

Práctica 1: Acondicionamiento de Señales y GNU Radio

1^{er} juan david camacho gonzalez

Código: 2210428

Universidad Industrial de Santander

Bucaramanga, Colombia

2^{do} Valentina Arguellez Angulo

Código: 2215670

Universidad Industrial de Santander

Bucaramanga, Colombia

DECLARACIÓN DE ORIGINALIDAD Y RESPONSABILIDAD

Los autores de este informe certifican que el contenido aquí presentado es original y ha sido elaborado de manera independiente. Se han utilizado fuentes externas únicamente como referencia y han sido debidamente citadas. Asimismo, los autores asumen plena responsabilidad por la información contenida en este documento.

Uso de IA: Se utilizó el modelo de lenguaje Gemini como asistencia técnica para la estructuración del código en LaTeX, resolución de errores de compilación en Ubuntu/WSL, sintaxis de comandos de Git y revisión gramatical de la justificación técnica. El diseño del filtro, la implementación en GNU Radio y el análisis de resultados fueron desarrollados íntegramente por los autores.

Resumen—Este informe muestra el desarrollo e implementación de bloques personalizados para el procesamiento digital de señales en GNU Radio utilizando Python. Se diseñaron módulos como acumuladores y diferenciadores, para analizar el comportamiento de los datos, se manejo la contaminación de señales por ruido Gaussiano mediante un filtro estadístico discreto de media móvil. La eficacia de este acondicionamiento se validó proponiendo su uso en la estabilización de biopotenciales, una etapa crítica en el procesamiento de señales mioeléctricas para sistemas de control de fuerza en la reproducción de movimientos de la mano. Los resultados demuestran que el promediado estadístico atenúa exitosamente las fluctuaciones del ruido, garantizando una señal estable y manteniendo una latencia adecuada para aplicaciones en tiempo real.

Index Terms—GNU Radio, Filtro Media Móvil, Procesamiento de Señales, Python.

I. INTRODUCCIÓN

El procesamiento digital de señales ha experimentado una evolución significativa con la consolidación de la Radio Definida por Software (SDR). Plataformas de código abierto como GNU Radio han transformado el análisis y la transmisión de datos, permitiendo a los desarrolladores trascender el uso de herramientas predefinidas [2], [3] para programar y compilar algoritmos personalizados en lenguajes como Python. Esta flexibilidad es crucial para adaptar el procesamiento de la información a las exigencias matemáticas y físicas de cada entorno.

En el desarrollo de la práctica, se exploran los fundamentos de la creación de bloques de procesamiento en tiempo real. En una primera etapa, se hace la implementación de operaciones

matemáticas (acumulación y la diferenciación). Estos módulos conforman la base para el análisis del comportamiento dinámico de los sistemas y la evaluación de tasas de cambio en las señales a lo largo del tiempo.

Por otro lado, un desafío crítico en la instrumentación y adquisición de señales físicas es la presencia de ruido. El acondicionamiento estadístico se vuelve obligatorio en implementaciones de alta precisión; por ejemplo, durante el diseño y validación de un sistema de control de fuerza en la reproducción de movimientos básicos de la mano, donde la presencia de componentes ruidosas en las señales mioeléctricas puede desestabilizar la etapa de inicial. Para solucionar este problema, se propone la implementación de técnicas estadísticas de suavizado.

Finalmente, el documento detalla los resultados obtenidos tras la integración de los módulos de acumulación, diferenciación y filtrado estadístico, en la Sección III se encuentran las conclusiones derivadas del análisis del sistema.

II. PROCEDIMIENTO

II-A. Acondicionamiento Estadístico: Filtro de Media Móvil

La presencia de ruido Gaussiano es un desafío inherente en la adquisición de señales físicas. Para mitigar las fluctuaciones aleatorias y mejorar la relación señal a ruido (SNR), se diseño un bloque de procesamiento estadístico personalizado. La estrategia seleccionada corresponde a un filtro discreto de media móvil, regido por la siguiente ecuación de diferencias [1]:

$$y[n] = \frac{1}{N} \sum_{i=0}^{N-1} x[n-i] \quad (1)$$

Donde $N = 10$ representa el tamaño de la ventana de promediado, $x[n]$ es la señal de entrada contaminada y $y[n]$ es la salida estabilizada.

