PROGRAMACIÓN EN RADIO DEFINIDA POR SOFTWARE (GNURADIO)

Santiago Andrés Ibañez Ramírez
Escuela de Ingeniería Electrónica
Universidad Industrial De Santander
Bucaramanga, Colombia
2211258
santiago2211258@correo.uis.edu.co

Nicolás David Martínez Cristancho
Escuela de Ingeniería Electrónica
Universidad Industrial De Santander
Bucaramanga, Colombia
2212269
nicolas2212269@correo.uis.edu.co

Jonathan Mauricio Rojas Rodriguez
Escuela de Ingeniería Electrónica
Universidad Industrial De Santander
Bucaramanga, Colombia
2212280
jonathan2212280@correo.uis.edu.co

Repositorio Github

***Resumen*—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. Metodología

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

II-A. Creación del los directorios

La primera parte del laboratorio constó 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.

II-B. Implementación de los bloques en GNURadio

Para esta parte se siguieron los pasos para la creción 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 parametros estadisticos básicos de la la señal.

II-C. Agregando una señal de tipo vector

Una ves teniendo todos los bloques que utilizaríamos se procedío 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

II-D. Implementando una aplicación de los bloques realizados

Por último se buscó una aplicación de los bloques, agregando un ruido gausiano a la señal de entrada que era de tipo senosoidal para luego ser procesada mediante el los bloques acumulador (integrador) y diferenciador (derivador), a estas señales resultantes se les pudo observar cambios notables y se les hicieron las medidas de los promedios de tiempo para ver que transformaciones tuvieron.

III. Análisis de resultados

*III-A. **Circuito integrador***

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 codigo planteado por el libro guia.

