PRACTICA 1:

Práctica 1: Implementación y Análisis de Bloques en GNU Radio para Acumulación y Diferenciación de Señales

Garzon Espejo Johan Esneider; johan2215588@correo.uis.edu.co

Meyer José Suarez Monroy; meyer2211601@correo.uis.edu.co

Abstract

This paper presents the implementation and validation of custom blocks in GNU Radio using Python. Accumulator and Differentiator blocks were developed to extract deterministic (amplitude, frequency, phase, period) and probabilistic (mean, variance, RMS) parameters from signals, allowing real-time analysis and improved noise handling. The results demonstrate the effectiveness of the approach and lay the foundation for future optimizations in software-defined radio systems.

1. INTRODUCCIÓN

Para diseñar un sistema de comunicaciones eficiente, es necesario comprender las características de la señal, conocidas como parámetros fundamentales: amplitud, frecuencia, fase y período, los cuales determinan el comportamiento del sistema [1].

Estos parámetros se identifican fácilmente en señales determinísticas, que son predecibles en el tiempo. En cambio, las señales aleatorias, al ser impredecibles, requieren del uso de parámetros probabilísticos, los cuales permiten describir con mayor precisión su comportamiento. Entre los más comunes se encuentran la media, varianza, valor RMS, potencia promedio y desviación estándar. Debido a la variabilidad de estas señales, su análisis debe realizarse en intervalos cortos, lo que constituye la base de la teoría de señales aleatorias [2].

En sistemas modernos, estos parámetros pueden implementarse en **GNU Radio** mediante bloques programados en **Python**, lo que brinda mayor flexibilidad que los bloques prediseñados. Esto permite personalizar el análisis, optimizar el rendimiento del sistema, incorporar algoritmos avanzados y procesar señales en tiempo real, ofreciendo soluciones más robustas en telecomunicaciones [1].

2. OBJETIVO PRINCIPAL

Implementar el uso de GitHub como herramienta de apoyo en el trabajo colaborativo, permitiendo la gestión, actualización y control de proyectos académicos de manera eficiente, mediante la creación de repositorios, el uso de tokens de acceso y la aplicación de buenas prácticas en el manejo de la información.

3. MARCO TEÓRICO

En la actualidad, el trabajo colaborativo en entornos académicos y profesionales requiere de plataformas que permitan la organización y el control de proyectos de manera eficiente. GitHub es una de las herramientas más utilizadas para este fin, ya que se basa en **Git**, un sistema de control de versiones distribuido que facilita el seguimiento de cambios, la integración de aportes de diferentes usuarios y la gestión del código fuente.

El uso de GitHub no solo se limita al almacenamiento en la nube, sino que también permite implementar flujos de trabajo colaborativos mediante ramas, commits y pull requests, lo cual favorece la coordinación en proyectos de gran escala. Además, el manejo de **tokens de acceso personal** garantiza la seguridad en las operaciones realizadas desde repositorios locales hacia la plataforma en línea.

De igual manera, la correcta gestión de la información es un aspecto esencial en el desarrollo de proyectos académicos y de investigación. Contar con repositorios organizados y documentados (por ejemplo, a través de archivos *Readme.md*) asegura la trazabilidad y la reutilización del conocimiento generado, apoyando tanto el proceso de formación como el desarrollo de futuras investigaciones.

4. METODOLOGÍA EXPERIMENTAL

Inicialmente se crea una rama para el desarrollo de la practica desde el repositorio, denominadaPráctica_1, la cual debe ser compartida con los demás integrantes del grupo. A esta rama se le asigna un repositorio con el mismo nombre, y dentro de dicho repositorio se crean dos carpetas: GNU Radio e Informe. Cada integrante del grupo establece su propia rama individual dentro de la rama Practica_1, nombrada como P1_Nombre1, P1_Nombre2, y P1_Nombre3. Se accede a estas ramas con el comando "git checkout P1_Nombre

Se implementaron dos bloques de Python en el entorno de GNU-Radio: un acumulador y un diferenciador, con ayuda del libro guía propuesto en la práctica, en la sección 1.2.0.1 [1], se obtuvieron los códigos guías para implementarlos en los respectivos bloques.

