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

José Gabriel Candamil Téllez- 2215587

Juan Camilo Romero González -2205606

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

7 de septiembre de 2024.
https://github.com/camiloromero01/CommunicationsII_2024_2_B1

Abstract

In this report, the knowledge acquired in class will be applied to identify the fundamental aspects of real-time systems. Likewise, the implementation process of statistics applied to real signals using GNU Radio will be detailed with a specific focus on the development of accumulator and differentiator blocks.

1. Introducción

En el análisis de señales, es fundamental saber y comprender que GNU Radio ofrece la posibilidad de programar nuestros propios algoritmos lo que abre un abanico de oportunidades y funcionalidades facilitando el estudio detallado de los sistemas de señales. En este informe se estudió el uso de los bloques acumulador, diferenciador y promedio para comprender mejor como estos pueden ser empleados en aplicaciones como lo es eliminar el ruido Gaussiano presente en una señal.

2. Metodología

El desarrollo de la práctica de laboratorio se llevo a cabo en tres fases metodológicas planteadas de la siguiente manera:

En la primera fase metodológica se generó un vector constante [5, 2, 2] (Vector Source) para corroborar a partir de la implementación de funciones y desarrollo de los bloques diferenciador (e_Diff), acumulador (e_Acum) y promedio (e_Average), el comportamiento de los mismos.
En esta implementación se incluyeron bloques (QT GUI Number Sink) para poder visualizar los resultados obtenidos.

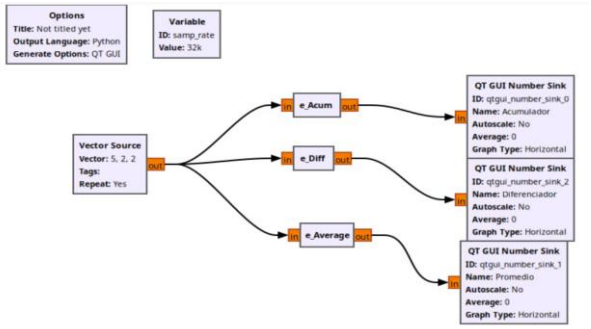

Fig. 1: Implementación del diagrama de bloques para la primera fase.

Con la ayuda de la programación de bloques en Python y tomando como referencia el libro guía se implementaron los códigos de los bloques estadísticos para el acumulador y el diferenciador, así mismo se creó el bloque para calcular el promedio.

```
Simport 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 ='e_Acum ', # will show up in GRC
         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
        y0 [:] = np. cumsum (x)
        return len (y0)
```

Fig. 2: Código para el bloque acumulador.

```python
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 ='e_Diff ', # will show up in GRC
            in_sig =[ np. float32 ],
            out_sig =[ np. float32 ]

        )
        self . acum_anterior = 0
    def work (self , input_items , output_items ):
        x = input_items [0] # Senial de entrada .
        y0 = output_items [0] # Senial acumulada diferencial
        N = len (x)
        diff = np. cumsum (x) - self . acum_anterior
        self . acum_anterior = diff [N -1]
        y0 [:] = diff
        return len (y0)
```

Fig. 3: Codigo bloque diferenciador.

```python
import numpy as np
from gnuradio import gr

class blk(gr.sync_block):
    def __init__(self):
        gr.sync_block.__init__(
            self,
            name='e_Average',
            in_sig=[np.float32],
            out_sig=[np.float32]
        )

    def work(self, input_items, output_items):
        x = input_items[0] # Señal de entrada.
        y0 = output_items[0] # Señal acumulada

# Calcula la suma acumulativa
        cumsum_x = np.cumsum(x)
        y0[:] = cumsum_x / (np.arange(len(x)) + 1)


        return len(output_items[0])
```

Fig. 4: Estructura de código para el bloque promedio.

Para la segunda fase metodológica se buscó recrear una señal más aproximada a la realidad, por ello se adiciono una fuente de ruido Gaussiano y se analizaron los cambios de las señales obtenidas con respecto a los resultados de la primera fase metodológica.

