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

JOSE DAVID FLOREZ RAMOS - 2174241

https://github.com/2174241/LabComu2_C1.git

8 de septiembre de 2024

Resumen

Este informe presenta la implementación de bloques personalizados en GNU Radio para el procesamiento de señales, proporcionar los conocimientos y habilidades para entender y programar radios definidas por software (SDR) utilizando GNURadio. Se enfoca en conceptos teóricos y su implementación práctica. Incluyendo comprender los fundamentos de SDR, implementar bloques en GNURadio y evaluar los resultados. En la práctica hicimos uso de software como Ubuntu Linux y GNURadio

Palabras clave: Python, GNU Radio, Estadística, Bloques.

1. Introducción

La radio definida por software (SDR, por sus siglas en inglés) ha permitido una evolución significativa en el desarrollo de sistemas de comunicaciones y procesamiento de señales. GNU Radio ofrece una plataforma flexible para diseñar y probar algoritmos mediante el uso de bloques funcionales y personalizados. En esta práctica, se exploraron los bloques acumulador y diferenciador, así como un bloque que calcula promedios estadísticos de señales. El propósito principal fue entender el comportamiento de las señales a lo largo del tiempo y aplicar estos conceptos a situaciones reales.

2. Marco teórico

- Bloque Acumulador: El bloque acumulador tiene como objetivo sumar los valores de una señal a lo largo del tiempo, lo cual es útil en aplicaciones de procesamiento de señales, demodulación y análisis de potencia.

- Bloque Diferenciador: Este bloque calcula la tasa de cambio de una señal respecto al tiempo, permitiendo detectar cambios rápidos o picos en la información transmitida.

- Promedios de Tiempo: Las señales estadísticas, como las senoidales, poseen características que pueden ser evaluadas a lo largo del tiempo. Algunas de estas métricas incluyen la media, la media cuadrática, la varianza, la desviación estándar y el valor RMS. Estas estadísticas son fundamentales para analizar y mejorar el rendimiento de sistemas de comunicaciones y procesamiento de señales.

Las fórmulas de estas métricas se pueden calcular de la siguiente manera:

- Media: $Xm=\langle x(t)\rangle$
- Media cuadrática: $Xc=\langle x^2(t)\rangle$
- Varianza: $\sigma 2=\langle |x(t)-Xm|^2\rangle$
- Desviación estándar: $\sigma x=sqrt(\langle |x(t)-Xm|2\rangle)$
- Valor RMS: $XRMS=\langle |x(t)|^2\rangle$

3. Procedimiento

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

Implementación del algoritmo en el bloque de Python: Se utilizó el libro guía en la sección 1.2.0.1 del cuálse extrajeron los respectivos códigos de Python con el algoritmo para armar los bloques y poder utilizarlos en las operaciones de promedio de tiempo.[1]

Implementación de bloques de acumulador y diferenciador: Se implementaron los bloques de acumulador y de diferenciador.

```
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]

                N = len(x)
                diff = np.cumsum(x) - self.acum_anterior
                self.acum_anterior = diff[N-1]
                y0[:] = diff
                return len(y0)
```

Figura 1. Código bloque diferenciador en GNU Radio.

```
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)

                return len(y)
```

Figura 2. Código bloque acumulador en GNURadio.

Implementación de un bloque para mostrar estadísticas, Teniendo en cuenta lo aprendido, se implementó un bloque para mostrar la estadística vista en la clase teórica anterior y se sugirió una aplicación.

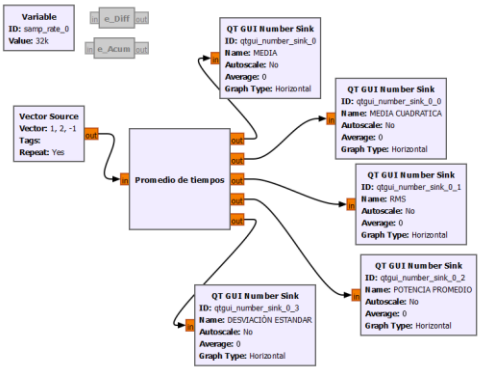

Figura 3. Diagrama de bloques diferenciador.

El experimento consistió en implementar un bloque de promedios de tiempo en GNU Radio, que recibe como entrada una señal compuesta por un vector sumado con una fuente de ruido. Este bloque tiene cinco salidas: media, media cuadrática, valor RMS, potencia promedio y desviación estándar, las cuales se visualizan utilizando el bloque QT GUI Number Sink.

