Práctica 1: SDR y GNURADIO

CAMILA ANDREA BELEÑO CABRALES - 2204280
DENILSON ALFONSO CARRASCAL PIÑERES - 2190400
LUIS FERNANDO ROMERO ROJAS - 2191663

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

8 de septiembre de 2024

Repositorio GitHub: `https://github.com/CamilaBeleno/ComunicationsII_2024_2_B1G3.git`

Resumen

In the first laboratory session, the fundamentals of block creation using the "GNU Radio"software, developed in Python, were explored. This software facilitates the exploration of new practical applications based on concepts learned in class, thereby promoting the development of skills in analysis, design, and the learning of advanced tools in the field of communications.
Palabras clave: Promedio, ruido, GNU Radio, SDR.

1. Introducción

La radio definida por software (SDR) es un sistema de radiocomunicaciones que a diferencia de las radios tradicionales, que dependen de hardware especializado para cada función, una SDR utiliza software para realizar muchas de las tareas que normalmente se realizarían mediante componentes electrónicos dedicados [1], de ese modo, un mismo dispositivo puede emplearse para múltiples tareas de AM, FM y CW, entre otros, obteniendo una radio flexible y reconfigurable [2]. Si bien el concepto de SDR no es nuevo, las capacidades de rápida evolución de la electrónica digital hacen prácticos muchos procesos que antes solo eran teóricamente posibles y siguen una tendencia hacia la posibilidad de ofrecer conexiones con velocidades crecientes para transmisión de información [3].

En esta práctica, se exploró el potencial de GNU Radio, un software diseñado para simular sistemas de comunicación mediante diagramas de bloques [4]. El enfoque principal fue la interacción con el software y la implementación de bloques fundamentales, como el acumulador y el diferenciador, utilizando el bloque Python. Este bloque es especialmente relevante porque permite la programación personalizada de sus funcionalidades a través de un editor de Python, lo que proporciona una gran flexibilidad y control en el diseño del sistema.

Adicionalmente, para optimizar la colaboración y el seguimiento del proyecto, se implementó una metodología que incluyó la creación de una rama específica en el repositorio de GitHub. Esta rama sirve para documentar y actualizar cada cambio realizado durante la implementación. La revisión y confirmación de estos cambios se realiza de manera eficiente mediante la interfaz web de GitHub, asegurando así una gestión y control adecuados del desarrollo del proyecto.

2. Procedimiento

Para ejecutar toda la practica se realizaron trabajos previos relacionados a github el cual es la herramienta fundamental para el trabajo cooperativo y plataforma optima como nube para guardar los avances de cada practica.

1. Primeramente en la práctica se aplicaron dos bloques de python, el primer bloque que se aplicó fue un acumulador en donde se añadió un código obtenido de un libro suministrado en el documento guía de la práctica, así mismo se realizó el bloque diferenciador.

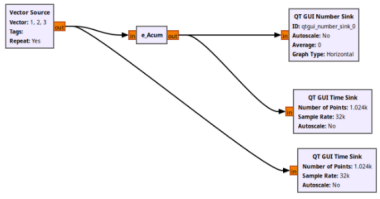

Figura 1. Implementación Bloque Acumulador

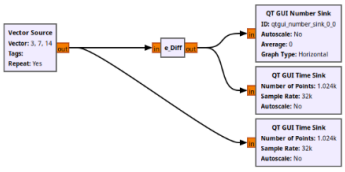

Figura 2. Implementación Bloque Diferenciador

2. Luego se añadió el código obtenido por el libro, teniendo en cuenta que éste tenía algunos errores y en equipo se le hicieron ciertas correcciones al código de tal manera que este funcionará acorde a como se quería.

```
import numpy as np
from gnuradio import gr
class blk(gr.sync_block):
    def __init__(self):
        gr.sync_block.__init__(
            self,
            name='e_Acum',
            in_sig=[np.float32],
            out_sig=[np.float32]
        )
    def work(self, input_items, output_items):
        x = input_items[0] # Senial de entrada
        y0 = output_items[0] # Senial acumulada
        limited_x = x[:3]
        y0 [:3] = np.cumsum(limited_x)
        return len(y0[:3])
```

Figura 3. Codigo - Bloque Acumulador

```
import numpy as np
from gnuradio import gr

class blk (gr.sync_block):

    def __init__(self):
        gr.sync_block.__init__(
            self,
            name ='e_Diff',
            in_sig=[np.float32],
            out_sig=[np.float32]
        )
        self.acum_anterior = 0
    def work (self, input_items, output_items):
        x = input_items[0]
        y0 = output_items[0]
        limited_x = x[:3]
        N = len(limited_x)

        diff = np.cumsum (limited_x) - self.acum_anterior
        self.acum_anterior = diff[N-2]
        y0[:3] = diff

        return len (y0[:3])
```

Figura 4. Codigo - Bloque Diferenciador

- Correcciones al bloque acumulador y diferenciador:
  En ambos codigos se ajusto el tamaño del vector de entrada ya que GNU hace un bucle con los datos del vector aproximadamente 8000 datos, al momento de acumular o derivar no tomaba solo los datos deseados si no todos los datos de ese bucle, en nuestra implementacion se limito el codigo a una entrada vector de 3 posiciones (limited_x =x[:3]).

