo sus principales características y usos en este proyecto.

3.1. Metodología de lectura y procesamiento para la identificación de transmisiones no deseadas

En la figura 3.1 se muestra un diagrama de bloques en donde están representados el y la Raspberry Pi, con todas las funciones que necesitan para la identificación de las trasmisiones ilegales. Dentro del bloque RTL-SDR existe el sub bloque que permite la configuración de los parámetros como la ganancia, el número de muestras del RTL-SDR y el rango de frecuencias a ser leído, este bloque de configuración es importante definirlo en el principio ya que el proceso de lectura del espectro radioeléctrico dependerá de los parámetros configurados anteriormente, para después convertir toda la información de las señales analógicas en señales digitales como se muestra en el último sub bloque del RTL-SDR. Una vez la información del espectro radioeléctrico ha sido digitalizada, pasa a la Raspberry Pi para iniciar con el procesamiento de todo el espectro leído. En esta etapa que se muestra en la figura 3.1 como el sub bloque de procesamiento de datos, se filtra el espectro radioeléctrico tanto de FM como de TV y se lo ordena por canales. Los siguientes sub bloques indican la comparación de la señal leída y la toma de decisiones por parte del algoritmo, usando comparadores estadísticos como la correlación y la raíz del error cuadrático medio, para finalmente enviar toda la información de las transmisiones no deseadas a la base de datos y su posterior visualización en la interfaz programada.

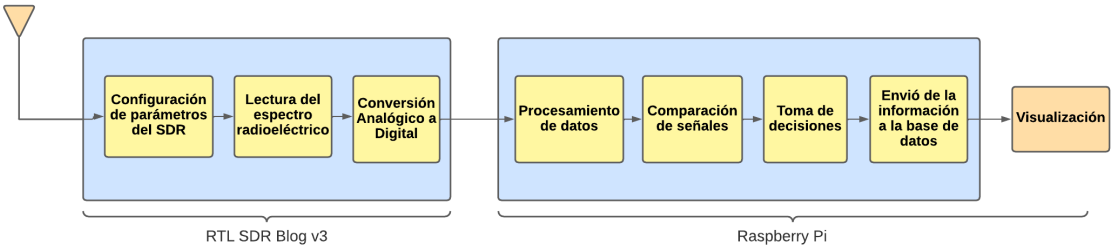

Figura 3.1: Diagrama de bloques de estructura del hardware y funciones del software

3.2. Funcionamiento y características técnicas de la radio definida por software

3.2.1. Radio definida por software

La tecnología de radio definida por software o por sus siglas en inglés es un sistema de radiocomunicaciones en la que la mayoría de los componentes electrónicos de radio se implementan por software en vez de hardware. La parte fundamental de estos sistemas de radiocomunicación radica en el receptor Zero-IF o Low-IF. Este receptor es configurable a tal punto, que permite diseñar distintos componentes como mezcladores, filtros, amplificadores, moduladores/demoduladores y también sistemas completos de radiocomunicación como transmisores, receptores, osciloscopios, analizadores de espectros, pudiendo configurar sus parámetros dinámicamente [15]

La nomenclatura radio definida por software no es un término nuevo, sino una designación que evolucionó con el tiempo. Una de las primeras apariciones de este término se remite a 1984 y hace alusión a un receptor digital de señales banda base desarrollado por un laboratorio del Departamento de Defensa de Estados Unidos. Posteriormente en 1991 el proyecto militar estadounidense *SpeakEasy* acuña la primera implementación importante del concepto de . Su objetivo era implementar más de 10 tipos de tecnologías de comunicaciones inalámbricas en un equipo programable, operando en la banda de frecuencias de 2 MHz a 200 MHz. [16] Es una tecnología que se proyecta como dominante en el campo de la radio comunicación a largo plazo.

3.2.2. Conceptos clave en la radio definida por software

Analogic Digital Converter (ADC): Convertidor analógico-digital donde se realiza la conversión asignando a cada nivel de tensión un numero digital para ser utilizado para el procesamiento de la señal.

- Tasa de muestreo: Es el número de veces por segundo que el convertidor analógico/digital toma una medida de la señal analógica y cuantifica el valor analógico utilizando un conjunto de bits. Mientras más bits se usen menor será el error de cuantificación, es decir el error entre la señal analógica medida y la salida del convertidor analógico - digital.
- Rango dinámico: Se refiere a la diferencia entre la señal más pequeña y la más grande que puede convertir el ADC.
- Tiempo de conversión: Es el tiempo que necesita el convertidor de analógico a digital para obtener un número digital a partir de un dato analógico
- Número de niveles: Indica la precisión con la que se cuantifica un dato analógico y depende del número de bits del convertidor analógico a digital.

Teoría de Nyquist: Esta teoría define que para evitar el efecto del solapamiento de señales cuando se convierta la señal de analógica a digital, la frecuencia de muestreo del convertidor debe ser de al menos dos veces el ancho de banda de la señal de interés. Para poder reconstruir una señal analógica de la salida del ADC, la señal analógica debe ser limitada en banda.

$$BW = f_{max} \rightarrow F_m = 2f_{max} \tag{3.1}$$

Digital Analogic Converter (DAC): El convertidor digital a analógico es el elemento que recibe información de una entrada digital, en forma de palabra de n bits y la transforma a una señal analógica.

Interfaz RF: Se encarga de trasladar adecuadamente y amplificar el centro de un rango de frecuencias a otro rango de frecuencias. La frecuencia central del rango de salida es la Intermediate frequency (IF) y generalmente será 0.

3.2.3. Funcionamiento general de la radio definida por software

Para explicar el funcionamiento del se utilizará un elemento gráfico. Como se puede apreciar en la figura 3.2 tanto para la recepción como la trasmisión de señales, el tiene una etapa de Radio Frecuencia (RF) donde la antena recibe la radiación electromagnética y la convierte en banda base. El bloque de cadena de RF hace referencia a todo este proceso hasta que las ondas recibidas son llevadas a una etapa de digitalización donde se convertirán en ondas digitales con una frecuencia de muestreo que cumpla con el teorema de Nyquist. Una vez digitalizada pasa por un proceso de demodulación del símbolo en donde los símbolos son mapeados a bits, y estos a su vez pasan por una etapa de decodificación del canal donde se remueve los datos de redundancia controlada, para pasar finalmente por la etapa de decodificación de la fuente donde se introducen datos de redundancia para la última etapa donde el computador recibe como entrada una representación de una señal del mundo real, como puede ser, la voz humana, música, video, imágenes, entre otros.

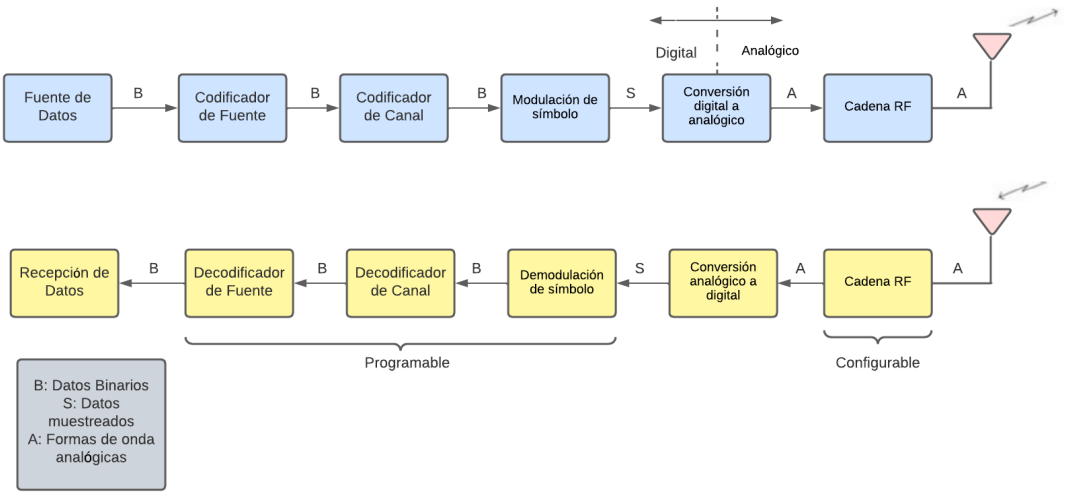

Figura 3.2: Diagrama de bloques elemental de todos los SDR.[16]

Simplificando la estructura básica que tienen todos los consiste en un convertidor analógico-digital ADC, un convertidor digital – analógico, DAC, una antena y otros módulos. Además, el puede emplear etapas de procesamiento digital de señales, (Digital Signal Processing (DSP)) y una unidad central de procesos, (Central Processing Unit (CPU)) de propósito general, como se muestra en la figura 3.3.

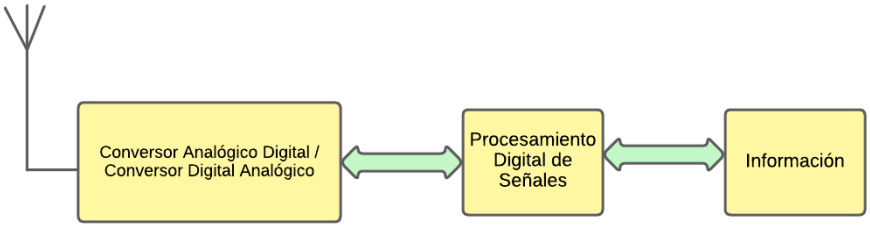

Figura 3.3: Diagrama de bloques simplificado del SDR. [16]

El bloque de DSP provee al sistema flexibilidad de desarrollo y es usado principalmente como contador de número de operaciones para los algoritmos de procesamiento de señales. Generalmente las técnicas de DSP eran usadas para una fase de pre modulación y post detección en receptores de radio [17]. El concepto ha ido cambiando con el tiempo, pero se sigue basando en el diagrama de bloques que se muestra en la figura 3.4, formada por 3 bloques fundamentales:

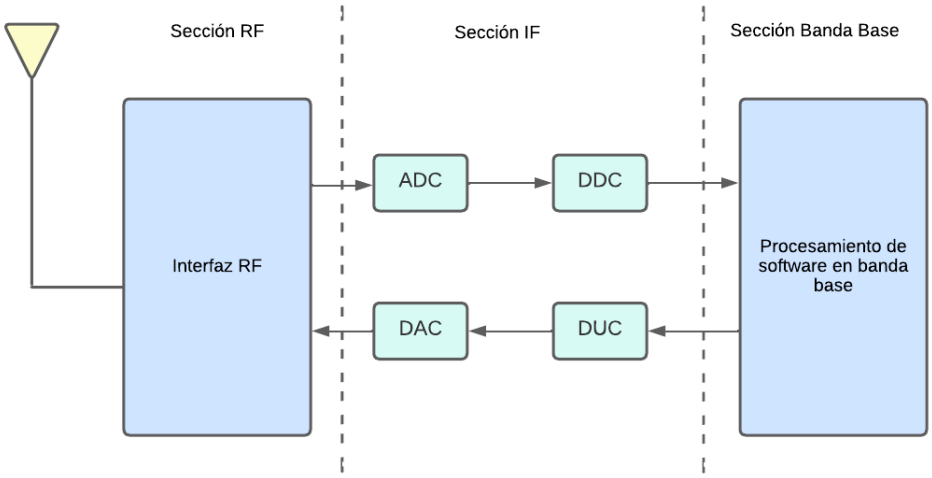

Figura 3.4: Secciones de bloques de recepción y transmisión de un SDR. [15]

La sección de RF, ubicada en la parte izquierda de la figura 3.4 también denominada cabecera de RF, es la encargada de transmitir o recibir señales de radiofrecuencia para adecuarlas y convertirlas en IF en recepción o amplificar y modular las señales de IF en el caso de transmisión. La frecuencia intermedia puede ser 0, dando a lugar al concepto de Zero-IF. La sección de IF es la encargada de pasar la señal de IF a banda base y digitalizarla en recepción o pasar la señal de banda base a IF y hacer la conversión analógica – digital en el caso de la transmisión. Tanto para la conversión analógica – digital y digital – analógica, los encargados son los módulos ADC y DAC respectivamente además que los módulos Digital Download Converter (DDC) y Digital Upload

Converter (DUC) se necesitan para bajar o subir la tasa de muestreo. La sección de banda base es la encargada de todo el procesamiento en banda base de la señal como la modulación o demodulación, análisis espectral de la señal, todo esto mediante software. [15]

3.3. Características técnicas y descripción del equipo RTL-SDR Blog V3

3.3.1. RTL-SDR Blog V3

Para este proyecto de titulación se emplea el dispositivo RTL-SDR blog V3 como radio definida por software. Este dispositivo fue diseñado originalmente para recepción Digital Video Broadcasting Terrestrial (DVB-T) High-definition television (HDTV), pero existen muchos desarrolladores que han empleado este dispositivo para múltiples propósitos en las radiocomunicaciones. En la figura 3.5 se muestra el dispositivo RTL-SDR blog v3, este dispositivo tiene una conexión USB para la comunicación con la computadora y la conexión SubMiniature version A (SMA) para el acople con la antena.

Figura 3.5: Dispositivo RTL-SDR Blog v3. [18]

El éxito de este equipo ha sido especialmente por su calidad y precio, pues este equipo solamente permite la recepción de señales radioeléctricas más no la transmisión de señales, siendo por defecto el analizador del espectro radioeléctrico más empleado por radio aficionados. El valor de este dispositivo ronda los $30 en la tienda en línea Amazon de Estados Unidos, y además, tiene muchos modelos genéricos del mismo tipo aún más económicos.

Los requerimientos de hardware del RTL-SDR Blog v3 son los mismos requerimientos que cualquier , pues este dispositivo es compatible desde Windows XP hasta Windows 10, Linux, MacOS y Android. Se recomienda para estos sistemas operativos emplear un computador mínimamente con doble núcleo en el procesador. También hay compatibilidad con los sistemas de Raspberry Pi, Odroid, C.H.I.P

3.3.2. Características técnicas del equipo RTL-SDR Blog V3

Las características técnicas del equipo empleado son:

- **Ancho de banda:** hasta 2.4 MHz
- **Convertidor Analógico Digital:** Chip RTL2832U 8-bits
- **Rango de Frecuencia:** 500 KHz – 1776 MHz
- **Impedancia de entrada:** 50 Ohm
- **Consumo de corriente:** 70 – 280 mA

3.3.3. Componentes electrónicos exclusivos del RTL-SDR Blog V3

TCXO: El modelo Blog v3 emplea un oscilador Temperature compensated crystal oscillator (TCXO) de 1 parts per million (PPM) para una excelente estabilidad de frecuencia.

Conector SMA: El modelo RTL-SDR Blog v3 emplea comúnmente conectores SMA a diferencia de otros dispositivos RTL-SDR donde emplean conectores específicos del modelo del productor. Esto hace que el modelo v3 sea adaptable al usuario empleado los mismos conectores para el como para las antenas.

Cubierta de aluminio: Exclusivo del modelo v3 la cubierta de aluminio permite bloquear cualquier interferencia de radiofrecuencia que entra al Printed Circuit Board (PCB), también está cubierta permite la disipación de calor en el PCB

Chip R820T2: Este chip creado para la conversión analógico digital ha sido mejorado por el productor con una más alta calidad en el silicio empleado, en comparación del modelo anterior R820T, permitiendo chips más seguros y con sensibilidades mejores y más estables.

Bias Tee de 4.5V: Este bias tee es accesible mediante software empleando la línea de comandos o el command (CMD) de Windows. Para la activación o desactivación es necesario emplear los siguientes comandos:

- ON: rtl biast –b 1
- OFF: rtl biast –b 0 [18]

3.4. Aspectos fundamentales de la Raspberry Pi

3.4.1. Raspberry Pi 3

Hace 15 años las únicas computadoras portátiles disponibles en el mercado eran las laptops, tanto de Windows como de MacOS. Haciendo su aparición en el mercado en el año 2011, la Raspberry Pi revoluciona el concepto de computadora portátil al tener una presentación muy diferente a la que los usuarios estaban acostumbrados, logrando llamar la atención a desarrolladores tanto de software como de hardware y haciéndose muy popular en los siguientes años.

El primer modelo de la Raspberry Pi contaba con dos modelos A y B, siendo este último un procesador Broadcom BCM2835 de un solo núcleo y a 700 MHz, también contaba con 512 MB de RAM además que el

sistema operativo tenía que instalarse en una tarjeta SD. Después se desarrolló el modelo B+, una revisión del modelo B que cambiaba de SD a Micro-SD, además de traer 4 puertos USB.

Para el año 2018 la Raspberry Pi Foundation saca el modelo Raspberry Pi 3 Model B+. Este modelo supuso un rediseño absoluto de la placa; sin embargo, mantuvo el tamaño de modelos anteriores, y la misma posición de los componentes que en el modelo Raspberry Pi 3. Además trae un procesador más potente que trabaja a 1.4 GHz, mejorando la conectividad en Bluetooth 4.2, BLE, WiFi a doble banda 2.4 GHz y 5 GHz, y la tarjeta de red, Gigabit Ethernet ya no está limitada a los 100 Mbps si no que puede alcanzar los 300 Mbps al funcionar sobre USB 2.0. [19].

El último modelo de la Raspberry Pi es el modelo 4, que se muestra en la figura 3.6, lo más resaltable de esta versión incluye mejoras en los puertos USB e incrementos en la memoria ram, con modelos desde 2 GB hasta los 8 GB.

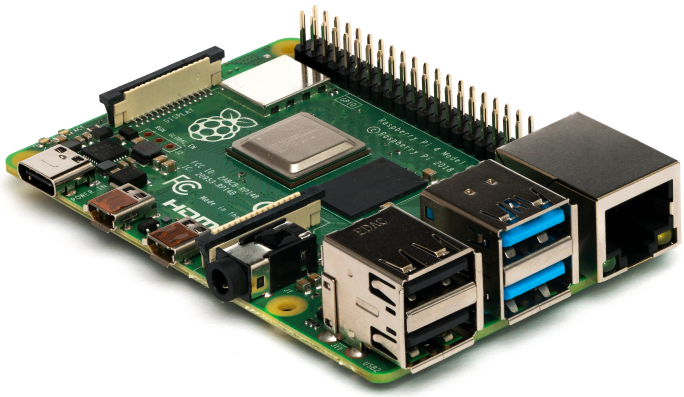

Figura 3.6: Raspberry Pi 4 [20]

3.4.2. Características técnicas del Raspberry Pi 3

El modelo que se empleará en este proyecto de titulación es el Raspberry Pi 3 Model B+, que ha sido desarrollado con un procesador quad-core de 64-bits corriendo a 1.4 GHz. Las especificaciones técnicas de este micro computador son:

- **Procesador:** Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
- **Memoria RAM:** 1GB LPDDR2 SDRAM
- **Acceso:** 40-pines GPIO
- **Video y sonido:** 1 Puerto HDMI
- **Multimedia:** H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30)
- **Soporte para tarjeta SD:** Formato Micro-SD para cargar el sistema operativo y almacenar información
- **Potencia de entrada:** 5V/2.5A DC via conector micro USB, 5V DC via GPIO header
- **Temperatura permitida:** 0°C a 50°C
- **Tiempo de vida de producción del modelo:** El modelo 3B+ dejará de producirse en enero del 2023.

3.5. Librerías Python para procesamiento de datos

A continuación, se señalan las principales librerías del lenguaje de programación Python a ser utilizadas para este proyecto.

3.5.1. Pandas

Pandas es una librería exclusiva de Python especializada en el manejo y análisis de conjuntos de datos. Pandas permite definir nuevas estructuras para los datos basadas en los arrays de la librería Numpy. El desarrollo de Pandas surgió como la necesidad de crear una herramienta única que posea todo lo necesario para estructuración y análisis de datos. [21] Pandas tiene una gran acogida en la comunidad de desarrolladores ya que permite leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL, además que usa índices o nombres para filas y columnas, para acceder a los datos.

Pandas dispone de tres estructuras de datos diferentes:

- Series: Estructura unidimensional etiquetada capaz de almacenar cualquier tipo de datos.
- Dataframes: Estructura bidimensional con columnas y filas, donde las columnas a su vez son series
- Paneles: Estructura de tres dimensiones (cubos)

Para poder instalar esta librería en el sistema operativo es necesario primero identificar que versión de Python se está usando, para ello es necesario emplear el comando:

python –versión

Figura 3.7: Versión empleada de Python

La versión de Python empleada es la 3.9.0 que es una versión estable y trae soporte para Pandas, para la instalación de pandas en el computador se emplea:

pip install pandas

Los comandos más empleados en Pandas son los de creación de estructura de datos, manipulación y visualización, siendo estos comandos la matriz de esta librería especializada en datos. Para importar la librería de Pandas:

import pandas as pd

Para crear un dataframe empleamos el comando:

df = pd.DataFrame()

Para seleccionar y extraer datos tenemos varias opciones donde necesitaremos el índice ya sea de la columna o de la fila

df.iloc[índice de fila, índice de columna]

3.5.2. Numpy

Numpy es una librería de Python especializada en el cálculo numérico y el análisis de datos. Numpy incorpora una nueva clase de objetos llamados arrays que permiten representar colecciones de datos de un mismo tipo en varias dimensiones, permitiendo trabajar con matrices y matrices multidimensionales.

