Práctica 1: INTRODUCCIÓN A GNU RADIO

JUAN DIEGO PEÑA SALINAS - 2195587
MILLER STEVEN GAMBA ARIZA - 2195584

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

6 de septiembre de 2023

RESUMEN

GitHub es una plataforma en línea que cuenta con una gran biblioteca de repositorios donde podemos encontrar diferentes códigos en distintos lenguajes de programación, con el cual podemos apoyarnos y basarnos para diferentes proyectos en un futuro. El software GNURADIO va a ser la principal herramienta de esta práctica de laboratorio y no solo de esta práctica sino de aquí en adelante, por lo cual familiarizarse con esta herramienta es de vital importancia. Una vez ya familiarizados con el uso de GNURADIO. Al tener estas dos herramientas vinculadas nos facilita la gestión y colaboración de proyectos en el área de comunicaciones.

Palabras clave: GNURADIO, Python , GitHub, repositorio, programación, ramas.

1. INTRODUCCIÓN

En el campo del trabajo moderno es de vital importancia el trabajo colaborativo entre diferentes especialistas de la materia o encargados de desarrollar diferentes tareas para cumplir con un objetivo específico. Teniendo en cuenta la importancia de estas competencias es imprescindible aprender a manejar herramientas que permitan trabajar de manera colaborativa y en paralelo, pero además permite organizar la información de la mejor manera para que se pueda llevar un control de lo que se está realizando y de quien está realizando dicha tarea. De esta necesidad surge la importancia del uso de la herramienta GitHub (figura 1.), la cual nos permite cumplir con los requerimientos de organización y registro del trabajo colaborativo, por esto en la parte inicial se aprenden a manejar de manera óptima y clara la herramienta GitHub para la recopilación y organización de la información desarrollada a lo largo de los laboratorios de la asignatura de Comunicaciones II en la universidad industrial de santander.

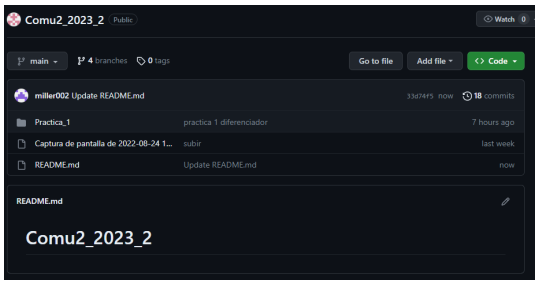

Figura 1. *Interfaz de un repositorio en GitHub.*

El software GNURADIO es ampliamente utilizado para el desarrollo de sistemas de radio y comunicaciones. El funcionamiento de GNURADIO consiste en la combinación de bloques, existen bloques que cumplen un rol en específico como lo son la modulación, la demodulación, filtrado, etc. Una vez realizada la combinación de bloques (figura 2.), se ejecuta el programa y se realiza el respectivo análisis (figura 3.).

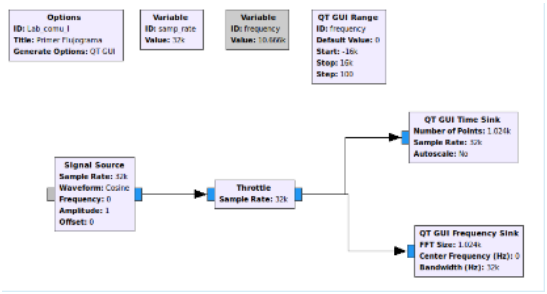

Figura 2. *Ejemplo del diagrama de bloques en GNURADIO.*

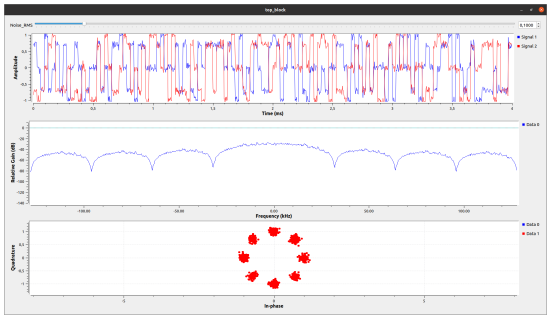

Figura 3. *Ejemplo de ejecución de un diagrama de bloques.*

A manera de profundizar en el SDR GNU Radio dentro de esta primera práctica se estudia una nueva forma de interactuar con la herramienta la cual consiste en darle prioridad a la programación por medio del lenguaje python. Gracias a que Python es un lenguaje de programación fácil de utilizar, de alto nivel y con una amplia biblioteca de módulos nos permite realizar la creación de nuevos módulos de cierta necesidad en específico de manera más sencilla.

2. PROCEDIMIENTO

En la necesidad de generar bloques con funciones específicas en GNU Radio se requiere el conocimiento de conceptos de programación en Python ya que este es el lenguaje usado por GNU Radio. Teniendo en cuenta esto lo primero es mirar la información consignada en el libro guía del curso ya que esta permite tener un primer acercamiento de cual el tipo de estructura que requiere la programación de un bloque para cumplir con una función específica en GNU Radio.

