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

JUAN SEBASTIAN AVILA DIAZ - 2200524
JENNIFER TATIANA CHAPARRO - 2191468
WILLIAM ANDRES PARRA RUEDA - 2184680

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

26 de Febrero de 2024

Resumen

En este informe se abordará el tema trabajado en el laboratorio de comunicaciones enfocado en la programación de bloques en GNU Radio utilizados para realizar cálculos estadísticos en las señales de entrada a los bloques programados. ***Palabras clave: Python, GNU Radio, Estadística, Bloques*** .

1. Introducción

La utilización de GNU Radio proporciona una plataforma robusta para el procesamiento de señales en la línea de comunicaciones, gracias a que permite la creación de bloques personalizados mediante el lenguaje de programación Python, los cuales se adaptan a diversas aplicaciones según los requisitos específicos de cada una. Con la integración de GitHub, simplifica el trabajo en grupo en proyectos, ya que permite a los desarrolladores almacenar, compartir y colaborar en proyectos de software.

2. Marco teórico

- Bloque acumulador:
  El bloque acumulador en GNU Radio es una componente con una aplicación significativa en el procesamiento de señales. Su función principal consiste en acumular o sumar valores de muestras de una señal o datos a lo largo del tiempo. Su propósito es mantener un registro de la suma de los valores de entrada, y puede ser utilizado en diversas aplicaciones como procesamiento de señales, demodulación, análisis de potencia, entre otras.

- Bloque diferenciador:
  El bloque diferenciador es una herramienta valiosa en el procesamiento de señales, ya que se utiliza para realizar la operación de derivación con respecto al tiempo. Esto implica calcular la tasa de cambio de una señal con respecto al tiempo. Puede emplearse para llevar a cabo demodulaciones de señales, detectando cambios rápidos o abruptos en la información transmitida, así como para identificar picos en una señal.

La señal senoidal posee características estadísticas llamadas promedios de tiempo, estos se conocen como la media, la media cuadrática, la varianza, la desviación estándar y la función de distribución de probabilidad de la función de entrada. Estas medidas en términos de instantes de tiempo se toman como promedios. Para poder sacar estos valores por medio del software, se deben implementar operaciones como la acumulación y el diferenciador.

A continuación se describen las fórmulas correspondientes para cada uno de estos procedimientos tomando x(t) como señal de entrada:

La media de una señal x(t) es:

$$X_m =< x(t) > \tag{1}$$

La media cuadrática de una señal de entrada es:

$$X_c =< x^2(t) > \tag{2}$$

La varianza de una señal de entrada x(t) es:

$$\sigma_x^2 =< |x(t) - X_m|^2 > \tag{3}$$

La desviación estándar de una señal de entrada x(t) es:

$$\sigma_x =< \sqrt{|x(t) - X_m|^2} > \quad (4)$$

El valor RMS de una señal de entrada x(t):

$$X_{RMS} =< |x(t)|^2 > \quad (5)$$

En el procedimiento se describe la manera en la que se construyeron estos bloques para hacer uso de los promedios de tiempo.

3. Procedimiento

La práctica comenzó configurando el GitHub con la terminal de Linux del computador de laboratorio. Se crearon las respectivas ramas y subcarpetas.

Para poder implementar los bloques acumulador y diferenciador por medio de la programación en GNU Radio, se utilizó el libro guía en la sección 1.2.0.1 del cuál se extrejeron los respectivos códigos de python con el algoritmo para armar los bloques y poder utilizarlos en las operaciones de promedio de tiempo.[1]

Se utilizó el código de la figura 1 para implementar el bloque de acumulador.

```
"""
Embedded Python Blocks:

Each time this file is saved, GRC will instantiate the first class it finds
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""

import numpy as np
from gnuradio import gr


class blk(gr.sync_block):  # other base classes are basic_block, decim_block, interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self):  # only default arguments here
        """arguments to this function show up as parameters in GRC"""
        gr.sync_block.__init__(
            self,
            name="bloque acumulador",  # will show up in GRC
            in_sig=[np.float32],
            out_sig=[np.float32]
        )

    def work(self, input_items, output_items):
        x = input_items[0]
        y = output_items[0]

        y[:] = np.cumsum(x)
        return len(y)
```

Figura 1. Programación del acumulador en GNU Radio

Después de generar el bloque del acumulador se ingresó un vector de N elementos al bloque, por medio del siguiente diagrama mostrado en la figura 2