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

return len(y0)
```

Figura 1. Modificacion Bloque Acumulador.

El bloque acumulador se le hizó 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 nada

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

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

Figura 2. Modificacion 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 aproximación procesa cada bloque de datos de forma independiente, evitando discontinuidades al garantizar que las diferencias se calculen únicamente dentro de los límites de cada segmento, sin depender de estados previos entre bloques. Al validar que la longitud de entrada (N) sea suficiente $(N2)$, se descarta el procesamiento de tramas incompletas, lo que previene artefactos como saltos o valores indeterminados.Al igual que en el código del integrador, este bloque diferenciador utiliza la variable de salida $y0$ (en lugar de una referencia errónea a y) para garantizar que los resultados se almacenen correctamente en el buffer de salida. 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.

El flujograma hecho indíca el procesamiento de una señal senoidal de entrada con ruido gaussiano superpuesto, simulando interferencias típicas en entornos reales como se muestra en la Figura 3. 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 cómo 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 robustez ante perturbaciones son fundamentales.

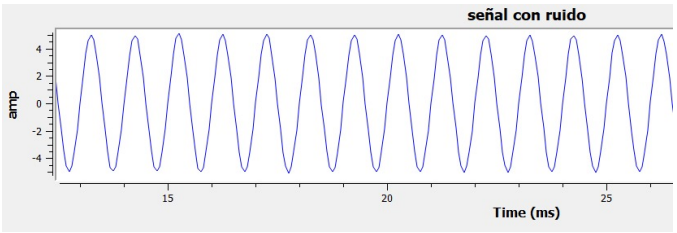

Figura 3. Señal Original con Adicionamiento de Ruido.

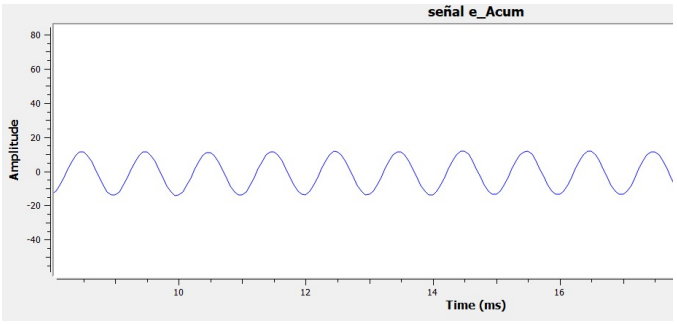

Figura 4. Señal Pasada por el Bloque Acumulador.

Aunque el bloque acumulador suaviza la señal con ruido generando una apariencia estable como se ve en la Figira 4, 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.

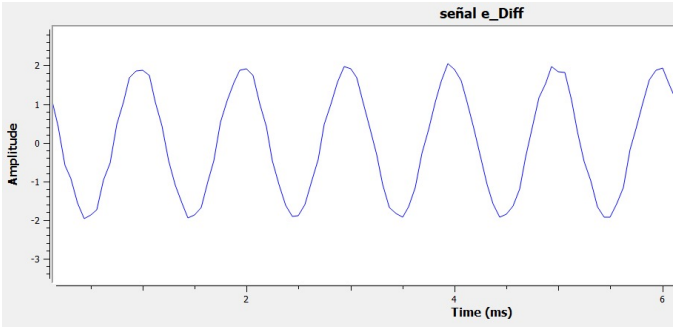

Figura 5. Señal Pasada por el Bloque Diferenciador.

La señal con ruido procesada por la rama del diferenciador mantiene una estructura temporal coherente (conservando la forma de la señal original) como se observa en la Figura 5, pero el cálculo de la tasa de cambio entre muestras acentúa los componentes de alta frecuencia del ruido gaussiano, haciendo evidente su presencia en la salida. Aunque el bloque de promedio de tiempos atenúa parcialmente estas fluctuaciones, la derivación revela cómo el ruido se manifiesta como perturbaciones rápidas superpuestas a la señal útil, un aspecto crítico en sistemas donde la detección precisa de variaciones instantáneas es esencial, como en demodulación o detección de transitorios.

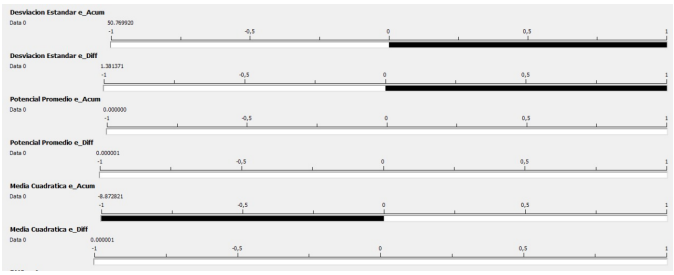

Figura 6. Resultados Promedio de Tiempo 1.

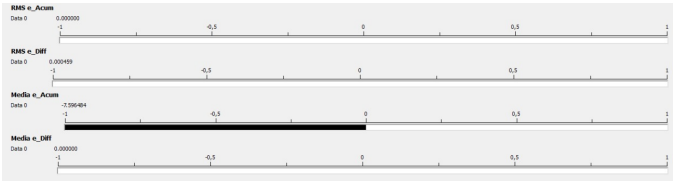

Figura 7. Resultados Promedio de Tiempo 2.

Los resultados de las Figuras 6 y 7 evidencian que el bloque acumulador (Acum) presenta una alta desviación estándar (50,77) y media negativa (-7,596), producto de la acumulación progresiva del ruido gaussiano, lo que genera deriva y fluctuaciones significativas (RMS: 8,87). En contraste, el bloque diferenciador (Diff) muestra una desviación estándar baja (1,38) y valores mínimos de RMS (0,000459) y media cuadrática (0,000001), indicando que la derivación mitiga la propagación del ruido, aunque resalta sus componentes de alta frecuencia como perturbaciones rápidas. Esto subraya que el integrador requiere técnicas adicionales para controlar el ruido acumulado, mientras el diferenciador preserva la estabilidad temporal, priorizando la detección de cambios instantáneos en aplicaciones de procesamiento de señales.

El bloque de promedios de tiempo complementa al integrador y diferenciador en sistemas de comunicacion: junto al integrador, permite suavizar senales con ruido en receptores de FM o en estimacion de parametros estadisticos (como potencia promedio en enlaces inalambricos), mientras que, acoplado al diferenciador, ayuda a estabilizar mediciones en esquemas de modulacion digital (ej. QAM) al reducir fluctuaciones de alta frecuencia. Ademas, este bloque es clave en aplicaciones como radar y sonar, donde promedia multiples pulsos para mejorar la relacion senal-ruido, o en sistemas de sincronizacion temporal para alinear tramas en redes opticas. Al combinar estos tres bloques, se logra un equilibrio

entre eliminacion de ruido acumulado (integrador), deteccion de cambios instantaneos (diferenciador) y estabilizacion de senales (promedios), optimizando desempeno en radio definida por software, telecomunicaciones 5G y procesamiento de senales en tiempo real.

Los bloques integrador y diferenciador tienen roles clave en sistemas de comunicacion: el primero se emplea en demodulacion de senales FM, donde la integracion recupera la informacion original al suavizar variaciones, o en sistemas de control de potencia para promediar consumos. Sin embargo, requiere filtros complementarios para mitigar el ruido acumulado. El diferenciador, por su parte, es util en deteccion de transiciones rapidas, como en esquemas de modulacion digital (ej. PSK) para identificar cambios de fase, o en ecualizacion de canales para corregir distorsiones. Ambos bloques, combinados con tecnicas de promediado, optimizan la precision en entornos con interferencias, equilibrando estabilidad temporal y respuesta dinamica en aplicaciones como radio definida por software o procesamiento de senales en tiempo real.

IV. CONCLUSIÓNES

- Una conclusion bastante acertada es que los promedios de tiempo estadisticos ayudan mucho a describir y desglozar el comportamiento la señal a analizar, un ejemplo de esto es el promedio que puede decirnos cual 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 indíca mediante una comparación de la potencia de la señal de entrada, si nuestra fase de amplificación en el sistema está bien o mal.

- Los bloques diferenciador y acumulador vimos que actúaron 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 validas, además de esto el bloque acumulador suavisa de cierta manera la señal cuando tiene un ruido gausiano, dando a entender por tanto que actua de cierta forma como un filtro a frecuencias altas (filtro pasa bajas).

- La aplicación de el acumulador y diferenciador a la señal como un proceso de filtrado, ademas de la comparacion de los promedios de tiempo de la señal antes y despues nos dio un amplio concepto sobre como los parametros matematicos-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 cominicación 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.

REFERENCIAS

[1] Homero Ortega Boda, Oscar Mauricio Reyes Torres. *Comunicaciones Digitales basadas en radio definida por software*. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones.

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

[3] Repositorio en GitHub. Disponible en: https://github.com/Jomao03/com2_B1_G3