Informe 1: PROGRAMACIÓN EN RADIO DEFINIDA POR SOFTWARE

Darwin Camilo Sánchez Gallo - 2202335
Humberto José Contreras Afanador - 2210404
Diego Andrés García Díaz - 2195533

URL Repositorio: `https://github.com/LabG1C02/CommunicationsII_2024_2_G1/tree/P1_Diego`

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

Septiembre 07 de 2024

Abstract

Block-oriented programming in GNU Radio allows the modeling of different systems by which it is possible to determine a response to a problem. The implementation of blocks such as accumulators, differentiators and averaging blocks through software allows us to reduce noise for a given signal and obtain a precise signal. Throughout this practice, these blocks are implemented in order to show a practical approach to processing a signal that has added distortion. [1].

Keywords: *Acumulador, Diferenciador, Media, Valor RMS, Promedios de tiempo, señales.*

1 Introducción

En el laboratorio de Programación en Radio Definida por Software (SDR), se exploran las técnicas de implementación de bloques personalizados en GNU Radio. GNU Radio es un entorno de desarrollo que permite la creación de bloques de procesamiento de señales mediante programación gráfica y Python. Este entorno es clave en la programación de SDR, ya que facilita la definición de módulos que pueden procesar, manipular y analizar señales de radio de manera flexible. En esta práctica, se han implementado bloques como acumuladores, diferenciadores y bloques para calcular valores estadísticos, con el objetivo de procesar y analizar señales con distorsiones añadidas. Además, se ha explorado la integración de scripts en Python para crear bloques más avanzados, demostrando la potencia de GNU Radio al permitir la combinación de programación gráfica y programación de bajo nivel en Python para obtener un control detallado sobre el procesamiento de señales. Este informe presenta la metodología seguida para la creación e implementación de estos bloques, así como el análisis de los resultados obtenidos a partir de señales generadas y su representación gráfica.

2 Metodología

Inicialmente se configuró el repositorio con diferentes ramas, siguiendo previas recomendaciones, este paso es fundamental ya que se puede tener una mejor organización del desarrollo de la práctica, también facilitando la colaboración entre los miembros del grupo para subir y descargar los diferntes archivos que se van almacenando en el repositorio.

2.1 Creación de bloques

Mediante la configuración del entorno GNU Radio se plantea la implementación de los diferentes tipos de bloques como lo son el acumulador y el diferenciador, estos bloques se obtienen mediante el libro guía [2], en este se nos entregan los códigos en Python. , ver figuras 1 y 2.

```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_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)
    #y0 [:] = len(x)
    return len (y0)
```

Fig. 1: Bloque Acumulador en Python.

```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
    # y0 [:] = len(y0)
    return len (y0)
```

Fig. 2: Bloque Diferenciador en Python.

2.2 Implementación de bloques acumulador y diferenciador

Se implementa los dos bloques acumulador y diferenciador para corroborar su funcionamiento, esto se conecta a un bloque generador de señales tipo triángulares y a dos tipos de bloques uno QT GUI Number Sink y QT GUI Time Sink, mediante los cuales queremos ver el comportamiento en forma gráfica.3

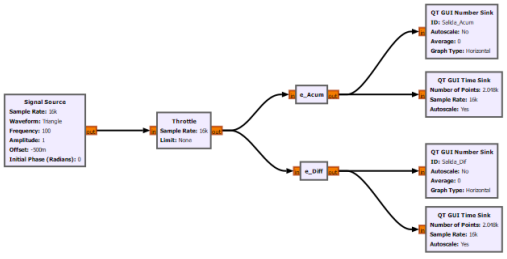

Fig. 3: Sistema de corroboración del funcionamiento del acumulador y diferenciador mediante una señal triangular.

2.3 Implementación de bloque de promedios

Se implementa un bloque para calcular diferentes tipos de valores como son la media, media cuadrada, valor RMS, potencia promedio y desviación estándar, para un conjunto de datos es este caso se genera un Vector Source [1, 2, -1] este se repite continuamente, ver figura 4.