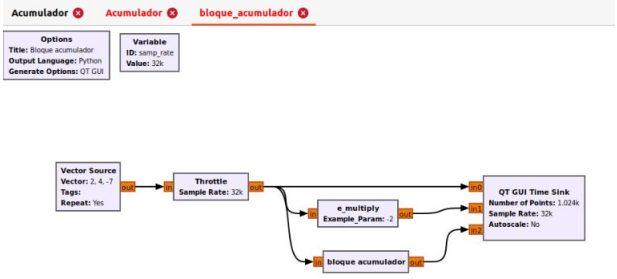

Figura 2. Diagrama de bloques acumulador

Luego, se procedió a implementar el bloque diferenciador por medio del código mostrado en la figura 3, extraído del libro guía.

```
import numpy as np
from gnuradio import gr

class blk(gr.sync_block):

    def __init__(self):

        gr.sync_block.__init__(
            self,
            name="Bloque_Diferenciador",
            in_sig=[np.float32],
            out_sig=[np.float32]

        )
        self.acum_anterior = 0
    def work(self, input_items, output_items):
        x = input_items[0]
        y = output_items[0]

        N = len(x)
        diff = np.cumsum(x) - self.acum_anterior
        self.acum_anterior = diff[N-1]
        y[:] = diff
        return len(y)n
```

Figura 3. Programación del diferenciador en GNU Radio

Para comprobar el correcto funcionamiento del bloque diferenciador, se propuso un diagrama de bloques. (Figura 4)

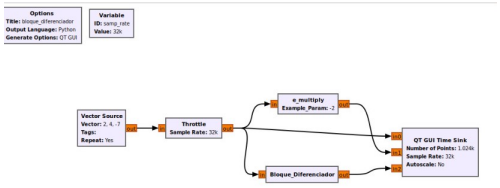

Figura 4. Diagrama de bloques diferenciador

Como se mencionó anteriormente, las señales de entrada tienen ciertas características estadísticas importantes. Ya teniendo en funcionamiento el bloque acumulador y el bloque diferenciador se podrán calcular con facilidad estas medidas estadísticas al igual que con los bloques anteriormente mencionados, la programación de estos bloques de medida estadística, se encuentran en el libro guía.

Al terminar de realizar los esquemas y tomar las respectivas pruebas, se agrupó la información en una carpeta la cuál fue subida desde el terminal de Linux a la rama "Práctica_1.en GitHub.

4. Análisis de Resultados

Para ver mejor el resultado de la implementación de los bloques, la señal de entrada para cada uno de ellos, es un vector de 3 elementos, al cuál se le calculan las 5 medidas estadísticas mencionadas en el marco teórico con sus respectivas fórmulas.

El resultado de simular el bloque acumulador con una señal de entrada x(t) = [2,4,-7] se muestra en la figura 5.

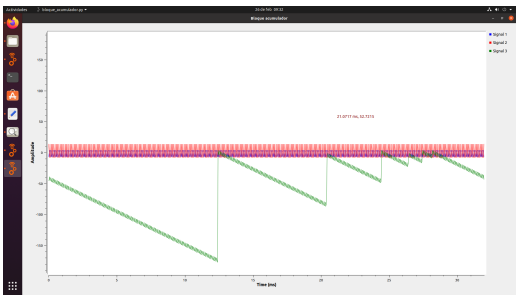

Figura 5. Resultado de simulación de acumulador

El resultado de simular el bloque diferenciador con una señal de entrada x(t) = [2,4,-7], se muestra en la figura 6.

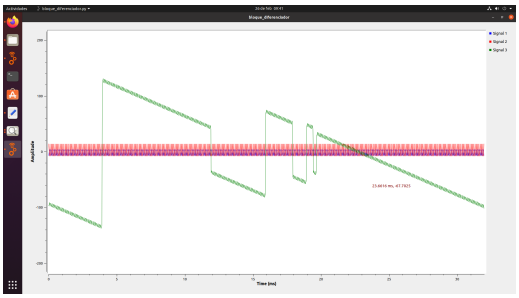

Figura 6. Resultado de simulación de diferenciador

EL resultado del promedio de una señal de entrada x(t) = [1,2,3] es de 2. (Figura 7).

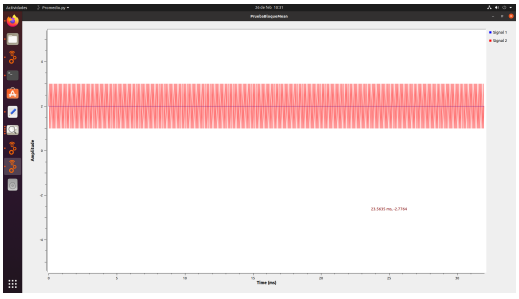

Figura 7. Resultado del promedio de la señal de entrada

El resultado de la media cuadrática para la señal de entrada x(t)=[1,2,3] se muestra en la figura 8.