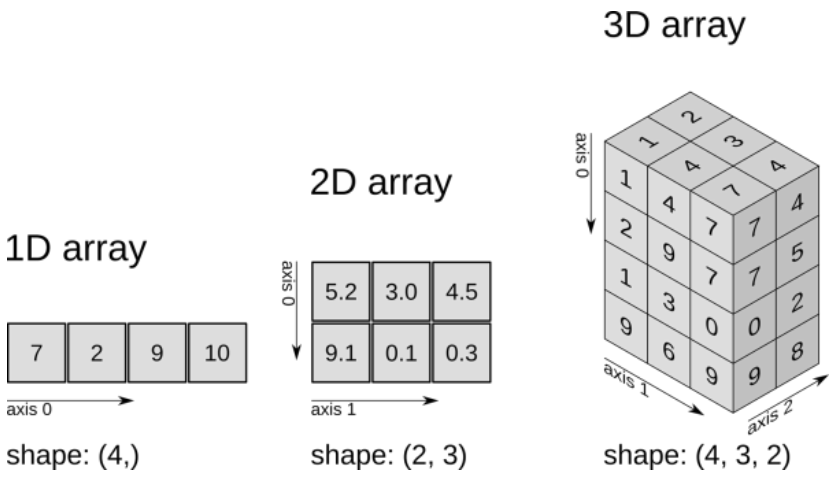

Figura 3.8: Diferentes tipos de arreglos en Numpy. [22]

Para su instalación se empleará pip con el siguiente comando:

pip install numpy

Para empezar a usarlo se tendrá que importar la librería con el comando:

import numpy as np

Donde *np* es el alias que por defecto toma numpy en el código. Para crear un array de una dimensión se emplea el comando:

array=np.array([n1,n2,n3])

Para crear un array de dos dimensiones se emplea el comando [22]:

array=np.array([n1,n2,n3],[n4,n5,n6])

Siendo estos los comandos de creación de numpy, muchas otras librerías como Scikit-learn o Pandas emplean como base numpy. Esto hace que la creación de estructuras de datos tanto de Sckit-Learn como Pandas son similares a Numpy. [23]

3.5.3. Matplotlib

Matplotlib es la librería más popular de Python de visualización de datos. Teniendo opciones de gráficos de barras, histogramas, diagramas de dispersión, diagramas de contorno, mapas de color entre otros. La visualización de datos es un proceso en donde a partir de los datos procesados, se busca la mejor representación gráfica del contexto que tengan los datos. Con esta premisa se requiere enfatizar la fácil personalización de los gráficos que tiene esta librería. Para la instalación de la librería se digita el siguiente comando:

pip install matplotlib

Para importar esta librería se emplea el comando:

import matplotlib.pyplot as plt

Donde el módulo pyplot es el encargado de la importación de los códigos de graficación de líneas. Para crear la figura y los ejes se emplea el comando:

fig, ax = plt.subplots()

Para ejemplificar el uso de matplotlib graficaremos una parábola en función del conjunto de datos definido en la variable "x", para definir este conjunto de datos se emplea la función np.linspace de numpy, que crea 1000 puntos equidistantes entre -10 y 10, estos datos son procesados por la variable "y" que será la función cuadrática de la parábola de ejemplo.

x = np.linspace(-10, 10, 1000)

y = x2 + 2*x + 2**

Para mostrar la gráfica de la parábola, descrita en las variables "x" y "y", se emplea el comando: [24]:

plt.show()

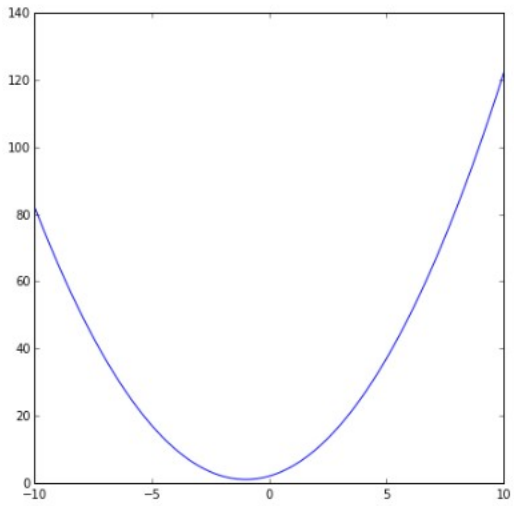

Figura 3.9: Gráfica de la parábola en Matplotlib

Finalmente, para guardar el gráfico de la parábola generado se emplea el comando:

plt.savefig('Gráfico de la parábola.png')

3.5.4. Pyrtlsdr

La librería Pyrtlsdr es una interfaz en Python de los dispositivos RTL-SDR que emplean el chip Realtek2832U. El dispositivo RTL-SDR al ser un dispositivo solamente de recepción de señales electromagnéticas la librería estará optimizada para la adquisición de los datos. Pyrtlsdr toma las funciones básicas de la librería librtlsdr-library y construye las propias para la fácil comunicación entre el RTL-SDR y el computador. Para instalar esta librería mediante pip se emplea el comando:

pip install pyrtlsdr

Y para importar la librería al código se emplea el comando:

from rtlsdr import RtlSdr

A continuación, se muestra un ejemplo de cómo esta librería adquiere y procesa los datos, primero se inicializará el objeto sdr con la función *RtlSdr*, es decir el dispositivo RTL-SDR se inicia para una captura de datos, posteriormente las variables asignadas como sample rate, center frequency y gain, son los parámetros de configuración básicos que necesita Pyrtlsdr para una lectura del espectro. A continuación la variable samples tiene como valor asignado a la función sdr.read_samples que será el comando para iniciar la lectura del espectro y finalmente el comando sdr.close() termina la sesión del objeto sdr creado. Para la etapa de graficación del espectro se usa Matplotlib, primero para hallar el Power Spectral Density (PSD) del espectro escaneado y segundo para la gráfica del mismo. [25]

```
from pylab import *
from rtlsdr import *

sdr = RtlSdr()

# configure device
sdr.sample_rate = 2.4e6
sdr.center_freq = 95e6
sdr.gain = 4

samples = sdr.read_samples(256*1024)
sdr.close()

# use matplotlib to estimate and plot the PSD
psd(samples, NFFT=1024, Fs=sdr.sample_rate/1e6, Fc=sdr.center_freq/1e6)
xlabel('Frequency (MHz)')
ylabel('Relative power (dB)')

show()
```

Figura 3.10: Código de ejemplo de lectura del espectro radioeléctrico con pyrtlsdr

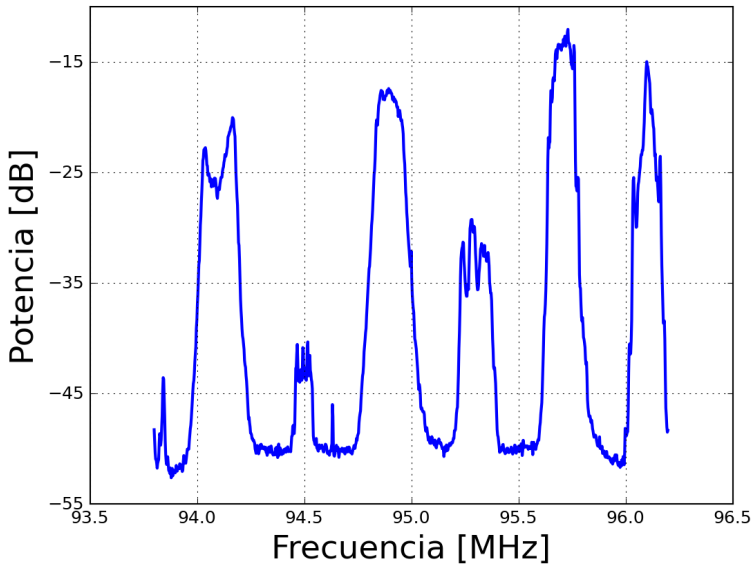

Figura 3.11: Gráfica del espectro radioeléctrico obtenido con Pyrtlsdr y Matplotlib

3.6. Comparadores estadísticos de señales

Para la identificación de las transmisiones no deseadas los datos procesados y filtrados por canales de FM y TV tienen que ser comparados con otra señal que sea de referencia y cuyo espectro no tenga emisiones radioeléctricas espurias, con esto la figura 3.12 muestra un diagrama de bloques que sigue la estructura de la comparación de señales y la posterior toma de decisiones en el algoritmo, en donde el primer bloque indica el filtrado por canales de FM y TV del espectro leído por el RTL-SDR, posterior a la filtración de los datos es creada la señal de referencia que será comparada con las señales leídas por RTL-SDR.

Para la creación de esta señal de referencia es necesario especificar la mínima señal detectable por el RTL-SDR, que es el mínimo valor de potencia que puede detectar el sistema, también llamado el umbral del sistema. La señal de referencia es una señal con todos sus valores iguales al valor del umbral. Cuando se define la mínima señal detectable del sistema, el espectro leído es filtrado en base a este valor, donde se tomarán solamente los valores que estén por encima del umbral o mínima señal detectable, esto permite filtrar el ruido que este por debajo del umbral permitiendo el cálculo con precisión de los comparadores de señales. Una vez creada la señal de referencia y la alisada la señal de cada canal se procede a realizar las comparaciones estadísticas con la correlación y la raíz del error cuadrático medio de estas dos señales, si la correlación es menor a 0.2 y la raíz del error cuadrático medio es mayor a 0.1 el algoritmo identifica a la señal como no deseada.

Estos valores se han escogido ya que la correlación mide la relación existente entre estas dos señales y si esta relación tiene un valor inferior a 0.20 indica una relación muy débil entre las señales, en cambio el valor del RMSE me indica la cantidad de error en el conjunto de datos de las señales, por lo que al escoger un valor de 0.01 buscaría un cambio mínimo en las señales.

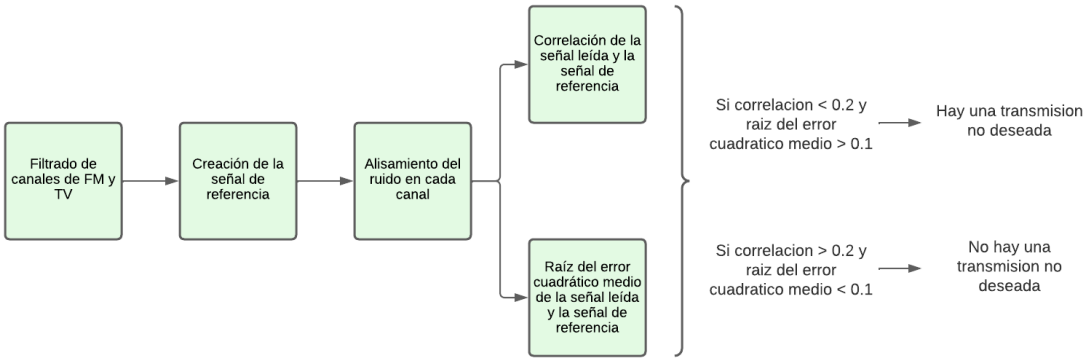

Figura 3.12: Diagrama de flujo de transformación y análisis de datos

Este diagrama de flujo muestra la metodología empleada para la comparación estadística de las señales y como el algoritmo toma la decisión de si una señal es ilegal o no. A continuación se detalla las definiciones de los operadores estadísticos usados, definiendo las propiedades más importantes de cada uno.

3.6.1. Correlación

Es una medida estadística que determina hasta qué punto dos variables están relacionadas linealmente, evalúa la tendencia creciente o decreciente en los datos y cuantifica la fuerza y dirección de la relación lineal y proporcionalidad entre dos variables estadísticas. La correlación permite definir relaciones simples sin hacer afirmaciones de causa y efecto y es que la correlación no indica causalidad. El coeficiente de correlación se describe como una medida sin unidades que va desde -1 a +1.

El signo define la dirección de la relación lineal donde:

- Mientras más se aproxima a cero, más débil es la relación lineal.
- Valores positivos indican una correlación positiva, indicando que los valores de ambas variables incrementan juntos.
- Valores negativos indican una correlación negativa, indicando que los valores de una variable incrementando mientras que los de la otra variable decrecen.

La fórmula del coeficiente de correlación es:

$$r = \frac{\sum[(x_i - x')(y_i - y')]}{\sqrt{\sum(x_i - x')^2 * \sum(y_i - y')^2}} \tag{3.2}$$

Esta magnitud indica la fuerza de la relación, cuanto más cercano el valor sea de +1 o -1 más fuerte será a la tendencia de las variables. Mientras que, si el coeficiente de correlación está cercano al cero, más débil será la tendencia, es decir existirá una mayor dispersión en la nube de puntos. [26]

- Si el coeficiente de correlación es igual a +1 o -1, existe una correlación perfecta.
- Si el coeficiente de correlación es igual a 0, las variables no están correlacionadas.

3.6.2. Raíz del error cuadrático medio

El RMSE permite cuantificar las diferencias entre los valores de muestra y los valores predichos por un modelo. También se puede describir al RMSE como la raíz cuadrada del promedio de los errores cuadrados, además que el RMSE es siempre positivo y tener un valor de 0 indicaría un ajuste perfecto a los datos.

La fórmula del RMSE es:

$$RMSE = \sqrt{\frac{\sum(Predecido_i - Actual_i)^2}{N}} \tag{3.3}$$

Lo que esta fórmula indica es la raíz cuadrada de promedio de la diferencia entre el valor predicho y el valor real. La magnitud del RMSE me indica el error que tiene el modelo predicho con el modelo real, o dicho de otra manera el error que tiene el modelo de referencia como el modelo real. Para los propósitos de este trabajo de titulación se empleará el RMSE como método de comparación entre señales. [27]

3.7. Base de datos y backend de la aplicación

Al detectar una transmisión el algoritmo almacena la información de la frecuencia, potencia y fecha de emisión. Para este propósito es necesario una base de datos que se comunique con el programa de detección de transmisiones y que permita la lectura y escritura de datos, por la estructura del programa se emplea una base de datos Non SQL (NoSQL), llamada Firestore que es propiedad de Google Cloud Platform. El encargado de la comunicación entre el programa de detección y la base de datos es el *backend* creado con Flask, *framework* hecho en Python. A continuación, se define los servicios de base de datos y *framework backend* empleados.

3.7.1. Backend con Flask

El *backend* es el encargado de la comunicación entre la base de datos, el algoritmo de detección y la interfaz del usuario. Por simplicidad del lenguaje se ha escogido Python como lenguaje de desarrollo. Flask es un *framework* de Python que proporciona herramientas y funciones útiles para la creación de aplicaciones web. Ofrece a los desarrolladores flexibilidad y un *framework* más accesible para los nuevos desarrolladores. Flask utiliza el motor de platillas de Jinja para crear dinámicamente páginas HTML usando conceptos de Python familiares como variables, bucles, listas, matrices. [28]

3.7.2. Base de datos NoSQL con Google Firestore

Los datos empleados necesita una estructura de almacenamiento de datos por usuario. Esto permitirá que se tenga las transmisiones no deseadas escaneadas por usuario. Para este propósito es necesario una base de datos NoSQL que definen modelos de datos específicos que se adapta a la aplicación desarrollada. La base de datos seleccionada para este proyecto es Google Firestore que es una plataforma para el desarrollo de aplicaciones web, propiedad de Google. [29]

CAPÍTULO 4

Metodología del desarrollo del algoritmo de detección de transmisiones no deseadas

La metodología implementada para el desarrollo del programa de detección de transmisiones no deseadas, tiene 3 etapas en donde se define:

- Funciones de extracción, transformación y carga de los datos.
- Funciones de conexión, lectura y escritura en la base de datos Google Firestore.
- Funciones de comunicación con la base de datos, infraestructura y diseño web.

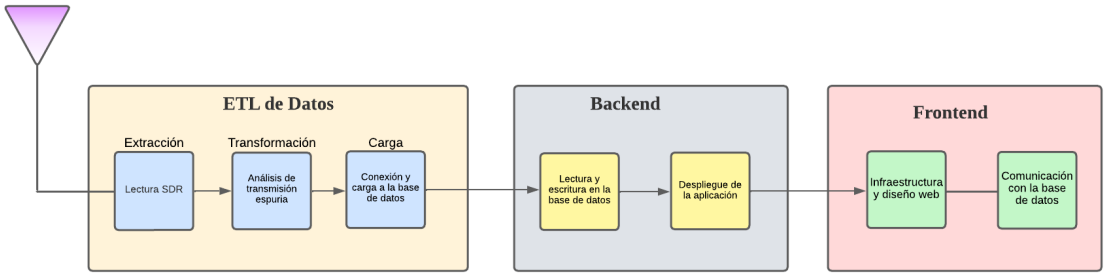

Figura 4.1: Diagrama de bloques de metodología del desarrollo del algoritmo

La figura 4.1 muestra el diagrama de bloques que define la metodología empleada en el desarrollo, desde la parte izquierda se tiene el bloque de la antena receptora definida por el triángulo invertido violeta; las señales radioeléctricas analógicas pasan al bloque amarillo definido como Extract Transform Load (ETL) de Datos, y cuyos sub bloques azules permiten la lectura y digitalización del espectro radioeléctrico, seguido de una transformación de la infraestructura de los datos para una posterior verificación de si la transmisión es deseada o no. Al tomarse esta decisión el algoritmo crea una variable que contiene todos estos datos y procede a una conexión a la base de datos de Firestore de Google.

El *backend* representado por el bloque de color gris, contiene el sub bloque de lectura y escritura en la base de datos. El algoritmo al detectar una transmisión no deseada envía y escribe en la base de datos la frecuencia, potencia y fecha de emisión de esta transmisión. En esta etapa también se despliega los servicios de administración de servidor web para una consecuente visualización del proyecto en la interfaz gráfica definida en el último bloque *frontend* que crea la infraestructura y diseño web; este bloque tendrá funciones específicas para conectarse con la base de datos implementadas con HTML y CSS

Para el despliegue de toda la aplicación, fue necesario definir funciones que permitirán la manipulación de datos y su visualización. Todas las funciones de manipulación de datos son desarrolladas en Python mientras que las de visualización de datos son desarrolladas en HTML y CSS. El desarrollo de las funciones se divide en dos partes, para una mejor comprensión se utilizarán el diagrama de bloques de la figura 4.2, donde la primera parte (recuadros verdes) corresponde al desarrollo del algoritmo de lectura, procesamiento de datos y toma de decisiones con respecto a si la señal es legal o ilegal. En esta parte se empleará solamente Python y sus librerías debido a que este lenguaje esta optimizado para el procesamiento de grandes cantidades de datos. Para esta parte se han creado las librerías *sdrscanfm* y *sdrscantv* que agrupan y organizan una estructura de las funciones desarrolladas. La segunda parte (recuadros azules) corresponde al desarrollo de la interfaz, comprende la creación del *backend* y *Frontend*. Para el *backend* se ha empleado el *framework* de Python Flask, ya que es un *framework* fácil de implementar y de desplegar en el sistema. Para el *frontend* se ha empleado HTML y CSS, lenguajes ampliamente utilizados para el desarrollo de interfaces web de usuario.

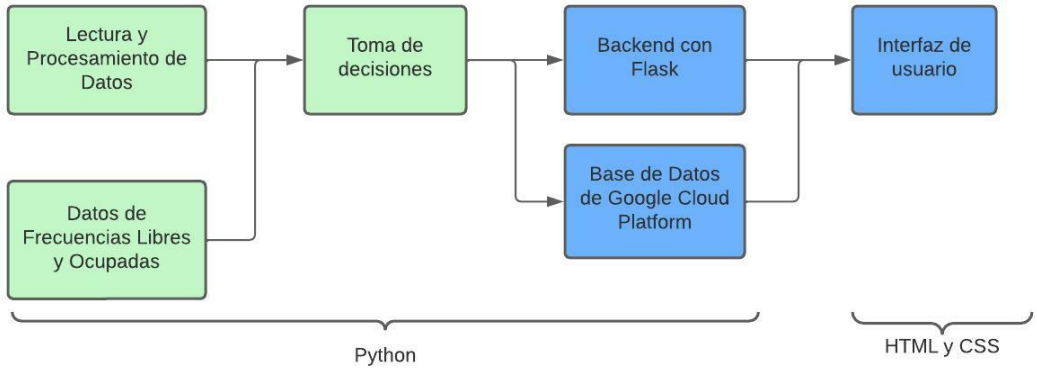

Figura 4.2: Diagrama de bloques de funcionamiento del algoritmo de detección e interfaz

Como se puede ver en la figura 4.2, los cinco primeros bloques de la parte izquierda están desarrollados con Python e interconectados entre sí. Su diagramación obedece a toma de decisiones sobre si la señal es legal o ilegal. Primero se ejecutan los bloques denominados Lectura y Procesamiento de Datos más el bloque que representa a la Información de las Frecuencias Libres y Ocupadas del espectro radioeléctrico en la región. A su vez el bloque de toma de decisiones envía un archivo JSON con toda la información previa a los bloques de *backend* con Flask y al bloque de la base de datos de Google Cloud Platform, en caso de existir una señal ilegal. Finalmente, estos dos últimos bloques se conectan a la interfaz de usuario desarrollada con HTML y CSS donde el usuario puede acceder a las opciones de escaneo del espectro y las alarmas del usuario.

4.1. Funciones de extracción, transformación y carga de los datos

La figura 4.3 muestra el diagrama de flujo que sigue el algoritmo para la lectura, procesamiento y visualización de datos. En la parte superior, los bloques en anaranjado muestran la etapa de configuración del dispositivo RTL-SDR y lectura del espectro radioeléctrico. Luego sigue la etapa de análisis por canal, expresada en los bloques de color celeste en donde se muestra las etapas de filtrado del canal, mínima señal detectable, creación de la señal a comparar, y finalmente la comparación estadística que arroja el resultado de si existe o no una transmisión no deseada. En el último bloque de color morado se muestra la etapa de visualización del espectro radioeléctrico analizado.

Todos estos bloques son creados con funciones especializadas para la lectura del espectro radioeléctrico y manejo de la infraestructura de los datos. El conjunto de estas funciones desarrolladas en Python crea las librerías *sdrscanfm* y *sdrscantv* que serán importadas en las demás funciones de despliegue de la interfaz de usuario.

Todas las funciones que se definen a continuación son creadas por el autor con el objetivo de procesar eficientemente el conjunto de datos leídos por el RTL-SDR. Las funciones programadas por el autor toman funciones de librerías populares de Python que se integran al código permitiendo el desarrollo del algoritmo de detección.

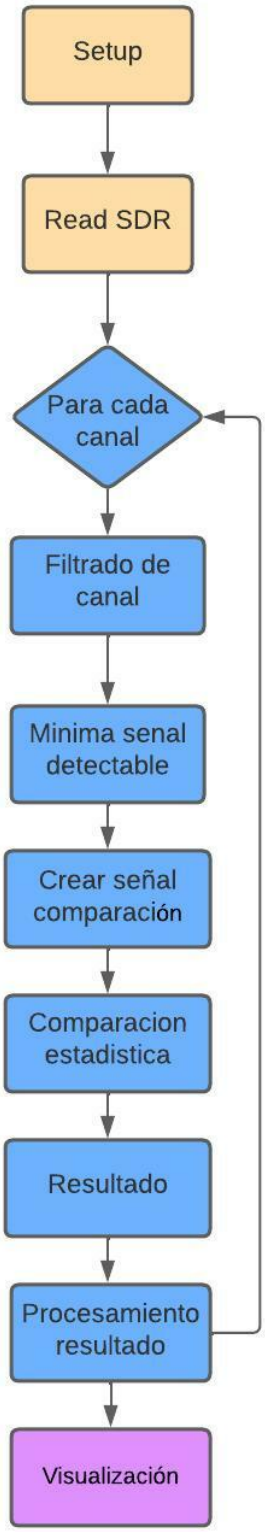

Figura 4.3: Diagrama de bloques lectura, procesamiento y visualización

4.1.1. Función setup

Esta función recibe como parámetros de entrada a la frecuencia máxima y mínima de cada canal en el espectro de FM y TV. En la figura 4.4 se muestra el diagrama de bloques de la función setup, donde una vez recibido los datos de entrada, se configuran todos los parámetros que se emplearán en la lectura del espectro radioeléctrico como la frecuencia de muestreo y cantidad de muestras, este proceso se muestra con el bloque de color azul. Finalmente, el bloque de color rojo indica la configuración de muestras para calcular la densidad espectral de potencia y configuraciones de la función *max hold*. En esta configuración inicial se definen las dimensiones de los arreglos de datos como el número de muestras en la lectura del , la cantidad de muestras en la PSD y la cantidad de veces que se escanea el espectro para la función de *max hold*.