3. Posteriormente se implemento un bloque titulado promediador el cual tenia como objetivo mirar la estadisticas vidualizada en clase.

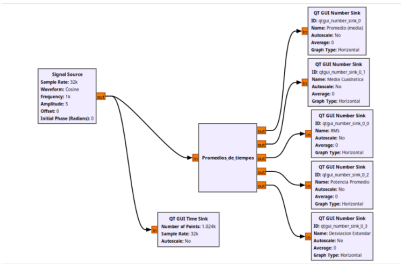

Figura 5. Implementación Bloque Promediador

```
import numpy as np
from gnuradio import gr
class blk (gr.sync_block ):

    def __init__ (self): # only default arguments here
        gr.sync_block.__init__ (
        self,
        name ='Promedios_de_tiempos', # will show up in GRC
        in_sig =[np.float32],
        out_sig =[ np.float32,np.float32,np.float32,np.float32,np.float32]
        )
        self.acum_anterior = 0
        self.Ntotales = 0
        self.Ntotales3 = 0
        self.acum_anterior1 = 0
        self.acum_anterior2 = 0
        self.acum_anterior3 = 0

    def work (self, input_items, output_items ):
        x = input_items [0] # Senial de entrada .
        y0 = output_items [0] # Promedio de la senial
        y1 = output_items [1] # Media de la senial
        y2 = output_items [2] # RMS de la senial
        y3 = output_items [3] # Potencia promedio de la senial
        y4 = output_items [4] # Desviacion estander de la senial

        # Calculo del promedio
        N = len (x)
        self.Ntotales = self.Ntotales + N
        acumulado = self.acum_anterior + np.cumsum (x)
        self.acum_anterior = acumulado [N -1]
        y0 [:]= acumulado / self.Ntotales

        # Calculo de la media cuadratica
        x2=np. multiply (x,x)
        acumulado1 = self.acum_anterior1 + np.cumsum (x2)
        self.acum_anterior1 = acumulado1 [N -1]
        y1 [:] = acumulado1 / self.Ntotales

        # Calculo de la potencia promedio
        # y3 [:] = np.multiply (y2,y2) esta mal

        # Calculo de la potencia promedio
        x4= np.multiply (x,x)
        self.Ntotales3 = self.Ntotales3 + N
        acumulado3 = self.acum_anterior3 + np.cumsum (x4)
        self.acum_anterior3 = acumulado3 [N-1]
        y3[:]= acumulado3 / self.Ntotales3

        # Calculo de la RMS
        y2 [:] = np.sqrt (y3)

        # Calculo de la desviacion estandar
        x3 = np.multiply (x-y0 ,x-y0)
        acumulado2 = self.acum_anterior2 + np.cumsum (x3)
        self.acum_anterior2 = acumulado2 [N -1]
        y4 [:] = np.sqrt ( acumulado2 / self.Ntotales )

        return len (x)
```

Figura 6. Codigo - Bloque Promediador

4. Despues de lo anterior de manera grupal se realizó una aplicación en donde a los bloques se le adicionaron ruido por medio del bloque de GNURadio "Noise Source τ de esta manera entender en comportamiento de ambos bloques y así mismo del bloque promediador.

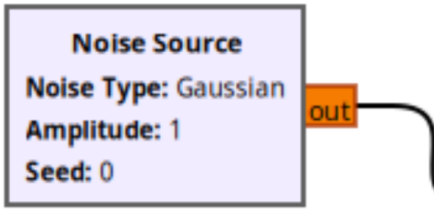

Figura 7. Implementación Bloque de Ruido

5. Finalmente se realizó una compracion del comportameinto de los bloques cuando se encontraban con presencia de ruido y cuando no, logrando así obtener resultados parera dar un analisís mas exacto referente a los comportamientos de cada uno de los bloques y su funcion en la aplicación seleccionada.

3. Resultados

Después de implementar los códigos en los bloques de Python y crear los diagramas de bloques correspondientes, se verificó que los resultados fueran los esperados al observar las gráficas de la simulación.

