PROGRAMACIÓ N EN RADIO DEFINIDA POR SOFTWARE (GNURADIO)

Correa Gómez Didier Manuel ,2212254, didier2212254@correo.uis.edu.co
Castellanos Mariaño Oscar Daniel,2205024, oscar2205024@correo.uis.edu.co
Escuela de Ingeniería Electrónica
Universidad Industrial De Santander Bucaramanga, Colombia
https://github.com/Daniel24-web/CommII_labB1_G1_DANIEL_DIDIER

***Abstract*-- This report presents the development of a practice in the GNU Radio environment, a software-defined radio (SDR) platform, with the aim of strengthening skills in programming and real-time signal processing. Through the creation of custom blocks in Python, fundamental concepts such as algorithm implementation, signal manipulation, and the application of statistics for data analysis are explored. The report details the process of creating branches in GitHub, implementing functional blocks in GNU Radio, and integrating them into a practical application. Additionally, the results obtained are discussed, and potential applications for the learned concepts are suggested, highlighting the importance of teamwork and efficient task management in software project development.**

I. INTRODUCCIÓN

La radio definida por software (SDR) ha revolucionado el campo de las telecomunicaciones, permitiendo la implementación de sistemas de comunicación flexibles y adaptables mediante software. GNU Radio es una de las plataformas más utilizadas en este ámbito, ofreciendo un entorno de desarrollo que combina bloques predefinidos con la posibilidad de crear funcionalidades personalizadas. Este informe documenta el proceso de aprendizaje y aplicación de GNU Radio en el contexto de Comunicaciones II, donde se busca no solo utilizar los bloques existentes, sino también desarrollar nuevos algoritmos que permitan expandir las capacidades de la plataforma.

El presente informe documenta el proceso de implementación de algoritmos en GNU Radio, siguiendo una serie de pasos que incluyen la creación de ramas en un repositorio de GitHub, la programación de bloques en Python, y la integración de estos bloques en una aplicación específica para el procesamiento de señales reales. Además, se exploran conceptos fundamentales como el uso de bloques de acumulador y diferenciador, así como la implementación de estadísticas para el análisis de señales. Finalmente, se propone una aplicación práctica de los conceptos vistos en clase, con el fin de demostrar la utilidad de estas herramientas en la solución de problemas reales de comunicación.

II. METODOLOGIA EXPERIMENTAL

En cuanto al procedimiento realizado para la ejecución de la primera práctica de laboratorio se procedió como sigue:

A. *Creación de los directorios*

La primera parte del laboratorio consto en crear directorios donde se guardaron los archivos y los avances del informe, luego de esto se crearon las ramas de cada uno de los integrantes del grupo para subir sus avances individuales.

B. *Implementación de los bloques en GNURadio*

Para esta parte se siguieron los pasos para la creación de los bloques con python sugeridos del libro guía de comunicaciones, el bloque acumulador y el bloque diferenciador, además del bloque promedios de tiempo que simplemente calculaba los parámetros estadísticos básicos de la señal.

C. *Agregando una señal de tipo vector*

Una vez teniendo todos los bloques que utilizaríamos se procedió a ingresar una señal de tipo vector tanto a los bloques diferenciador y acumulador como al de promedios de tiempo para analizar el funcionamiento de los mismos.

III. ANÁLISIS DE RESULTADOS

En esta sección se estudiarán la programación y funcionamiento de los bloques Integrador, Derivador y Promedios de Tiempo, con el objetivo de analizar el comportamiento de señales al aplicar estas operaciones matemáticas, así como determinar los valores resultantes en
cada caso. Adicionalmente, se buscará identificar posibles aplicaciones prácticas de estos bloques en sistemas de comunicaciones. Antes de analizar distintos valores obtenidos en la aplicación, se toma a consideración la modificación que se le hizo al código planteado por el libro guía.

```
for i in range(len(x)):
    self.acumulated_value += x[i]
    y0[i] = self.acumulated_value

return len(y0)
```