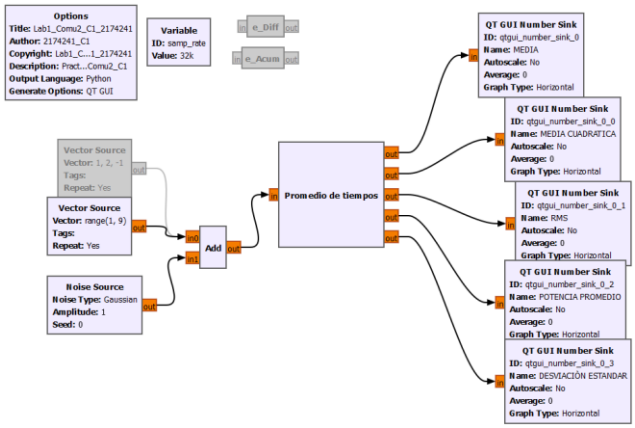

Figura 4. Diagrama de bloques en GNU Radio.

Finalizando de realizar los esquemas, y tomando las evidencias de cada uno, se guardó esta información en el repositorio en GitHub, en una rama dedicada a guardar los avances y resultados obtenidos “Práctia-1”.

4. Análisis de Resultados

Para ver mejor el resultado de la implementación de los bloques, se utilizó con entrada un vector de tres elementos [1, 2, -1] y luego un rango de 1 a 9. Los resultados fueron comparados con las fórmulas teóricas. En ambos casos, las salidas numéricas en los bloques de media, media cuadrática, RMS, potencia promedio y desviación estándar fueron consistentes con los cálculos esperados, lo que valida el funcionamiento correcto del bloque.

Resultados Obtenidos:

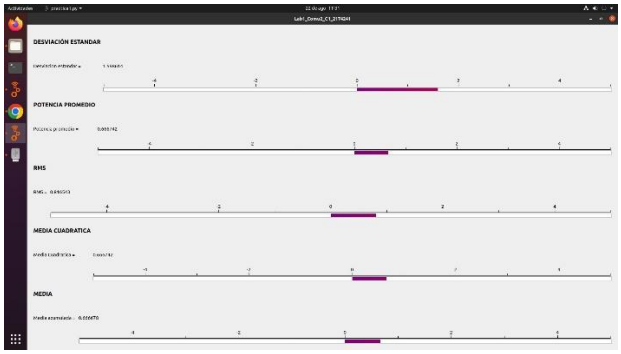

Figura 5. Resultado con vector de tres elementos.

Figura 6. Resultado con números del 1 al 9.

Media: Consistente con el valor esperado para las entradas.
Media cuadrática: Resultado cercano a la teoría.
RMS: Valores precisos de acuerdo a los datos ingresados.
Potencia promedio y desviación estándar: Correctamente calculados.

Desde la figura 1 y figura 2 está el código de los bloques diferenciador y acumulador respectivamente; en la figura 3, el diagrama del bloques para el promedio de tiempos, trabajando sobre ese esquema para lograr la implementación de estos se aprecia en la figura 4 los bloques de entrada con el vector y con el rango de números, ahora en las figuras 6 y 7 son las respectivas visualizaciones de los resultados con cada una de las entradas, estos diagramas de bloques cumplieron la función de realizar los cálculos de medidas estadísticas vistas previamente en clase, usando la opción del bloque de Python que proporciona GNURaido [2]

POSIBLES APLICACIONES:

- **Filtrado de Ruido:** El bloque de "promedio de tiempos" puede filtrar el ruido de una señal al promediar varias muestras a lo largo del tiempo, útil en sistemas de comunicación inalámbrica donde la señal está afectada por ruido.
- **Monitoreo de Sensores:** Este bloque puede aplicarse en entornos de sensores para obtener lecturas más estables, como en sistemas de monitoreo ambiental.
- **Seguimiento de Movimiento:** En sistemas de seguimiento, como en vehículos o personas, el bloque mejora la precisión de las posiciones estimadas al reducir el impacto de mediciones ruidosas.
- **Análisis de Señales Biológicas:** El bloque ayuda a filtrar el ruido en señales biológicas como electrocardiogramas (ECG) o electroencefalogramas (EEG), permitiendo una interpretación más clara.

5. Conclusiones

Esta práctica con GNU Radio me permitió aplicar y comprender de manera tangible conceptos fundamentales del procesamiento de señales. La implementación de los bloques acumulador y diferenciador me permitió observar cómo las señales se acumulan y varían con el tiempo, destacando su relevancia en aplicaciones como demodulación y análisis de potencia.

El bloque de promedios de tiempo me permitió calcular y visualizar en tiempo real estadísticas clave como la media, media cuadrática, valor RMS, potencia promedio y desviación estándar. Este análisis es crucial para mejorar la calidad de las señales, como en el filtrado de ruido y el monitoreo de sistemas de comunicación.

En general, la práctica no solo reforzó conceptos teóricos, sino que me dio una visión más clara de cómo los ingenieros aplican estas herramientas para optimizar sistemas de comunicaciones en escenarios reales.

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." [On-line]. Available: https://wiki.gnuradio.org/index.php/Embedded_Python_Block