Práctica 1: Acondicionamiento de Señales y GNU Radio

1er juan david camacho gonzalez
Código: 2210428
Universidad Industrial de Santander
Bucaramanga, Colombia

1er Jordy Pabon Carrillo
Código: 2210397
Universidad Industrial de Santander
Bucaramanga, Colombia

2do 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 soluionar 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

El filtrado 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 diseñó un bloque de procesamiento estadístico personalizado en GNU Radio mediante un *Embedded Python Block*. La estrategia seleccionada correspondió a un filtro discreto de media móvil, regido por la siguiente ecuación de diferencias [1]:

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

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

II-A1. Validación y Aplicación Práctica: La capacidad de programar filtros a medida fue un requerimiento crítico evaluado para aplicaciones de alta complejidad, tales como el diseño y validación de un sistema de control de fuerza en la reproducción de movimientos básicos de la mano [4]. En este contexto, dado que las bioseñales crudas presentan una alta varianza que desestabiliza los actuadores mecánicos,

la aplicación de un promedio estadístico resultó ser un paso previo obligatorio para garantizar un control suave.

Para validar el bloque, la señal original se sometió a una fuente de ruido Gaussiano. El desempeño del filtro se sintetiza en la Tabla I.

Cuadro I: Síntesis de validación del filtro de media móvil.

Condición	Efecto en la Salida	Conclusión
Ruido Gaussiano	Atenuación de picos de alta frecuencia.	Mejoró la relación SNR estabilizando la señal base.
Ventana corta ($N = 10$)	Leve rizado residual.	Minimizó la latencia computacional, vital para el control en tiempo real.

Al ejecutar el sistema, los resultados obtenidos corroboraron la eficacia del bloque. Tal como se observó en la Fig. 1, la estadística aplicada atenuó significativamente las variaciones aleatorias. Aunque persistió un rizado debido al tamaño reducido de la ventana, este diseño garantizó la baja latencia exigida para la reproducción instantánea de movimientos, logrando un equilibrio óptimo entre suavidad de la señal y velocidad de respuesta.

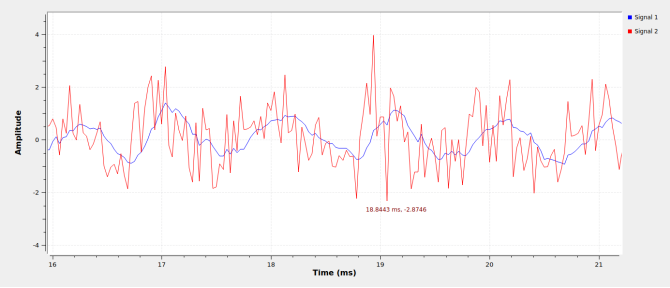


Figura 1: Comparación temporal entre la señal contaminada con ruido Gaussiano y la salida estabilizada por el filtro.

II-B. Acumulador discreto con memoria

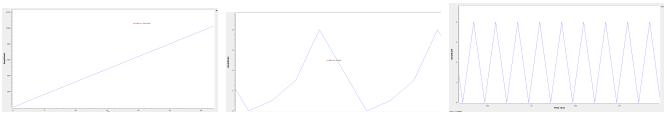
El acumulador discreto ideal, definido recursivamente como $y[n] = y[n-1]+x[n]$, requirió adaptación para su implementación en GNU Radio debido a su arquitectura de procesamiento por bloques de tamaño N. Dado que computar únicamente la suma acumulada local reiniciaba el valor en cada iteración de la función `work()`, se diseñó un *Embedded Python Block* para preservar el estado del sistema. Matemáticamente, la salida del bloque se definió como $\mathbf{y} = A + \text{cumsum}(\mathbf{x})$, donde el estado interno se actualizó guardando el último valor procesado ($A \leftarrow y_{N-1}$) como condición inicial del siguiente bloque.

II-B1. Validación del bloque: El comportamiento del sistema se evaluó mediante un flujograma con un *Vector Source* cíclico acoplado al bloque acumulador. Para comprobar la continuidad temporal, se ejecutaron tres pruebas fundamentales, cuyos resultados y conclusiones se sintetizan en la Tabla II.

Las formas de onda resultantes de estas tres pruebas se graficaron en la Fig. 2. Tal como se observó en las gráficas, el sistema mantuvo la coherencia del estado interno a lo largo del tiempo sin importar el tipo de entrada, lo que permitió concluir que la memoria compartida entre los bloques de GNU Radio operó correctamente sin pérdidas de datos.

