UNIVERSIDAD CATOLICA
DE TEMUCO

Laboratorio #2

Programación 2

Alumno: Cristian Beltrán Concha

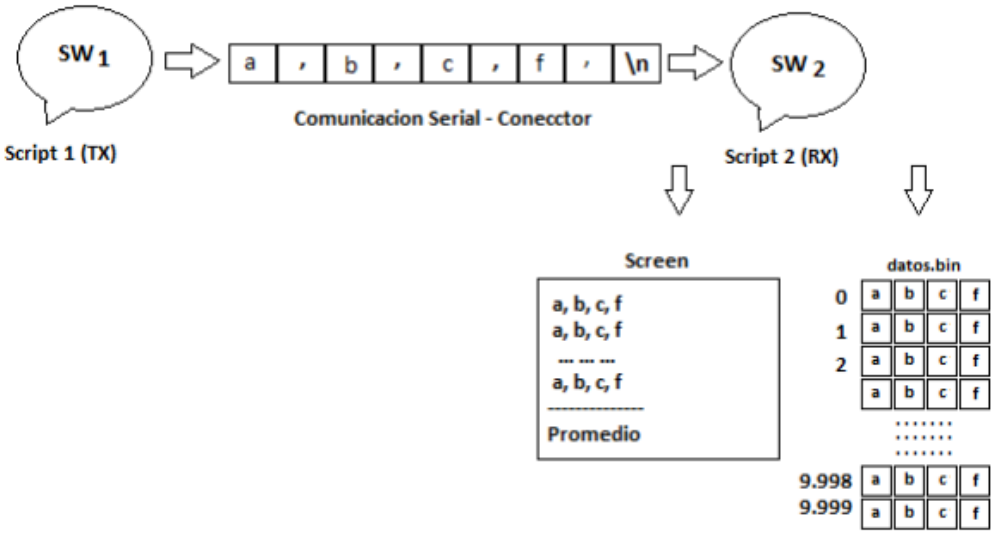
Profesor: Luis Caro Saldivia

Fecha: 28-10-2015

1.- Se tiene un Script en Python que envía por el puerto serial **10.000** datos estructurados: {randint[1,100], randint[1,1.000], randint[1,10.000], random()}. Los datos se envían como String y van separados con coma y con salto de linea. Un ejemplo valido de una secuencia de datos seria así: 23,450,6.527,0.19096663981190465,\n. Los datos se envían cada 0.1 segundo.

Por el lado del Script en Python que recibe los datos, estos se deben convertir al tipo de dato original y ser almacenados en el archivo binario "**datos.bin**". Una vez que se han recibido todos los datos, estos se deben imprimir por pantalla y calcular el promedio de los datos de cada columna. Estas operaciones se hacen sobre el archivo binario.

Para simular la conexión serial, se debe utilizar el programa **Virtual Serial Port Emulator** y configurar un **Connector**, con emulación de baudios a **9.600-8N1**. Aqui un esquema que clarifica lo que debe hacer.



Código que Envía Información por serial

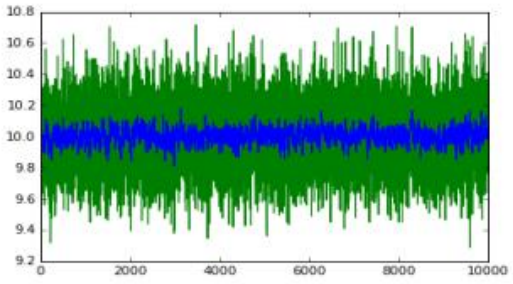
```
import random as RA
import time
import serial

s = serial.Serial(0)    #COM1
s.baurate = 9600
for i in range(10000):
    a =  str(RA.randint(1,100)) + "," + str(RA.randint(1,1000)) +","+ str(RA.randint(1,10000)) +","+str(RA.random())
    s.write(a+"\n")
    #print a
    time.sleep(.1)  # espera .1 segundos
```

Recibe la información y crea archivo

```
import serial

s = serial.Serial(0)     #COM1
s.baurate = 9600

f = open("datos.bin","wb")  # abre el archivo en modo escritura

dato = []
for i in range(10000)       # espera los 10.000 datos
    a = s.readline()
    f.write(a)
f.close()

#  =========== Archivo
f = open("datos.bin","rb")
for l in f:
    dato.append(l[:-1])     # agrega datos al array
f.close()


a = 0.0
b = 0.0
c = 0.0
f = 0.0

for d in dato:
    e = d.split(",")    # trozea la info
    a += int(e[0])
    b += int(e[1])
    c += int(e[2])
    f += float(e[3])
    print d         # imprime datos

print "\nPromedio a: " + str(a/len(dato))
print "Promedio b: " + str(b/len(dato))
print "Promedio c: " + str(c/len(dato))
print "Promedio f: " + str(f/len(dato))
```

2.- Genere un arreglo de **10.000** datos aleatorios con valores de una distribución Normal con media **10** y desviación estándar de **0.5**. Aplique el siguiente filtro y obtenga el siguiente gráfico. Realice un total de **5** gráficos probando valores de **a** con [**0.1, 0.3, 0.5, 0.7, 0.9**].

```
// Filtro Complementario...
def lowpass(v,a):
 f[0] = v[0]
 for i in range(1,10000):
    f[i] = a*v[i] + (1.0 - a ) * f[i-1]
 return f
```



```
import pylab as plt
import random as RA


def lowpass(v, a):
    f[0] = v[0]
    for i in range(1, 10000):
        f[i] = a*v[i] + (1.0 - a)*f[i-1]
    return f


a = [.1, .3, .5, .7, .9]
ve = []
f = []

# completa array con distribucion normal media 10 desviacion estandar 0.5
for i in range(10000):
    ve.append(RA.gauss(10, .5))
    f.append(0.0)

# ---- Grafico 1 ---- #
f = lowpass(ve, a[0])
plt.figure(1)
plt.subplot(211)
plt.plot(ve)
plt.plot(f)
# ------------------ #
# ---- Grafico 2 ---- #
f = lowpass(ve, a[1])
plt.subplot(212)
plt.plot(ve)
plt.plot(f)
# ------------------ #

# ---- Grafico 3 ---- #
f = lowpass(ve, a[2])
plt.figure(2)
plt.subplot(211)
plt.plot(ve)
plt.plot(f)
# ------------------ #
# ---- Grafico 4 ---- #
f = lowpass(ve, a[3])
plt.subplot(212)
plt.plot(ve)
plt.plot(f)
# ------------------ #
# ---- Grafico 5 ---- #
f = lowpass(ve, a[4])
plt.figure(3)
plt.subplot(211)
plt.plot(ve)
plt.plot(f)
```