Para probar el bloque acumulador, se utilizó como entrada un vector de magnitud 3 con el contenido [1, 2, 3]. Se esperaba que la salida mostrara la acumulación progresiva de estos tres valores, comenzando con un impulso de magnitud 1, seguido por otro de magnitud 1+2=3, y finalmente un impulso de 1+2+3=6. Al analizar los resultados obtenidos en la Figura [8], se pudo constatar que los valores eran correctos, lo que confirma que el bloque acumulador funciona de manera adecuada.

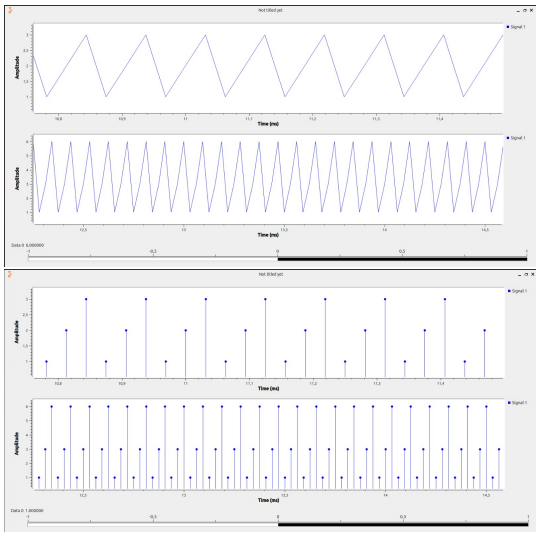

Figura 8. Resultados acumulador

Para probar el bloque diferenciador, se utilizó un vector de entrada de magnitud 3, con los valores [3, 7, 14]. Se espera que la salida refleje la acumulación de estos tres números menos la entrada seleccionada del bloque acumulador. En este caso, se obtendrán 6 impulsos.En el primer impulso, el acumulador se encuentra en 0, por lo que la salida será la acumulación directa de los números de entrada: primero 3, luego 3 + 7 = 10, y finalmente 3 + 7 + 14 = 24. Posteriormente, el bloque acumulador adopta el valor de la posición N-2 de la acumulación anterior, que en este caso es 10. A partir de aquí, los resultados de salida son los siguientes: 3 - 10 = -7, luego 3 + 7 - 10 = 0, y finalmente 3 + 7 + 14 - 10 = 14.Al verificar estos resultados en la Figura [9], se confirma que son los esperados, lo que demuestra que el bloque diferenciador está funcionando correctamente.

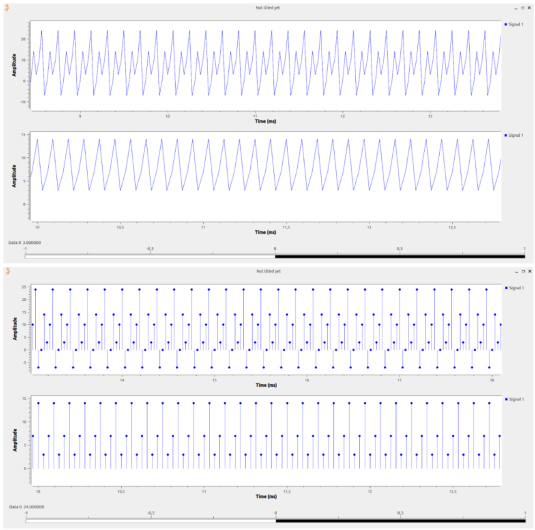

Figura 9. Resultados diferenciador

Para probar el correcto funcionamiento del bloque promediador se uso una entrada coseno de magnitud 5. Donde se espera que los resultados sean:

La potencia promedio de una señal $f(t) = 5\cos(\omega t)$ es:

$$P_{\text{avg}} = \frac{1}{T}\int_0^T f^2(t)\,dt$$

Sustituyendo $f(t) = 5\cos(\omega t)$:

$$P_{\text{avg}} = \frac{1}{T}\int_0^T 25\cos^2(\omega t)\,dt = 25 \cdot \frac{1}{2} = 12{,}5$$

La desviación estándar se calcula como:

$$\sigma = \sqrt{\frac{1}{T}\int_0^T (f(t)-\mu)^2\,dt}$$

Dado que $\mu = 0$ (la media de $f(t) = 5\cos(\omega t)$ es cero):

$$\sigma = \sqrt{12{,}5} \approx 3{,}54$$

La media cuadrática o RMS se define como:

$$f_{\text{rms}} = \sqrt{\frac{1}{T}\int_0^T f^2(t)\,dt}$$

Esto es equivalente a la raíz de la potencia promedio:

$$f_{\text{rms}} = \sqrt{12{,}5} \approx 3{,}54$$

El valor promedio de la función $f(t) = 5\cos(\omega t)$ es:

$$f_{\text{avg}} = \frac{1}{T}\int_0^T 5\cos(\omega t)\,dt = 0$$

Al verificar estos resultados en la Figura [10], se confirma que son los esperados, lo que demuestra que el bloque diferenciador está funcionando correctamente.