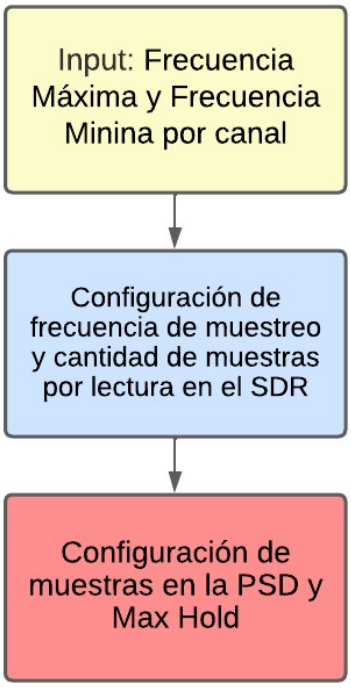

Figura 4.4: Función de configuración de parámetros

4.1.2. Función readsdr

La segunda función implementada recibe como parámetros las variables configuradas en la función setup, esta función es la encargada de inicializar el objeto sdr (sdr=rtlsdr.RtlSdr( )) dentro del código de acuerdo a los parámetros configurados anteriormente. Adicionalmente, una vez iniciado el objeto sdr, se puede configurar la ganancia del RTL-SDR con la variable *sdr.gain*. El objetivo de esta función es realizar una lectura de las muestras obtenidas con el RTL-SDR y procesarlas con la función *psd* para obtener la PSD. En la figura 4.5 se puede observar este proceso en donde se trabaja con dos bucles *for*. El primero es un bucle for anidado inicialmente recorre el arreglo *freq*, obteniendo los valores de las muestras en cada frecuencia asignada para después volver a realizar este proceso un determinado número de veces. El segundo bucle *for* trasforma las muestras en valores de la PSD y la gráfica. El parámetro *sdr.close* permite cerrar el objeto *sdr*, para poder realizar otra lectura con el mismo objeto. Es importante incluir este comando debido a que si no se cierra el objeto, el RTL-SDR seguirá

activado y no se podrán realizar nuevas lecturas. Finalmente, todos estos valores son almacenados, procesados y retornados en el df data.

```
def readsdr(rate_best, freqs, nfreq, npsd_res, npsd_avg, nsamp, nfreq_spec, samples,
    psd_array, freq_array, relative_power_array, psd_total,veces):
    #Initializing SDR
    sdr = rtlsdr.RtlSdr()
    sdr.sample_rate = rate_best
    sdr.gain = 2
    samp_rate = sdr.sample_rate
    for k in range(veces):
        for i,freq in enumerate(freqs):
                sdr.center_freq = freq
                samples[:,i] = sdr.read_samples(nsamp)
    for i,freq in enumerate(freqs):
        fc_mhz = freq/1e6
        bw_mhz = sdr.sample_rate/1e6
        psd_array[:,i],freq_array[:,i] = plt.psd(samples[:,i], NFFT=npsd_res, Fs=bw_mhz, Fc=fc_mhz)
    freq_series=np.concatenate(freq_array)
    psd_series=np.concatenate(psd_array)
    psd_total=np.insert(psd_total,k,psd_series,axis=1)
    sdr.close()
    psd_total=pd.DataFrame(psd_total)
    psd_new=psd_total.loc[:,0:veces-1]
    psd_max=psd_new.max(axis=1)
    max_hold=psd_max.apply(hacer_potencia)
    data_array = np.stack((freq_series, max_hold), axis=1)
    df=pd.DataFrame(data_array,columns=['Frecuencia','Potencia'])
    data= df.sort_values('Frecuencia',ascending=True)

    return data
```

Figura 4.5: Función de lectura del espectro radioeléctrico

4.1.3. Función hacer potencia

En la función anterior los datos obtenidos son procesados por una función hacer_potencia. Esta función permite transformar los datos de PSD obtenidos a decibelios mili vatios (dBm).

4.1.4. Función canal filter

Esta función recibe como parámetros los datos obtenidos de la función readsdr, además de los valores de la frecuencia máxima y mínima del canal. Esta función tiene como objetivo filtrar los datos de cada canal a analizar y almacenarlos en la variable *data_canal*. Este proceso es importante ya que permite analizar canal por canal, y no el conjunto de datos entero, generando menos carga computacional y menos tiempo de procesamiento del algoritmo.

4.1.5. Función mínima señal detectable

Esta función recibe como parámetros a los datos por canal obtenidos en la lectura del espectro radioeléctrico, y los procesa con la función *detection_limit*; esto con el fin de crear una señal de referencia que es la señal que se va a comparar para la toma de decisiones posteriores en el código.

4.1.6. Función detección de limite

Esta función es creada específicamente para la función *minima_senal_detectable_canal*. Los parámetros de entrada en esta función, son cada valor de potencia dentro del dataframe *data_canal* y el umbral corresponde al valor de la potencia mínima que debe tener el canal para que sus valores de potencia no sean cambiados. Es decir, esta función permite un alisamiento del espectro del canal a partir del valor del umbral. Si el valor de la potencia es menor al umbral este valor se convierte en el valor de la constante, en caso contrario, el valor de la potencia no cambia. El objetivo de esta función es eliminar el ruido dentro del canal debido a que el ruido no permite una correcta comparación de las señales.

4.1.7. Función crear señal comparación

Esta función toma como entrada a la señal de referencia y el umbral de la mínima señal detectable. El propósito de esta función es crear una señal que permita la comparación estadística con la señal de referencia y determinar si la señal es legal o ilegal. Para esto se necesita que la señal de comparación tenga las mismas dimensiones que la señal de referencia y que los valores dentro de esta señal creada sean los valores del umbral que se definieron anteriormente.

4.1.8. Función comparación señales

Esta función toma como entrada a las señales que van a compararse para tomar la decisión si existe una señal ilegal o no. Lo primero que se realiza es una comparación de dimensiones de los *dataframes* a comparar y en caso de no ser de la misma dimensión se los convierte a la misma dimensión. La función comparación implícita en el código de esta función da los valores de la correlación y el RMSE. Finalmente se devuelven los valores tanto de la correlación como del RMSE.

4.1.9. Función comparación

Esta función es la encargada de calcular los valores de la correlación y el RMSE de las señales de referencia y comparación. Para este proceso se ha implementado una validación de la correlación debido a que si las señales no varían entre si el valor de la correlación va a ser indeterminado originando errores en el código. Además, para el RMSE se ha implementado una validación en donde se calcula este valor múltiples veces y se toma el mínimo. Estas validaciones permiten que la comparación de las señales sea más robusta dando así menos errores al momento de tomar decisiones.

4.1.10. Función procesamiento de diccionarios

Esta función toma como entrada el diccionario retornado de la función procesamiento y devuelve una estructura de diccionario apta para cargar estos datos en la base de datos de Google Cloud Platform.

4.1.11. Función procesamiento

La función procesamiento es la función principal del algoritmo. Es aquí donde se ejecutan todas las demás funciones anteriormente descritas, y almacena los valores de la correlación y RMSE por canal. Posteriormente esta función ejecuta la comparación para determinar si la transmisión es deseada o no. Para esto se utiliza el condicional *if*, en donde si la correlación es menor a 0.2 y el RMSE es mayor a 0.01 el algoritmo determina

que la señal es no deseada y procede a almacenar la información de la potencia y frecuencia en un diccionario para posteriormente retornar esta variable. El valor de comparación en la correlación es asignado dado que se requiere buscar semejanzas entre una señal de referencia con la señal leída por el , la correlación mide la relación existente entre estas dos señales y si esta relación decae un 80 % indica que las dos señales son distintas entre sí. Para confirmar la comparación de señales, el valor del RMSE indica la cantidad de error que existe entre el conjunto de datos de estas dos señales. El valor asignado de 0.01 indica un error muy pequeño en las señales, este valor se ha escogido ya que en el procesamiento de las señales se elimina el ruido y en canales donde no exista una transmisión no deseada el RMSE será 0, por lo que el valor optimo seria el menor valor con el que el sistema detecta una transmisión no deseada.

Además, esta función crea la gráfica del canal analizado y de la frecuencia ilegal en caso de existir. Esta gráfica es almacenada en un directorio del proyecto y es empleada para mostrarla en el *frontend* de la interfaz web. Finalmente, el diccionario que se retorna tiene los valores de los datos de la potencia y el espectro radioeléctrico, la señal no deseada y la decisión si existe o no señal ilegal. Retornando el valor de 0 si no existe señal ilegal y 1 si existe señal ilegal.

4.1.12. Función sdrscantv

La librería creada para el análisis de las frecuencias de TV, emplea las mismas funciones de la librería *sdrscanfm* excepto la función setup. Esto sucede debido a que las longitudes de los arreglos para almacenar las muestras leídas difieren entre sí, porque existen canales con un mayor ancho de banda y por ende un mayor número de muestras. El cambio de dimensiones se observa en la longitud del arreglo de *psd_total*, en donde se multiplica el valor de las filas por un entero para evitar el error de las dimensiones. Este proceso es repetido en las 5 iteraciones que se realiza para analizar el espectro radioeléctrico de televisión, teniendo 5 valores diferentes para cada iteración.

4.2. Funciones del backend de la interfaz de usuario

Para el desarrollo de la interfaz web se implementa un *backend* que se conecte al servidor de la Raspberry Pi y a la base de datos de Google. Para esto, el *framework* Flask permite crear un *backend* desde cero e implementar las funciones necesarias para la conexión tanto de la interfaz que se muestra al usuario como de las librerías de escaneo del espectro radioeléctrico.

La librería *app.firestore_service* desarrollada permite la conexión del *backend* con la base de datos de Google, y las librerias anteriormente creadas *sdrscanfm* y *sdrscantv* son importadas para que obtener la información del espectro radioeléctrico analizado. Flask emplea código de Python para ejecutar las funciones de conexión al servidor y a la base de datos, para esto las funciones implementadas se describen a continuación.

4.2.1. Función index

Esta función permite obtener la dirección IP del usuario y la almacena dentro de la variable *user_ip*. Esto sirve para crear una sesión de usuario permitiendo navegar en la página con mayor seguridad, ya que la IP del usuario se oculta dentro de la variable *session*.

4.2.2. Función hello

Esta función permite la conexión con la página de inicio que se muestra en la aplicación web. La función acepta los métodos *POST* y *GET*, además la función *render template* permite renderizar una página web creada con , que en ese caso es *hello.html*, también se envía la variable contexto que contiene toda la información del usuario.

4.2.3. Función fm

Esta función permite la renderización de la página que emite la orden del escaneo del espectro radioeléctrico FM. Partiendo de la información del usuario, se ejecutan las funciones de la librería *sdrscanfm* en donde por partes se analiza todo el espectro FM y en caso de existir una frecuencia ilegal la almacena en una variable. La variable contexto contiene toda la información sobre el espectro analizado y envía a la página que muestra al usuario si existe o no una frecuencia ilegal. Finalmente, en caso de existir frecuencias ilegales, se ejecuta la función *put_alarma* que envía los datos de la transmisión ilegal a la base de datos.

4.2.4. Función tv vhf

Esta función permite la renderización de la página que ejecuta el escaneo del espectro radioeléctrico TV VHF. Usando la librería sdrscantv se escanea el espectro TV en dos iteraciones. En caso de existir una transmisión ilegal los datos son almacenados en la variable contexto y enviados a la página de TV . Al igual que la función de fm, en caso de existir una transmisión ilegal la función *put_alarma* envía la información de la transmisión ilegal a la base de datos.

4.2.5. Función tv uhf

Al igual que la función anteriormente descrita para *tv uhf*; esta función permite la renderización de la página que escanea el espectro radioeléctrico de TV UHF. Las funciones ejecutadas permiten el análisis del espectro y en caso de existir una transmisión ilegal, almacena los datos y los envía a la base de datos de Google.

4.3. Funciones del frontend de la interfaz de usuario

El código de la interfaz está dividido en varios archivos html que modularizan el código y permiten un fácil despliegue de la aplicación web. A continuación, se analiza el código implementado que despliega la interfaz de usuario.

4.3.1. Función base.html

Esta función es replicada en toda la interfaz de usuario, y contiene la información del encabezado y mensajes para el usuario. Esta función se ha creado con el fin de tener menos líneas de código para la interfaz ya que se repite en todas las demás páginas html creadas.

4.3.2. Función signup.html

Esta función html muestra la página de registro del usuario dentro de la aplicación. La función extiende o trae el archivo base.html y crea un formulario en donde la información del nuevo usuario es enviada a la base de

datos, para posteriores inicios de sesión del mismo usuario.

4.3.3. Función login.html

Esta función es la encargada de mostrar la página de inicio de sesión del usuario. Para esto se implementa un formulario en donde el usuario digita su nombre de usuario y la contraseña. Posteriormente se comparan estos valores con los valores de la base de datos, en caso de que el usuario no haya digitado correctamente sus credenciales la página retornará un mensaje de error de inicio de sesión. Tal como en la función *signup.html* se extiende la función *base.html* y se importa la librería que crea los formularios para que el usuario digite sus credenciales.

4.3.4. Función hello.html

Esta función muestra la página de inicio de la aplicación, en donde el usuario encontrará la información necesaria para utilizar el escaneo del espectro tanto de FM como de TV.

4.3.5. Función fm.html

Esta función muestra al usuario toda la información necesaria del escaneo FM. En el código existe un condicional en el que en caso de existir una transmisión ilegal se mostrará la gráfica del espectro analizado y también se mostrará una alerta que se ha detectado la transmisión ilegal. En la figura 4.6 se muestra la interfaz del usuario con la función de detección en el espectro FM, la interfaz muestra una alarma de color rojo si existe una transmisión no deseada y un mensaje en color amarillo en caso de no existir.

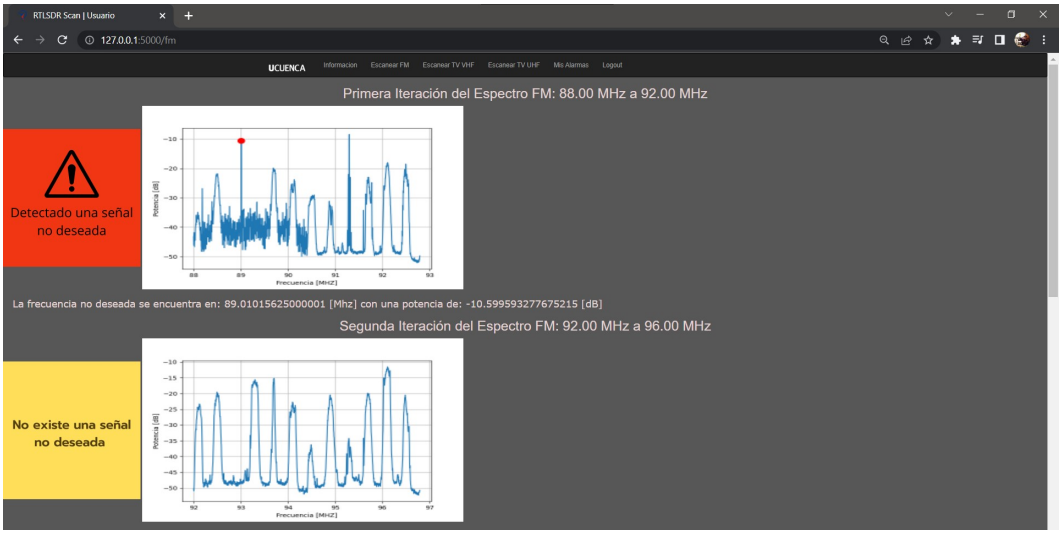

Figura 4.6: Página fm.html

4.3.6. Función tv.html

Esta función muestra la información del escaneo en el espectro radioeléctrico de TV. Además, posee la misma estructura que el archivo fm.html, en donde se divide en condicionales que permiten mostrar una imagen de alerta en caso de existir una transmisión no deseada.

4.3.7. Función alarmas.html

Esta función muestra las alarmas del usuario que se han generado por transmisiones no deseadas. Accede a la variable alarmas y crea una tabla en donde se visualiza la información de la alarma. Esta información incluye la frecuencia, potencia, fecha y a qué servicio del espectro radioeléctrico pertenece.

CAPÍTULO 5

Implementación de hardware y pruebas de laboratorio

En este apartado se describe la metodología empleada para corroborar el correcto funcionamiento del algoritmo. La figura 5.1 a continuación expone el proceso empleado en las pruebas de laboratorio. En la parte izquierda de la figura los bloques de color naranja dentro del recuadro gris, representan a los generadores de señal empleados para la emisión de la transmisión no deseada, estos equipos tienen variables como el nivel de transmisión que incrementan o disminuyen la potencia de transmisión en los equipos. La transmisión no deseada emitida es detectada en el bloque de detección de color azul, donde comprende la configuración y ejecución de los procesos de lectura del espectro, así como el análisis de datos, toma de decisiones y visualización de resultados. En la última etapa de análisis de resultados, toda la información obtenida de la efectividad del sistema de detección es visualizada en gráficas e interpretadas en función de su porcentaje de efectividad.

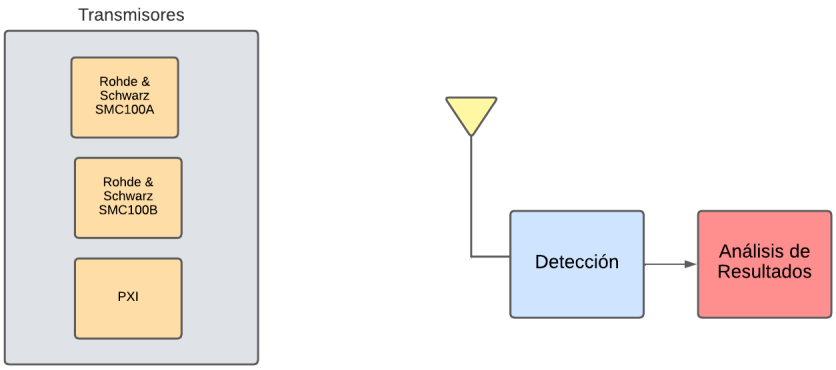

Figura 5.1: Diagrama de bloques de la secuencia de procesos en las pruebas de laboratorio

Las pruebas de laboratorio realizadas siguen una serie de instrucciones en donde se recopilan los datos de la potencia de recepción, nivel de transmisión del generador de señales, valor de la ganancia del , y si el sistema detectó correcta o incorrectamente la transmisión inducida. Con este propósito se define un conjunto de pasos a seguir para realizar estas pruebas.

1. Inicialización y configuración de parámetros de transmisión del generador de señales.
2. Configuración y registro de la frecuencia y nivel de potencia de transmisión de la señal no deseada a generarse.
3. Configuración y registro del valor de la ganancia del configurada en el algoritmo.
4. Transmisión de la señal configurada por parte del generador de señales.
5. Ejecución del sistema de detección.
6. Observación y registro de la potencia de recepción en el sistema de detección.
7. Observación de la fiabilidad del algoritmo, verificar si es capaz de detectar la transmisión no deseada correctamente y registrar el intento.
8. Volver al paso 2 para configurar un nuevo nivel de potencia de transmisión y una nueva frecuencia.
9. Análisis de resultados.

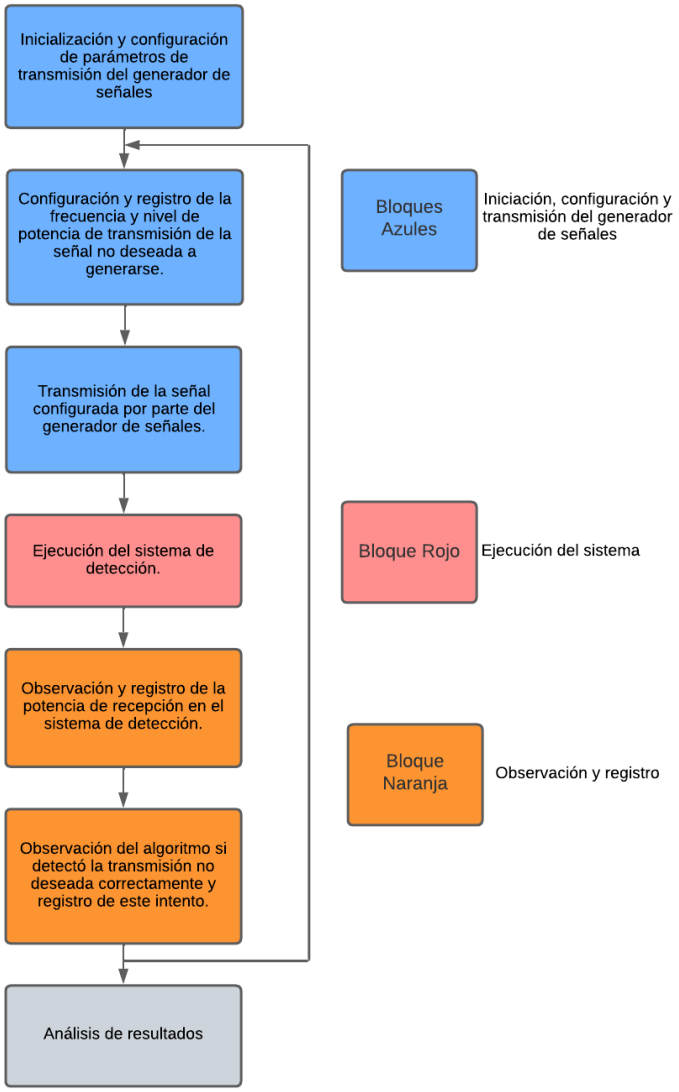

Figura 5.2: Proceso de configuración del transmisor, ejecución del sistema detector y observación de resultados.

El gráfico que representa este proceso se muestra en la figura 5.2. Los bloques azules representan a la etapa de iniciación, configuración y transmisión del generador de señales. El bloque rojo indica la etapa de ejecución del sistema de detección. Los bloques naranjas muestran la etapa de observación y registro de la prueba realizada. En este paso se registra si la detección fue correcta o en su defecto incorrecta, además de registrar los valores de la frecuencia de emisión, potencia de recepción y fecha de detección. Esta serie de pasos da la información suficiente para identificar los parámetros óptimos tanto de configuración del algoritmo de detección, así como de recepción del espectro radioeléctrico. La visualización de estos datos permite ver con facilidad las relaciones existentes entre la potencia de recepción, el nivel de transmisión del generador de señales con el punto mínimo de detección del algoritmo o umbral. De la misma manera con la visualización de datos se identificará el valor óptimo de la ganancia del relacionando este número con el número de pruebas realizadas con éxito o fracaso. Finalmente se concluirá con el análisis de resultados donde se definirán los mejores parámetros de configuración del sistema de detección y se concluirá con gráficos la eficiencia del sistema.

La toma de datos en esta etapa permite definir le eficacia del algoritmo con diferentes parámetros, como la ganancia del o la mínima señal detectable por el algoritmo. Estos parámetros son la "columna vertebral" del proyecto, ya que a partir de estos valores el algoritmo toma la decisión de si la transmisión es deseada o no. En el código de estos parámetros están representados como:

- **Ganancia del SDR:** Este parámetro es configurado dentro de la función *read_sdr* que se detalló en el capítulo 4. Este parámetro maneja la ganancia [dB]. La máxima ganancia que puede llegar a tener el dispositivo es de 24 [dB]. Dentro de la función *read_sdr*, el valor óptimo de la ganancia, permitiendo que las muestras leídas por el puedan incrementar su valor de potencia y sean detectables al algoritmo desarrollado.
- **Umbral:** Esta variable se refiere al mínimo valor detectable de potencia de recepción que el algoritmo puede detectar. Depende de factores como la cantidad de muestras para la lectura, la capacidad de procesamiento de la Raspberry Pi, así como el piso de ruido del espectro radioeléctrico.

Para encontrar el valor óptimo de ganancia del SDR, se registra en qué valores de la ganancia, el algoritmo se desempeña mejor. Para el desarrollo de este caso se emplea las nomenclaturas de verdadero positivo y falso negativo de tal manera que reflejen lo siguiente:

- **Verdadero positivo:** Existe una transmisión no deseada y el algoritmo lo detectó.
- **Falso negativo:** Existe una transmisión no deseada y el algoritmo no lo detectó.

De esta manera se podrá cuantificar con precisión la eficacia del algoritmo en donde se buscará el valor de la ganancia del SDR que tenga mayor cantidad de verdaderos positivos (VP) y menor cantidad de falsos negativos. A partir de estos valores se puede calcular la sensibilidad del sistema a partir de la siguiente ecuación:

$$VPR = \frac{VP}{VP + FN} \tag{5.1}$$

Donde la sensibilidad representa la razón de verdaderos positivos o también llamado la razón de éxitos. Con esta información se define con visualización de datos el mejor valor de ganancia para el algoritmo. El valor óptimo del umbral está definido por relaciones entre los valores de la potencia de recepción del sistema y el nivel de potencia de transmisión del generador de señales.