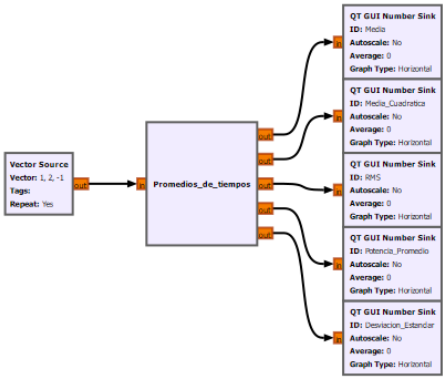

Fig. 4: Implementación de bloque de valores estadísticos para un vector.

2.4 Implementación en una aplicación

En general, los bloques Acumulador y Diferenciador pueden aplicarse en sistemas de monitoreo donde se desea detectar eventos bruscos que afecten el comportamiento normal, como interrupciones o fallas o en el análisis de señales de comunicación, la diferenciación y acumulación de señales puede ayudar a detectar ruido, interferencias o la calidad de la señal.

3 Análisis de resultados

El flujo de procesamiento en GNU Radio utiliza un bloque personalizado llamado *Promedios_de_tiempos* para calcular diversas métricas estadísticas de una señal de entrada. El bloque se conecta a un Vector Source que proporciona un flujo continuo de datos basado en el vector, con una longitud creciente, generando resultados acumulativos que pueden variar ligeramente.

El bloque *Promedios_de_tiempos* está diseñado para tomar una señal de entrada continua y calcular varias métricas estadísticas, como la media (Promedio), media Cuadrática, RMS, potencia promedio y desviación estándar.

Cada métrica se calcula acumulativamente a lo largo del tiempo, utilizando los valores anteriores para actualizar los resultados actuales, esto asegura que los resultados reflejen de manera precisa el comportamiento de la señal a lo largo del tiempo. La variación mínima en los resultados se debe al crecimiento del tamaño de la señal de entrada y a cómo se manejan los cálculos acumulativos en el código.

Al incluir el bloque "Noise Source" junto con el bloque "Add", permitió evaluar el ruido en la medición de las señales procesadas. La adición de ruido simula condiciones reales de transmisión, donde las señales se ven afectadas por el ruido que hay en el ambiente y otras perturbaciones que también afectan las señales. El uso del bloque acumulador, seguido por el bloque diferenciador y el bloque de promedios de tiempo, ayudó a filtrar la señal y obtener mediciones estables, lo que demuestra la utilidad de estos bloques para la reducción de ruido en señales digitales.

Bloque Acumulador:

El bloque toma la señal de entrada y devuelve la suma acumulativa de sus valores, es útil en aplicaciones donde se necesita integrar una señal o seguir su evolución en el tiempo, como en la estimación de tendencias o la detección de cambios de comportamiento en sistemas dinámicos. En el contexto de señales de sensores, el acumulador puede usarse para integrar valores de un sensor, como la aceleración, para calcular velocidad o desplazamiento, al igual que puede ser útil para el control integral, donde se acumulan errores a lo largo del tiempo para corregir desviaciones del objetivo.

Bloque Diferenciador:

Este bloque calcula la diferencia acumulada de la señal respecto a un estado anterior, el diferenciador es útil para detectar cambios bruscos en la señal o variaciones rápidas, también en sistemas eléctricos o electrónicos para detectar cambios rápidos en señales, como transitorios de corriente o voltaje, otra aplicacion importante es que se usa para detectar bordes o cambios en imágenes.