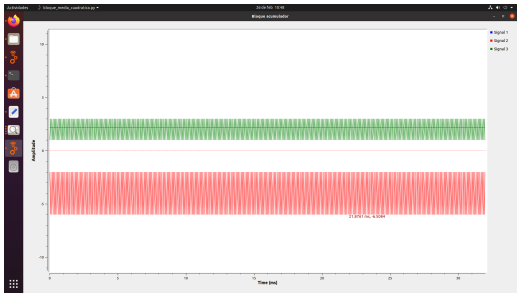

Figura 8. Resultado de la media cuadrática de la señal de entrada

El resultado de la varianza para la señal de entrada x(t)=[1,0,1] es de 0.125. (Figura 9).

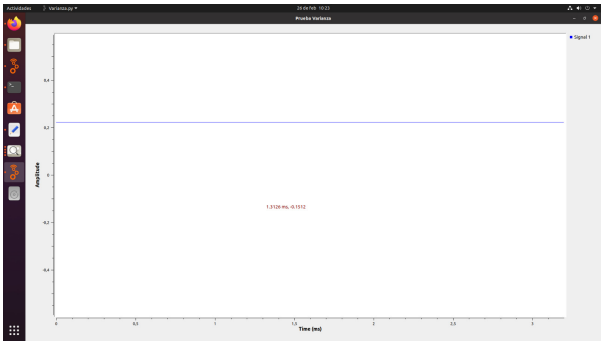

Figura 9. Resultado de la varianza de la señal de entrada

El resultado de la desviación estándar para la señal de entrada x(t)=[1,0,1] es de 0.48. (Figura 10).

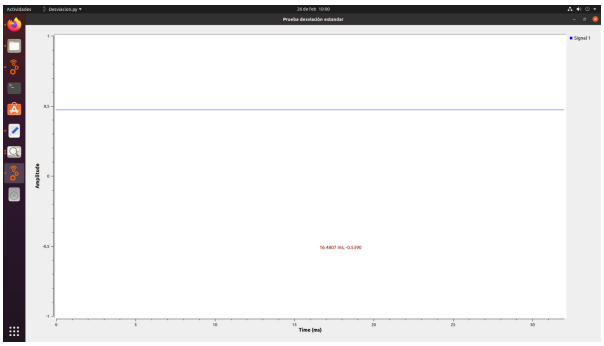

Figura 10. Resultado de la desviación estándar de la señal de entrada

En la figura 11 se muestra la señal coseno en rojo y su valor RMS en azul, siendo este 0.707.

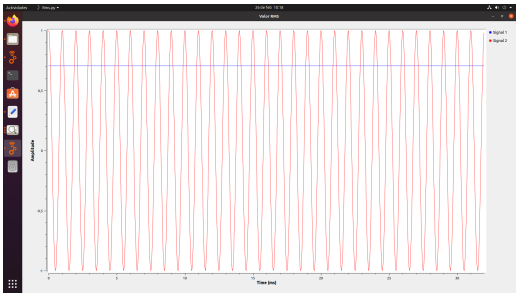

Figura 11. Valor RMS de la señal de entrada

Después de desarrollar los pasos recomendados para la practica, se puede notar como se obtuvieron los resultados que deseaban tanto para el bloque diferenciador como para el acumulador(lo anterior se aprecia en la figura 5 y 6 respectivamente). Como parte final se nos pedía construir ciertos diagramas de bloques que cumplieran la función de realizar cálculos de medidas estadísticas que se habían visto previamente en clase, para esto se uso la opción del bloque Python que proporciona GNURADIO[2].

Antes de realizar un analisis resulta muy importante aclarar que las medidas estadísticas que se calcularon fueron el promedio, la media cuadrada, varianza, valor RMS y desviación estándar. Para comprobar su correcto funcionamiento se agrego a la entrada de los bloques creados un vector de números a excepción del valor RMS para el cual fue mas conveniente poner una sinusoidal. Los resultados de las medidas anteriormente mencionadas se pueden apreciar en las figuras 7,8,9,10 y 11 y de estos podemos concluir que son los que se deseaban, ya que si los comparamos con los que se debían obtener con las formulas presentadas en el marco teórico encontramos que no existe ningún margen de error.

5. Conclusiones

- GNU Radio es una herramienta flexible y de código abierto que permite desarrollar y simular sistemas, facilitando la implementación de esquemas mediante bloques, los cuales pueden ser definidos y programados.
- La implementación de los bloques acumulador y diferenciador en GNU Radio ayuda a comprender conceptos aplicados a las comunicaciones digitales y algunos conceptos vistos en clase.
- La implementación de un bloque para mostrar las estadísticas de las señales es una herramienta necesaria al realizar análisis en tiempo real; esto ayuda a entender las tendencias y el comportamiento de las señales.

Referencias

[1] U. I. d. S. Homero Ortega, Oscar Reyes, “Comunicaciones digitales basadas en radio definida por software,” p. 19, 2019.

[2] “Embedded Python Block - GNU Radio.” [Online]. Available: https://wiki.gnuradio.org/index.php/Embedded_Python_Block