5.1. Implementación del hardware del sistema

El hardware necesario para el despliegue del sistema de detección se definió en el Capítulo 3, en donde se especificó las características técnicas más esenciales, tanto del equipo RTL-SDR como de la Raspberry Pi. Esta microcomputadora además de ejecutar el algoritmo de detección, muestra la interfaz de usuario. Por esta razón tendrá que tener periféricos de entrada y salida de información, como cualquier otro computador. En la figura 5.3 se muestran las conexiones de la Raspberry Pi con el monitor, ratón, teclado y modem, además de la memoria micro SD con el sistema operativo instalado.

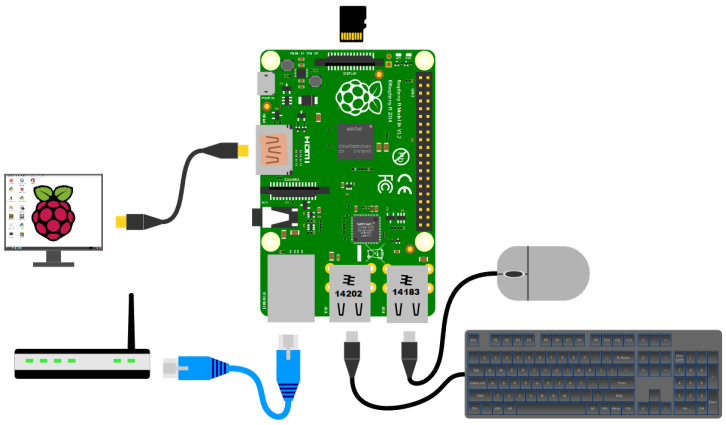

Figura 5.3: Conexiones de periféricos a la Raspberry Pi [30]

Una vez realizadas las conexiones de los periféricos, se enciende la Raspberry Pi y se espera a que muestre la pantalla de inicio. Para la descarga del programa de detección de transmisiones es necesario bajar el repositorio desde GitHub. Con este propósito abrimos una nueva terminal, creamos y entramos a un nuevo directorio que alojara el algoritmo y digitamos el comando:

git clone https://github.com/NeoGeoXL/Tesis.git

Esto descargará la última versión del algoritmo de detección, posteriormente se crea y se habilita el ambiente virtual del proyecto. Finalmente se instalan las librerías necesarias con el comando **pip install -r requeriments.txt.** Este archivo contiene todos los nombres y versiones de las librerías empleadas en el desarrollo del algoritmo. El sistema desplegado en el laboratorio se muestra en la figura 5.4:

Figura 5.4: Sistema de detección de transmisiones no deseadas instalado en el Laboratorio

De esta manera los equipos electrónicos empleados en las pruebas de laboratorio son:

- Sistema de detección de transmisiones no deseadas
- Generador de señales Rohde Schwarz SMC100A
- National Instruments PXI

A continuación, se desarrollan las ideas de las pruebas de laboratorio, analizando el funcionamiento del sistema de detección con diferentes configuraciones y con varios transmisores.

5.2. Despliegue del sistema de detección y pruebas de laboratorio

La primera prueba de laboratorio realizada se desarrolla empleando el generador de señales Rohde Schwarz SMC100A con mínimos parámetros de *LowFrequency Generator / Output* (1V de *LF output Voltaje* y 1 KHz de *Generation Frequency*, de esta manera siguiendo la metodología del diagrama de bloques de la figura 5.2, se realiza la toma de datos de los valores de la potencia de recepción y se identifica si el algoritmo detectó correctamente la transmisión inducida por el generador de señales. La figura 5.5 a continuación permite identificar la relación existente entre estas dos variables, en donde para cada valor de potencia de recepción existe un porcentaje de éxito de detección; de está manera se identifica gráficamente el valor óptimo de detección y hasta que valores de potencia de recepción es eficiente el algoritmo. En la gráfica se muestra con una "X" en rojo el punto de la señal mínima detectable o umbral del algoritmo, ubicado en -29 dBm; en este punto el porcentaje de éxito de detección es del 100 %; esto significa que los niveles del umbral superiores a -29 dBm detectarán todas las transmisiones no deseadas receptadas por el RTL-SDR. Caso contrario configurando el umbral con valores de potencia de recepción menores a -29 dBm el algoritmo tendrá menos porcentaje de éxito en la detección como se muestra en la figura 5.5.

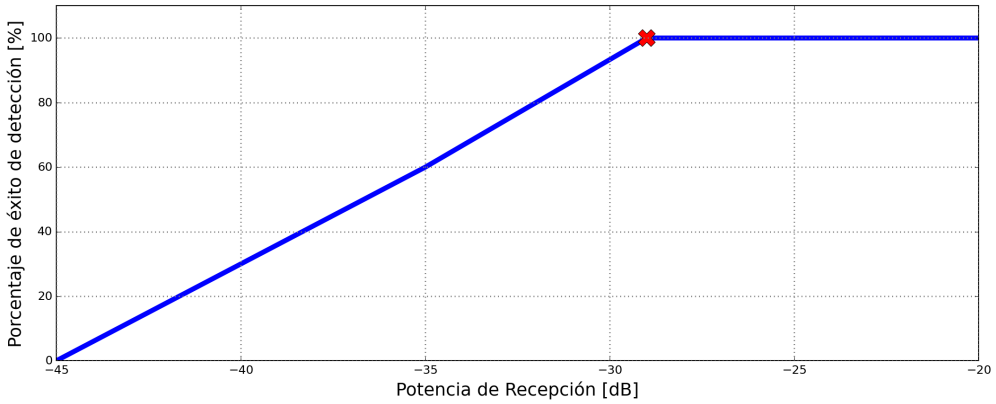

Figura 5.5: Relación de la potencia de recepción del RTL-SDR con el porcentaje de éxito de detección

Para definir el valor óptimo de la ganancia es necesario realizar pruebas que permitan registrar si el algoritmo detecto o no la transmisión no deseada. Para el desarrollo de estas pruebas se registran con diferentes valores de ganancia, cuantas veces el algoritmo detectó bien y a partir de esta información calcular la sensibilidad del valor de la ganancia expuesta en la formula (5.1) y también calcular el porcentaje de éxito de detección por ganancia. Con este fin y de acuerdo a los valores registrados y mostrados en las tablas expuestas en el anexo D, se calcula la sensibilidad del sistema a partir de los verdaderos positivos y falsos negativos.

Figura 5.6: Matriz de confusión para prueba de laboratorio con ganancia de SDR de 2 dB

La figura 5.6 muestra la matriz de confusión con la cantidad de verdaderos positivos y falsos negativos en la prueba de laboratorio con una ganancia de de 2 dB. Se observa que la cantidad de verdaderos positivos es ligeramente mayor que la cantidad de falsos negativos, dándonos valores de sensibilidad mayores al 50 %. Para ajustar la ganancia al valor optimo es necesario encontrar un valor de sensibilidad alto ya nos indica un mayor porcentaje de éxito de detección.

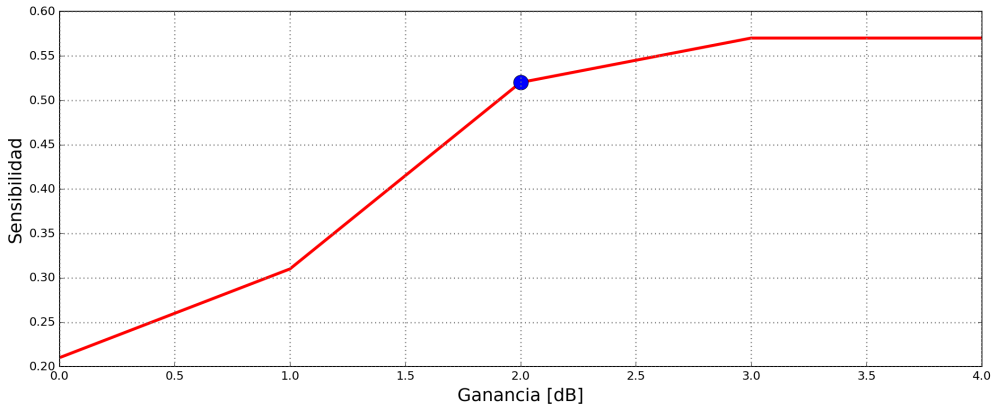

Figura 5.7: Relación de la ganancia del SDR con la sensibilidad que provee en el sistema

Con los valores de los verdaderos positivos y falsos negativos registrados, se puede calcular la sensibilidad que mide el mayor porcentaje de resultados positivos en un evento. Los datos de las pruebas del anexo D registran todos estos valores variando la ganancia del que se configura desde el código. Para cada valor de ganancia se tiene un valor de sensibilidad que indica el porcentaje de resultados positivos en el experimento. A partir de la formula 5.1 se calculan los valores de sensibilidad para cada valor de ganancia. Estos valores se muestran en la figura 5.7, donde se observa que para un valor de 2 dB de ganancia, se tiene una sensibilidad mayor a 0.50 y sigue una tendencia creciente. A partir de este valor de ganancia el sistema de detección tiende a identificar transmisiones legales como no deseadas, disminuyendo su eficiencia, por lo que no sería óptimo tener una ganancia mayor a 2 dB.

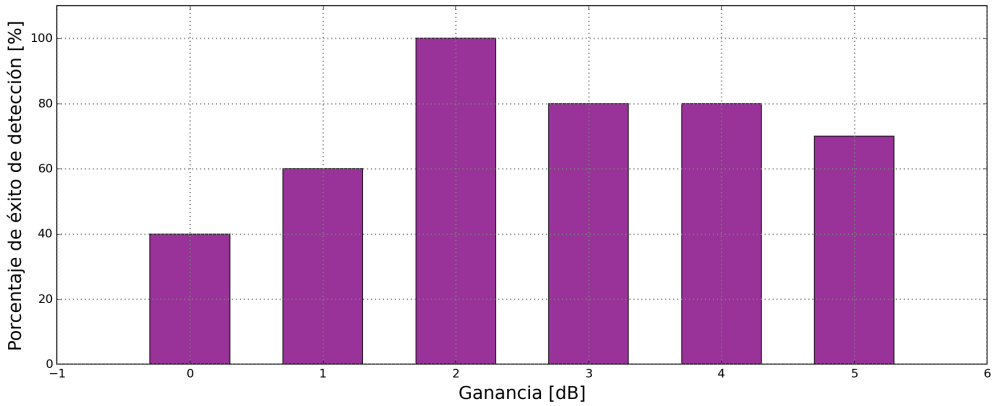

Figura 5.8: Diagrama de barras de la ganancia del SDR con el porcentaje de éxito de detección

El 100 % de éxito de detección del diagrama de barras de la figura 5.8 se ubica en el valor óptimo de ganancia que es 2 dB y a partir de este valor se define los porcentajes de éxito de detección para cada valor de ganancia. Realizando 10 pruebas con diferentes niveles de potencia de recepción y para cada nivel de ganancia definido, se observa que al configurar una ganancia de 1 dB o inferior, el sistema tendrá un porcentaje de éxito de detección menor al 60 %; a partir de los valores de ganancia de 2 dB se observa un decrecimiento en el porcentaje de éxito de detección debido a que el algoritmo etiquetará transmisiones legales de canales de FM o de TV como transmisiones no deseadas. De esta manera visualmente se define el valor adecuado de la ganancia como 2 dB.

En el proceso de lectura del espectro radioeléctrico, uno de los parámetros de configuración en el RTL-SDRes la variable encargada de configurar cuantas muestras del espectro obtener y otra la cantidad de muestras que se necesita para la lectura del espectro por el RTL-SDR, la cantidad de muestras es el resultado de la multiplicación del número de muestras en la *Power Spectral Density* (PSD) y el número de PSD que se van a promediar.

$$Muestras = npsd\_res * npsd\_avg \tag{5.2}$$

La cantidad de muestras para el escaneo del espectro se define con el producto de la fórmula 5.2. Para el procesamiento y cálculo del espectro se emplea la función plt.psd() y esta función necesita la variable *NFFT* que es el número de puntos o datos empleados en la Fast Fourier Transform (FFT). A este valor de *NFFT* se le asigna la variable *npsd_res* (el número de muestras para la PSD). La variable *npsd_res* asignada anteriormente para el cálculo de las muestras en el escaneo será ahora la cantidad de muestras del espectro que se desea obtener. El que las muestras sean definidas con este producto permite variar el número de muestras en el espectro y también variar el número de PSD a promediar, llegando así a un valor optimo del producto de estas dos variables.

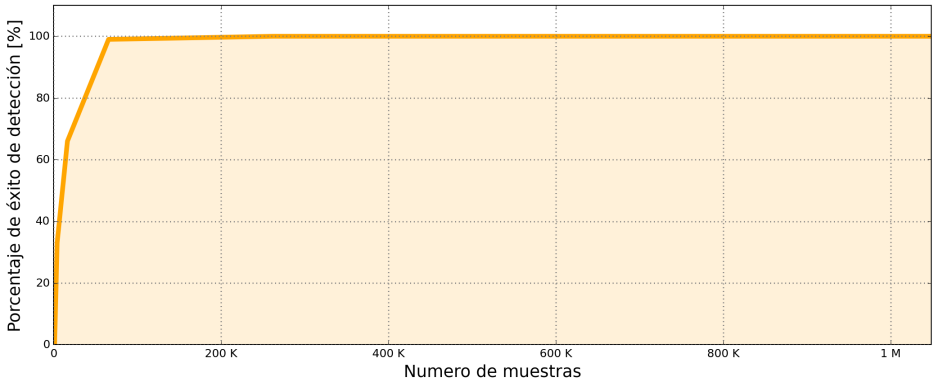

Figura 5.9: Relación de la cantidad de muestras usadas en la lectura del espectro y el porcentaje de éxito de detección

Como se ve en la figura 5.9, el número óptimo para que el sistema alcance un 100 % de éxito de detección es alrededor de 100 000. Esto es asignando a un valor de 512 para la variable *npsd_res* y 256 para la variable npsd_avg. Es decir para tener un porcentaje de éxito de detección del 100 %, es necesario que el sistema tenga por lo menos 512 valores escaneados de su espectro por cada señal con su frecuencia. Por debajo de las 100 000 muestras, el sistema empieza a distinguir señales de ruido y señales autorizadas como señales ilegales, por lo que el punto óptimo es sobrepasar las 100 000 muestras.
A mayor número de muestras asignadas se requiere mayor capacidad de computacional, una Raspberry Pi 3, llegará a su límite de procesamiento y carga computacional alrededor de 512 en *npsd_res* y 512 en *npsd_avg*, por lo que para bajar el umbral y seguir teniendo un 100 % de éxito de detección es necesario aumentar el número de muestras por encima de las 200 000.

La capacidad de procesamiento que se tiene en la Raspberry no es suficiente para alcanzar un punto más bajo de detección por lo que al ejecutar el sistema en una computadora de gama media con Windows

permite niveles más bajos del nivel de detección teniendo un 100 % de éxito de detección con señales de -50 dBm.

Las siguientes figuras muestran la operación del sistema de detección con una frecuencia no deseada inducida en 91.10 MHz.

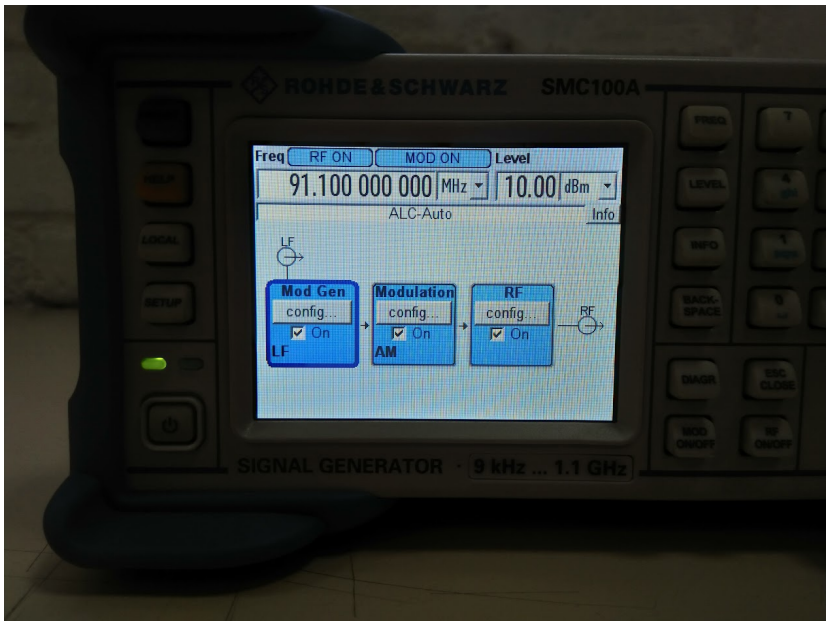

Figura 5.10: Configuración del generador de señales

El algoritmo desarrollado detecta la señal no deseada enviada desde el generador de señales en la frecuencia de 91.10 MHz. Como se puede ver en la figura 5.11, la detección hecha por el algoritmo tiene éxito definiendo la alerta en color rojo; además se indica con un punto rojo, en la gráfica en que parte del espectro se ubica la transmisión no deseada.

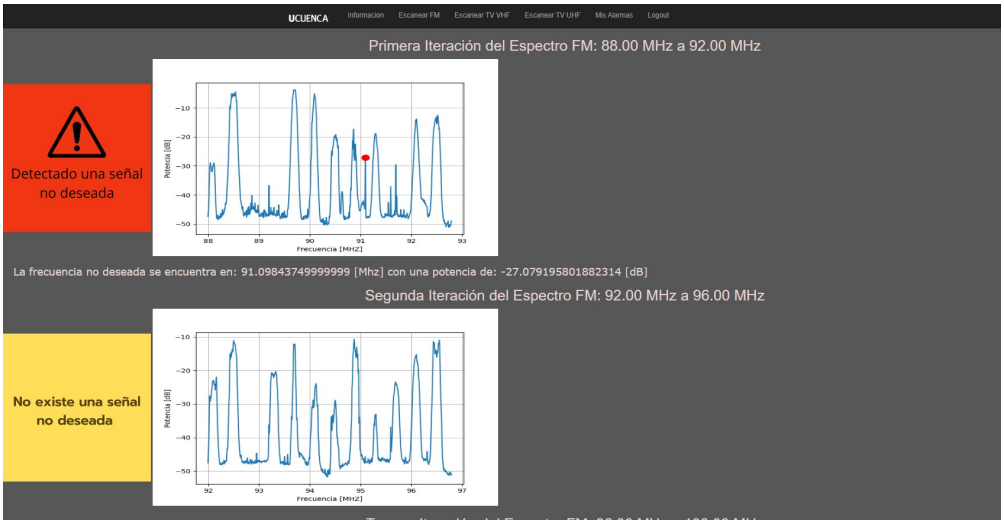

Figura 5.11: Detección de transmisión no deseada en 91 MHz

Como se vio en el capítulo anterior, el sistema desarrollado toma decisiones en base al diccionario de Python que tiene la información de si un canal está legalmente en uso o está libre. La ejecución del sistema en cada grupo de canales permite la detección de la transmisión no deseada en ese grupo permitiendo al sistema detectar más transmisiones en los grupos de canales restantes. La modularización de las funciones principales del sistema en el código permite la ejecución del proceso de detección una cantidad igual al número de canales que se desea monitorear.

Para corroborar el funcionamiento del sistema con varios transmisores, se emplea el generador de señales Rohde Schwarz SMC100A y el generador de señales del PXI. Para el primer generador el nivel de transmisión es de 16 dBm y en el PXI el nivel de transmisión es de 4 dBm. El sistema al detectar estas dos transmisiones asegura que la potencia de recepción superó el umbral de -29 dBm. Evidenciando que el nivel de transmisión de cualquier generador de señales es independiente de la eficiencia del sistema.

La figura 5.12 muestra el despliegue de los equipos generadores del señal en el laboratorio. Las transmsiones generadas por estos equipos se muestra en la figura 5.13 en donde el sistema las identifica como no deseadas y las muestra en pantalla.

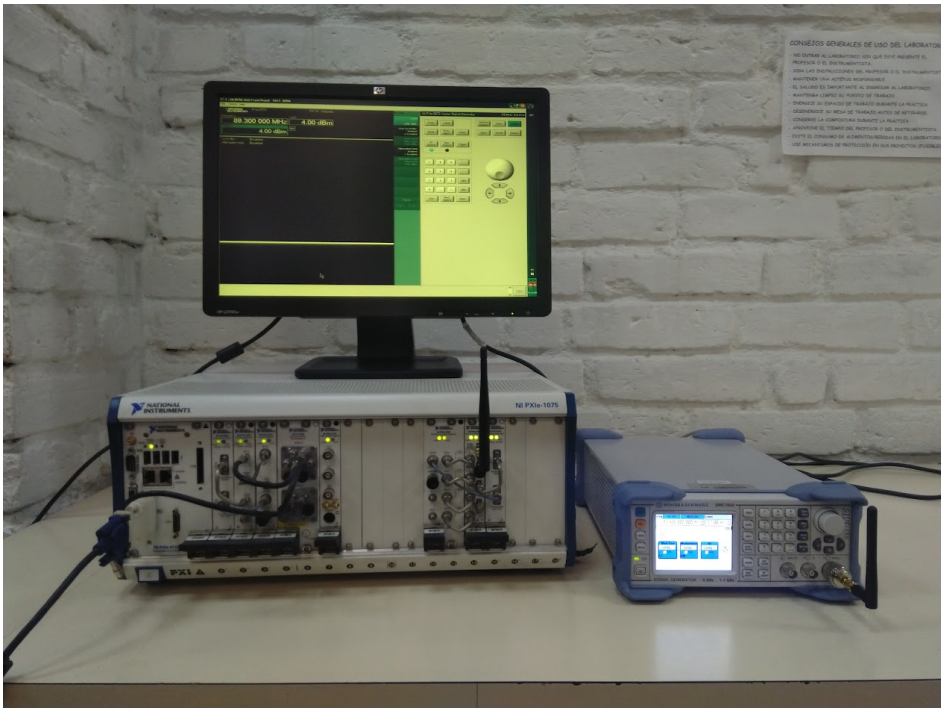

Figura 5.12: Configuración del PXI y del generador de señales Rohde Schwarz SMC100A

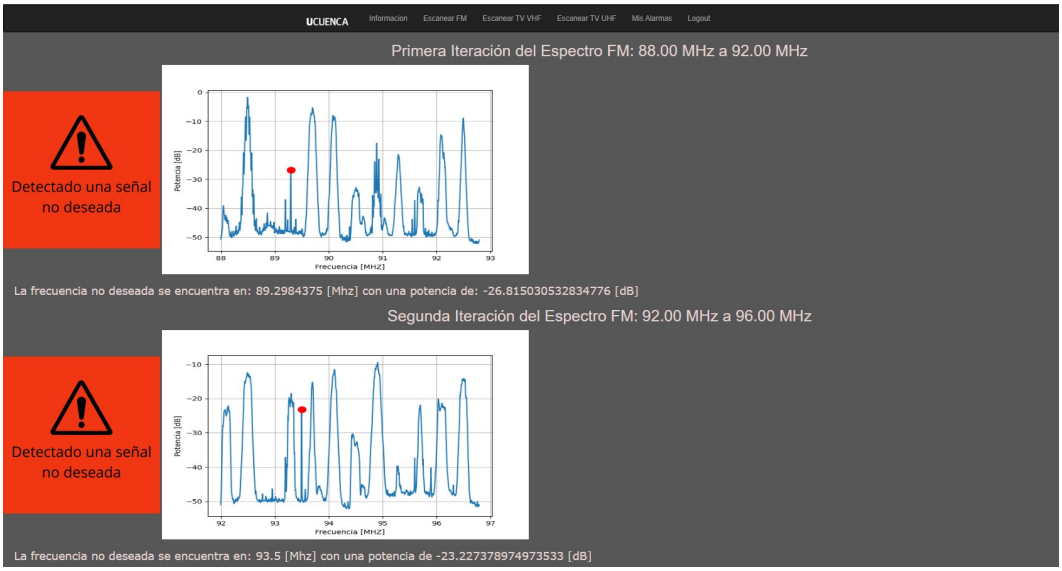

Figura 5.13: Transmisiones de FM detectadas por el algoritmo

El sistema al identificar estas transmisiones asegura que las señales enviadas superan el umbral de detección e inicia el proceso del registro de la transmisión. El umbral también puede variarse para detectar transmisiones de más baja potencia.