Figura 1. Modificación bloque acumulador

El bloque acumulador se le hizo la modificación mostrada en la Figura 1, utiliza una variable persistente self.acumulated_value que suma secuencialmente cada muestra de entrada x[i], almacenando el resultado en ‘y0[i]‘. Esta estrategia garantiza la continuidad del valor

acumulado entre bloques de procesamiento, evitando discontinuidades abruptas como saltos o caídas artificiales en la señal. Al preservar el estado interno del acumulador entre iteraciones, se mantiene la coherencia temporal al integrar señales muestreadas en bloques discretos, lo que es crítico en aplicaciones de procesamiento de señales donde la integridad de la forma de onda y la eliminación de artefactos por segmentación son prioritarias, como en sistemas de comunicación con procesamiento por tramas. La corrección de la referencia a la variable de salida y0 (en lugar de una señal y inexistente) asegura que los resultados se escriban correctamente en la salida, manteniendo la integridad de la señal en aplicaciones de procesamiento por tramas, como en sistemas de comunicación donde la coherencia temporal es crítica.

```
def work(self, input_items, output_items):
    x = input_items[0]
    y0 = output_items[0]

    N = len(x)

    if N < 2:
        return 0  # Si hay muy pocos datos, no se hace r

    diff = x[1:] - x[:-1]  # Calcula las diferencias

    y0[:len(diff)] = diff  # Copia los datos en la salid
    return len(diff)
```

Figura 2. Modificación bloque diferenciador

El bloque diferenciador se modificó como se muestra en la Figura este calcula la diferencia entre muestras consecutivas de la señal de entrada x mediante la operación $x[1:] - x[:-1]$, almacenando el resultado en y0. Esta coherencia en la asignación evita inconsistencias durante el procesamiento por tramas, asegurando la continuidad de la señal derivada y eliminando artefactos como saltos o valores no definidos, aspectos esenciales en sistemas de comunicaciones donde la precisión temporal y la integridad de la forma de onda son prioritarias.

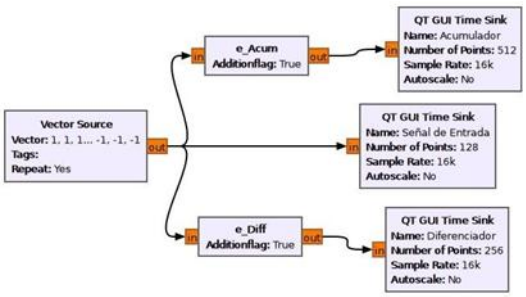

Figura 3. Implementación de bloques

Esta señal se ramifica en dos trayectorias: la primera utiliza un bloque acumulador (integrador) para suavizar variaciones abruptas mediante suma acumulativa, seguido de un bloque de promedio de tiempos que reduce fluctuaciones residuales; la segunda emplea un bloque diferenciador (derivador) para calcular la tasa de cambio entre muestras, también acoplado a un bloque de promedio de tiempos que estabiliza la salida. Este diseño permite comparar como la integración y la derivación, combinadas con promediado temporal, mitigan el ruido y preservan las características esenciales de la señal, destacando su aplicabilidad en sistemas de comunicación donde la precisión temporal y la robustezante perturbaciones son fundamentales.

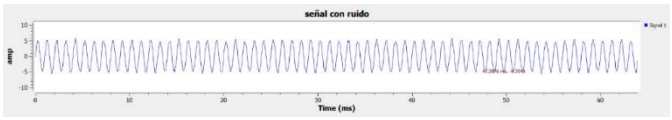

Figura 4. Señal original con adicionamiento de ruido

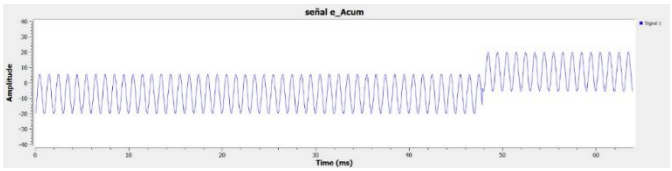

Figura 5. Señal pasada por el bloque Acumulador.

Aunque el bloque acumulador suaviza la señal con ruido generando una apariencia estable como se ve en la Figura 5, la acumulación progresiva del ruido gaussiano induce oscilaciones residuales o deriva en la salida. Este fenómeno evidencia que, al integrar tanto la señal útil como las perturbaciones, el ruido no se elimina, sino que se propaga, resaltando la importancia de técnicas complementarias para mitigar interferencias en aplicaciones prácticas como el procesamiento de señales en comunicaciones.

