Práctica 1: PROGRAMACIÓN EN RADIO DEFINIDA POR SOFTWARE (GNURADIO)

Natalia Estefanía Cruz Castillo - 2214536
Oscar Andrés Meza Ortíz- 2202341
Valeria de los Ángeles Sarmiento Clavijo - 2202326

https://github.com/ValeriaS57/CommunicationsII_2024_2_B1_ONV/tree/Practica_1/Practica_1

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones
Universidad Industrial de Santander

8 de septiembre de 2024

Abstract

In this practice, fundamental concepts in the field of communications were explored using GNU Radio, developing blocks that simulate an accumulator and a differentiator applied to time-based signals or repetitive vectors. A block for statistical analysis was also included, evaluating both interference-free signals and signals with Gaussian distortion that were filtered to mitigate the effects of noise, demonstrating the effectiveness of the filter application for signals affected by disturbances.

Palabras clave: Blocks simulate, GNU Radio, Accumulator, Differentiator

1 Introducción

Para la correcta construcción de un sistema de comunicación, es esencial comprender las diversas características de la señal, lo que permite realizar los tratamientos adecuados y mejorar la eficiencia del sistema. Estas características se conocen como parámetros fundamentales, entre estos se encuentran la amplitud, frecuencia, fase y período. Cada uno de estos factores influye directamente en el comportamiento y rendimiento del sistema de comunicación[1].

Sin embargo, estos parámetros fundamentales son más fáciles de identificar en señales determinísticas, es decir, aquellas que se mantienen consistentes y predecibles a lo largo del tiempo, siendo de esta forma predecibles. En su contraparte tendremos las señales aleatorias, requieren de diversos procesos para ser captados, ya que como su nombre lo indica son impredecibles. Una solución óptima es trabajar con parámetros probabilísticos, permitiéndonos tener un rango más claro sobre el comportamiento de la señal y realizar aproximaciones precisas de sus características[2].

Los parámetros probabilísticos que se utilizan incluyen la varianza, la media, la media cuadrática(M.C.), la potencia promedia, el valor RMS y desviación estándar(D.E.). Debido a la naturaleza variable de la señal, es necesario analizar la señal en segmentos de tiempos pequeños. Al hacerlo, se pueden calcular estadísticas locales que reflejan el comportamiento de la señal con mayor precisión, que si analizáramos intervalos mucho más grandes, a esto se le conoce como la teoría de las señales aleatorias[2].

Para desarrollar sistemas de comunicación avanzados, es posible implementar parámetros probabilísticos mediante el uso de bloques programados en Python dentro del entorno de GNU Radio. Esta capacidad de programación ofrece una flexibilidad mucho mayor en comparación con los bloques predefinidos del software, ya que permite personalizar y ajustar parámetros de manera específica según las necesidades del proyecto. Esto no solo facilita la optimización del rendimiento del sistema, sino que también permite la integración de algoritmos complejos, el procesamiento de señales en tiempo real y el control preciso de aspectos críticos, como el filtrado de señales. Al aprovechar la programación en Python, se pueden diseñar soluciones más robustas y adaptadas a escenarios específicos, lo que resulta fundamental en aplicaciones modernas de telecomunicaciones[1].

2 Procedimiento

- Inicialmente, se crea una rama para el desarrollo de la práctica desde el repositorio, denominada

Práctica_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".

- En GNU Radio, se deben crear dos bloques: un acumulador y un diferenciador, utilizando el bloque 'python block'. Para editar el código Python, se hace doble clic en el bloque. El código para el acumulador y el diferenciador se implementa dentro de este bloque, con base al libro guía en la sección 1.2.0.1 [1], posteriormente se implementan los bloques, tal como se observa en la Fig.2.1

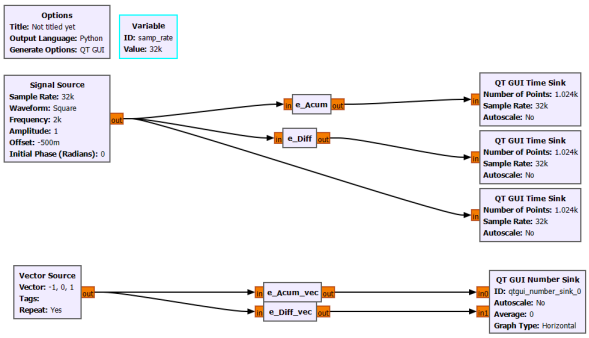

Fig.2.1 Implementación de los bloques acumulador y diferenciador tanto para una señal continua y vectores.