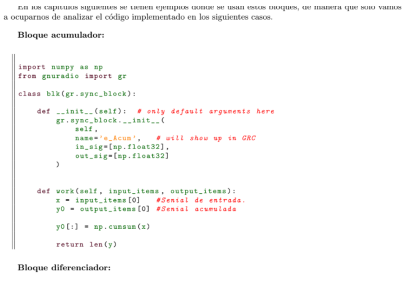

Figura 4. *Ejemplo del esquema para la programación de bloques en GNU Radio.*

Ya teniendo una noción de cómo se debe programar un bloque en GNU Radio se procede a programar un bloque acumulador el cual tiene como principal objetivo sumar o acumular datos de una señal de entrada a través del tiempo y entregar a la salida la suma acumulativa. Las aplicaciones de un bloque como este en el tratamiento de señales son diversas entre las cuales puede usarse para el cálculo de la potencia de una señal o el cálculo de las estadísticas temporales.

Para desarrollar este bloque se usó como base el ejemplo presente en la figura # el cual corresponde a un bloque acumulador sin embargo no funciona así que se le realizaron las respectivas modificaciones para obtener su correcto funcionamiento; obteniendo como resultado la programación de bloque presente en la figura 5.

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

class e_Diff(gr.sync_block):
    def __init__(self, example_param=1.0):  # only default arguments here
        gr.sync_block.__init__(
            self,
            name="e_accum",  # will show up in GRC
            in_sig=[np.float32],
            out_sig=[np.float32]
        )
        self.acum_anterior = 0
        self.example_param = example_param

    def work(self, input_items, output_items):
        x = input_items[0]  # Input signal.
        y0 = output_items[0]  # Differential cumulative signal
        N = len(x)
        summ = np.cumsum(x) + self.acum_anterior
        self.acum_anterior = summ[-1]
        y0[:] = -summ
        return len(y0)
```

Figura 5. *Código para la programación de un bloque acumulador en GNU Radio.*

Una vez se pudo obtener el bloque acumulador se procede a implementar un bloque diferenciador el cual tiene distintas aplicaciones en el tratamiento de señales que está implícito dentro del área de las telecomunicaciones; entre las aplicaciones un bloque con este funcionamiento destacan: detección de cambios de una señal, filtrado de alta frecuencia.

Para desarrollar este bloque se usó como base el ejemplo presente en la figura 8 el cual corresponde a un bloque diferenciador sin embargo no funciona así que se le realizaron las respectivas modificaciones para obtener su correcto funcionamiento; obteniendo como resultado la programación de bloque presente en la figura 7.

Bloque diferenciador:

```
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]   #Sensal de entrada.
        y0 = output_items[0] #Sensal acumulada diferencial

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

Figura 6. *Ejemplo del esquema para la programación de un bloque diferenciador en GNU Radio.*

```
1   import numpy as np
2   from gnuradio import gr
3
4   class e_Diff(gr.sync_block):
5       def __init__(self, example_param=1.0):  # only default arguments here
6           gr.sync_block.__init__(
7               self,
8               name="e_Diff",  # will show up in GRC
9               in_sig=[np.float32],
10              out_sig=[np.float32]
11          )
12          self.acum_anterior = 0
13          self.example_param = example_param
14
15      def work(self, input_items, output_items):
16          x = input_items[0]  # Input signal.
17          y0 = output_items[0]  # Differential cumulative signal
18          N = len(x)
19          diff = np.cumsum(x) - self.acum_anterior
20          self.acum_anterior = diff[-1]
21          y0[:] = diff
22          return len(y0)
```

Figura 7. *Código para la programación de un bloque acumulador en GNU Radio.*

3. CONCLUSIONES

- El uso de un repositorio en GitHub es de gran importancia ya que esta plataforma brinda un banco de códigos para proyectos posteriores.
- Es importante tener en cuenta simuladores como GNURADIO es cual nos puede ser de mucha ayuda al momento de diseñar sistemas de comunicaciones.
- El uso de software GnuRadio es primordial para la programación en radio, con GnuRadio podemos afianzar los conceptos teóricos. Además, con GnuRadio podemos probar algoritmos y técnicas de procesamiento de comunicaciones de manera eficiente antes de su implementación en tiempo real.

REFERENCIAS

[1] Phil Lapsley, Jeff Bier, Amit Shoham and Edward A. Lee, “DSP Processor Fundamentals: Architectures and Features”, Berkeley, California: Berkeley Design Technology, Inc., 1996.

[2] ‘’Wikipedia: Procesamiento digital de señales” [Online]. Aviable: https://es.wikipedia.org/wiki/Procesamiento_digital_de_se%C3%B1ales

[3] ‘’Wikipedia: Tiempo real” [Online]. Aviable: https://es.wikipedia.org/wiki/Tiempo_real