En las pruebas con los equipos generadores de señal, el mínimo nivel de potencia identificable en el PXI es de 4 dBm mientras que para el generador de señales Rohde Schwarz SMC100A, es alrededor de los 16 dBm. Estos valores no influyen en la eficiencia del sistema, pues el algoritmo necesita que se envién señales por encima de los -29 dBm independientemente del nivel de transmisión del generador de señales empleado; por lo que, los parámetros que se configuren en los transmisores no importaran siempre y cuando superen el umbral.

Capítulo 6

Análisis de resultados

Las pruebas realizadas al sistema de detección arrojan como resultados el valor óptimo del umbral, visualizado las relaciones existentes entre la potencia de recepción y el porcentaje de éxito de detección. En la figura 6.1 se muestra el valor óptimo del umbral en -29 dBm, esto indica que para valores de umbral superiores a -29 dBm el algoritmo detectará las transmisiones no deseadas receptadas por el RTL-SDR con un 100 % de efectividad.

El porcentaje de éxito de detección se obtiene de las pruebas realizadas al sistema, variando la potencia de recepción de la señal mínima detectable y registrando si para ese valor de potencia de recepción la señal pudo detectarse o no. El cálculo del porcentaje se lo realiza empleando 10 pruebas de detección y registrando cuantas veces para ese el valor de potencia se pudo detectar. Para señales por encima de los -29 dBm de cada 10 pruebas en las 10 el sistema detecta la transmisión no deseada. A medida que se baja la potencia de recepción de la señal mínima detectable, el sistema disminuirá su efectividad a tal punto que si la potencia de recepción de la señal mínima detectable el sistema solamente identificara 6 de cada 10 transmisiones no deseadas.

La correlación y el RMSE intervienen en cálculo del porcentaje de detección ya que estos valores definirán si la transmisión no es deseada, para ello en la correlación se busca un valor que defina una relación débil entre las señales y en el RMSE se busca un error pequeño en los conjuntos de datos de las señales. Caso contrario si se define una correlación alta y un RMSE alto el porcentaje de éxito de detección disminuirá

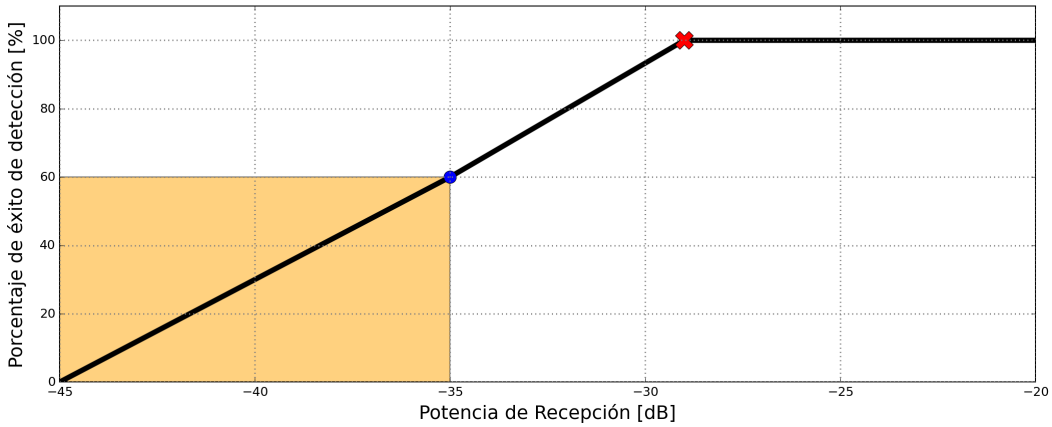

Figura 6.1: Relación de potencia de recepción con el porcentaje de éxito de detección

Además se puede concluir que, a niveles inferiores del punto óptimo del umbral, la eficiencia del algoritmo decae considerablemente, dado que, si este umbral se decrementa en 6 dB el algoritmo tendrá una eficiencia menor al 60 %, lo que indica que el sistema deja de ser eficiente. El área de color naranja muestra la zona donde el algoritmo no es eficiente debido a que no detecta las transmisiones no deseadas por debajo de un nivel de recepción de -35 dBm.

El porcentaje de éxito de detección para cada valor de ganancia del definido se muestra en la figura 6.2, donde se identificó que el mejor valor de ganancia para el sistema de detección es de 2 dB. Esto es debido a que niveles superiores de este valor, el sistema identifica transmisiones legales como no deseadas, generando errores en la toma de decisiones y disminuyendo la eficacia de detección del algoritmo.

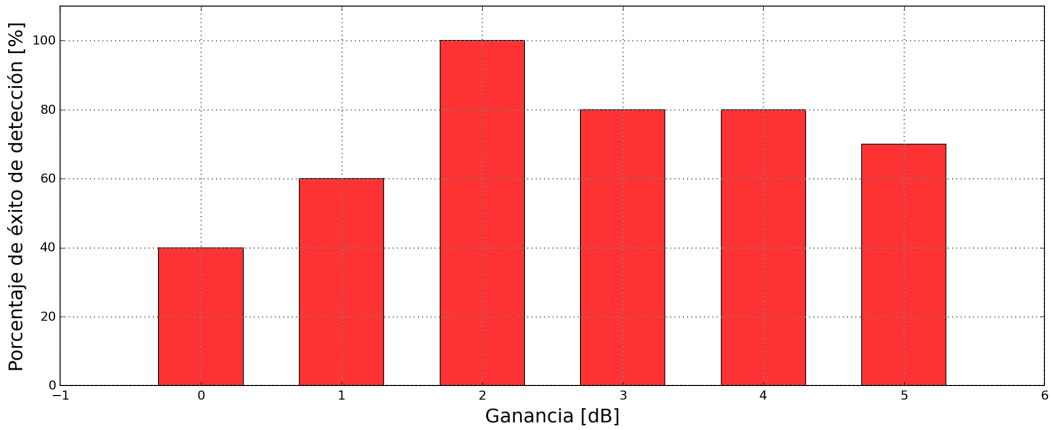

Figura 6.2: Relación de la ganancia con el porcentaje de éxito de detección

Para el proceso de lectura del espectro radioeléctrico y cálculo de la *Power Spectral Density* (PSD) es necesario definir el número de muestras que tendrá cada señal leída. Para este caso el número de muestras se define en la ecuación 6.1

$$Muestras = npsd\_res * npsd\_avg \tag{6.1}$$

En donde la cantidad de muestras es el producto del número de PSD a promediar y el número de muestras que se envía como parámetro para el cálculo de la PSD en Matplotlib. Las variaciones de estos parámetros permiten lecturas del espectro más precisas y por ende un mejor procesamiento del algoritmo de detección. Como se ve en la figura 6.3 para que el sistema tenga un porcentaje de éxito de detección se necesita mínimamente que ambos parámetros tengan más de 200 muestras.

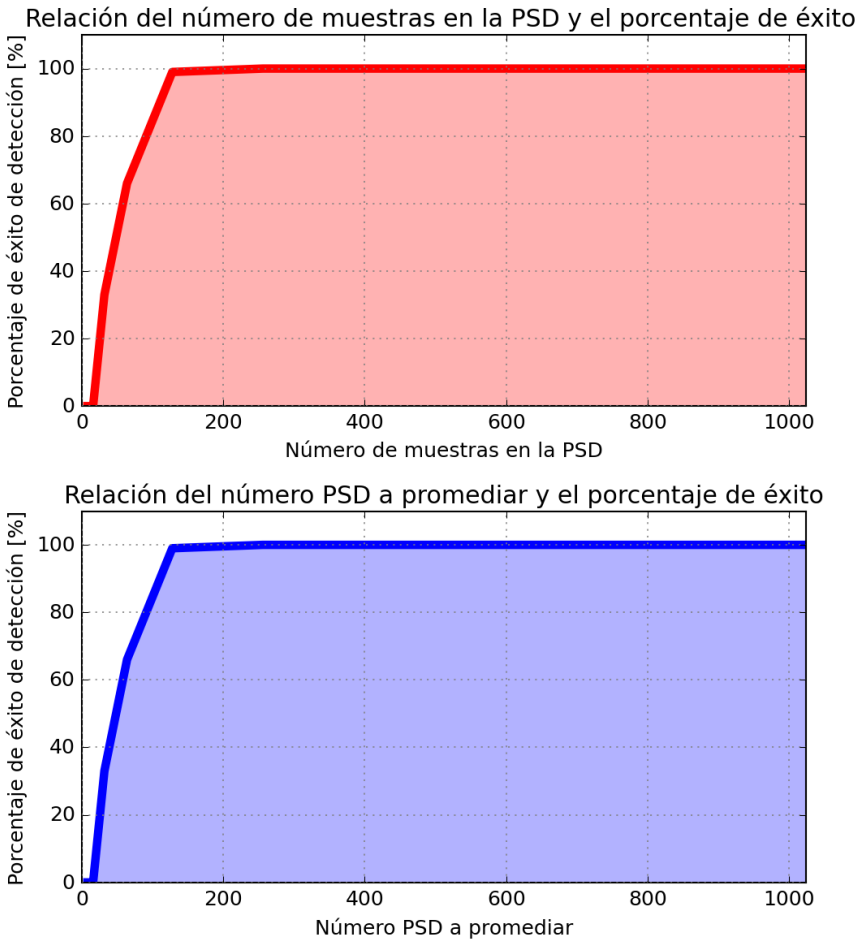

Figura 6.3: Relación del porcentaje de éxito de detección con el número de muestras en la PSD y el número de PSD a promediar

Entonces para el número de PSD a promediar se toma 256 muestras y para el número de muestras en la PSD se toma el doble o 512 muestras. Configurado este valor de muestras en el código del sistema de detección en la Raspberry Pi, la eficiencia del sistema alcanza su máximo valor y detectará todas las señales no deseadas que sean mayores o iguales al umbral o mínima señal detectable. Al intentar bajar el umbral de detección en la Raspberry Pi el sistema disminuirá su porcentaje de éxito de detección a la mitad. La capacidad de procesamiento de la Raspberry Pi 3 no es suficiente para procesar un número de muestras mayor que permita leer e identificar transmisiones por debajo de los -30 dBm.

Las muestras al ser un producto, sus variables pueden variarse de acuerdo al número de muestras que se quieran en la lectura del espectro radioeléctrico y para poder alcanzar el 1 millón de muestras en la lectura del espectro es necesario que las variables del producto número de PSD a promediar y número de muestras

en PSD tengan 1024 muestras cada una. Para poder ejecutar el algoritmo con este número de muestras y no tener problemas con la capacidad computacional, se ejecuta el sistema en Windows que además es el sistema operativo donde se desarrolló el sistema de detección.

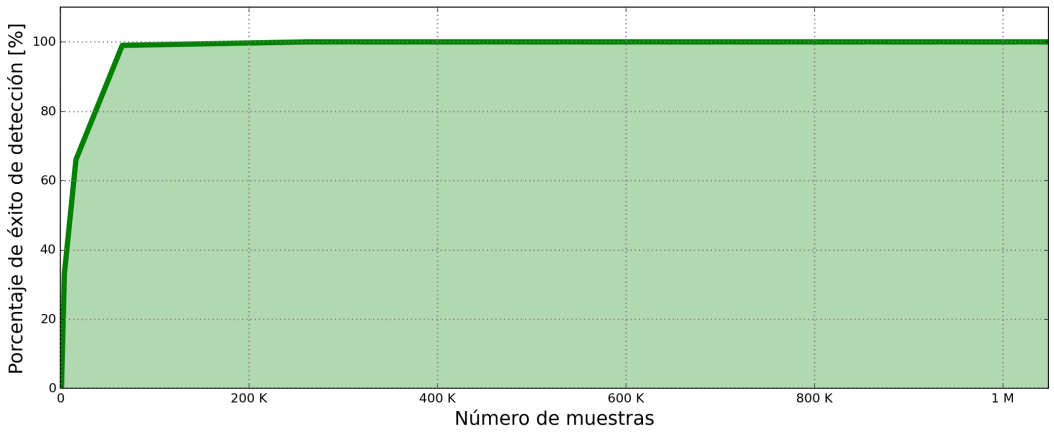

Figura 6.4: Relación del número de muestras para el cálculo del espectro radioeléctrico y el porcentaje de éxito de detección

En la figura 6.4 se visualiza que a partir de las 100 000 muestras el sistema de detección tiene un porcentaje de éxito de detección del 100 % ya que aumentando el número de muestras, aumentará el porcentaje de éxito de detección. Esto se evidencia también en el número de muestras en la PSD y la cantidad de PSD a promediar. Los mejores resultados se encuentran cuando estas dos variables son mayores o iguales que 256 muestras. Además, el número de muestras está relacionado con la capacidad computacional del hardware, porque para detectar transmisiones muy pequeñas o cerca del piso de ruido del espectro es necesario trabajar con un valor de 1 millón de muestras para la lectura del espectro radioeléctrico por parte del . Este valor indica que mínimamente se debe tener 1024 muestras para el cálculo de la PSD por parte de Matplotlib.

Para alcanzar el 1 millón de muestras en la lectura del espectro es necesario trabajar en Windows con un computador de gama media, en este caso al desarrollar el sistema en Windows no es necesario repetir la descarga del programa de GitHub y la instalación de las librerías. Ejecutando el sistema de detección en Windows, se obtienen mejores resultados que en la Raspberry Pi 3, pues la capacidad de computacional de una computadora de gamma media es mayor a la capacidad computacional de una Raspberry Pi 3.

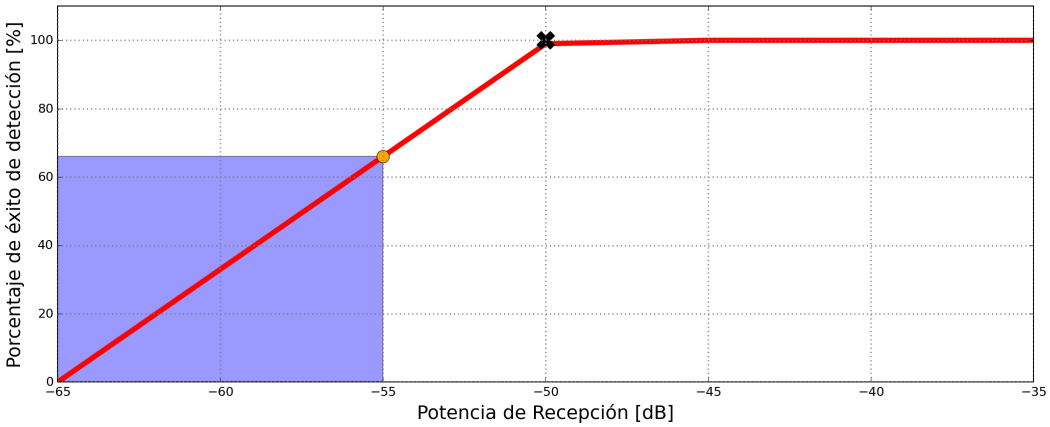

Figura 6.5: Relación del umbral de detección en Windows con el porcentaje de éxito de detección

Con 1 millón de muestras configurado el sistema en Windows, se detecta transmisiones no deseadas de más baja potencia, hasta un nivel de -50 dBm con un 100 % de éxito de detección. Esta relación se visualiza en la figura 6.5 del sistema ejecutado en Windows, donde el umbral ha alcanzado un nivel de -50 dBm con un 100 % de éxito de detección y a partir de este nivel de potencia de recepción la eficiencia del sistema decae paulatinamente hasta la zona marcada por el recuadro azul indicándonos los valores de potencia de recepción en los que el sistema es ineficiente al detectar menos del 60 %.

Al realizar pruebas, el sistema puede identificar varias transmisiones no deseadas en una misma ejecución, ya que el código desarrollado procesa los canales de FM y TV en grupos y ejecuta las funciones de lectura y procesamiento del espectro radioeléctrico. La modularización del código permite identificar varias transmisiones no deseadas siempre que estas señales se encuentren en diferentes grupos de canales. Tanto para FM como de TV el algoritmo se divide en 5 grupos de canales permitiendo un procesamiento del espectro en partes. El sistema detecta varias transmisiones siempre que estas sean mayores o iguales al umbral definido. La figura 6.6 muestra la detección de 3 transmisiones no deseadas en el espectro radioeléctrico de FM.

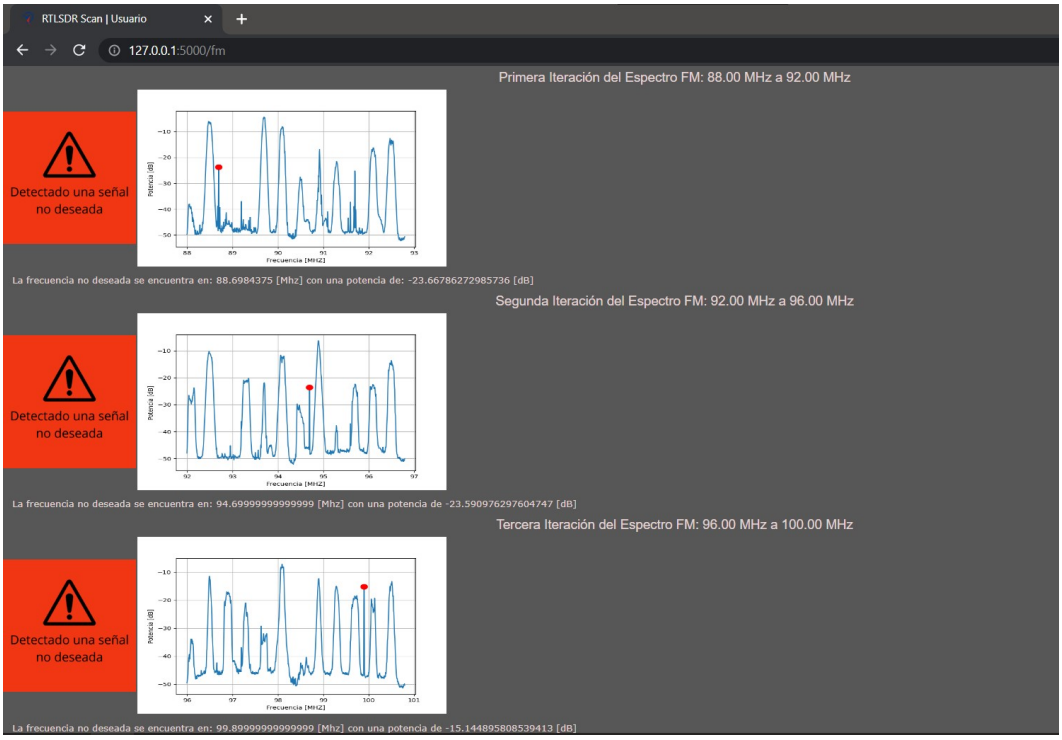

Figura 6.6: Identificación de 3 transmisiones no deseadas en el espectro radioeléctrico de FM

Todas las pruebas realizadas arrojan resultados positivos en la detección de las transmisiones no deseadas inducidas por un generador de señales. Como se pudo ver en las imágenes, el algoritmo identifica solamente señales que estén por encima del umbral configurado, ya que este umbral permite un procesamiento eficiente en el algoritmo al eliminar el ruido radioeléctrico presente en todo el espectro radioeléctrico. El algoritmo es capaz de detectar varias transmisiones no deseadas; el código está desarrollado para que se procese los datos por grupos de canales. Esto permite que el análisis planteado pueda ser replicado en todos los grupos de canales tanto de FM como de TV. Finalmente, el algoritmo presenta como una alerta todas las transmisiones no deseadas identificadas, las muestra en la interfaz, y almacena toda la información de la transmisión no deseada identificada en una base de datos que es mostrada en la interfaz del usuario en la pestaña Mis Alarmas.

CAPÍTULO 7

Conclusiones y Recomendaciones

7.1. Conclusiones

Después del análisis, planteamiento, desarrollo y pruebas del algoritmo de detección de transmisiones no deseadas, se puede concluir que:

- Los estudios revisados en el estado del arte, indican que la técnica mayormente empleada para la detección de transmisiones no deseadas es el método de detección de energía o de potencia de la señal. Este método provee un umbral o límite de potencia en donde las señales leídas pueden ser procesadas paralelamente para identificar la transmisión no autorizada.
- El desarrollo del algoritmo toma el método de detección de potencia y conjuntamente con operadores estadísticos como la correlación y el error cuadrático medio, permite la identificación de transmisiones no deseadas.En la correlación se busca un valor que defina una relación débil entre las señales y en el RMSE se busca medir la diferencia entre los conjuntos de datos de las señales, buscando un valor minino que permita comparar señales con eficiencia.
- Las librerías de Python empleadas en el desarrollo permiten la fácil lectura, manipulación y visualización de los datos, esto conjuntamente con lenguajes de programación de desarrollo web como HTML y CSS permiten que el algoritmo desarrollado sea presentado de manera amigable para el usuario.
- El valor de la ganancia del RTL-SDR configurado en el algoritmo ha sido escogido mediante las pruebas del algoritmo en el laboratorio en donde se ha considerado que este valor no altere las lecturas del espectro radioeléctrico en los demás canales que procesa el algoritmo. El valor de la ganancia configurado en el algoritmo es de 2 dB, dándonos los mejores resultados con respecto a la cantidad de verdaderos positivos, falsos negativos y falsos positivos.
- El algoritmo es eficiente al momento de identificar transmisiones que superen el valor del umbral, este valor se ha escogido mediante la caracterización del espectro radioeléctrico en la región y el análisis del piso del ruido dentro del espectro.
- Para tener niveles de detección más bajos es necesario aumentar el número de muestras en la lectura del espectro y por ende aumentar la capacidad computacional del hardware. El sistema alcanzar a detectar transmisiones por los -50 dBm ejecutando el código en una computadora de gama media con

Windows. El sistema de detección por defecto se ejecuta en la Raspberry Pi modelo 3 B+, teniendo un porcentaje de éxito de detección del 100 % con potencias de recepción mayores a -29 dBm, para poder alcanzar potencias de recepción menores que se aproximen a los resultados de la ejecución del sistema en Windows, es necesario incrementar la capacidad computacional de la microcomputadora. Con los nuevos modelos de Raspberry Pi 4 se puede alcanzar resultados próximos al del ordenador ya que incrementan significativamente su poder de computo frente a modelos anteriores como el que se usa en este trabajo de titulación.

- A un mayor número de muestras para lectura del espectro radioeléctrico mayor es el porcentaje de éxito de detección en el sistema, siendo el sistema eficiente a partir de las 100 mil muestras. Se configuran el mayor número de muestras posibles para la Raspberry Pi y Windows.
- Las pruebas realizadas indican que el algoritmo es capaz de identificar cualquier tipo de transmisión con cualquier modulación, pues al solo depender la toma de decisiones de los valores de la potencia, los distintitos tipos de modulación son identificables por el algoritmo.
- El procesamiento de los canales se hace en grupo tanto para FM como de TV. Esto permite que el algoritmo la detección de varias transmisiones no deseadas en una misma ejecución, siempre que cada señal procesada e identificada sea mayor al umbral.

