Práctica 2

Conquista de gnuradio a nivel de programación

Prácticas de programación en Python

Autores:

Lugo Vásquez Vanny Sofia - 2154709
Angarita Pertuz Jose David - 2142992
Cancino Rey William Andrés - 2151534

Grupo:
B1A.g2

ESCUELA DE INGENIERIAS ELECTRICA ELECTRONICA Y TELECOMUNICACIONES– E3T
UNIVERSIDAD INDUSTRIAL DE SANTANDER – UIS

Aspectos a mejorar en la guia

Los siguientes son apuntes del profesor para introducir mejoras a futuras prácticas:

- Por ahora no hay apuntes

Enlace a materiales de apoyo

- Manual de manuales
- El libro de la asignatura
- Página del libro

El Problema:

Por ahora el problema a resolver consiste en que el estudiante no tiene las suficientes bases de programación por objetos en Python para pasar a realizar desarrollos usando GNU Radio y herramientas profesionales.

El objetivo general es:

Retar al estudiante a construir su propio conocimiento para programar por objetos usando python y herramientas profesionales como Github y Visual Studio Code.

Preparativos

- Baje una version actualizada del libro para concentrarse en el capítulo 1.5 y 1.6 donde está la teoría necesaria. Tenga en cuenta que: en el libro de la asignatura. Observe que en los capítulos del libro ofrecen enlaces a código de software, a flujogramas y otros recursos que son parte del libro. Por ejemplo, observa que debajo de cada gráfica con flujogramas hay una nota que dice: “Flujograma usado”. Esos recursos usados en el libro están en la página del libro: https://sites.google.com/saber.uis.edu.co/comdig/sw

Apuntes de interés

- Python es un lenguaje interpretado
- Python es un lenguaje indentado
- En Python los tipos de las variables se deduce de manera automática, por ejemplo:
  - Si el programador escribe x=0.4 el lenguaje decide que x es una variable de tipo flotante
  - Si se escribe x=4 el lenguaje decide que x es una variable de tipo entero, pero si escribe x=4. decide que es una variable de tipo flotante.
  - Si se escribe x=[1,2,d,2,4,fuerte,9] el lenguaje decide que x es una variable de tipo lista flotante
  - El uso de vectores y matrices es común cuando se trabaja con GNU Radio. Para trabajar con vectores se debe usar una librería llamada numpy:
    - import numpy as np # para importar la librería
    - x=np.array([1,2,3,4,5]) # crea un vector.
    - x=np.linspace(2.0, 3.0, num=5) # es otro ejemplo para crear un vector
    - ver más ejemplos de trabajo con vectores en: Manual de manuales, sección “Python para desmemoriados”

Objetivos específicos

Nota: para todos los casos use Python3

1. Programación por Terminal de Ubuntu. Descubra la utilidad de programar por terminal de Ubuntu para realizar cálculos rápidos o para comprobar que algunas sentencias o métodos funcionan correctamente antes de pensar en usarlas en un programa más complejo. Para ello, abra un terminal de Ubuntu. Use la Internet para consultar el atajo para abrir una terminal de Ubuntu, pero también para buscar ayuda y ejemplos cuando lo requiera. Compruebe que puede realizar los siguientes cálculos:
   a. Programación tradicional: x, y son números escalares, por ejemplo x=0.4, y=1.3
      i. z=x+y

ii. h=cos(x)

iii. $g = e^{j2\pi t}$

b. Programación vectorial: x, y, t,z,h,g son números vectores. Por ejemplo x=(0,1, 0.2, 0.4), y=(2., 3., 5.1)
   i. z=x+y
   ii. h=cos(x)
   iii. $g = e^{j2\pi t}$

2. Programación de python con archivos. Práctica de variable aleatoria. Para esta tarea use el editor “gedit” para crear un archivo con extensión .py. Hallar la esperanza a una variable aleatoria.
   a. Hemos realizado un experimento para recoger las alturas y los pesos aleatoriamente a un volumen de N estudiantes varones que pasan por la entrada de la UIS. Los datos recogidos se encuentran en la siguiente tabla, donde X es la variable aleatoria de las alturas, Y son los pesos.