- 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.

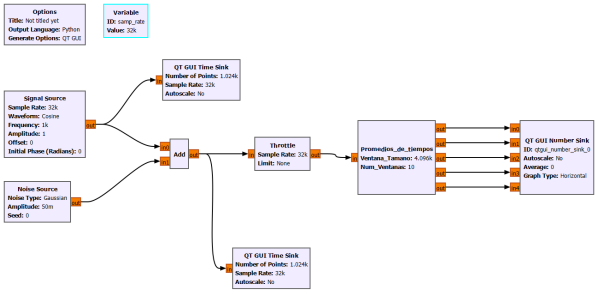

Fig.2.2 Implementación de la aplicación de reducción de ruido Gaussiano con las modificaciones realizadas.

- Para subir los cambios hechos en el dispositivo se usa el comando "git status" para verificar que hubo cambios de manera local. Si los cambios realizados se hicieron correctamente se debe utilizar el comando "git add .", este comando añade todos los archivos y cambios al área de preparación, por lo que deben ser confirmados con "git commit -m "mensaje"". Finalmente, con el comando "git push" los cambios son subidos a la nube en GitHub.

3 Resultados

Se ejecutó los bloques de la Fig.2 con una señal cuadrada de amplitud 1, offset -0.5 y frecuencia 2 [kHz], obteniendo como resultado para el acumulador una señal triangular de la misma frecuencia, es decir, 2 [kHz] y una amplitud diferente, lo cual corresponde con el comportamiento teórico donde la integral de una señal cuadrada periódica resulta en una señal triangular. Mientras que en el diferenciador se observa la variación de la señal entre 2 valores que van creciendo en el transcurso del tiempo y corresponde a la razón, se cambió de la señal cuadrada.

Mientras en el caso de la señal vectorial se utilizó el vector=[-1,0,1] con el fin de visualizar un correcto funcionamiento de los bloques, en el caso del acumulador varía entre los valores y al sumarse se anulan obteniendo cero, repitiendo este procedimiento la cantidad de veces que se repita el vector. Mientras que en el caso del diferenciador observamos que va variando su signo y aumentando su magnitud en el transcurso del tiempo, tal como se visualiza en la Fig. 3.3

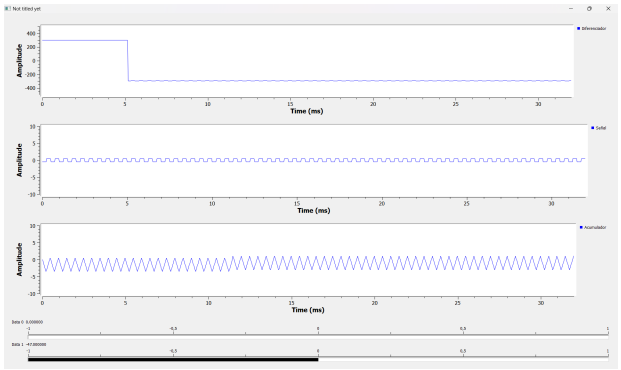

Fig.3.3 Resultados obtenidos del bloque acumulador y diferenciador tanto para una señal periódica cuadrada y un vector repetitivo.

Mientras que en el caso del bloque estadístico y su aplicación (Ver Fig.2.2), se realizó la evaluación de los bloques, variando el ruido Gaussiano desde 0 hasta 0.1, y registrando sus valores para su posterior comparación.