7.2. Recomendaciones

- Al instalar las librerías necesarias para la ejecución del equipo RTL-SDR Blog v3 es necesario realizar varias pruebas con diferente software de visualización del espectro radioeléctrico como Sharp de tal manera que se corrobore que el equipo esta correctamente configurado en la computadora en donde va a correr el algoritmo.
- Asegurarse que todos los conectores, cables y puertos que tiene el RTL-SDR este en buenas condiciones, así como sus conexiones estén debidamente aseguradas, ya que puede existir lecturas de datos erróneas por parte del RTL-SDR si no se tiene un buen uso.
- Se puede reemplazar el dispositivo RTL-SDR Blog v3 por otro SDR que cumpla las características técnicas que se ha planteado en este trabajo de titulación, además se tiene que tener en cuenta que al momento del reemplazar por otro dispositivo este tenga compatibilidad con la librería Pyrtlsdr que es la encargada de la toma de datos en el algoritmo.
- Al instalarse el sistema de detección en un equipo diferente ya sea Raspberry Pi o una PC con Windows, es necesario que el operador encuentre los valores óptimos de número de muestras para el escaneo del espectro, ya que esta es la única variable que depende de la capacidad computacional del equipo con la que se trabaje.

7.3. Trabajos futuros

- Desarrollo de una función que permita el análisis del espectro radioeléctrico por región.
- Empleo y configuración de equipos de lectura de espectro más complejos que permitan una mejor captura de datos y procesamiento.
- Desarrollo de una interfaz de usuario más compleja.
- Análisis y mejora del algoritmo de toma de decisión de si una transmisión es legal o no.
- Desarrollo de una versión portátil de este proyecto.

ANEXO A

Manual de Usuario

La primera vez que el usuario va a emplear el algoritmo es necesario que se cree una cuenta digitando el nombre de usuario y la contraseña. El proceso descrito sirve para que los datos de las transmisiones identificadas se guarden en la cuenta del usuario y para que el algoritmo enseñe todas las alarmas que el usuario ha detectado con el algoritmo.

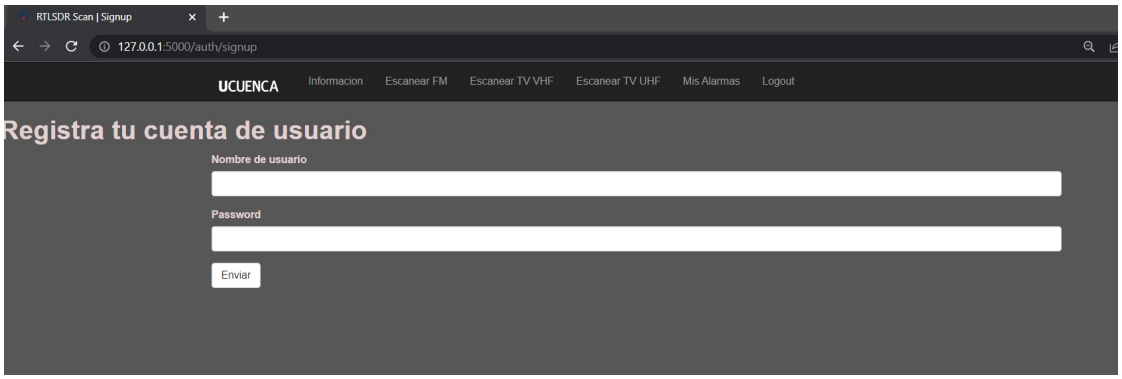

Figura A.1: Ventana de registro de usuario

Para que el usuario pueda acceder a su cuenta es necesario que primero inicie la sesión, digitando el nombre de usuario y la contraseña que ha introducido al momento de crear la cuenta.

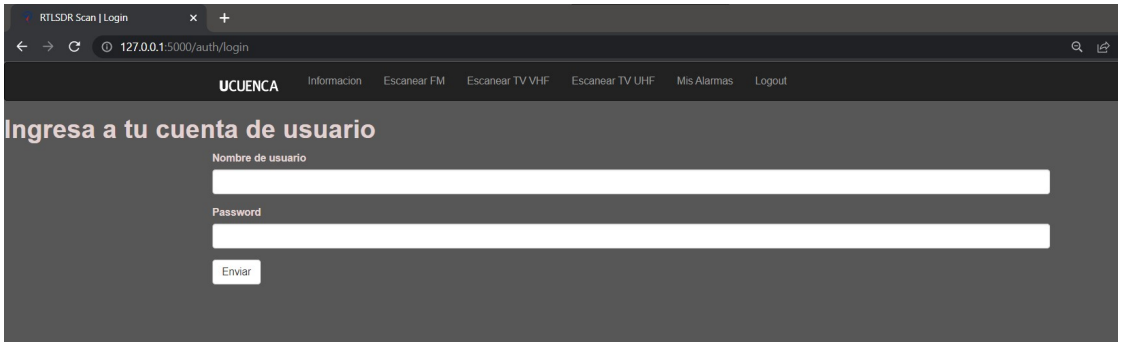

Figura A.2: Ventana de ingreso a la cuenta de usuario

Una vez iniciada la sesión, el usuario accederá a la pestaña de información, en donde se muestra una breve introducción del funcionamiento y propósito del algoritmo. Esta página es solamente de información.

Figura A.3: Ventana de información

La primera opción de escaneo de frecuencias es la opción de Escanear FM como se muestra en la siguiente figura, al dar clic en esta opción el usuario ejecutara el algoritmo desarrollado en Python de detección de transmisiones no deseadas.

Figura A.4: Opción de Escaneo FM

Al no existir una transmisión no deseada el algoritmo mostrara un mensaje en el recuadro amarillo indicando que no existe una señal no deseada, este proceso es repetido para las 5 iteraciones de los grupos de canales.

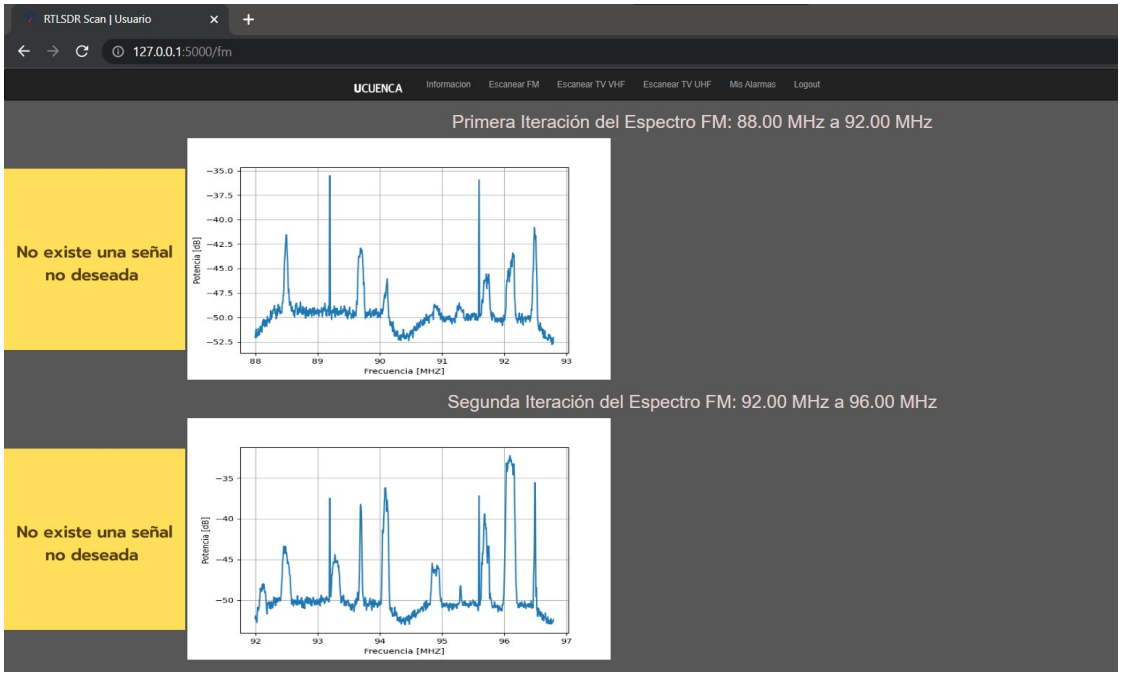

Figura A.5: Ventana de Escaneo FM al no encontrar transmisiones no deseadas

En caso de que exista una transmision no deseada aparecera un mensaje en un recuadro de color rojo y con una alerta especificando que existe una señal no deseada, segidamente se mostrara un mensaje al usuario indicando la frecuencia y potencia de esta señal no deseada. También aparecerá un punto rojo en el grafico del espectro radioelectrico de FM.

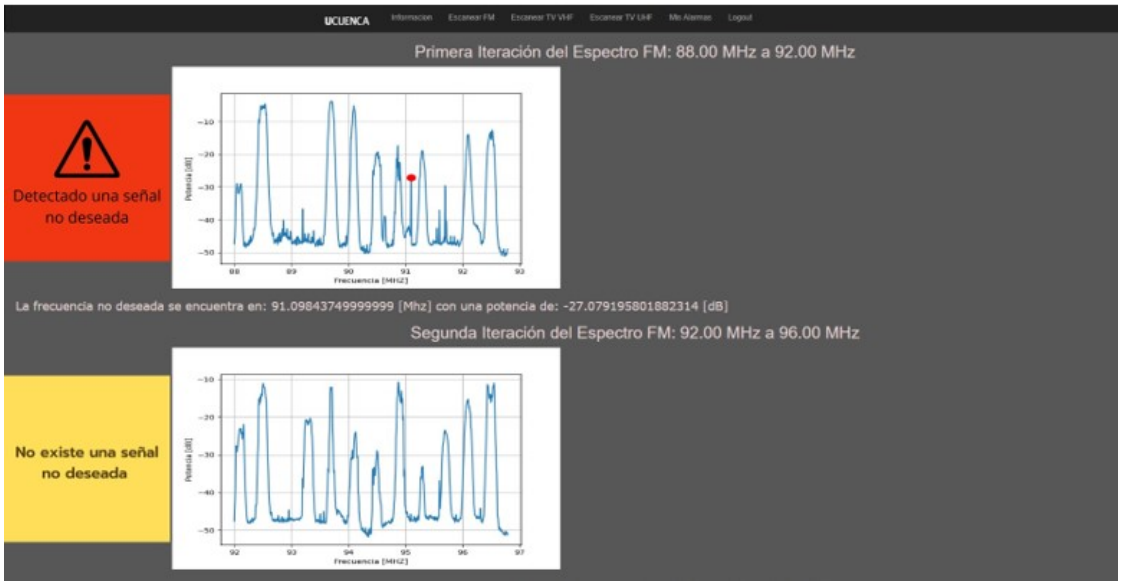

Figura A.6: Ventana de Escaneo FM al encontrar transmisiones no deseadas

La segunda opción de Escanear TV VHF, ejecuta el algoritmo de detección de transmisiones no deseadas para el espectro radioeléctrico de TV VHF, para este caso, se tiene dos iteraciones que engloban los 13 canales existentes en el país de transmisión de TV VHF.

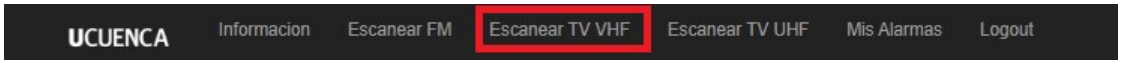

Figura A.7: Opción de Escaneo TV VHF

De igual manera que en el análisis del espectro radioeléctrico de FM, el algoritmo mostrara un mensaje señalando que no existen señales no deseadas, en caso de no haberlas, caso contrario mostrará una alerta de color rojo, como se puede ver en las siguientes figuras.

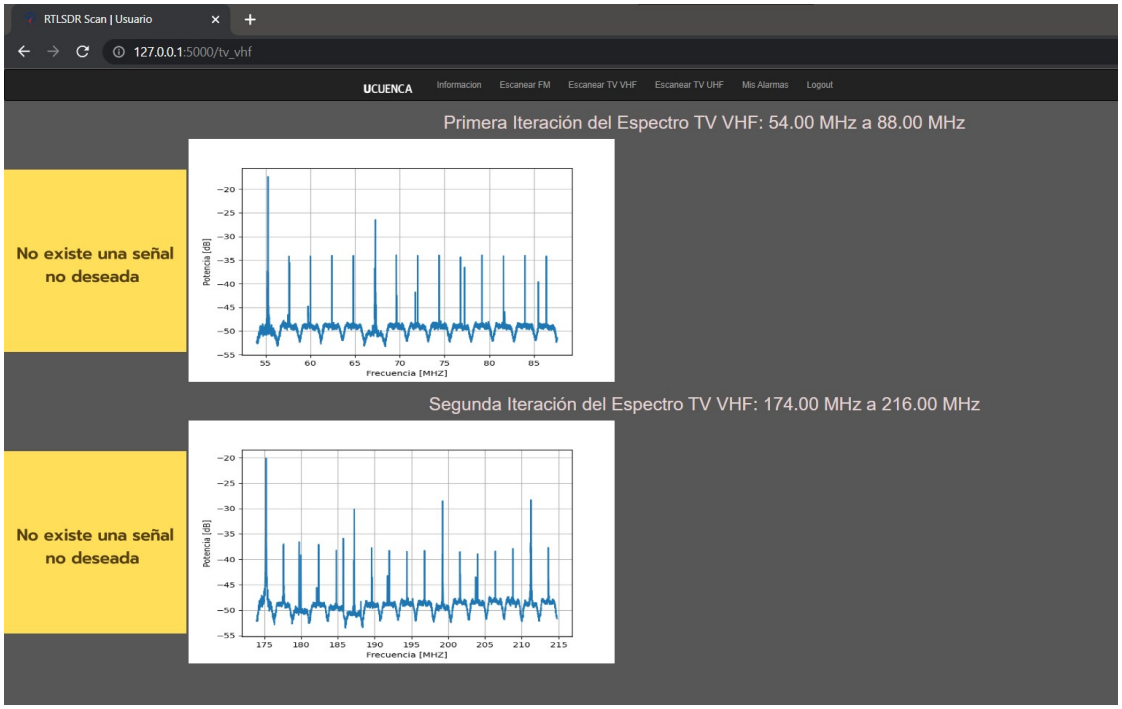

Figura A.8: Ventana de Escaneo TV VHF al no encontrar transmisiones no deseadas

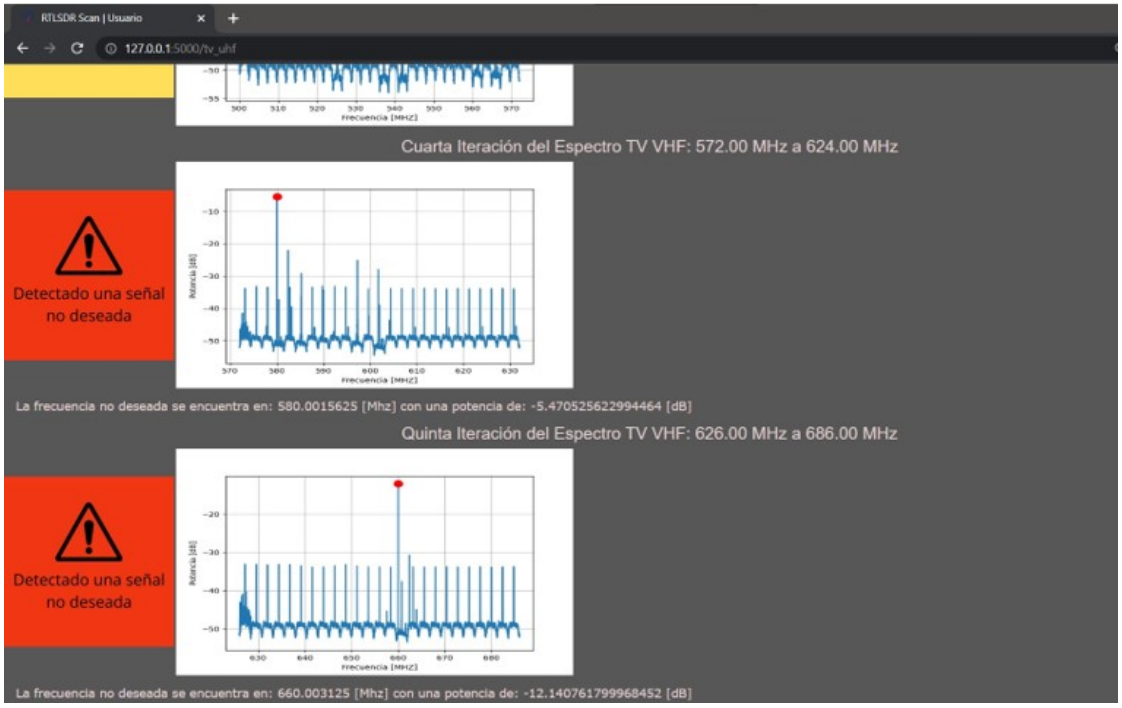

Figura A.9: Ventana de Escaneo TV VHF al encontrar transmisiones no deseadas

La última opción de escaneo es la del Escaneo TV UHF, en la que el algoritmo ejecutará el análisis en esta parte del espectro radioeléctrico y como en casos anteriores mostrará una alerta en caso de existir una o más transmisiones no deseadas en los canales de TV UHF.

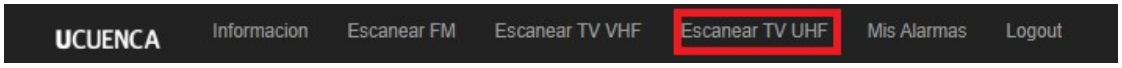

Figura A.10: Opción de Escaneo TV UHF

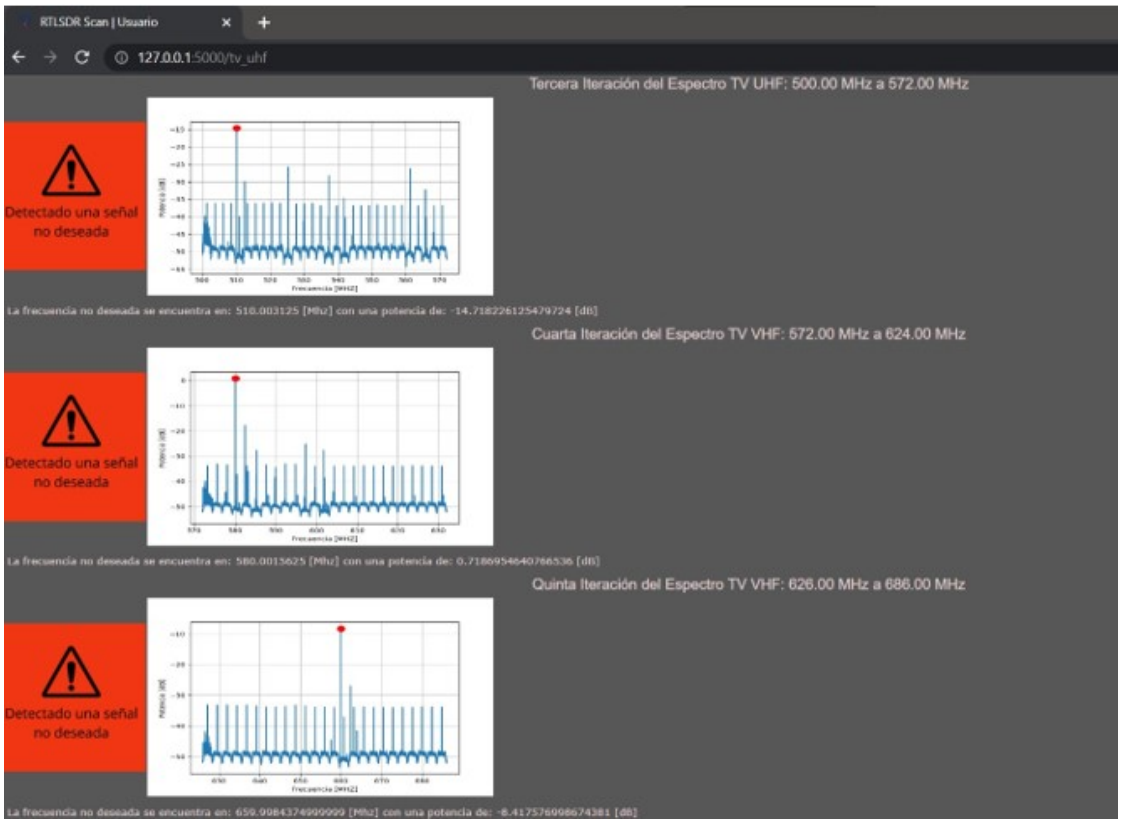

Figura A.11: Ventana de Escaneo TV UHF al encontrar transmisiones no deseadas

La sección de Mis Alarmas enseñará todas las alarmas que el usuario ha identificado. Esta información es almacenada en la base de datos de Google Cloud Platform y es mostrada en la interfaz del usuario.

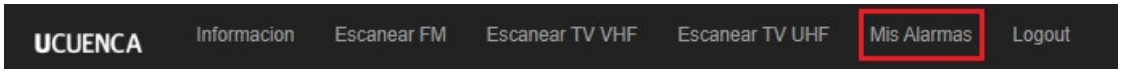

Figura A.12: Opción de Mis Alarmas

Como se puede ver en la figura A.13, la página de Mis Alarmas muestra una tabla en donde está almacenada la información de todas las alertas de las transmisiones no deseadas que ha detectado el algoritmo. La información es presentada en orden de Frecuencia, Potencia, Fecha de la detección y si la señal identificada pertenece al espectro de radio FM, TV VHF o TV UHF.

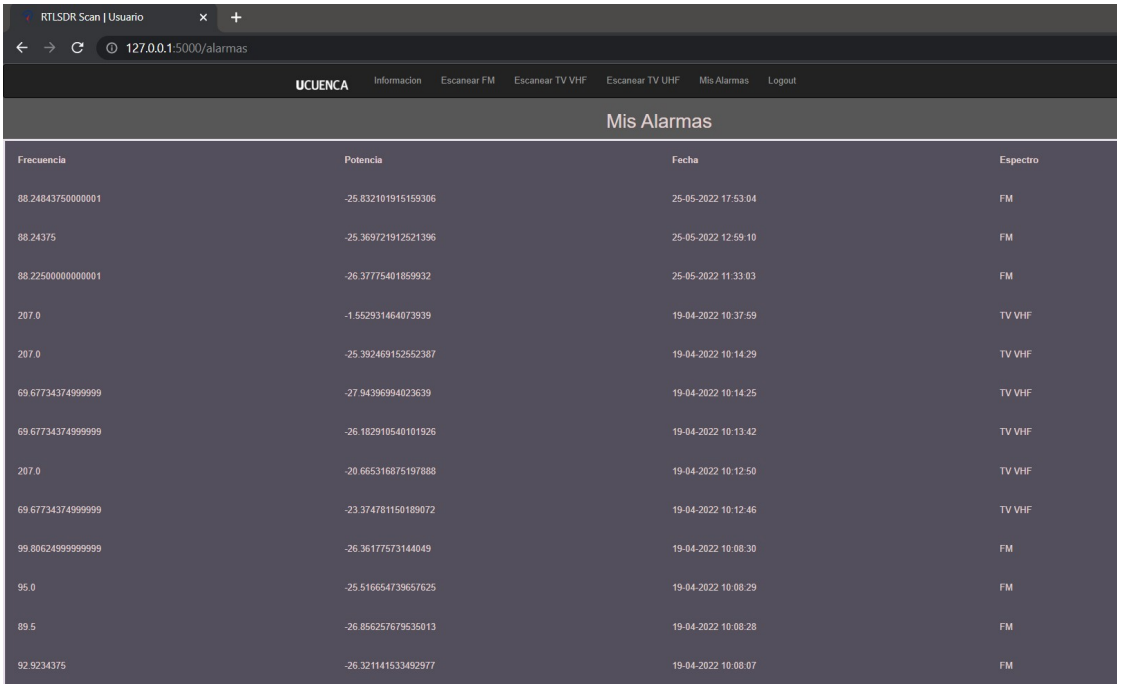

Figura A.13: Ventana de Mis Alarmas

La última opción que presenta la interfaz del usuario es la de cierre de sesión, con esta opción el usuario cierra la sesión y lleva a la página de inicio de sesión.