Ensayo	X (metros)	Y (kilos)
1	1.7	60
2	1.78	66.3
3	1.68	58
4	1.8	75
5	1.67	57
6	1.75	62.3

   b. Preparativos para hallar diferentes características de las variables aleatorias como la media, la media cuadrática, la varianza, la correlación. Hay que tener en cuenta que las funciones que se implementen deben poder procesar cientos de miles de ensayos. Por eso es importante asegurarse muy bien que están bien hechas. Una buena práctica es crear una tabla de excel (o google sheet), como la de este enlace que muestre un ejemplo muy claro de datos precalculados. De manera que un primer reto es lograr que con Python se logre resultados iguales que con la tabla que sirve de patrón, pero usando y re-usando objetos y/o funciones de Python.
   c. En Python, en el archivo, pase los valores de las variables aleatoria, para obtener X=(1.7, 1.78, ...) y Y=(60, 66.3, 58, ….). Luego cree una función que permita hallar la esperanza de cualquier variable aleatoria y compruebe que funciona con X y con Y

d. re-utilice la función anterior para hallar la media cuadrática de cualquier variable aleatoria.
e. siga re-utilizando para hallar la varianza
f. siga re-utilizando para hallar la correlación entre X y YEnsa

3. Programación usando un editor profesional como el “Visual Studio Code”. Práctica de promedio de tiempo. Ahora lo que se tiene es una señal discreta x[n] que resulta de muestrear una señal x(t) con una frecuencia de muestreo determinada. Adapte la solución anterior para hallar. Use el ejemplo de promedios de tiempo que se tiene en la tabla de este enlace para entender cómo se espera calcular promedios de tiempo de una señal x[n]. Si se compara con la tabla que vimos en punto b, vemos que los cálculos son los mismos, es más cuestión de uso de nuevos términos y de asociarlos al tiempo. Ahora debe usar Visual Studio Code como editor, los datos de entrada son los de la señal x[n] y debe poder obtener la media, la media cuadrática, la varianza, el valor RMS, la potencia promedio y función de Autocorrelación $R_X(2)$.

comando	explicación
git clone URL	para clonar su repositorio. Desde su cuenta copia la URL del repositorio; en terminal Ubuntu envía el comando; en tu computador aparecerá la carpeta del repositorio
git add .	para agregar nuevo contenido a una lista
git commit -m “comentario”	para pasar el nuevo contenido a base local
git push	para que el contenido suba al repositorio en la nube

Informe de resultados

Desarrollo del Objetivo 1. Presente a continuación los resultados del objetivo 1.

```
>>> import numpy as np
>>> from numpy import pi
>>> x = np.array([0,1,0.2,0.4])
>>> y = np.array([2,3,5.1,6])
>>> t = np.linspace(0,1,101)
>>> z = x+y
>>> h = np.cos(x)
>>> g = np.exp(2j*pi*t)
>>> z
array([2. , 4. , 5.3, 6.4])
>>> h
array([1.        , 0.54030231, 0.98006658, 0.92106099])
>>> g
array([ 1.00000000e+00+0.00000000e+00j,  9.98026728e-01+6.27905195e-02j,
        9.92114701e-01+1.25333234e-01j,  9.82287251e-01+1.87381315e-01j,
        9.68583161e-01+2.48689887e-01j,  9.51056516e-01+3.09016994e-01j,
        9.29776486e-01+3.68124553e-01j,  9.04827052e-01+4.25779292e-01j,
        8.76306680e-01+4.81753674e-01j,  8.44327926e-01+5.35826795e-01j,
        8.09016994e-01+5.87785252e-01j,  7.70513243e-01+6.37423990e-01j,
        7.28968627e-01+6.84547106e-01j,  6.84547106e-01+7.28968627e-01j,
        6.37423990e-01+7.70513243e-01j,  5.87785252e-01+8.09016994e-01j,
        5.35826795e-01+8.44327926e-01j,  4.81753674e-01+8.76306680e-01j,
        4.25779292e-01+9.04827052e-01j,  3.68124553e-01+9.29776486e-01j,
        3.09016994e-01+9.51056516e-01j,  2.48689887e-01+9.68583161e-01j,
```

Figura 1. Programación en python desde el terminal de ubuntu.

