Reconocimiento de voz automático de una señal FM

1° Cristopher Sánchez Yup
Ingeniería en Electrónica y Telecomunicaciones
Universidad Rafael Landívar
Ciudad de Guatemala, Guatemala
linkedin.com/in/csanchezyu

2° César Andrés Dávila Contreras
Ingeniería en Electrónica y Telecomunicaciones
Universidad Rafael Landívar
Ciudad de Guatemala, Guatemala
linkedin.com/in/cesardavilacon

***Resumen*—Este documento es un modelo e instrucciones de un sistema de captura de señales FM y reconocimiento de voz para la asistencia de personas con problemas auditivos, la implementación del sistema surgió con la idea de mostrar de forma visual a las personas con discapacidad auditiva los temas expuestos en las emisoras de radio.**

***Palabras Clave*—SDR, LimeSDR, TCP, GNU Radio.**

I. Introducción

La discapacidad auditiva es una limitación que afecta a una gran cantidad de personas en todo el mundo y puede impedir que estas disfruten de ciertos contenidos, como programas de radio. Con el objetivo de proporcionar una solución para esta problemática, se ha desarrollado un sistema de captura de señales FM y reconocimiento de voz que permitirá a las personas con problemas auditivos tener acceso a la información transmitida por las emisoras de radio. El presente documento tiene como objetivo proporcionar un modelo e instrucciones para la implementación de este sistema, el cual permitirá mostrar de forma visual los temas expuestos en las emisoras de radio a las personas con discapacidad auditiva. Este sistema ofrece una alternativa innovadora y efectiva para mejorar la calidad de vida de las personas con problemas auditivos y facilitar su acceso a la información.

II. Software Definido por Radio

La tecnología de software definido por radio (SDR) ha evolucionado significativamente en los últimos años y se ha convertido en una alternativa viable a la tecnología de radio tradicional. SDR es una tecnología que permite la configuración y la programación de dispositivos de radio para realizar diferentes tareas de procesamiento de señales. En este sentido, la tecnología SDR se ha utilizado en una amplia gama de aplicaciones, desde redes inalámbricas hasta comunicaciones militares.

En la literatura, hay una amplia variedad de trabajos que tratan sobre el tema del SDR. Los trabajos más relevantes se centran en los siguientes aspectos [1]:

- Arquitecturas SDR: varios trabajos han investigado diferentes arquitecturas SDR para mejorar el rendimiento y la flexibilidad de los sistemas de radio. Entre las arquitecturas estudiadas se incluyen la arquitectura de canalización, la arquitectura de procesamiento de alta velocidad y la arquitectura de canal de aire.
- Implementación de SDR: muchos trabajos han investigado diferentes técnicas de implementación de SDR para mejorar el rendimiento y la eficiencia de los sistemas de radio. Entre las técnicas estudiadas se incluyen el uso de FPGA, la programación de alto nivel y la implementación de hardware dedicado.
- Estándares y protocolos: varios trabajos se han centrado en el desarrollo de estándares y protocolos para la comunicación inalámbrica SDR. Estos trabajos incluyen el desarrollo de estándares como IEEE 802.22 y protocolos como el protocolo de acceso al medio cognitivo (MAC) y el protocolo de control de acceso al medio cognitivo (MAC) [2].
- Aplicaciones de SDR: varios trabajos han investigado diferentes aplicaciones de SDR en campos como la comunicación inalámbrica, la monitorización del espectro, la radiodifusión y la defensa. Estos trabajos han demostrado que la tecnología SDR tiene un gran potencial en una amplia gama de aplicaciones.

En conclusión, la revisión bibliográfica sobre SDR ha demostrado que la tecnología SDR es una alternativa viable a la tecnología de radio tradicional. Hay una amplia variedad de trabajos que tratan sobre diferentes aspectos de SDR, desde arquitecturas hasta implementación y aplicaciones. La literatura sugiere que la tecnología SDR tiene un gran potencial en una amplia gama de aplicaciones y se espera que su uso continúe creciendo en el futuro.

III. Metodología de Desarrollo

A. Requerimiento de Hardware

a) LimeSDR: es una plataforma de radio definida por software de bajo costo, de código abierto y habilitada para aplicaciones, con soporte para admitir casi cualquier tipo de estándar de comunicación inalámbrica. LimeSDR puede enviar y recibir UMTS, LTE, GSM, LoRa, Bluetooth, Zigbee, RFID y transmisión digital [3].

b) Requisitos Mínimos del Ordenador: se recomienda una computadora con mínimo 8 GB de RAM, 4 CPUS y 2 núcleos. Si el sistema operativo host es Windows, se recomienda que sea Windows10.