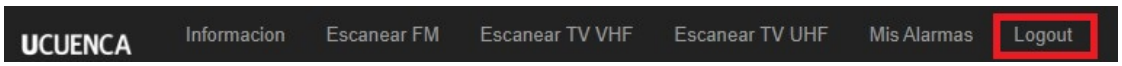

Figura A.14: Opción de Logout

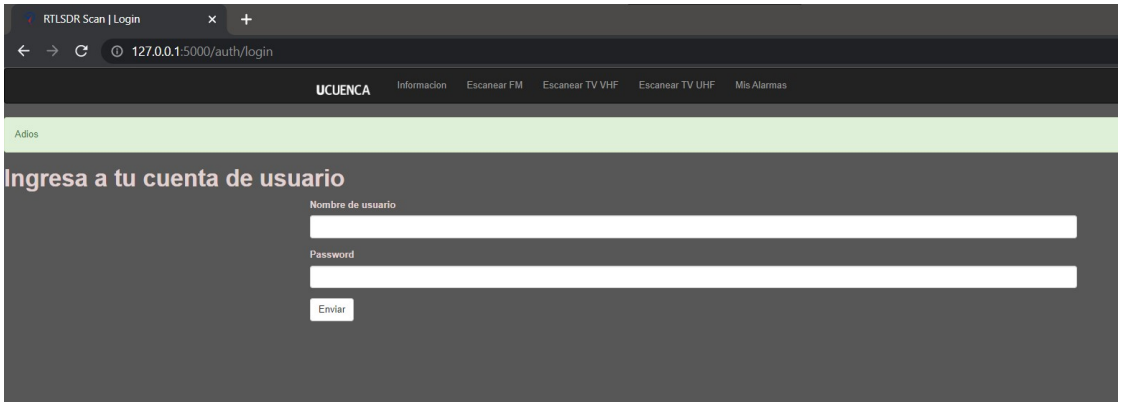

Figura A.15: Reingreso a la ventana de ingreso a la cuenta de usuario

ANEXO B

Instalación del RTL-SDR Blog V3 y Raspbian

B.1. Instalación del RTL-SDR Blog V3 en el computador

El software más empleado y simple de configurar para los dispositivos RTL-SDR en Windows es SDR Sharp. Este es un analizador de espectro gratuito, en donde se pueden configurar los parámetros como la frecuencia central, ganancia del , así como instalar plugins que permiten análisis más profundos.

Para la instalación del SDR Sharp es necesario descargar la última versión estable de la página oficial www.airspy.com. Una vez que se han descargado los archivos, extraemos el contenido de la carpeta, y buscamos entre los archivos el ejecutable Zadig, que permite la instalación de los drivers del RTL-SDR en nuestro computador. Para esto ejecutamos el Zadig con permisos de administrador y en opciones marcamos la opción List All Devices como se muestra en la figura B.1.

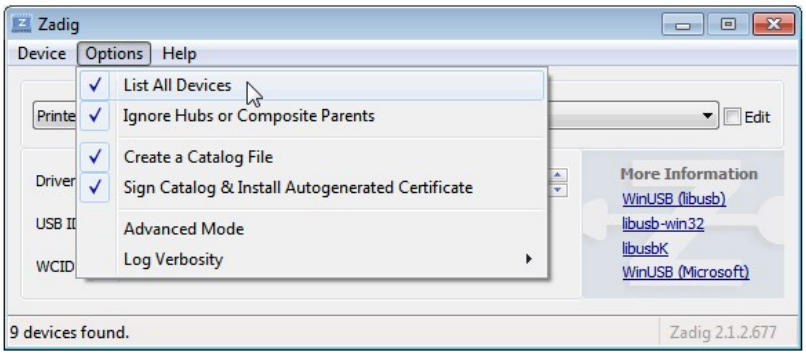

Figura B.1: Opciones de dispositivos en Zadig

Se selecciona la opción "Bulk-In, Interface (Interface 0)", que es nuestro RTL-SDR, se tiene que tener atención en este punto que la opción sea Interface 0 y no 1, además que en algunos computadores se mostrará como RTL2832UHIDIR o RTL2832U en vez de Bulk-In Interface. También la opción de USB ID debe ser "0BDA 2838 00".

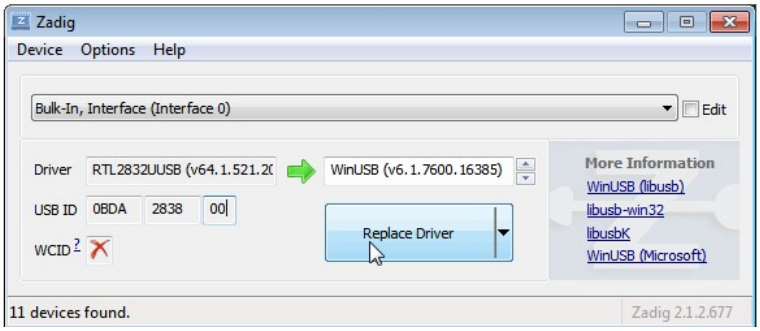

Figura B.2: Configuración de drivers para RTL-SDR en Zadig

Una vez seleccionado el dispositivo RTL-SDR en Zadig, se da clic en *Replace Driver* y se espera a la instalación del driver. Para corroborar que la instalación de los drivers del RTL-SDR ha sido correcta se puede emplear cualquier software de visualización del espectro radioeléctrico como puede ser SDR Sharp, siendo este software el más empleado por radio aficionados, tiene múltiples carencias pues solamente permite la visualización del espectro radioeléctrico escaneado. Para el propósito de este trabajo de titulación se necesita herramientas de desarrollo de algoritmos con el RTL-SDR por lo que se empleará Python y sus librerías específicas para el RTL-SDR para el desarrollo del algoritmo principal de este proyecto.

B.2. Instalación Raspbian en Raspberry Pi 3

Raspberry Pi al correr bajo Linux tiene múltiples distribuciones orientadas a diferentes industrias, el sistema operativo por defecto se llama Raspbian y se lo encuentra en la página oficial de Raspberry Pi https://www.raspberrypi.com/software/.

Una vez descargado el archivo se procede a preparar la tarjeta Micro-SD, formateándola con el formato FAT32, usando con la herramienta de Windows o el programa SD Card Formatter. Formateada la tarjeta Micro-SD es necesario cargar la imagen del sistema operativo Raspbian a la Micro-SD, para este proceso se emplea el programa Raspberry Pi Imager, donde seleccionaremos el archivo del Raspbian descargado, seleccionaremos la tarjeta Micro-SD formateada y daremos clic en Write.

Una vez terminado este proceso se procede a desconectar la memoria Micro-SD y ponerla en la Raspberry Pi 3, también se colocan los cables de alimentación, HDMI e Internet. El sistema operativo iniciará y podremos configurar por primera vez el usuario y contraseña dentro de esta micro computadora. [20]

ANEXO C

Tablas de frecuencias

C.1. Tablas de frecuencias libres y ocupadas

El algoritmo desarrollado procesa la información de todo el espectro radioeléctrico en una serie de ejecuciones de la función procesamiento, a cada ejecución se define con una iteración. Así cada iteración con un grupo de 20 canales de FM dividirá la carga computacional que genera el algoritmo en la Raspberry Pi. Para el monitoreo de todo el espectro FM serán necesario 5 “iteraciones” o ejecuciones del algoritmo y la misma cantidad para el espectro de TV. Para un óptimo procesamiento es necesario especificar que canales están libres y cuales están ocupados. Una vez realizado esto es posible definir que el algoritmo analice aquellos canales en donde no existen transmisiones legales de estaciones de radio FM.

C.1.1. Canales libres y ocupados de FM

A continuación, se detallará en varias tablas como están definidas cada iteración, su frecuencia máxima y mínima de cada iteración y que grupo de canales tiene asignado del servicio de radio FM.

Primera iteración:

Frecuencia Mínima: 88 MHz

Frecuencia Máxima: 92 MHz

Tabla C.1: Canales correspondientes a la primera iteración del análisis del espectro FM

Canal FM	Frecuencia máxima [MHz]	Frecuencia mínima [MHz]	Disponibilidad
canal 1	88	88.19	usado
canal 2	88.2	88.39	libre
canal 3	88.4	88.59	usado
canal 4	88.625	88.79	libre
canal 5	88.8	88.99	libre
canal 6	89	89.19	libre
canal 7	89.25	89.39	libre
canal 8	89.4	89.59	libre
canal 9	89.6	89.79	usado
canal 10	89.85	89.99	libre
canal 11	90	90.19	usado
canal 12	90.2	90.39	libre
canal 13	90.4	90.59	usado
canal 14	90.6	90.79	libre
canal 15	90.8	90.99	usado
canal 16	91	91.19	libre
canal 17	91.2	91.39	usado
canal 18	91.4	91.59	libre
canal 19	91.6	91.79	usado
canal 20	91.8	91.95	libre

Segunda iteración:

Frecuencia Mínima: 92 MHz

Frecuencia Máxima: 96 MHz

Tabla C.2: Canales correspondientes a la segunda iteración del análisis del espectro FM

Canal FM	Frecuencia máxima [MHz]	Frecuencia mínima [MHz]	Disponibilidad
canal 21	92	92.19	usado
canal 22	92.25	92.39	usado
canal 23	92.4	92.59	usado
canal 24	92.69	92.79	libre
canal 25	92.8	92.99	libre
canal 26	93	93.15	libre
canal 27	93.2	93.39	usado
canal 28	93.45	93.59	libre
canal 29	93.6	93.79	usado
canal 30	93.8	93.95	libre
canal 31	94	94.19	usado
canal 32	94.2	94.39	libre
canal 33	94.4	94.59	usado
canal 34	94.6	94.79	libre
canal 35	94.8	94.99	usado
canal 36	95	95.19	libre
canal 37	95.2	95.39	usado
canal 38	95.4	95.59	libre
canal 39	95.6	95.79	usado
canal 40	95.8	95.99	libre

Tercera iteración:

Frecuencia Mínima: 96 MHz

Frecuencia Máxima: 100 MHz

Tabla C.3: Canales correspondientes a la tercera iteración del análisis del espectro FM

Canal FM	Frecuencia máxima [MHz]	Frecuencia mínima [MHz]	Disponibilidad
canal 41	96	96.19	usado
canal 42	96.27	96.39	libre
canal 43	96.4	96.59	usado
canal 44	96.6	96.79	libre
canal 45	96.8	96.99	usado
canal 46	97	97.19	libre
canal 47	97.2	97.39	usado
canal 48	97.4	97.59	libre
canal 49	97.6	97.79	usado
canal 50	97.8	97.94	libre
canal 51	98	98.19	usado
canal 52	98.225	98.39	libre
canal 53	98.4	98.59	usado
canal 54	98.6	98.79	libre
canal 55	98.8	98.99	usado
canal 56	99.025	99.19	libre
canal 57	99.2	99.39	usado
canal 58	99.425	99.59	libre
canal 59	99.6	99.79	usado
canal 60	99.8	99.99	libre

Cuarta iteración:

Frecuencia Mínima: 100 MHz

Frecuencia Máxima: 104 MHz

Tabla C.4: Canales correspondientes a la cuarta iteración del análisis del espectro FM

Canal FM	Frecuencia máxima [MHz]	Frecuencia mínima [MHz]	Disponibilidad
canal 61	100	100.19	usado
canal 62	100.2	100.39	libre
canal 63	100.4	100.59	usado
canal 64	100.65	100.79	libre
canal 65	100.8	100.99	libre
canal 66	101	101.19	libre
canal 67	101.2	101.39	usado
canal 68	101.4	101.59	libre
canal 69	101.6	101.79	usado
canal 70	101.8	101.99	libre
canal 71	102	102.19	usado
canal 72	102.2	102.39	libre
canal 73	102.4	102.59	usado
canal 74	102.6	102.79	libre
canal 75	102.8	102.99	usado
canal 76	103	103.19	libre
canal 77	103.2	103.39	usado
canal 78	103.41	103.59	libre
canal 79	103.6	103.79	usado
canal 80	103.8	103.99	libre

Quinta iteración:
Frecuencia Mínima: 104 MHz
Frecuencia Máxima: 108 MHz

Tabla C.5: Canales correspondientes a la quinta iteración del análisis del espectro FM

Canal FM	Frecuencia máxima [MHz]	Frecuencia mínima [MHz]	Disponibilidad
canal 81	104	104.19	usado
canal 82	104.2	104.39	libre
canal 83	104.4	104.59	usado
canal 84	104.65	104.79	libre
canal 85	104.8	104.99	usado
canal 86	105	105.19	libre
canal 87	105.2	105.39	usado
canal 88	105.4	105.59	libre
canal 89	105.6	105.79	usado
canal 90	105.8	105.99	libre
canal 91	106	106.19	usado
canal 92	106.2	106.39	libre
canal 93	106.4	106.59	usado
canal 94	106.6	106.79	libre
canal 95	106.8	106.99	usado
canal 96	107	107.19	libre
canal 97	107.2	107.39	usado
canal 98	107.4	107.59	libre
canal 99	107.6	107.79	usado
canal 100	107.8	107.99	libre

C.1.2. Canales libres y ocupados de TV

De la misma manera que el espectro radioeléctrico de FM, el espectro de TV se ha dividido en 5 iteraciones de 10 canales cada una. Este proceso es necesario ya que al igual que en FM esta división permite disminuir la complejidad computacional además que en el espectro radioeléctrico de TV, existen saltos de frecuencia considerables entre los espectros radioeléctricos de y UHF. Debido a lo anterior, emplear recursos computacionales en rangos de frecuencias que no son utilizados en el servicio de TV, provocaría que el dispositivo RTL-SDR tarde tiempos considerables para la lectura del espectro. Cabe recalcar que en el servicio de TV se usan dos portadoras, una para el video y otra para el audio, con una distancia de frecuencia considerables entre sí, dejando un espacio libre en el canal. De esta manera las frecuencias a analizar en el algoritmo son asignadas de acuerdo a si no están en el rango de las portadoras de audio y video de un canal de TV. Las siguientes tablas muestran con detalle los rangos de frecuencia que el algoritmo analiza y la disponibilidad de estos canales.

Primera iteración:

Frecuencia Mínima: 54 MHz

Frecuencia Máxima: 88 MHz

Tabla C.6: Canales correspondientes a la primera iteración del análisis del espectro TV

Canal FM	Frecuencia máxima [MHz]	Frecuencia mínima [MHz]	Disponibilidad
canal 2 - Libre 1	54	54.249	libre
canal 2 - Video	54.25	56.25	usado
canal 2 - Libre 2	56.251	58.49	libre
canal 2 - Armonicos	58.5	59.25	usado
canal 2 - Libre 3	59.251	59.68	libre
canal 2 - Audio	59.69	59.81	usado
canal 2 - Libre 4	59.82	60	libre
canal 3 - Libre 1	60.001	61.149	libre
canal 3 - Armonicos	61.15	61.5	usado
canal 3 - Libre 2	61.51	66	libre
canal 4 - Libre 1	66.001	66.749	libre
canal 4 - Video	66.75	68.75	usado
canal 4 - Libre 2	68.751	70.749	libre
canal 4 - Armonicos	70.75	71.01	usado
canal 4 - Libre 3	71.02	71.7	libre
canal 4 - Audio	71.71	71.77	usado
canal 4 - Libre 4	71.78	72	libre
canal 5 - Libre 1	76	76.249	libre
canal 5 - Video	76.25	78.75	usado
canal 5 - Libre 2	78.751	80.249	libre
canal 5 - Armonicos	80.25	81.25	usado
canal 5 - Libre 3	81.251	81.68	libre
canal 5 - Audio	81.69	81.81	usado
canal 5 - Libre 4	81.82	82	libre
canal 6 - Libre 1	82.01	85.39	libre
canal 6 - Armonicos	85.4	85.6	usado
canal 6 - Libre 2	85.7	88	libre

Segunda iteración:

Frecuencia Mínima: 174 MHz

Frecuencia Máxima: 216 MHz

Tabla C.7: Canales correspondientes a la segunda iteración del análisis del espectro TV

Canal FM	Frecuencia máxima [MHz]	Frecuencia mínima [MHz]	Disponibilidad
canal 7 - Libre 1	174	180	usado
canal 8 - Libre 1	180.01	180.49	libre
canal 8 - Video	180.5	182.25	usado
canal 8 - Libre 2	182.26	184.49	libre
canal 8 - Armonicos	184.5	185.25	usado
canal 8 - Libre 3	185.26	185.68	libre
canal 8 - Audio	185.69	185.81	usado
canal 8 - Libre 4	185.82	186	libre
canal 9 - Libre 1	186.01	186.99	libre
canal 9 - Video	187	187.5	usado
canal 9 - Libre 2	187.6	190.5	libre
canal 9 - Armonicos	190.6	191	usado
canal 9 - Libre 3	191.01	191.68	libre
canal 9 - Audio	191.69	191.81	usado
canal 9 - Libre 4	191.82	192	libre
canal 10 - Libre 1	192.01	198	libre
canal 11 - Libre 1	198.01	198.99	libre
canal 11 - Video	199	199.75	usado
canal 11 - Libre 2	199.76	202.74	libre
canal 11 - Armonicos	202.75	203	usado
canal 11 - Libre 3	203.01	203.68	libre
canal 11 - Audio	203.69	203.81	usado
canal 11 - Libre 4	203.82	204	libre
canal 12 - Libre 1	204.01	210	libre
canal 13 - Libre 1	210.01	210.749	usado
canal 13 - Video	210.75	212	usado
canal 13 - Libre 2	212.01	214.6	usado
canal 13 - Armonicos	214.7	215.1	usado
canal 13 - Libre 3	215.2	215.67	usado
canal 13 - Audio	215.68	215.81	usado
canal 13 - Libre 4	215.82	216	usado

Tercera iteración:

Frecuencia Mínima: 500 MHz

Frecuencia Máxima: 572 MHz

Tabla C.8: Canales correspondientes a la tercera iteración del análisis del espectro TV

Canal FM	Frecuencia máxima [MHz]	Frecuencia mínima [MHz]	Disponibilidad
canal 19 - Libre 1	500	500.04	libre
canal 19 - Armonicos	500.05	500.25	usado
canal 19 - Libre 2	500.26	506	libre
canal 21 - Armonicos	512.5	513.75	usado
canal 21 - Libre 2	513.76	518	libre
canal 22 - Libre 1	518.01	518.349	libre
canal 22 - Armonicos	518.35	518.45	usado
canal 23 - Video	524.5	526.5	usado
canal 23 - Armonicos	528.5	529.125	usado
canal 23 - Libre 3	529.13	529.6	libre
canal 23 - Audio	529.625	529.85	usado
canal 23 - Libre 4	529.86	530	libre
canal 24 - Libre 1	530.01	532.774	libre
canal 25 - Libre 1	536.01	536.249	libre
canal 25 - Video	536.25	538.25	usado
canal 25 - Libre 2	538.26	540.49	libre
canal 25 - Armonicos	540.5	541.25	usado
canal 25 - Libre 3	541.26	541.64	libre
canal 25 - Audio	541.65	541.825	usado
canal 25 - Libre 4	541.83	542	libre
canal 26 - Libre 1	542.01	547.174	libre
canal 27 - Libre 1	548.01	548.249	libre
canal 27 - Video	548.25	550.5	usado
canal 28 - Libre 1	554.01	560	libre
canal 29 - Libre 1	560.01	560.249	libre
canal 29 - Video	560.25	562	usado
canal 29 - Libre 2	562.01	564.49	libre
canal 29 - Armonicos	564.5	565.05	usado
canal 29 - Libre 3	565.06	565.67	libre
canal 29 - Audio	565.68	565.81	usado
canal 29 - Libre 4	565.82	566	libre
canal 30 - Libre 1	566.01	572	libre

Cuarta iteración:

Frecuencia Mínima: 572 MHz

Frecuencia Máxima: 624 MHz

Tabla C.9: Canales correspondientes a la cuarta iteración del análisis del espectro TV

Canal FM	Frecuencia máxima [MHz]	Frecuencia mínima [MHz]	Disponibilidad
canal 31 - Libre 1	572.01	572.49	libre
canal 31 - Video	572.5	574	usado
canal 31 - Libre 2	574.01	575.92	libre
canal 31 - Armonicos	575.925	577.25	usado
canal 31 - Libre 3	575.3	577.69	libre
canal 31 - Audio	577.7	577.81	usado
canal 31 - Libre 4	577.82	578	libre
canal 32 - Libre 1	578.01	584	libre
canal 33 - Libre 1	584.01	584.49	libre
canal 33 - Video	584.5	586	usado
canal 33 - Libre 2	586.01	588.649	libre
canal 33 - Armonicos	588.65	589.1	usado
canal 33 - Libre 3	589.2	589.68	libre
canal 33 - Audio	589.69	589.8875	usado
canal 33 - Libre 4	589.9	590	libre
canal 34 - Libre 1	590.01	590.34	libre
canal 34 - Armonicos	590.35	590.45	usado
canal 34 - Libre 2	590.46	596	libre
canal 35 - Libre 1	596.01	596.24	libre
canal 35 - Video	596.25	598.55	usado
canal 35 - Libre 2	598.56	599.95	libre
canal 35 - Armonicos	599.96	601.1	usado
canal 35 - Libre 3	601.2	601.68	libre
canal 35 - Audio	601.69	601.825	usado
canal 35 - Libre 4	601.9	602	libre
canal 36 - Libre 1	602.01	608	libre
canal 37 - Libre 1	608.01	614	libre
canal 38 - Libre 1	614.01	620	libre
canal 39 - Libre 1	620.01	623.69	libre
canal 39 - Armonicos	623.7	623.9	usado
canal 39 - Libre 2	624	624	libre

Quinta iteración:

Frecuencia Mínima: 626 MHz

Frecuencia Máxima: 686 MHz

Tabla C.10: Canales correspondientes a la quinta iteración del análisis del espectro TV

Canal FM	Frecuencia máxima [MHz]	Frecuencia mínima [MHz]	Disponibilidad
canal 40 - Libre 1	626.01	632	libre
canal 41 - Libre 1	632.01	638	libre
canal 42 - Libre 1	638.01	644	libre
canal 43 - Libre 1	644.01	650	libre
canal 44 - Libre 1	650.01	656	libre
canal 45 - Libre 1	656.01	662	libre
canal 46 - Libre 1	662.01	668	libre
canal 47 - Libre 1	668.01	674	libre
canal 48 - Libre 1	674.01	680	libre
canal 49 - Libre 1	680.01	686	libre

ANEXO D

D.1. Tablas de registros de datos de pruebas de laboratorio

En este apéndice se exponen las tablas con la información registrada de las pruebas de laboratorio realizadas al sistema de detección, se comienza registrando las tablas de las relaciones entre potencia de recepción y nivel de transmisión, para luego analizar los datos de los registros obtenidos en las distintas configuraciones de la ganancia del SDR.