Principalmente, se logra importar librerías y reconocer la utilidad de estas. Por ejemplo, la librería “math” se utiliza para acceder a funciones trigonométricas y exponenciales. La librería “cmath” permite trabajar las funciones que se mencionan anteriormente con números complejos. Finalmente, la librería “numpy”, la cual se utiliza para el cálculo numérico y la manipulación de matrices, como se muestra en Figura 1.
También se evidencia la importancia de realizar programación vectorial. Esto permite trabajar con conjuntos de datos grandes de manera eficiente y rápida.

Desarrollo del Objetivo 2. Presente a continuación los resultados del objetivo 2.

Ensayo	X [metros]	Y [kilos]	X^2	$(X\text{-media})^2$	X*Y
1	1,7	60	2,89	0,0009	102
2	1,78	66,3	3,1684	0,0025	118,014
3	1,68	58	2,8224	0,0025	97,44
4	1,8	75	3,24	0,0049	135
5	1,67	57	2,7889	0,0036	95,19
6	1,75	62,3	3,0625	0,0004	109,025

Media de X	1,73
Media cuadrática de X	2,995366667
Varianza	0,002466667
Correlación	109,4448333
N	6

Figura 2. Implementación previa en excel.

```python
import numpy as np

def media(x):
        x = np.array(x)
        a = np.sum(x)
        b = x.shape
        N = b[0]
        media0 = a/N
        return media0

def mediacua(x):
        x = np.array(x)
        mediacua0 = media(x**2)
        return mediacua0

def varianza(x):
        x = np.array(x)
        a2 = (x - media(x))**2
        varianza0 = media(a2)
        return varianza0

def correl(x,y):
        x = np.array(x)
        y = np.array(y)
        a = x*y
        correl0 = media(a)
        return correl0

X = [1.7, 1.78, 1.68, 1.80, 1.67, 1.75]
Y = [60, 66.3, 58, 75, 57, 62.3]

print("Media de X = ", media(X))
print("Media de Y = ", media(Y))
print("Media cuadrática de X = ", mediacua(X))
print("Media cuadrática de Y = ", mediacua(Y))
print("Varianza de X = ", varianza(X))
print("Varianza de Y = ", varianza(Y))
print("Correlación entre X e Y = ", correl(X,Y))
```

Resultados en el terminal de ubuntu

```
(tfcuda) william@william:~/Documentos/Comunicaciones_2$
python codigo.py
Media de X =  1.7299999999999998
Media de Y =  63.1
Media cuadrática de X =  2.995366666666667
Media cuadrática de Y =  4019.1633333333334
Varianza de X =  0.0024666666666666713
Varianza de Y =  37.553333333333335
Correlación entre X e Y =  109.44483333333334
```

Figura 3. Script en python y resultados para el cálculo de algunos conceptos de variable aleatoria.

Con los datos que se presentan de las variables aleatorias “X” e “Y” en Figura 2., se calcula lo siguiente: Media, media cuadrática, varianza y correlación. Dichos cálculo se realizan previamente en excel, y finalmente se implementa en python.
Como se evidencia en Figura 2. y Figura 3., los datos con iguales, lo cual demuestra que la implementación en python es correcta.

Desarrollo del Objetivo 3. Presente a continuación los resultados del objetivo 3.

Inicialmente, para implementar el script que compone los cálculos de los promedios de tiempo de una señal discreta, se realiza una implementación básica en excel, con el fin de comprobar que dichos cálculos son correctos y de igual forma obtener la secuencia de pasos que involucra su implementación en python, según se muestra en Figura 4.

n	x[n]	$x^2[n]$	Acumulador	$(x[n]-promedio)^2$	x[n]*x[n+2]
1	0	0	0	0,08027777777778	0
2	0,5	0,25	0,5	0,04694444444444	0,25
3	1,2	1,44	1,7	0,84027777777778	0
4	0,5	0,25	2,2	0,04694444444444	-0,25
5	0	0	2,2	0,08027777777778	0
6	-0,5	0,25	1,7	0,613611111111111	0

Promedio de x[n]	0,283333333
Media cuadrática de x[n]	0,365
Varianza	0,284722222
Valor RMS	0,604152299
Potencia promedio	0,365
Función de autocorrelación	0
N	6

Figura 5. Implementación básica en excel de los promedios de tiempo.