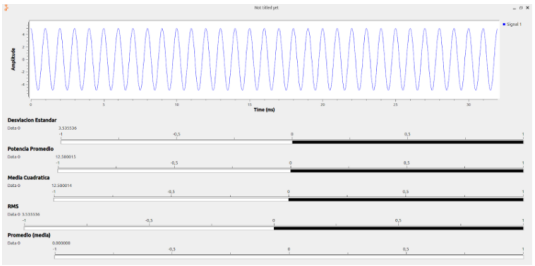

Figura 10. Resultados fuente coseno sin ruido

Finalmente para realizar una aplicación al bloque diferenciador, se sumaron una señal seno de amplitud 5 y una señal de ruido gaussiano de amplitud 1, por medio de los resultados obtenidos del bloque diferenciador se puede identificar si una señal contiene ruido.

La potencia promedio de una señal $f(t) = 5\sin(\omega t) + n(t)$ donde $n(t)$ es un ruido gaussiano con desviación estándar 1, es:

$$P_{\text{avg}} = P_{\text{seno}} + P_{\text{ruido}}$$

Para la señal senoidal:

$$P_{\text{seno}} = \frac{1}{T}\int_0^T (5\sin(\omega t))^2\,dt = 25 \cdot \frac{1}{2} = 12{,}5$$

Para el ruido gaussiano:

$$P_{\text{ruido}} = 1^2 = 1$$

Entonces, la potencia promedio total es:

$$P_{\text{avg}} = 12{,}5 + 1 = 13{,}5$$

La desviación estándar de la señal es:

$$\sigma = \sqrt{P_{\text{avg}}} = \sqrt{13{,}5} \approx 3{,}67$$

El valor RMS de la señal es igual a la desviación estándar:

$$f_{\text{rms}} = \sqrt{P_{\text{avg}}} = \sqrt{13{,}5} \approx 3{,}67$$

El valor promedio de la señal es:

$$f_{\text{avg}} = \frac{1}{T}\int_0^T (5\sin(\omega t) + n(t))\,dt = 0$$

A partir de los resultados obtenidos, podemos observar que la potencia promedio de una señal compuesta puede revelar información sobre la amplitud del ruido que se le suma. En el caso de una señal senoidal con amplitud 5, la potencia promedio es 12.5. Cuando se añade un ruido gaussiano con una desviación estándar de 1, la potencia promedio total de la señal se incrementa a 13.5.

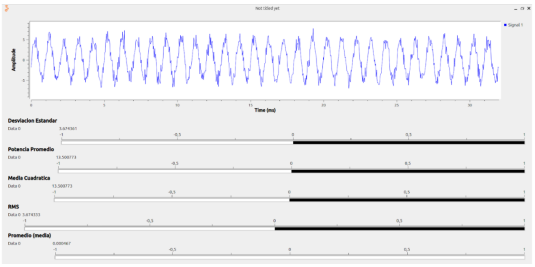

Figura 11. Resultados fuente seno con ruido

4. Conclusiones

- La integración del Bloque Python en GNU Radio no solo simplifica la implementación de sistemas de comunicación complejos, sino que también amplifica la capacidad para medir y analizar variables cruciales. Esta versatilidad no solo acelera el proceso de desarrollo, sino que también refuerza la habilidad del diseñador para tomar decisiones fundamentadas basadas en análisis detallados y precisos. De esta manera, se mejora significativamente la calidad y la efectividad del diseño de sistemas de comunicación.

Referencias

[1] Wikiwand, "Radio definida por software." [Online]. Available: https://www.wikiwand.com/es/articles/Radio_definida_por_software

[2] K. C. P. L. F. G. I. Carlos Vicente Niño Rondón, Wilfrey Andrés Contreras Gómez, "Radio definida por software: Una mirada a las tendencias y aplicaciones," *Ingenierías USBMed, vol. 14, n.◦ 1*, p. 58–69, 2023.

[3] AcademiaLab, "Radio definida por software." [Online]. Available: https://academia-lab.com/enciclopedia/radio-definida-por-software/

[4] A. Lab, "Radio gnu," 2024. [Online]. Available: https://academia-lab.com/enciclopedia/radio-gnu/