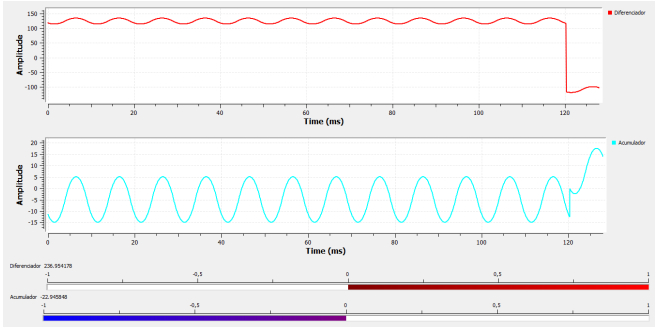

Fig. 5: a nice plot

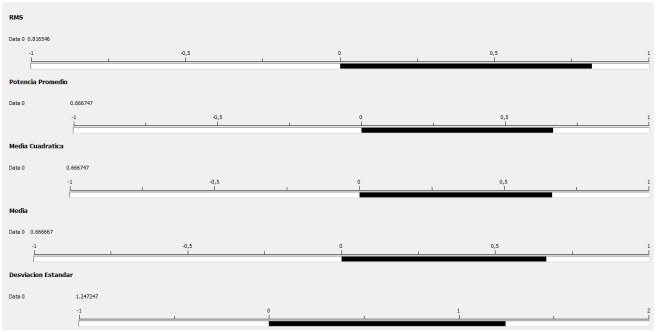

Fig. 6: a nice plot

4 Conclusiones

- El uso de GitHub como entorno de trabajo colaborativo es fundamental para llevar un orden y trabajar al mismo tiempo sin afectar a los demás usuarios, usando ramas y otras características que este tiene, también se ha fortalecido el uso de Git para diferentes funciones como lo es clonar el repositorio, demás de cargar y descargar los diferentes archivos, entre otras más. [3]

- Se ha fotalecido el manejo de Git [4] y los diferentes comandos que se usan en el terminal para hacer el manejo de los archivos y de lo que se carga y descarga del respositorio [5].

- Con esta práctica se logró observar el efecto del ruido en las señales porcesadas, cuando se usa un generador de ruido (Noise Source Block), se observan cambios en los valores promedio de la señal, además se puede hacer un filtrado y mejora de las señales, con el fin de mantener una buena calidad.

- Otra ventaja de usar GNU Radio es que se pueden crear bloques personalizados con *Embedded Python Block* esto con el fin de tener mas flexibilidad y control en sistemas de procesamiento de señales más específicos [6].

- Mediante el estudio de sistemas en tiempo real se presentaron varios inconvenientes en el instante de procesar dichas señales, esto dependiendo de la aplicación o entorno que se desea modelar, es necesario contar con ciertas etapas que nos aseguren en la salida una señal confiable, sin embargo el software tiene deficiencias o problemas al procesar señales, por tanto para prevenir estos inconvenientes, a una señal se pueden tomar muestras en un intervalo del cual se pasara por el sistema de

promedios y entregara una señal más acorde a lo esperado ya que el pasar un intervalo por el bloque promediado, esta cuenta con una cantidad de muestras reducida y evita la toma de muestras distorsionadas entregando una señal mejor procesada.

References

[1] D. Pu, "Digital communication systems engineering with software-defined radio." [Online]. Available: http://ebookcentral.proquest.com/lib/bibliouis-ebooks/detail.action?docID=3002022.

[2] H. Ortega, B. Bodá, O. Mauricio, and R. Torres, "Comunicaciones digitales basadas en radio definida por software." [Online]. Available: https://sites.google.com/saber.uis.edu.co/comdig

[3] D. García, D. Sánchez, and H. Contreras, "Repositorio CommunicationsII_2024_2_G1." [Online]. Available: https://github.com/LabG1C02/CommunicationsII_2024_2_G1/tree/P1_Diego/Practica_1/Practica1

[4] B. S. Scott Chacon, "Git - book." [Online]. Available: https://git-scm.com/book/en/v2

[5] N. G. Ciordia, "10 comandos de git que todo desarrollador debería saber." [Online]. Available: https://www.freecodecamp.org/espanol/news/10-comandos-de-git-que-todo-desarrollador-deberia-saber/

[6] T. G. Radio, "Creating your first block." [Online]. Available: https://wiki.gnuradio.org/index.php?title=Creating_Your_First_Block