Una vez se realiza la comprobación de los cálculos y con los pasos claros para realizarlos, se implementa un código en python, el cual recibe una señal discreta (vector), de tipo real o compleja, para hallar los promedios de tiempo: Media (promedio), media cuadrática, varianza, valor RMS, potencia promedio y función de autocorrelación. Dicha implementación se realiza en el editor de código “Visual Studio Code”, según se evidencia en Figura 5.

```python
import numpy as np

def promedio(x):
    x = np.array(x)
    a = np.sum(x)
    b = x.shape
    N = b[0]
    promedio0 = a/N
    return promedio0

def mediacua(x):
    x = np.abs(np.array(x))
    mediacua0 = promedio(x**2)
    return mediacua0

def varianza(x):
    x = np.array(x)
    a2 = (x - promedio(x))**2
    varianza0 = promedio(a2)
    return varianza0

def rms(x):
    x = np.abs(np.array(x))**2
    rms0 = np.sqrt(promedio(x))
    return rms0

def potenpro(x):
    x = np.abs(x)**2
    potenpro0 = promedio(x)
    return potenpro0

def autocor(x,tao):
    x = np.array(x)
    xnext = np.concatenate((x[tao:],np.zeros(tao)))
    a = x*xnext
    if(x.dtype=='complex128'):
        a = np.abs(a)
    autocor0 = promedio(a)
    return autocor0

x = [0, 0.5, 1.2, 0.5, 0, -0.5]
tao = 2

print "Promedio de x[n] = " + str(promedio(x))
print "Media cuadrática de x[n] = " + str(mediacua(x))
print "Varianza = " +  str(varianza(x))
print "Valor RMS = " +  str(rms(x))
print"Potencia promedio = " +  str(potenpro(x))
print "Función de autocorrelación = " +  str(autocor(x,tao))
```

Figura 6. Implementación en python para el cálculo de los promedios de tiempo de una señal discreta.

Por último, desde el terminal que proporciona este editor, se pueden observar los resultados de cada uno de los cálculos según Figura 7.