Bloque acumulador:

El bloque de Python denominado Acumulador se programó para sumar de manera consecutiva las muestras de la señal de entrada según la ecuación:

$$y[n] = y[n-1] + x[n]$$

Para evitar que la salida permaneciera cercana a cero con una señal cuadrada simétrica, se aplicó un desplazamiento de -0.5 a la fuente. Este offset impide la cancelación entre valores positivos y negativos, lo que permite apreciar de manera clara la evolución acumulada. Finalmente, se conectó la salida del bloque y la señal de entrada a un visualizador en tiempo real, a fin de comparar la forma de onda original y la forma procesada.

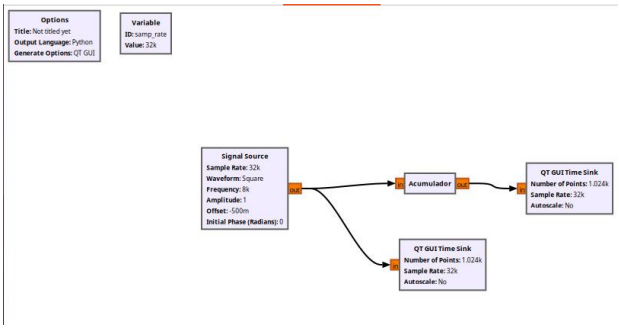

Figura 1. *Diagrama de flujo en GNU Radio que muestra un bloque acumulador con una señal cuadrada de entrada y un offset de −0.5.*

Bloque diferenciador:

El bloque diferenciador es aquel que realiza la operación de la diferenciación, es decir, calcula la derivada con respecto al tiempo, de manera matemática se puede ver de la siguiente manera.

$$y[n] = x\,[n] + x\,[n-1]$$

Nuevamente se emplea un bloque de Python en GNURadio para implementar un algoritmo que me haga esta operación, en donde se le ingresa una señal con el fin de obtener su derivada.

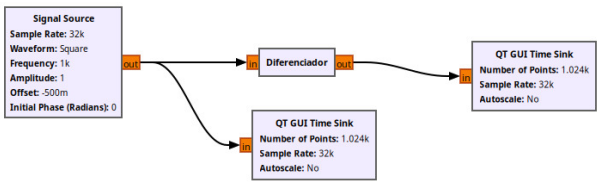

Figura 2. *Implementación del bloque diferenciador.*

A continuación, se debe crear un tercer bloque para mostrar las estadísticas de una señal. Este bloque, al igual que los anteriores, se implementa utilizando el bloque 'Python block' en GNU Radio, donde se realiza con base al libro guía sección 1.2.0.1 [1]. A partir de esta implementación se debe realizar una aplicación, en este caso disminuir la afectación del ruido Gaussiano en la estadística, para ello se divide la señal en 10 ventanas de tamaño 4096 muestras y se decide utilizar funciones propias de Python para intentar comparar con las sugeridas por el libro.

Bloque promedio del tiempo

El bloque desarrollado, llamado "Promedios_de_tiempos", fue definido como un bloque de sincronización (gr.sync_block). Se inicializó con variables acumulativas para mantener un seguimiento del cálculo en tiempo real, evitando pérdida de información en flujos continuos de datos.

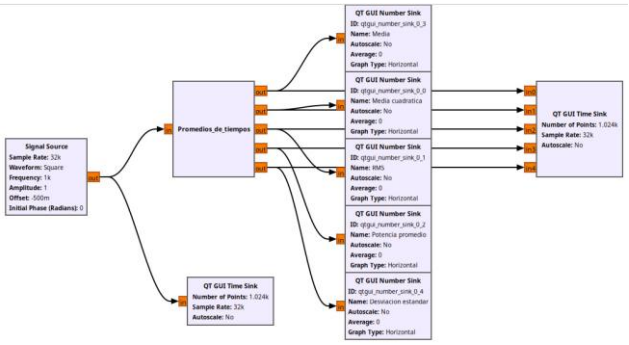

Figura 3. *Diagrama de flujo estadísticas de una señal.*

Dentro del método **work()** se implementaron los siguientes cálculos:

- **Media:** se obtiene sumando todos los valores de la señal y dividiéndolos entre el número total de muestras.
- **Media cuadrática:** se calcula de forma análoga a la media, pero considerando los valores de la señal al cuadrado.
- **Valor RMS:** corresponde a la raíz cuadrada de la media cuadrática.
- **Potencia promedio:** resulta de elevar al cuadrado el valor RMS.
- **Desviación estándar:** se determina a partir de la suma de las diferencias al cuadrado entre la señal y su media, dividiendo entre la cantidad de muestras y extrayendo finalmente la raíz cuadrada.