Calculado sus valores teóricos obtenemos:

$$\text{Media} = \frac{1}{\frac{1}{2000}} \int_0^{\frac{1}{2000}} sin(4000\pi x)\, dx = 0 \qquad (1)$$

$$\text{M. C.} = \frac{1}{\frac{1}{2000}} \int_0^{\frac{1}{2000}} sin(4000\pi x)^2\, dx = \frac{1}{2} \qquad (2)$$

$$\text{RMS} = \sqrt{\frac{1}{\frac{1}{2000}} \int_0^{\frac{1}{2000}} sin(4000\pi x)^2\, dx} = \frac{1}{\sqrt{2}} \qquad (3)$$

$$\text{Potencia} = \frac{1}{\frac{1}{2000}} \int_0^{\frac{1}{2000}} sin(4000\pi x)^2\, dx = \frac{1}{2} \qquad (4)$$

$$\text{D.E.} = \sqrt{\frac{1}{\frac{1}{2000}} \int_0^{\frac{1}{2000}} (sin(4000\pi x) - 0)^2\, dx} = \frac{1}{\sqrt{2}} \qquad (5)$$

Como se observa en la Tabla 3.1 los valores más cercanos a los teóricos obtenidos en las ecuaciones (1,2,3,4,5) se obtienen a través de la implementación de la aplicación (2) cuando el ruido es cero, siendo estos iguales, aunque se destaca que con los bloques de estadística (1) varía levemente solo en el 4 o 3 decimal, si aumentamos el ruido se mantiene ese patrón con excepción de la media.

Ruido	Media	Media cuadrada	RMS	Potencia promedio	Desviación estandar
0 (1)	0	0.5002	0.7073	0.5002	0.7073
0 (2)	0	0.5000	0.7071	0.5000	0.7071
0.25 (1)	0	0.5003	0.7073	0.5003	0.7073
0.25 (2)	0	0.5001	0.7071	0.5002	0.7072
0.05 (1)	0.0001	0.5028	0.7091	0.5028	0.7088
0.05 (2)	0	0.5008	0.7077	0.5008	0.7077
0.1 (1)	0	0.5113	0.7152	0.5112	0.7141
0.1 (2)	0.0002	0.5100	0.7138	0.5100	0.7137

Tabla 3.1. Resultados de los bloques de estadística(1) y aplicación(2) para diferentes niveles de ruido en una señal seno.

4 Conclusiones

Referente al desarrollo de los códigos propuestos se le hicieron pequeñas modificaciones como arreglar la indentación, cambiar unas comillas y corregir el orden de unas líneas de código para su correcto funcionamiento. En el caso del bloque acumulador y diferenciador si tenemos una señal que su media no es cero, y sus valores están sobre el eje positivo los valores de salida de los bloques tienden a infinito, pero en la realidad de la implementación llegan a un punto donde se desbordan y reinicia la operación como ocurría en sistemas digitales.

Comparando la programación del bloque estadístico que utilizó los códigos propuestos en el libro guía y el código de la aplicación modificado con funciones propias de Python obtenidas de la librería NumPy, tal como np.mean(), así como el uso de ventanas de tamaño 2^{12} muestras para mejorar los valores, se observa que presenta una gran exactitud cuando su ruido es cero, pero mientras más crece sus valores varían un poco entre ellos y tarda más para estabilizarse en comparación al código propuesto.

Como consecuencia, se considera que el bloque de aplicación no hace un aporte significativo para mitigar los efectos del ruido, debido a que no se visualiza una diferencia notable entre los resultados, pero si se evidencia un contraste en el tiempo en el que demora en estabilizarse la respuesta de la señal. Por lo que para una próxima aplicación se sugiere la búsqueda de una estrategia diferente para reducir el efecto de las distorsiones en la señal y que a su vez logre estabilizarse su respuesta en el menor tiempo posible.

De igual manera, respecto a la visualización de las salidas continuas, se concluye que en el caso del acumulador y el diferenciador es mejor utilizar bloques QT GUI Time Sink para visualizar como varía la forma de la señal a través del tiempo, mientras que en el caso del bloque de estadística y vectores es mejor QT GUI Number Sink para poder visualizar sus valor numérico.

References

[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 SystemsSignalsand systems, and of Probability and Random Process*. Cham: Springer International Publishing, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-57706-3_9