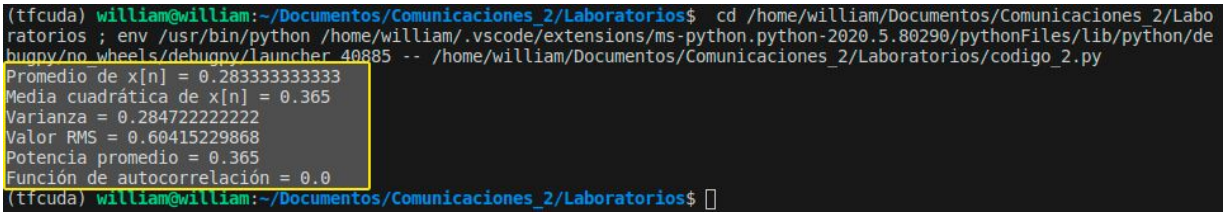


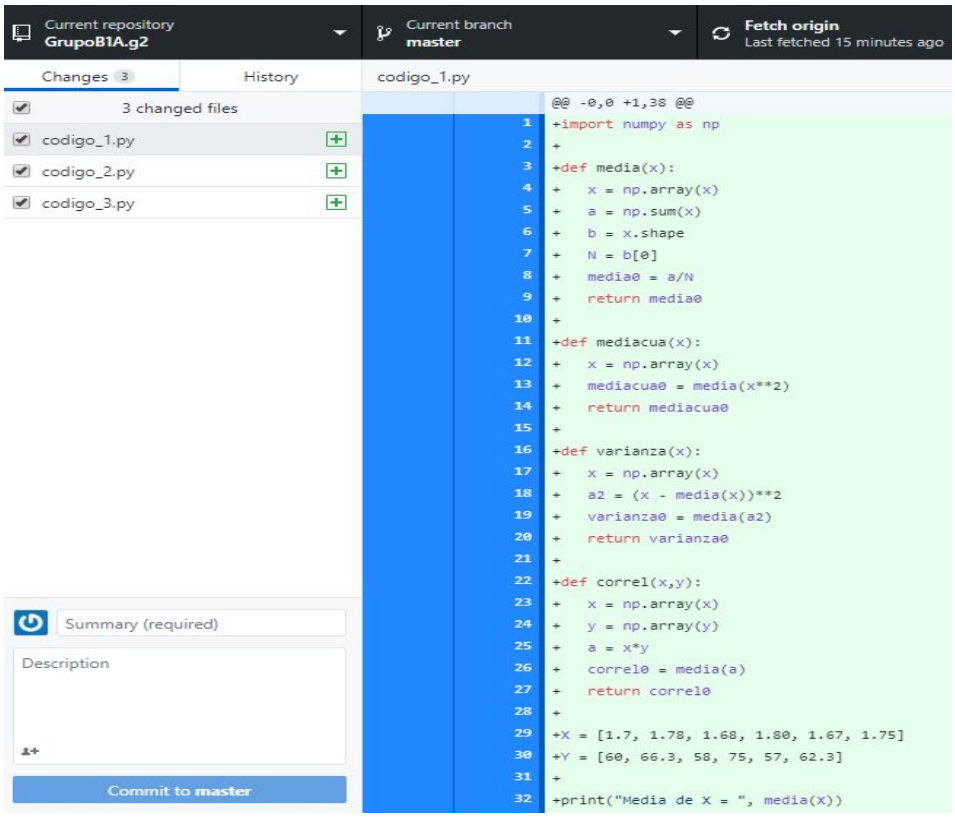
Figura 7. Resultados de la implementación en python.

Como se observa en Figura 5. y Figura 7., los resultados son idénticos, lo cual indica que la implementación en python es satisfactoria. Adicional a ello, es importante mencionar que dicho script está en capacidad de procesar señales discretas complejas.

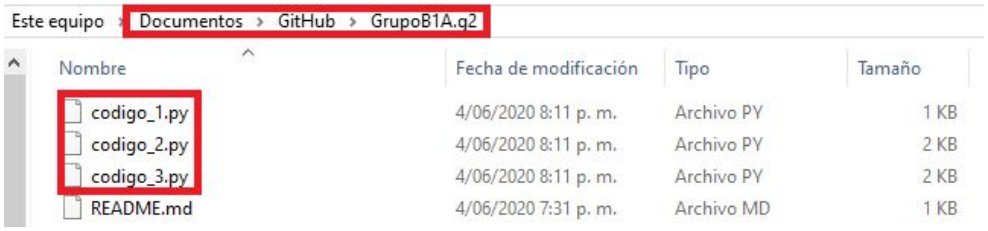
Los códigos que se presentan anteriormente se encuentran en [1]. El archivo “codigo.py” corresponde a la implementación para el cálculo de los conceptos referentes a la variable aleatoria. El archivo “codigo_2.py” es la implementación para realizar los cálculos de promedios de tiempo de una señal discreta. Finalmente, “codigo3.py” presenta una implementación para el cálculo de los promedios de tiempo utilizando un acumulador de la señal.

Desarrollo del Objetivo 5. Presente a continuación los resultados del objetivo 5.

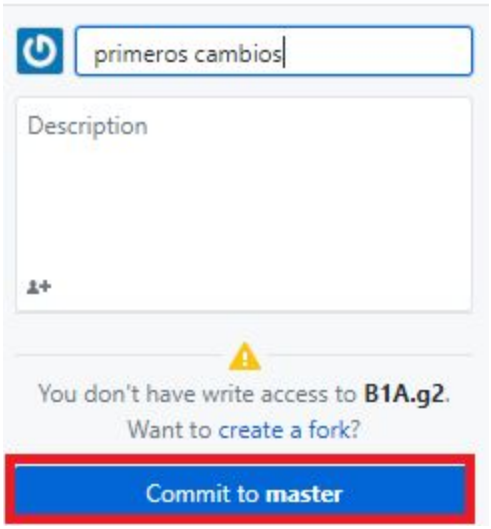
Github es un sistema de gestión de proyectos y control de versiones de código, así como una plataforma de red social diseñada para desarrolladores.
Inicialmente, se crea un repositorio en github con el nombre “Grupo B1A.g2”.