B. Requerimiento de Software

a) GNU Radio: es un conjunto de herramientas de desarrollo de software gratuito y de código abierto que proporciona

bloques de procesamiento de señales para implementar radios definida por software. En este caso, se utilizará con LimeSDR para crear radio definida por software, para este proyecto, se recomienda utilizar GNU Radio en Ubuntu, ya que es mucho más sencillo obtener los pluggins de LimeSDR para dicho sistema operativo [4].

b) Python 3.9: lenguaje de programación utilizado para el desarrollo de software, las dependencias más importantes para este prototipo son:

- TCP Sockets: Transmission Control Protocol (TCP) sockets, se utiliza para realizar la transmisión y recepción de datos entre GNU Radio y el script de Python. ¿Por qué se recomienda usar TCP sobre UDP? La transmisión de datos es confiable, el servidor vuelve a enviar los paquetes perdidos, el receptor le dice al servidor el espacio que tiene para recibir los paquetes. TCP guarda los paquetes hasta que haya espacio. [5].
- Modelo de Reconocimiento de Voz: el habla primero se convierte de sonido físico a energía eléctrica usando un micrófono y luego a datos digitales usando un convertidor analógico a digital. Estos datos digitales se pueden convertir en texto usando varios algoritmos. Uno de los requerimientos es tener Python 3.8+, PyAudio 0.2.11+, Google Api Client Library, Vosk y Whisper [6].
- Threading: consiste en poner los dos subprocesos a trabajar de forma simultánea, haciendo un mejor uso de los recursos de máquinas multinúcleos [7].
- Streamlit: convierte scripts de datos en aplicaciones web de forma muy sencilla, todo en Python, no se requiere experiencia de front-end [8].

IV. Modelo del Sistema

El desarrollo de este prototipo comenzó con la recepción de señales FM en GNU Radio utilizando la tarjeta LimeSDR. El algortimo del prototipo consiste en la recepción de señales de radiofrecuencia de banda comercial, dicha señal se somete a un demodulador FM el cual permite escuchar lo que la estación de radiofrecuencia está transmitiendo. Dicho sonido se desea convertirlo a texto en tiempo real. La data recibida por parte del demodulador FM se envía a través de la ip 127.0.0.1 puerto 40868 a través de un servidor TCP. Dicha información es recuperada dentro de un script de Python utilizando TCP sockets, la data es recibida, decodificada y concatenada en un arreglo. Cuando este arreglo tenga una longitud considerable de datos, se genera un archivo de sonido .wav. Esta parte del algoritmo es considerada un subproceso. Luego se utiliza un modelo de reconocimiento de voz entrenado por Google para convertir el archivo de sonido a texto. Dicho texto es mostrado en una interfaz gráfica web. Esta parte final del algoritmo es considerada otro subproceso.

En Python, debido al bloqueo de intérprete global, solo un subproceso puede ejecutar código de Python a la vez. ¿Qué significa esto? Al realizar pruebas, se pudo apreciar la pérdida de información por parte del socket TCP. El texto final se muestra en la interfaz gráfica web una vez el modelo de reconocimiento de voz finalice su tarea, lo que presenta una ventana de tiempo en dónde el socket deja de recibir información. Para solucionar este error, se implementó ”Paralelismo basado en subprocesos”, el cual consiste en poner los dos subprocesos a trabajar de forma simultánea, haciendo un mejor uso de los recursos de máquinas multinúcleos [7].

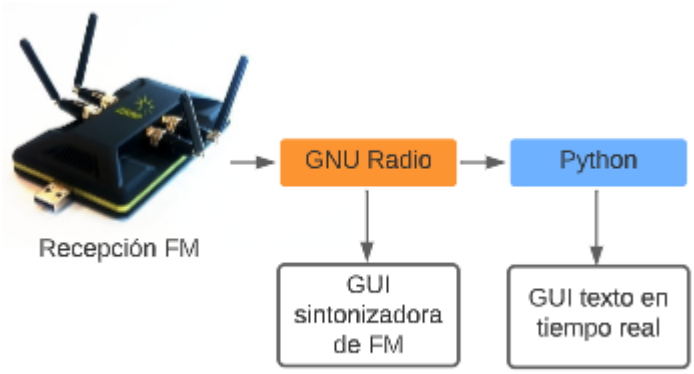

Fig. 1. Esquema del sistema de recepción FM e interfaces gráficas.

Fig. 2. Esquema de GNU Radio.

Fig. 3. Esquema de Python.

V. Resultados

Una vez que todas las unidades del sistema se prueban y validan individualmente, se armó el sistema físico.

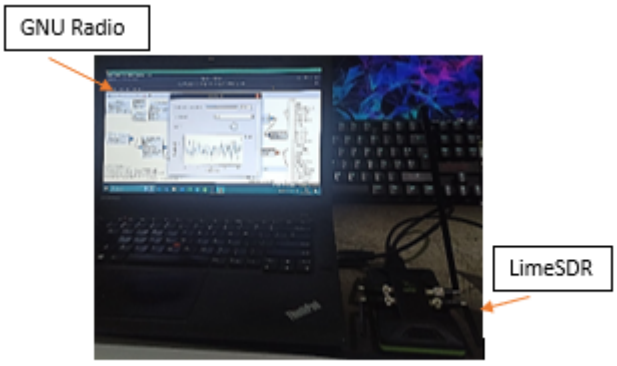

Fig. 4. Hardware del prototipo.