Con esta implementación se plantea una aplicación orientada a reducir la influencia del ruido gaussiano en los cálculos estadísticos. Para ello, la señal se segmenta en 10 ventanas de 4096 muestras cada una y se emplean funciones nativas de Python, con el fin de compararlas frente a las propuestas en el libro.

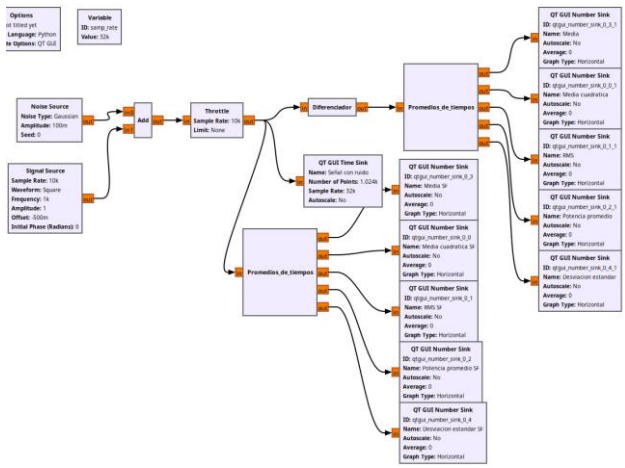

Figura 4. *Implementación de la aplicación de reducción del ruido Gaussiano.*

5. ANÁLISIS DE RESULTADOS

Bloque acumulador:
Para la prueba se utilizó una señal cuadrada con un desplazamiento de 0.5 como entrada. Al pasarla por el bloque **Accumulator**, la salida resultante tomó la forma de una onda triangular, debido a que cada nivel constante de la señal cuadrada se integra de manera lineal mientras permanece activo. Además, el desplazamiento aplicado evita la cancelación entre valores positivos y negativos, lo que provoca que la salida vaya aumentando progresivamente.

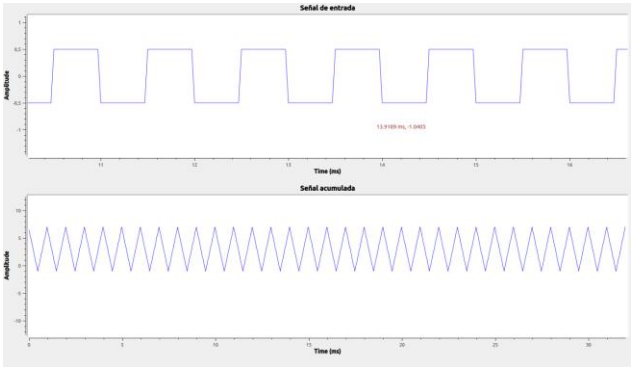

Figura 5. *Resultados de simulación bloque Acumulador.*

Bloque diferenciador:
Para comprobar el correcto funcionamiento del bloque diferenciador, se aplicó una señal cuadrada. Como era de esperar, la salida correspondió a una serie de impulsos, obteniéndose el siguiente resultado.