II-A1. Implementación en GNU Radio: La arquitectura del sistema se construyó utilizando el entorno de GNU Radio Companion. El filtro se programó desde cero mediante un *Embedded Python Block*. Como se evidencia en el diagrama de flujo (Fig. 1), la señal original se somete a una fuente de ruido antes de ingresar al bloque diseñado, permitiendo evaluar su desempeño en tiempo real.

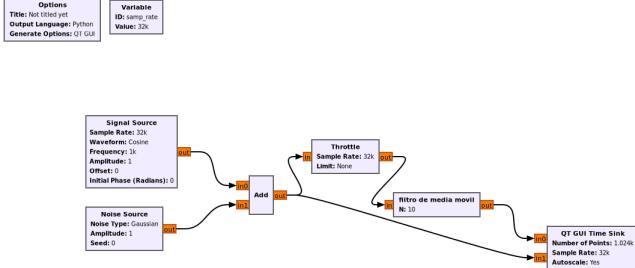


Figura 1. Diagrama de bloques implementado en GNU Radio, destacando el módulo de filtro estadístico en Python.

II-A2. Validación y Aplicación Práctica: La capacidad de programar filtros a medida es un requerimiento crítico en aplicaciones de alta complejidad, como el diseño y validación de sistemas de control de fuerza para la reproducción de movimientos básicos de la mano [4]. En este contexto, las señales mioeléctricas crudas presentan una alta varianza que puede desestabilizar a los actuadores mecánicos. Aplicar un promedio estadístico es el paso previo obligatorio para garantizar un control suave y preciso.

Al ejecutar el sistema, los resultados obtenidos (Fig. 2) demuestran la eficacia del bloque. La estadística aplicada atenúa significativamente los picos del ruido Gaussiano. Aunque persiste un leve rizado debido a una ventana pequeña ($N = 10$), este diseño garantiza una baja latencia computacional, un factor indispensable para el tiempo de respuesta inmediato que exige el control de una prótesis real.

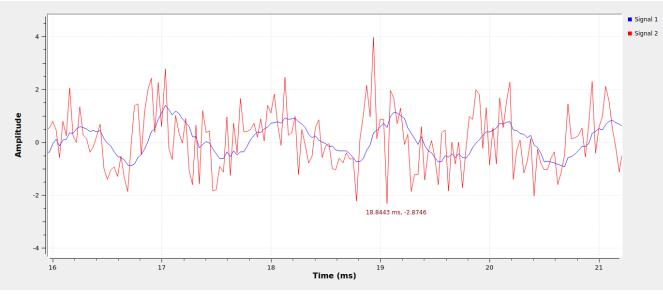


Figura 2. Resultados del procesamiento: Comparación entre la señal contaminada con ruido Gaussiano y la señal estabilizada.

II-B. Diferenciador Discreto

Un diferenciador es un sistema cuya salida corresponde a la derivada temporal de la señal de entrada. En tiempo continuo, si la señal de entrada es $x(t)$, el diferenciador ideal está definido como:

$$y(t) = \frac{d}{dt}x(t) \quad (2)$$

Este operador mide la tasa de cambio instantánea de la señal. Por ejemplo, si $x(t) = \sin(\omega t)$, entonces:

$$\frac{d}{dt} \sin(\omega t) = \omega \cos(\omega t) \quad (3)$$

lo cual implica que la derivada de una señal senoidal es otra señal senoidal desfasada 90° y con amplitud proporcional a la frecuencia angular. En sistemas discretos, la derivada se approxima mediante diferencias finitas. La forma más sencilla del diferenciador discreto es:

$$y[n] = x[n] - x[n - 1] \quad (4)$$

Esta expresión calcula la variación entre muestras consecutivas, permitiendo estimar la pendiente local de la señal. Si la señal permanece constante, la salida es cero; si la señal presenta un incremento o decremento, la salida toma valores positivos o negativos respectivamente.