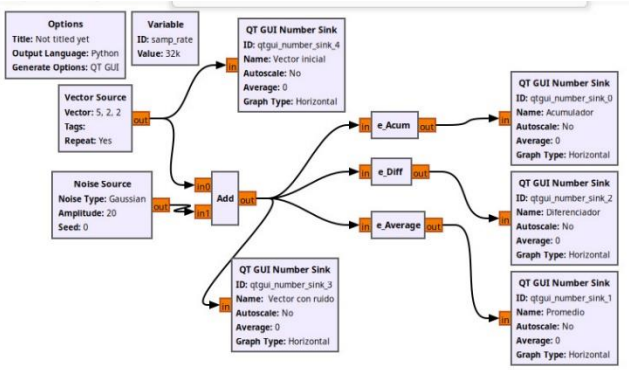

Fig. 5: Diagrama de bloques para la segunda fase.

Para la tercera fase metodológica se propuso una aplicación en donde a partir de una señal continua (tren de pulsos) con adición de ruido gaussiano permitiera representar una señal más acorde a la realidad, se buscó por medio de los bloques estadísticos eliminar el ruido presente planteando a la salida de la señal un integrador, aprovechando las características estadísticas del ruido gaussiano para reducirlo significativamente; luego se deriva la señal obtenida para recuperar la forma original de la señal generada en un principio.

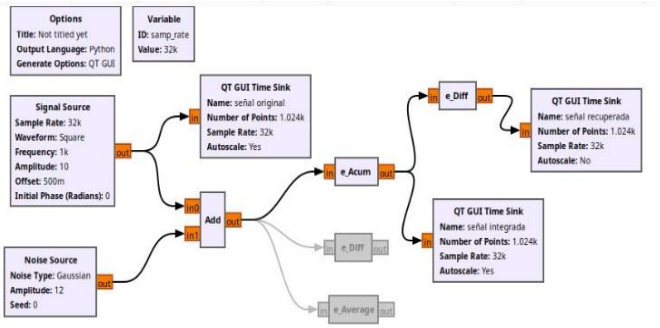

Fig. 6: Diagrama de bloques para la tercera fase.

3. Análisis de resultados

Acorde a lo planteado en la primera fase metodológica se realizó el diagrama de bloques como se ve en la Fig. 1, permitiendo observar que el bloque acumulador repite el vector de tres elementos múltiples veces llegando a manejar valores muy altos, alcanzando a obtener más de 8000 elementos.

En el caso del bloque diferenciador este detecta cambios bruscos en la señal y también se evidencia una gran variación en los valores que arroja GNU radio.

Para el caso del bloque promedio se obtuvieron los resultados esperados del promedio para el vector ingresado [5, 2, 2], donde se estimaba que fuera un valor de 3 unidades, concordando con lo obtenido en el software.

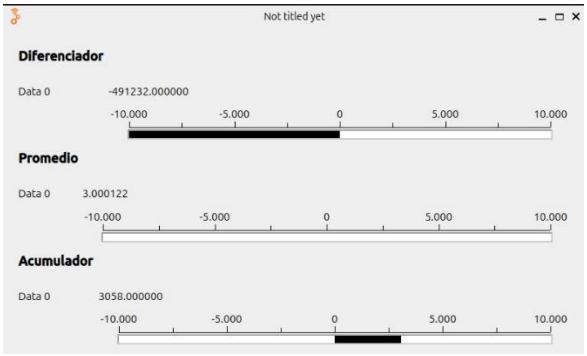

Fig. 7: Resultados obtenidos en la primera fase.

En la segunda fase se obtuvo que el ruido gaussiano distorsiona los valores del vector ingresado, al realizar la comparación con los resultados de la Fig.7 se observo que en el caso del acumulador y del diferenciador los resultados son significativamente mas grandes y para el promedio se obtuvo una leve variación.