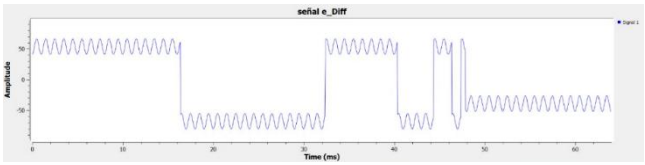

Figura 6. Señal pasada por el bloque Diferenciador

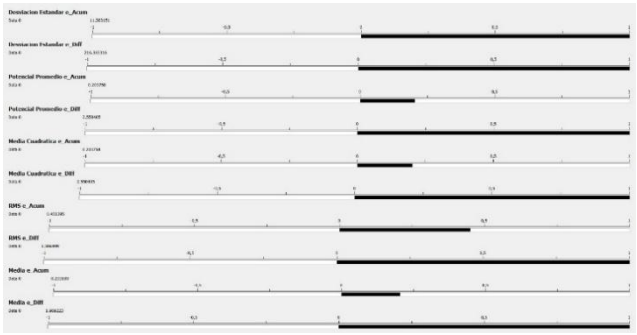

Figura 7. Resultados promedios

Los resultados evidencian que el bloque acumulador (Acum) presenta una desviación estándar considerable (11,30), acompañada de un potencial promedio bajo (0,2038) y una media cuadrática también reducida (0,2038). El valor de RMS (0,4514) y la media (0,2037) indican la presencia de una acumulación progresiva del ruido, que puede generar deriva con el tiempo y fluctuaciones sostenidas.

En contraste, el bloque diferenciador (Diff) muestra una desviación estándar mucho mayor (216,88), junto con un potencial promedio elevado (2,5504) y una media cuadrática igualmente alta (2,5504). Asimismo, sus valores de RMS (1,5970) y media (1,8982) reflejan una mayor sensibilidad a las variaciones rápidas del ruido, lo cual puede ser útil para detectar eventos instantáneos, pero también implica mayor vulnerabilidad frente a perturbaciones de alta frecuencia.

IV. Conclusiones

- ✓ Una conclusión bastante acertada es que los promedios de tiempo estadísticos ayudan mucho a describir y desglosar el comportamiento la señal a analizar, un ejemplo de esto es el promedio que puede decirnos cuál es la media de la señal original (fuente) si sabemos que el ruido que posee la señal es de tipo Gaussiano y la medición de la potencia de una señal también nos indica mediante una comparación de la potencia de la señal de entrada, si nuestra fase de amplificación en el sistema esta´ bien o mal.
- ✓ Los bloques diferenciador y acumulador vimos que actuaron como un derivador e integrador respectivamente, esto debido a que la señal vector se puede tomar como una señal discreta y por tanto las operaciones son válidas, además de esto el bloque acumulador suaviza de cierta manera la señal cuando tiene un ruido gausiano, dando a entender por tanto que actúa de cierta forma como un filtro a frecuencias altas (filtro pasa bajas).
- ✓ La aplicación del acumulador y diferenciador a la señal como un proceso de filtrado, además de la comparación de los promedios de tiempo de la señal antes y después nos dio un amplio concepto sobre como los parámetros matemáticos estadísticos de las señales predicen el comportamiento que estas pueden tener o que tuvieron antes, es por eso tan importante el buen uso de softwares como GNURadio que permiten al usuario ver todas las etapas de la señal en un sistema de comunicación ideal o real para poder tomar las medidas en el diseño necesarias.
- ✓ La interacción entre el integrador, diferenciador y promedios de tiempo permite diseñar sistemas de comunicación adaptativos. Por ejemplo, en receptores SDR, el integrador puede compensar distorsiones de canal en bajas frecuencias, mientras el diferenciador detecta transiciones rápidas en esquemas de modulación, y el promediado estabiliza métricas para la toma de decisiones.
- ✓ Los resultados experimentales correlacionan con teorías de procesamiento de señales (acumulación de ruido en integración vs. amplificación de alta frecuencia en derivación), reforzando la importancia de validar algoritmos en entornos realistas antes de su despliegue en hardware dedicado.
- ✓ Estos resultados destacan que el integrador tiende a acumular ruido de manera más estable, mientras que el diferenciador amplifica componentes de alta frecuencia, por lo que su uso requiere filtros o técnicas adicionales para mitigar este efecto no deseado en aplicaciones sensibles al ruido.

Referencias

[1] Homero Ortega Boda, Oscar Mauricio Reyes Torres. *Comunicaciones Digitales basadas en radio definida por software*. Escuela de Ingenier´ıas Ele´ctrica, Electro´nica y Telecomunicaciones.

[2] GNU Radio. *Python Module - GNU Radio*. Disponible en: https://wiki.gnuradio.org/index.php?title=Python Module

[3] https://github.com/Daniel24-web/CommII_labB1_G1_DANIEL_DIDIER