II-B1. Implementación en GNU Radio: La implementación del diferenciador se realizó mediante un bloque personalizado en GNU Radio utilizando la opción *Embedded Python Block*. Este bloque fue programado para ejecutar la operación de diferencia discreta entre muestras consecutivas, de acuerdo con la expresión:

$$y[n] = x[n] - x[n - 1] \quad (5)$$

El bloque fue diseñado como un *sync_block*, con una entrada y una salida de tipo *float32*, permitiendo procesar la señal muestra a muestra y mantener continuidad entre bloques mediante el almacenamiento del valor previo de la señal. Para

la validación del diferenciador se construyó un diagrama de bloques en GNU Radio compuesto por:

- **Signal Source:** Generador de señal, configurado con distintas formas de onda (senoidal, cuadrada) para evaluar el comportamiento del sistema.
- **Throttle:** Bloque utilizado para controlar la tasa de procesamiento y evitar el consumo excesivo de CPU en simulación.
- **Diferenciador (bloque Python):** Encargado de realizar la operación de diferencia discreta.
- **QT GUI Time Sink:** Utilizado para visualizar simultáneamente la señal original y la señal diferenciada en el dominio del tiempo.

La Figura 3 muestra el diagrama de bloques implementado en GNU Radio con la señal senoidal.

La Figura 4 muestra el diagrama de bloques implementado en GNU Radio con la señal cuadrada.

II-B2. Validación y Aplicación Práctica: El bloque diferenciador implementado fue evaluado utilizando dos tipos de señales de entrada: una señal senoidal y una señal cuadrada. El objetivo fue verificar experimentalmente el comportamiento del sistema frente a variaciones suaves y cambios abruptos en



Figura 3. Diagrama de bloques del diferenciador implementado en GNU Radio señal seno.

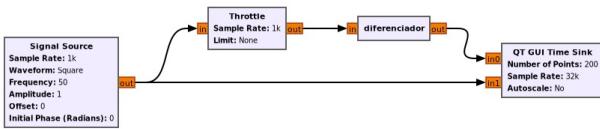


Figura 4. Diagrama de bloques del diferenciador implementado en GNU Radio señal cuadrada.

el tiempo, contrastando los resultados obtenidos con el modelo teórico de la derivada.

En el caso de la señal cuadrada, se observó que la salida del diferenciador presentó impulsos de alta amplitud en los instantes donde la señal experimenta transiciones entre sus niveles máximo y mínimo. Durante los intervalos donde la señal permanece constante, la salida fue aproximadamente cero. Este comportamiento es coherente con la teoría, ya que la derivada de una función constante es nula, mientras que en los puntos de cambio brusco la pendiente es elevada. Por lo tanto, el diferenciador actúa como detector de flancos, generando picos positivos en las transiciones ascendentes y picos negativos en las descendentes. La Figura 5 muestra la señal original cuadrada y la señal diferenciada implementada en GNU Radio.

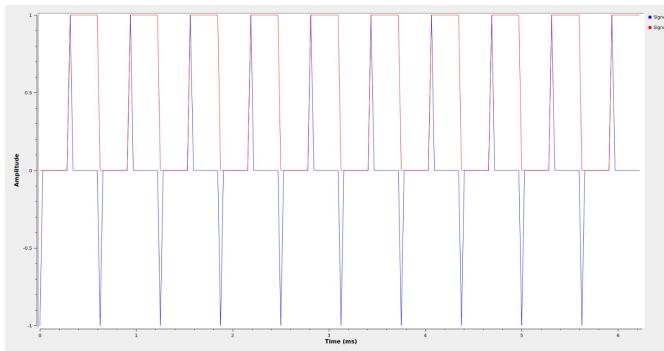


Figura 5. Resultado implementando el diferenciador para una señal cuadrada.