La última fase del desarrollo se dedica a caracterizar la implementación de reconocimiento de voz automático de una señal FM. El desarrollo de software se ejecuta en una máquina virtual de Ubuntu con 4 GB de RAM y 4 procesadores lógicos. La computadora portátil posee 8 GB de RAM y un procesador IntelCore 5 Duo i5 a 1,9 GHz. Se realizó la captura de una señal de FM por medio del uso de la tarjeta LimeSDR. Se realizó la demodulación y generación de la señal de audio de un programa de radio.

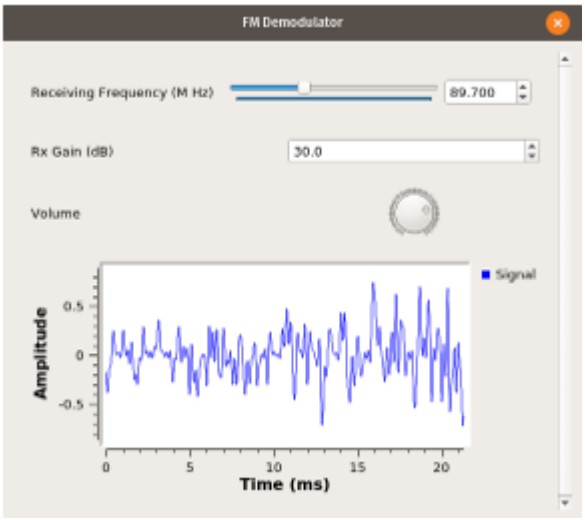

Fig. 5. Interfaz gráfica de GNU Radio.

Se realizó la implementación del sistema de reconocimiento de voz para detectar al/los hablantes del programa de radio y también la presentación de la conversación en tiempo real en una interfaz gráfica web.

Fig. 6. Interfaz gráfica web.

VI. Conclusiones y Recomendaciones

Existen varios factores de los cuales dependen los resultados de este prototipo. Uno de ellos, y el más importante, es el tipo de contenido que se encuentra transmitiendo la radiodifusora al momento de realizar la recepción de la señal. Este prototipo funciona siempre y cuando se reciba una conversación pura, es decir, que no cuente con música de fondo, ya que la música de fondo, altera la tasa de muestreo final de la señal demodulada. Otro factor importante es la ubicación del hardware del prototipo (Fig 4.), lo recomendable es que se encuentre al aire libre, con la antena apuntando hacia la ubicación de la radiodifusora. Debe considerarse también que el lugar en donde se encuentre el hardware del prototipo sea un ambiente con pocos equipos eléctricos, por ejemplo, si se tienen teléfonos celulares, equipos electrodomésticos y lámparas que posean balastros, estos crearán interferencia dañando así la recepción de la señal. Debido a que la tarjeta LimeSDR cuenta con un chip transceptor, conforme aumente el tiempo de uso aumenta la temperatura del mismo, lo que produce una saturación en el canal de recepción de la señal y por ende fallas en la recepción de las señales RF, por lo que se recomienda agregar un sistema de ventilación para la tarjeta LimeSDR.

Agradecimientos

Los autores agradecen al Ingeniero Manuel Ríos Rivas, director del departamento de Electrónica y Telecomunicaciones de la Universidad Rafael Landívar por sus valiosas contribuciones y orientación durante todo el proceso de investigación.

Referencias

[1] Wyglinski, A. M., Getz, R., Collins, T., & Pu, D. (2018d). Software-Defined Radio for Engineers. Artech House.
[2] IEEE 802.22 WRAN WG Website. (s.f.). Disponible: https://www.ieee802.org/22/
[3] Lime Microsystems. (2018, 14 agosto). LimeSDR. Disponible: https://limemicro.com/products/boards/limesdr/.
[4] Gr-limesdr Plugin for GNURadio - Myriad-RF Wiki. (n.d.). Disponible: https://wiki.myriadrf.org/Gr-limesdr_Plugin_for_GNURadio.
[5] Socket — Low-level networking interface. (s. f.). Python documentation. Disponible: https://docs.python.org/3/library/socket.html.
[6] SpeechRecognition. (2023, 13 marzo). PyPI. Disponible: https://pypi.org/project/SpeechRecognition/.
[7] Threading — Thread-based parallelism. (s.f.). Python Documentation. Disponible: https://docs.python.org/3/library/threading.html.
[8] Streamlit. (s. f.). The fastest way to build and share data apps. Disponible: https://streamlit.io/.