manejará

densidad espectral de potencia o

Árbol

señal

mínima

detección de limite

comparación

vistas en el capítulo anterior

código especializado

decibelios mili vatios

Tablas de frecuencias libres y ocupadas

Tablas de frecuencias FM

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.

Canales libres y ocupados de TV

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.

Funciones

Metodología del desarrollo del algoritmo de detección de transmisiones no deseadas

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.

Funciones del backend de la interfaz de usuario

Funciones del frontend de la interfaz de usuario

en varias funciones

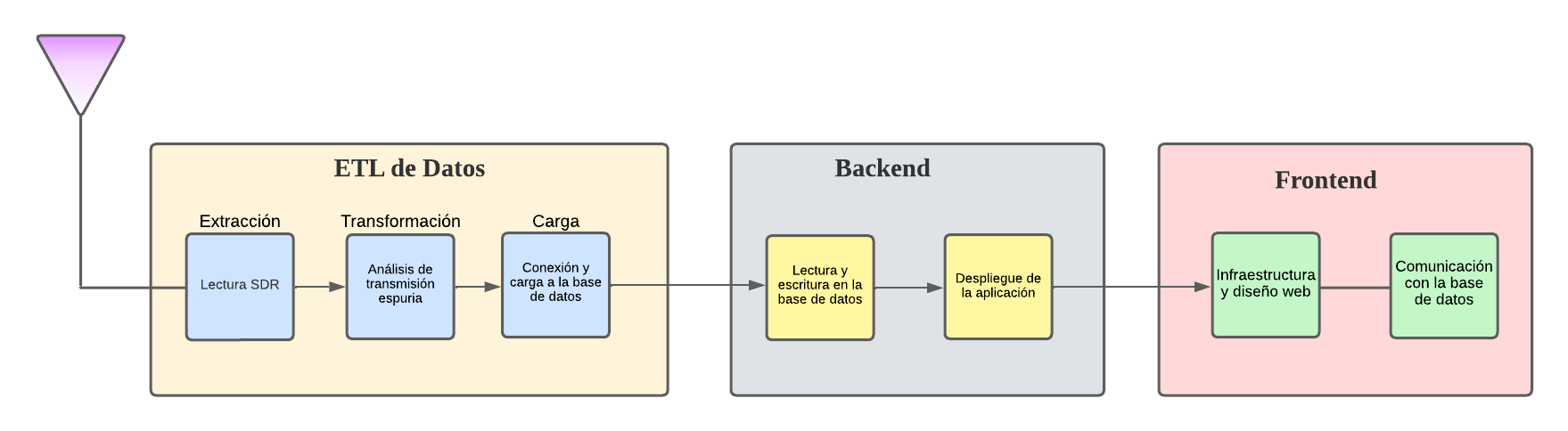
el código implementado

Función

Extiende

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.



La figura X muestra el diagrama de bloques que define la metodología empleada en el desarrollo, iniciando 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 \gls{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 \gls{HTML} y \gls{CSS}

Para el despliegue de toda la aplicación, fue necesario definir funciones que permitirán la manipulación de datos y su visualización, para el desarrollo de estas funciones es necesario que se va a emplear lenguajes de programación como Python, HTML y CSS. El desarrollo de las funciones se divide en dos partes, para una mejor comprensión utilizaremos el diagrama de bloques de la figura \ref{fig24}, 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.

\begin{figure}[H]

\centering

\includegraphics[width=0.9\textwidth]{capitulos/figuras/24.jpeg}

\caption{Diagrama de bloques de funcionamiento del algoritmo de detección e interfaz }

\label{fig24}

\end{figure}

Como se puede ver en la figura \ref{fig24} 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. Partimos desde 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.