Por otro lado, al utilizar una señal senoidal como entrada, se obtuvo una señal cosenoidal como salida, lo cual coincide con el resultado matemático esperado:

$$\frac{d}{dt} \sin(\omega t) = \omega \cos(\omega t) \quad (6)$$

Se evidenció además que la señal diferenciada presenta un desfase de $\pi/2$ radianes respecto a la señal original, lo cual es característico del proceso de derivación en funciones trigonométricas. Asimismo, la amplitud de la señal resultante depende de la frecuencia angular ω y de la frecuencia de muestreo utilizada en la implementación discreta.

La Figura 6 muestra la señal original seno y la señal diferenciada implementada en GNU Radio.

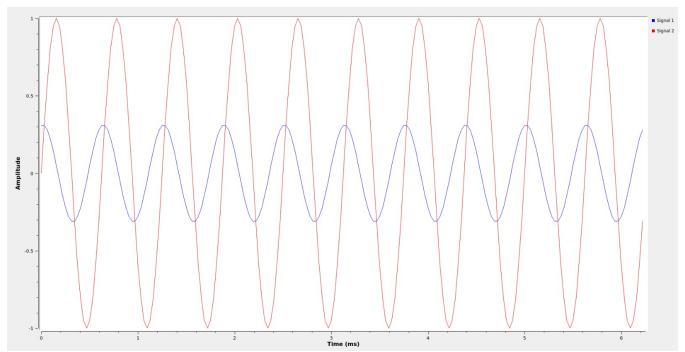


Figura 6. Resultado implementando el diferenciador para una señal seno.

Es importante mencionar que el diferenciador en tiempo discreto se implementa mediante una aproximación basada en diferencias entre muestras consecutivas. Debido a este proceso de discretización, pueden presentarse pequeñas variaciones en amplitud o ligeras irregularidades cuando la frecuencia de la señal de entrada aumenta o cuando la frecuencia de muestreo no es suficientemente alta.

En general, los resultados experimentales obtenidos confirman el comportamiento teórico de un diferenciador digital. La prueba con señal cuadrada permitió verificar la detección de cambios abruptos, mientras que la prueba con señal senoidal validó la relación matemática entre una función y su derivada. Esto demuestra que el bloque implementado cumple correctamente la función esperada dentro del entorno GNU Radio.

III. CONCLUSIONES

- El análisis del filtro estadístico de media móvil mostró que la selección del tamaño de la ventana de promediado es muy importante en el diseño. Si bien aumentar la cantidad de muestras atenua de manera más eficiente la varianza del ruido Gaussiano, esto introduce inevitablemente un retardo temporal en la salida, lo cual representa un parámetro restrictivo en sistemas que operan en tiempo real.

- El acondicionamiento digital de señales mioelectrocas se confirma como una etapa crucial para el diseño y validación de sistemas de control de protesis electromecánicas. La estabilización de estas señales, mediante la reducción de ruido estadístico, es lo que garantiza que los actuadores mecánicos reciban comandos limpios, evitando oscilaciones o respuestas erráticas en la prótesis.

IV. REFERENCIAS

REFERENCIAS

- [1] A. V. Oppenheim y R. W. Schafer, *Tratamiento de señales en tiempo discreto*, 3^{ra} ed. Madrid, España: Pearson Educación, 2011.
- [2] A. M. Wyglinski y D. Okin, *Software-Defined Radio for Engineers*. Norwood, MA, USA: Artech House, 2018.
- [3] GNU Radio Project, “Embedded Python Block,” *GNU Radio Wiki*, 2024. [En línea]. Disponible en: https://wiki.gnuradio.org/index.php/Embedded_Python_Block
- [4] M. B. I. Reaz, M. S. Hussain, y F. Mohd-Yasin, “Techniques of EMG signal analysis: detection, processing, classification and applications,” *Biological Procedures Online*, vol. 8, pp. 11-35, 2006.