Tabla D.1: Datos de registrados de la potencia de recepción en relación con los niveles de transmisión del generador de señales

Potencia de Recepción [dBm]	Nivel de Transmisión [dBm]
-44.02	-5
-43.78	-4
-43.12	-3
-41.89	-2
-41.52	-1
-40	0
-39.89	1
-39.02	2
-38.14	3
-37.52	4
-37	5
-36.85	6
-36.46	7
-36.25	8
-36	9
-34.04	10
-34.02	11
-33.2	12
-32.03	13
-31.16	14
-30.06	15
-29	16
-28.2	17
-26.12	18
-24.78	19

Las tablas visualizadas a continuación representan la toma de datos en las pruebas de distintas configuraciones de la ganancia del SDR.

Tabla D.2: Resultados con una ganancia del SDR de 0 dB

Potencia de Recepción [dBm]	Ganancia SDR [dB]	Detecto (VP)	No Detecto (FN)
-45	0	0	1
-44.23	0	0	1
-43.1	0	0	1
-42.02	0	0	1
-41.12	0	0	1
-40.2	0	0	1
-39.1	0	0	1
-38.2	0	0	1
-37.4	0	0	1
-36.6	0	0	1
-35	0	0	1
-33.2	0	0	1
-32.5	0	0	1
-31.2	0	0	1
-30.12	0	0	1
-29	0	1	0
-27.5	0	1	0
-26.3	0	1	0
-24.12	0	1	0

La sensibilidad del algoritmo desarrollado con una ganancia de 0 dB es de 0.21. Si aumentamos un dB a la ganancia del SDR tenemos los siguientes resultados:

Tabla D.3: Resultados con una ganancia del SDR de 1 dB

Potencia de Recepción [dBm]	Ganancia SDR [dB]	Detecto (VP)	No Detecto (FN)
-44.6	1	0	1
-43.2	1	0	1
-42.5	1	0	1
-41.52	1	0	1
-40.15	1	0	1
-39.12	1	0	1
-38.26	1	0	1
-37.6	1	0	1
-36.5	1	0	1
-35.7	1	0	1
-34.6	1	0	1
-33.13	1	0	1
-31.58	1	0	1
-30.23	1	0	1
-29	1	1	0
-28.12	1	1	0
-27.15	1	1	0
-25.2	1	1	0
-23	1	1	0

Como se puede apreciar en la tabla D.3 los valores de nivel detectable por el algoritmo llegan hasta 14. Esto representaría la transmisión con una potencia mayor al umbral definido. Como se puede apreciar en la tabla D.3, al incrementar los valores de la ganancia del SDR también se incrementan los niveles de potencia del transmisor que son detectables por el algoritmo desarrollado.

Tabla D.4: Resultados con una ganancia del SDR de 2 dB

Potencia de Recepción [dBm]		Ganancia SDR [dB]	Detecto bien (VP)
Detecto mal (FN)			
-41.43	2	0	1
-40.17	2	0	1
-39.20	2	0	1
-38.19	2	0	1
-37.4	2	0	1
-36.02	2	0	1
-35.16	2	0	1
-33.12	2	0	1
-31.2	2	0	1
-29	2	1	0
-27.15	2	1	0
-26.2	2	1	0
-25.32	2	1	0
-24	2	1	0
-23.48	2	1	0
-22.41	2	1	0
-20.8	2	1	0
-19.7	2	1	0
-18.3	2	0	1

La sensibilidad del algoritmo desarrollado con una ganancia de 2 dB es de 0.52, para una ganancia de 3 dB se tiene los siguientes resultados:

Tabla D.5: Resultados con una ganancia del SDR de 3 dB

Potencia de Recepción [dBm]	Ganancia SDR [dB]	Detecto bien (VP)	Detecto mal (FN)
-40.49	3	0	1
-39.74	3	0	1
-38	3	0	1
-37.05	3	0	1
-36.7	3	0	1
-35.89	3	0	1
-34.52	3	0	1
-33.15	3	0	1
-31.20	3	0	1
-29	3	1	0
-28.41	3	1	0
-27.26	3	1	0
-26.15	3	1	0
-24.58	3	1	0
-23.96	3	1	0
-22.58	3	1	0
-21.74	3	1	0
-20.62	3	1	0
-19.78	3	1	0

La sensibilidad del algoritmo desarrollado con una ganancia de 3 dB es de 0.57, para una ganancia de 4 dB se tiene los siguientes resultados:

Tabla D.6: Resultados con una ganancia del SDR de 4 dB

Potencia de Recepción [dBm]	Ganancia SDR [dB]	Detecto bien (VP)	Detecto mal (FN)
-40.08	4	0	1
-39.46	4	0	1
-38.16	4	0	1
-37.52	4	0	1
-36.79	4	0	1
-34.48	4	0	1
-33.06	4	0	1
-32.73	4	0	1
-30.97	4	0	1
-29	4	1	0
-28.18	4	1	0
-27.96	4	1	0
-26.78	4	1	0
-24.78	4	1	0
-23.13	4	1	0
-21.57	4	1	0
-20.49	4	1	0
-19.52	4	1	0
-18.74	4	1	0

La sensibilidad del algoritmo desarrollado con una ganancia de 5 dB es de 0.57, para una ganancia de 5 dB se tiene los siguientes resultados:

Tabla D.7: Resultados con una ganancia del SDR de 5 dB

Potencia de Recepción [dBm]	Ganancia SDR [dB]	Detecto bien (VP)	Detecto mal (FN)
-40.17	5	0	1
-39.06	5	0	1
-38.64	5	0	1
-38.12	5	0	1
-37.52	5	0	1
-33.45	5	0	1
-32.07	5	1	0
-31.03	5	0	1
-30.03	5	0	1
-29	5	1	0
-28.74	5	1	0
-27.63	5	1	0
-26.88	5	1	0
-24.29	5	1	0
-22.48	5	1	0
-21.71	5	1	0
-20.35	5	1	0
-20.02	5	1	0
-18.02	5	1	0

Desarrollando una tabla que permita visualizar los valores de la sensibilidad a partir de la ganancia:

Tabla D.8: Datos de la sensibilidad para cada ganancia del SDR

Ganancia SDR [dB]	Sensiblidad
0	0.21
1	0.31
2	0.52
3	0.57
4	0.57
5	0.57

Para relacionar la potencia de recepción del RTL-SDR y el nivel de transmisión del PXI es necesario registrar estos datos, para poder realizar visualizaciones que permitan definir la ubicación del punto óptimo del umbral o mínima señal detectable. A continuación, se detalla el registro de toda esta información:

Tabla D.9: Datos de registrados de la potencia de recepción en relación con los niveles de transmisión del generador de señales PXI

Potencia de Recepción [dBm]	Nivel de Transmisión [dBm]
-27.14	5
-29	4
-30.75	3
-32.49	2
-33.75	1
-36.02	0
-36.79	-1
-38.89	-2
-42.12	-3
-42.65	-4
-43.65	-5

Finalmente, se registra la información de pruebas del sistema con parámetros en el transmisor de LF Generator Output Voltage al máximo, de esta manera podremos visualizar en un diagrama la eficacia del sistema.

Tabla D.10: Resultados con una ganancia del SDR de 2 dB y parámetros de LF Generator Output Voltage al máximo

Potencia de Recepción [dBm]		Ganancia SDR [dB]	Detecto bien (VP)
Detecto mal (FN)			
-40.32	2	0	1
-38.15	2	0	1
-37.19	2	0	1
-35.05	2	1	1
-34.17	2	1	1
-32.78	2	1	1
-31.62	2	1	1
-30.25	2	1	1
-29.51	2	1	1
-29	2	1	1
-28.12	2	1	0
-27.17	2	1	0
-26.69	2	1	0
-24.45	2	1	0
-23.89	2	1	0
-21.81	2	1	0
-20.15	2	1	0
-18.68	2	1	0
-17.07	2	1	0
-16.15	2	1	0

ANEXO E

Programacion implementada

E.1. Código de escaneo y análisis de transmisiones no deseadas

```
import matplotlib.pyplot as plt
import rtlsdr
import numpy as np
import time
import pandas as pd
from datetime import datetime
from sklearn.metrics import mean_squared_error
import pathlib
import time

def hacer_potencia(psd_max):
    potencia=10*np.log10(psd_max)
    return potencia

def setup(f_min, f_max,veces):
    #Frecuency range and step
    rate_best = 2.4e6
    df = rate_best
    # Set up the scan
    freqs = np.arange(f_min + df/2.,f_max,df)
    nfreq = freqs.shape[0]
    npsd_res = 512
    npsd_avg = 256
    nsamp = npsd_res*npsd_avg
    nfreq_spec = nfreq*npsd_res
```

```
    samples = np.zeros([nsamp,nfreq],dtype='complex128')
    #Setting the data lists
    psd_array = np.zeros([npsd_res,nfreq])
    freq_array = np.zeros([npsd_res,nfreq])
    #time_array = np.zeros([npsd_res,nfreq],dtype='datetime64[s]')
    relative_power_array = np.zeros([npsd_res,nfreq])
    #Configuracion de dataframes para el MAXHOLD
    len=freq_array.shape[0]
    psd_total=np.empty([len*2,veces])
    return rate_best, freqs, nfreq, npsd_res, npsd_avg, nsamp, nfreq_spec, samples,
    psd_array, freq_array, relative_power_array, psd_total
#db microvoltio / metro

def readsdr(rate_best, freqs, nfreq, npsd_res, npsd_avg, nsamp, nfreq_spec, samples, psd_array,
freq_array, relative_power_array, psd_total,veces):
    #Initializing SDR
    sdr = rtlsdr.RtlSdr()
    sdr.sample_rate = rate_best
    sdr.gain = 2
    samp_rate = sdr.sample_rate
    for k in range(veces):
        for i,freq in enumerate(freqs):
                sdr.center_freq = freq
                samples[:,i] = sdr.read_samples(nsamp)
    for i,freq in enumerate(freqs):
        fc_mhz = freq/1e6
        bw_mhz = sdr.sample_rate/1e6
        psd_array[:,i],freq_array[:,i] = plt.psd(samples[:,i],
        NFFT=npsd_res, Fs=bw_mhz, Fc=fc_mhz)
    #print(psd_array)
    freq_series=np.concatenate(freq_array)
    psd_series=np.concatenate(psd_array)
    #print(k)
    #psd=pd.DataFrame(psd_series)
    psd_total=np.insert(psd_total,k,psd_series,axis=1)
    sdr.close()
    psd_total=pd.DataFrame(psd_total)
    psd_new=psd_total.loc[:,0:veces-1]
    psd_max=psd_new.max(axis=1)
    max_hold=psd_max.apply(hacer_potencia)
    data_array = np.stack((freq_series, max_hold), axis=1)
    df=pd.DataFrame(data_array,columns=['Frecuencia','Potencia'])
    data= df.sort_values('Frecuencia',ascending=True)
```

```
    return data

def canal_filter(data,f_min_canal,f_max_canal):
    data_canal=data[(data['Frecuencia']>=f_min_canal) & (data['Frecuencia']<=f_max_canal)]
    data_canal=data_canal.reset_index(drop=True)
    return data_canal

def ploteo_senal_comparacion_y_referencia(data_canal,senal_referencia,senal_comparacion,key):

    data_canal_freqs=data_canal['Frecuencia']
    data_canal_freqs = data_canal_freqs.to_numpy()
    referencia_numpy = senal_referencia.to_numpy()
    comparacion_numpy = senal_comparacion.to_numpy()
    referencia_signal = np.stack((data_canal_freqs,  referencia_numpy), axis=1)
    comparacion_signal = np.stack((data_canal_freqs,  comparacion_numpy), axis=1)
    #print(referencia_signal)
    #print(comparacion_signal)
    plt.subplot(2,1,1)
    plt.plot(referencia_signal[:,0],referencia_signal[:,1])
    plt.title('Senal de referencia')
    #plt.xlabel('Frecuencia [MHz]')
    plt.ylabel('Potencia [dB]')
    plt.subplot(2,1,2)
    plt.plot(comparacion_signal[:,0],comparacion_signal[:,1])
    plt.title('Senal de comparacion')
    plt.xlabel('Frecuencia [MHz]')
    plt.ylabel('Potencia [dB]')
    plt.suptitle('Senales de referencia y comparacion del: {}'.format(key))
    plt.show()

def comparacion(data_canal,senal_referencia,senal_comparacion,key):
    senal_comparacion=senal_comparacion.squeeze()
    #ploteo_senal_comparacion_y_referencia(data_canal,senal_referencia,senal_comparacion,key)
    '''    senal_referencia_numpy = senal_referencia.to_numpy()
    senal_comparacion_numpy = senal_comparacion.to_numpy()
    correlacion_signal = np.stack((senal_referencia_numpy,  senal_comparacion_numpy), axis=1)
    correlacion_df = pd.DataFrame(correlacion_signal,columns=['Referencia','Comparacion'])
    print(correlacion_df)'''
    '''senal_referencia_numpy = senal_referencia.to_numpy()
    senal_comparacion_numpy = senal_comparacion.to_numpy()
    corre = np.corrcoef(senal_referencia_numpy,senal_comparacion_numpy)
    print('La correlacion con numpy es : {}'.format(corre))'''
    corr=senal_referencia.corr(senal_comparacion)
```

```
    #print(corr)
    corr_validation=np.isnan(corr)
    if corr_validation==True:
        corr=0.2
    rmse=mean_squared_error(senal_referencia,senal_comparacion,squared=True)
    if rmse > 10:
        rmse_list=[]
        for i in range(50):
            rmse=mean_squared_error(senal_referencia,senal_comparacion,squared=True)
            rmse_list.append(rmse)
        rmse=min(rmse_list)
    return corr,rmse

def detection_limit(n,umbral,constante):
    #umbral = -45
    if n <= umbral:
        return constante
    else:
        return n

def minima_senal_detectable_canal(data):
    senal_referencia = data['Potencia'].apply(detection_limit,args=(-29,-29))
    #plt.plot(senal_referencia)
    return senal_referencia

def crear_senal_comparacion(senal_referencia,umbral):
    senal_comparacion=np.empty(senal_referencia.shape[0])
    senal_comparacion[:] = umbral
    '''senal_comparacion = senal_comparacion.squeeze()
    print(' el shape de senal comparacion: ')
    print(senal_comparacion.shape)'''
    senal_comparacion=pd.DataFrame(senal_comparacion)
    #print(senal_comparacion)
    return senal_comparacion

def filtrado_canal(data,f_min_canal,f_max_canal):
    data_canal=canal_filter(data,f_min_canal,f_max_canal)
    #umbral=data_canal['Potencia'].max()
    #senal_referencia=data_canal['Potencia'].apply(detection_limit,args=(umbral,umbral))
    return data_canal

def comparacion_senales(data_canal,senal_referencia,senal_comparacion,key):
    if senal_comparacion.shape[0] != senal_referencia.shape[0]:
```

```
        senal_comparacion=senal_comparacion[0:senal_referencia.shape[0]]
    corr,rmse = comparacion(data_canal,senal_referencia,senal_comparacion,key)
    return corr,rmse

def procesamiento(f_min,f_max,canales,idx):
    veces=1
    rate_best, freqs, nfreq, npsd_res, npsd_avg, nsamp, nfreq_spec
    relative_power_array, psd_total= setup(f_min, f_max,veces)
    data=readsdr(rate_best, freqs, nfreq, npsd_res, npsd_avg, nsamp, nfreq_spec, samples,
    psd_array, freq_array, relative_power_array, psd_total,veces)
    #umbral,senal_referencia=minimun_signal_detectable(canales,data)
    for key in canales:
        values=canales[key]
        condicicon=values[2]
        if condicicon=='libre':
            f_min_canal=values[0]
            f_max_canal=values[1]
            data_canal=filtrado_canal(data,f_min_canal,f_max_canal)
            senal_referencia=minima_senal_detectable_canal(data_canal)
            senal_comparacion = crear_senal_comparacion(senal_referencia,-29)
            corr, rmse = comparacion_senales(data_canal,senal_referencia,senal_comparacion,key)
            #data_canal.to_csv(r'C:\Users\ggarc\Desktop\Tesis\matrizfm')
            CURRENT_DIR = pathlib.Path().resolve()
            IMAGE_DIR=CURRENT_DIR.joinpath("app","static","images","fm")
            IMAGE_DIR = str(IMAGE_DIR)
            date = time.strftime('%d-%m-%Y %H:%M:%S', time.localtime())
            espectro = 'FM'
            if corr < 0.25 and rmse > 0.01 :
                maxim=data_canal['Potencia'].max()
                idmax=data_canal['Potencia'].idxmax()
                #print(rmse)
                if maxim > -200 and maxim < 200:
                    parasita = data_canal.loc[idmax]
                    max_freq=parasita['Frecuencia']
                    max_pot=parasita['Potencia']
                    espuria={
                        'Fecha':date,
                        'Espectro':espectro,
                        'Potencia':max_pot,
                        'Frecuencia':max_freq,
                    }
                    descision = 1
                    '''plt.subplot(3,1,1)
```

```
                plt.plot(data_canal['Frecuencia'],data_canal['Potencia'])
                plt.title('Senal Electromagnetica de '+str(key))
                plt.subplot(3,1,2)
                plt.plot(senal_referencia)
                plt.title('Senal Referencia')
                plt.subplot(3,1,3)
                plt.plot(senal_comparacion)
                plt.title('Senal Comparacion')
                plt.show()'''
                print('La transmision no deseada se encuentra en
                {} con una potencia de: {}'.format(max_freq,max_pot))
                data_numpy=data.to_numpy()
                plt.switch_backend('agg')
                plt.clf()
                plt.plot(data_numpy[:,0],data_numpy[:,1])
                plt.plot(espuria['Frecuencia'],espuria['Potencia']
                ,marker='o',color='r',markersize=10)
                plt.grid()
                plt.xlabel('Frecuencia [MHZ]')
                plt.ylabel('Potencia [dB]')
                plt.savefig(IMAGE_DIR+'\espectro_iteracion_{}.png'.format(idx))
                #plt.show()
                return data,espuria,descision

        else:
            print('No hay interferencia en el ' + str(key))
            print('El rmse es: '+ str(rmse))
            print('La correlacion es ' + str(corr))
            print('*'*50)
            espuria={
                    'Espectro':espectro,
                    'Fecha': date,
                    'Potencia':0,
                    'Frecuencia':0,
                }
            descision = 0
data_numpy=data.to_numpy()
plt.switch_backend('agg')
plt.clf()
plt.plot(data_numpy[:,0],data_numpy[:,1])
plt.grid()
plt.xlabel('Frecuencia [MHZ]')
plt.ylabel('Potencia [dB]')
```

```
    plt.savefig(IMAGE_DIR+'\espectro_iteracion_{}.png'.format(idx))
    #plt.show()
    return data,espuria,descision

def procesamiento_diccionarios(datos):
    datos.set_index('Frecuencia',inplace=True)
    datos=datos.rename_axis('Frecuencia')
    datos_potencia = datos['Potencia'].tolist()
    datos=datos.to_dict(orient='split')
    del datos['columns']
    espectro = {
        'Frecuencia': datos['index'],
        'Potencia': datos_potencia,
    }
    return espectro
```

Bibliografía

[1] M. Saber, H. K. Aroussi, A. E. Rharras, y R. Saadane, "Raspberry pi and rtl-sdr for spectrum sensing based on fm real signals," pp. 1–6, 2018.

[2] S. Haykin, D. J. Thomson, y J. H. Reed, "Spectrum sensing for cognitive radio," pp. 849–877, 2009.

[3] S. Majumder, "Energy detection spectrum sensing on rtl-sdr based iot platform," *IEEE Conference on Information and Communication Technology (CICT)*, 2018.

[4] S. Zheng, S. Chen, L. Yang, J. Zhu, Z. Luo, J. Hu, y X. Yang, "Big data processing architecture for radio signals empowered by deep learning: Concept, experiment, applications and challenges," *IEEE Access*, 2018.

[5] ——, "Big data processing architecture for radio signals empowered by deep learning: Concept, experiment, applications and challenges," *IEEE Access*, vol. 6, pp. 55 907–55 922, 2018.

[6] F. Zhou, Y. Wu, Y.-C. Liang, Z. Li, Y. Wang, y K.-K. Wong, "State of the art, taxonomy, and open issues on cognitive radio networks with noma," *IEEE Wireless Communications*, vol. 25, num. 2, pp. 100–108, 2018.

[7] Y. B. I. Goodfellow y A. Courville, *MIT Press*, 2016.

[8] UIT, "Reglamento de radiocomunicaciones, art. 1, términos y definiciones, sección 1, 1.5, «ondas radioeléctricas u ondas hertzianas»," 2001.

[9] A. N. del Ecuador, "Ley orgánica de comunicación del ecuador," 2019.

[10] R. Z. R. Zones, "Que es el espectro radioeléctrico," 2021.

[11] A. Llanos, "Gestión del espectro radioeléctrico en ecuador nueva modalidad para radiodifusión y televisión abierta," 2013.

[12] A. Technologies, "Spectrum monitoring techniques, using a spectrum analyzer for unattended spectrum monitoring," 2015.

[13] K. Technologies, "Spectrum analysis basics, application note 150," 2016.

[14] V. K. Q. Rodríguez, "Mapa de control del espectro radioeléctrico," 2010.

[15] J. J. M. F. Ivan Pinar Dominguez, *Laboratorio de Comunicaciones Digitales Radio Definida por Software*, 2011.

[16] D. P. Travis F. Collins, Robin Getz y A. M. Wyglinsk, *Software-Defined Radio for Engineers*, 2018.

[17] M. Sadiku y C. Akujuobi, "Software-defined radio: a brief overview," *IEEE Potentials*, vol. 23, num. 4, pp. 14–15, 2004.

[18] "Rtl-sdr-blog-v3-datasheet." [En línea]. Disponible: https://www.rtl-sdr.com

[19] N. S. Yamanoor y S. Yamanoor, "High quality, low cost education with the raspberry pi," pp. 1–5, 2017.

[20] "Setting up your raspberry pi, raspberry pi projects." [En línea]. Disponible: https://www.raspberrypi.com/products/

[21] I. Stancin y A. Jovic, "An overview and comparison of free python libraries for data mining and big data analysis," pp. 977–982, 2019.

[22] "Numpy documentation." [En línea]. Disponible: https://numpy.org/doc/

[23] S. van der Walt, S. C. Colbert, y G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science Engineering*, vol. 13, num. 2, pp. 22–30, 2011.

[24] "Pyrtlsdr documentation," 2021. [En línea]. Disponible: https://pyrtlsdr.readthedocs.io/en/latest/Overview.html#description

[25] "Matplotlib documentation." [En línea]. Disponible: https://matplotlib.org/

[26] A. G. González, *The Correlation Coefficient: An Overview*, 2007.

[27] W. Wang y Y. Lu, "Analysis of the mean absolute error (MAE) and the root mean square error (RMSE) in assessing rounding model," *IOP Conference Series: Materials Science and Engineering*, vol. 324, p. 012049, 2018.

[28] "Flask documentation." [En línea]. Disponible: https://flask-es.readthedocs.io/

[29] "Firestore documentation."

[30] "Projects of raspberry pi," 2021. [En línea]. Disponible: https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started/3