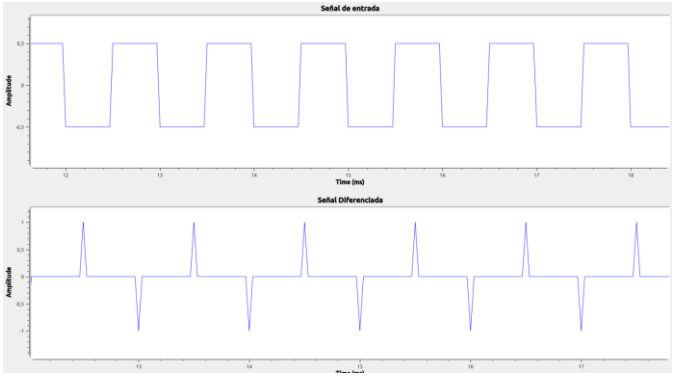

Figura 6. *Resultados de simulación bloque Diferenciado.*

Promedio de tiempo y aplicación:
El bloque **"Promedios_de_tiempos"** permitió comprobar el cálculo en tiempo real de las principales estadísticas de la señal. En la primera simulación, con una señal cuadrada de entrada, se observó que los valores obtenidos para la media, media cuadrática, RMS, potencia promedio y desviación estándar fueron coherentes con el comportamiento esperado de la señal periódica sin ruido.

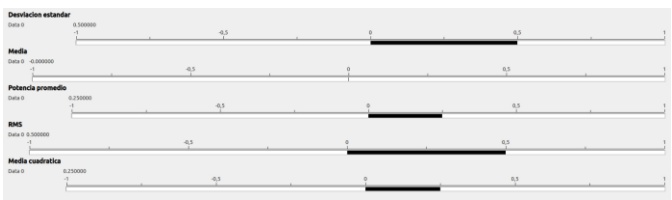

Figura 7. *Resultados de simulación promedio de tiempo.*

Posteriormente, al implementar la aplicación con la adición de ruido gaussiano, se evidenció un aumento en la variabilidad de la señal, lo que se reflejó principalmente en la desviación estándar y en ligeras variaciones del RMS y la potencia promedio. Esto demuestra que el bloque desarrollado no solo cumple con el cálculo correcto de los parámetros estadísticos, sino que también es capaz de evidenciar el efecto del ruido en el análisis de señales, validando así su utilidad en escenarios de procesamiento más realistas.

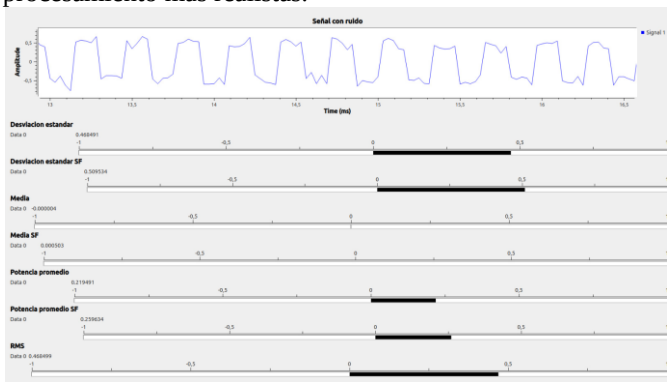

Figura 8. *Resultados de simulación, aplicación.*

6. CONCLUSIONES

- La comparación entre las funciones propias de Python y las sugeridas en el libro permitió comprobar la validez de los conceptos teóricos vistos en clase, mostrando que diferentes enfoques pueden llegar a resultados equivalentes en la reducción de ruido.

- Durante la implementación práctica fue necesario realizar ajustes en los códigos, lo que evidencia la importancia de la depuración y adaptación del software para garantizar un correcto funcionamiento en entornos reales.

- Se identificó que los bloques acumulador y diferenciador presentan limitaciones cuando la señal tiene un promedio distinto de cero y valores siempre positivos, ya que tienden a crecer indefinidamente; sin embargo, en la práctica digital estos sistemas llegan a un desbordamiento y reinician su operación.

- La implementación del bloque acumulador en GNU Radio resaltó la necesidad de introducir un offset en señales cuadradas simétricas, evitando así la cancelación entre valores positivos y negativos y permitiendo observar adecuadamente la evolución de la señal.

7. REFERENCIAS

[1] H. O. Boada and O. M. R. Torres, Comunicaciones Digitales basadas en radio definida por software, primera ed. Bucaramanga, Colombia: Editorial UIS, 2019.

[2] S.-M. M. Yang, Review of Signals and Sys-temsSignalsand systems, and of Probability and Random Process. Cham: Springer Interna- tional Publishing, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-57706-3_9