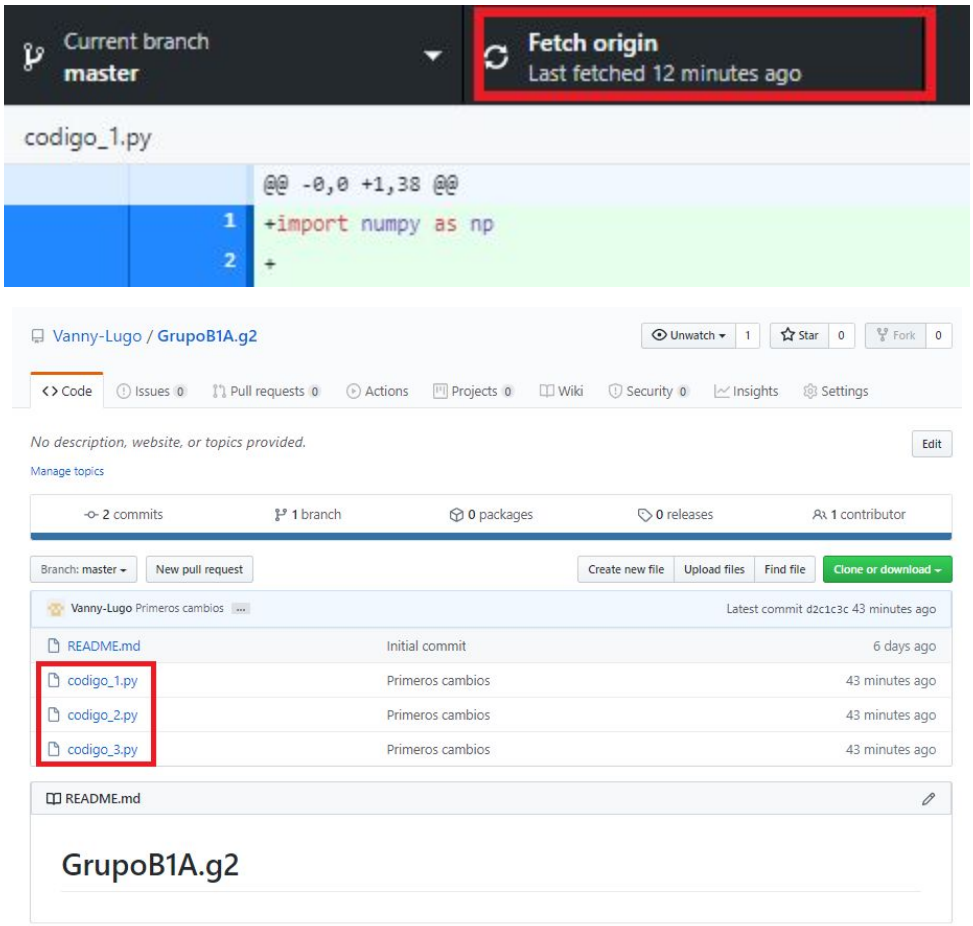
En este objetivo se crea un repositorio local y se clona el repositorio web.



Además, para agregar el contenido al repositorio web se realiza un comentario al cambio a desarrollar.



Finalmente, se envía la información al repositorio web.



REFERENCIAS

[1] William Cancino. Códigos en python.
https://github.com/WilliamCancino/B1A.g2/tree/master/Laboratorio/02/codigos

[2] José David Angarita. Códigos en python.
https://github.com/josdeivi90/lab1_comunicaciones

[3] Vanny Sofia Lugo Vásquez. Códigos en python.
https://github.com/Vanny-Lugo/GrupoB1A.g2

[4] Homero Ortega Boada. Comunicaciones digitales basadas en radio definida por software.
https://drive.google.com/file/d/1vGuvcL-3BPdIOqpK-VqGGDYXiDD_SOVE/view

[5] Github.
https://www.hostinger.co/tutoriales/que-es-github/