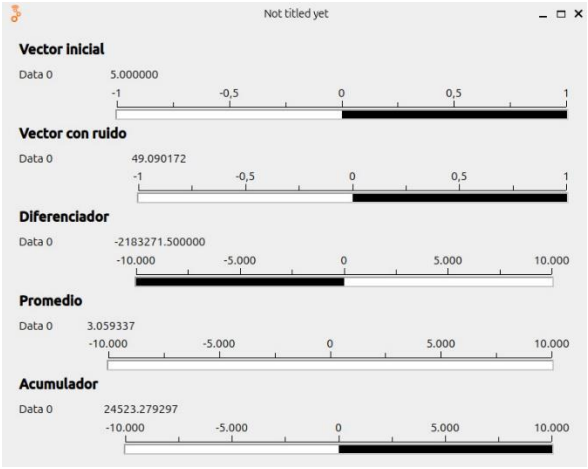

Fig. 8: Resultados obtenidos en la segunda fase.

Para la tercera fase metodológica se pudo obtener los datos esperados, ya que, al agregar una señal cuadrada con ruido gaussiano se puede aprovechar que este ruido posee un valor medio de cero, lo cual permite eliminarlo realizando un proceso de integración con el bloque acumulador, en donde se obtiene una señal triangular, que al ser pasada por un bloque diferenciador nos permite regresar a la señal original sin ruido, además dependiendo el punto de inicio de la señal cuadrada la salida del acumulador presenta cierto nivel de offset.

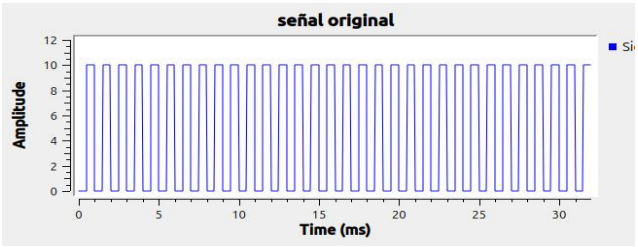

Fig. 9: Resultados obtenidos en la tercera fase.

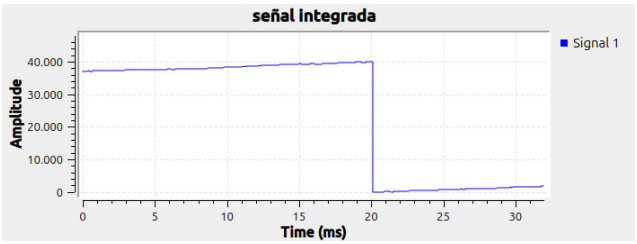

Fig. 10: Resultados obtenidos en la tercera fase.

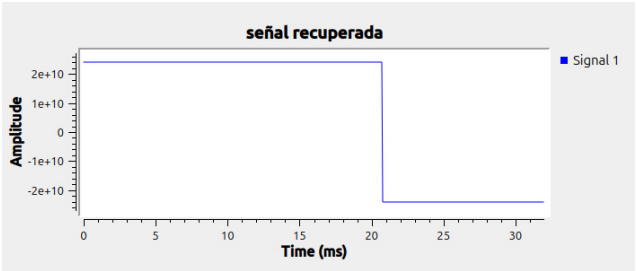

Fig. 11: Resultados obtenidos en la tercera fase.

4. Conclusiones

Los bloques implementados en GNU Radio, específicamente el acumulador, el diferenciador y el bloque de promedio, demostraron ser efectivos en el procesamiento de señales y en la reducción de ruido, logrando recuperar la forma original de la señal a pesar de la adición de ruido gaussiano.

Si conocemos las propiedades estadísticas de una señal, estas se pueden usar para diversos requerimientos, en este caso se aplicó el conocimiento estadístico para atenuar una señal con ruido gaussiano mejorando la calidad de la señal de interés.

5. Referencias

[1] Colaboradores de los proyectos Wikimedia. “Ruido gaussiano - Wikipedia, la enciclopedia libre”. Wikipedia, la enciclopedia libre. Accedido el 8 de septiembre de 2024. [En línea]. Disponible: https://es.wikipedia.org/wiki/Ruido_gaussiano.