Cuadro II: Síntesis de pruebas de validación del acumulador con memoria.

Señal de Entrada	Forma de Salida	Conclusión del Comportamiento
Constante ($x[n] = 1$)	Rampa creciente.	Evidenció la deriva temporal del sistema ante señales con componente DC.
Media cero ($[1, 2, 5, -4, -4]$)	Señal acotada.	Validó la conservación de la memoria en régimen continuo sin desbordamientos.
Bipolar (± 1)	Onda triangular.	Comprobó el comportamiento del bloque como un integrador discreto ideal.



(a) Componente DC. (b) Media cero. (c) Bipolar (± 1).

Figura 2: Respuesta del acumulador con memoria ante diversas secuencias cíclicas.

II-C. Diferenciador Discreto

El diferenciador discreto aproxima la derivada temporal de una señal midiendo la tasa de cambio entre muestras consecutivas. Para su implementación en GNU Radio, se programó un *Embedded Python Block* basado en la ecuación de diferencias finitas $y[n] = x[n] - x[n-1]$. El sistema se configuró como un `sync_block` de tipo `float32`, el cual requirió almacenar el valor de la muestra previa ($x[n-1]$) en la memoria interna del bloque para mantener la continuidad matemática durante el procesamiento por vectores que realiza el software.

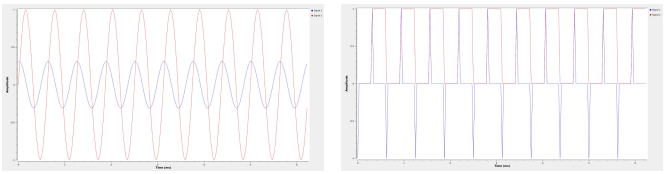
II-C1. Validación y Aplicación Práctica: El comportamiento del bloque se evaluó estimulando el sistema con un *Signal Source* y visualizando el contraste entre la señal original y la derivada en un sumidero temporal. Se ejecutaron dos pruebas fundamentales para analizar la respuesta ante variaciones suaves y cambios abruptos. Los resultados teóricos y prácticos se sintetizan en la Tabla III.

Cuadro III: Síntesis de pruebas de validación del diferenciador discreto.

Señal de Entrada	Forma de Salida	Conclusión del Comportamiento
Senoidal	Cosenoidal.	Evidenció el desfase teórico de $\pi/2$ radianes, validando la derivación en señales trigonométricas continuas.
Cuadrada	Tren de impulsos.	Actuó como detector de flancos, generando picos en las transiciones y valores nulos en los tramos constantes.

El comportamiento descrito se corroboró visualmente en las gráficas de la Fig. 3. Tal como se observó, la salida

del sistema respondió fielmente a las transiciones de las señales de entrada. En la prueba de la onda cuadrada, los picos de amplitud confirmaron la elevada pendiente en los puntos de cambio brusco. Por otro lado, la prueba senoidal demostró la dependencia de la amplitud resultante frente a la frecuencia angular. Adicionalmente, se concluyó que, debido a la naturaleza de la discretización por diferencias finitas, la fidelidad de la aproximación dependió fuertemente de contar con una frecuencia de muestreo suficientemente alta para evitar irregularidades.



(a) Prueba con señal senoidal. (b) Prueba con señal cuadrada.

Figura 3: Respuesta del diferenciador discreto ante señales de entrada continua y abrupta.

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 atenúa 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.
- A partir de los resultados se concluye que el acumulador implementado reproduce el comportamiento de una **integración discreta** en ejecución continua. Cuando la entrada tiene una componente DC (media distinta de cero), el acumulado presenta **deriva** y crece aproximadamente de forma lineal (Prueba 1). Cuando la entrada tiene **media cero**, el acumulado se mantiene **acotado** y refleja únicamente la estructura de la suma parcial (Prueba 2). Para una entrada cuadrada bipolar (± 1), la salida presenta un comportamiento aproximadamente **triangular**, coherente con la integración discreta por tramos (Prueba 3). Finalmente, la inclusión de la memoria (`acum_anterior`) resulta esencial para garantizar continuidad entre llamadas a `work()` y evitar reinicios del acumulado debido al procesamiento por bloques.
- El acondicionamiento digital de señales mioelectrocas se confirma como una etapa crucial para el diseño y validación de sistemas de control de protesis electromecanicas. 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.
- 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.

IV. Referencias

Referencias

[1] A. V. Oppenheim y R. W. Schafer, *Tratamiento de señales en tiempo discreto*, 3ra 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.