```
"""
Programa final para realizar mediciones en la muestra por medio de
rampas.
"""

import visa
import time
import matplotlib.pyplot as plt
import numpy as np
import parallel



class handler():
    """
                Maneja los eventos de matplotlib y el control de
temperatura
    """
    def __init__(self,line,puerto):
        self.line = line
        self.puerto = puerto
        self.setP = 0 #set point
        self.setPex = 0 #punto decimal del set point
        self.new = True # habilita la entrada de un nuevo dato
        self.xs = line.get_xdata()
        self.ys = line.get_ydata()
        self.STOP = False # frena la iteracion
        self.cid =
line.figure.canvas.mpl_connect('close_event',self.stop)
        self.cidkey =
line.figure.canvas.mpl_connect('key_press_event',self.setPoint)

    def stop(self,event):
        self.STOP = True

    def cambiarSetP(self):
        print('nuevo set point: ', self.setP)
        self.puerto.write('SMON;')
        self.puerto.write('SETP1,'+str(self.setP)+';')# setea el
setPoint a 80
        self.puerto.write('SCONT;')
                #self.puerto.write('SCONT;')# modo control

    def setPoint(self, event):
        try:
            n = int(event.key)
            if self.new:
                self.setP = n
                self.setPex = 0
                self.new = False
            elif self.setPex==0 :
                self.setP = self.setP*10 + n
            else:
                self.setP = self.setP + n*(10**self.setPex)
                self.setPex -= 1
        except:
```

```
            if event.key == '.':
                self.setPex = -1
            if event.key == 'enter':
                self.new = True
                self.cambiarSetP()
            if event.key == 'backspace':
                self.new = True
            if event.key == 'r':
                plt.cla()
                print ('reset grafico')


class fuente():
        def __init__(self):
                self.p = parallel.Parallel()  # open LPT1 or
/dev/parport0
                self.curr = 0 # 0-> corriente positiva; 1-> corriente
negativa
                self.medicion = 0 # 0-> tension; 1-> termocupla
                self.p.setData(2*self.medicion+self.curr)
        def set(self):
                self.p.setData(2*self.medicion+self.curr)
        def invertir(self):
                if(self.curr == 1):
                        self.curr = 0
                else:
                        self.curr =1
        def medir(self,entrada):
                if entrada=='t':
                        self.medicion = 1
                elif entrada=='v':
                        self.medicion = 0




"""
La rampa funciona cuando quiere, el problema radica en que
cuando se setea no se setea el punto de inicio en el comando
Encontramos que para que empiece a funcionar suele ser util
setear una rampa manualmente en el equipo.
"""
def rampa(T_fin,paso,tc,nanovolt,fuente,t0,file,axTc,axNv,axNt):
        tc.write('SCONT;') # modo control

        tc.write('STUNE1;') # rampa, va desde temperatura actual a
80.0 K a 1.0 K/min

        T_aux = tc.query('QSAMP?1;')
        T = float(T_aux[0:len(T_aux)-3])

        tc.write('SETP1,'+str(T)+';') # setea el setPoint a la T
actual
        #time.sleep(5.)

        print('SRMP1,1,'+str(T_fin)+','+str(paso)+';')
```

```
        tc.write('SRMP1,1,'+str(T_fin)+','+str(paso)+';') # rampa, va
desde temperatura actual a 80.0 K a 1.0 K/min

        #time.sleep(2.)

        #tc.write('SETP1,'+str(T)+';') # setea el setPoint a 10

        seguir = True

        while not H.STOP and seguir:
                T_aux = tc.query('QSAMP?1;')
        #    V = nanovolt.query('MEAS:VOLT:DC?;')
                V1 = float(nanovolt.query('READ?;'))
                fuente.invertir()
                fuente.set()
                time.sleep(0.2)
                V2 = float(nanovolt.query('READ?;'))

                t = time.time()

                fuente.medir('t')
                fuente.set()
                time.sleep(0.2)
                T1 = float(nanovolt.query('READ?;'))
                fuente.invertir()
                fuente.set()
                time.sleep(0.2)
                T2 = float(nanovolt.query('READ?;'))

                fuente.medir('v')
                fuente.set()

                T = float(T_aux[0:len(T_aux)-3])

                #print(T_aux+str(t-t0))
                file.write(str(t-t0)+ " " +str(T)+ " " + str(V1)+ " "
+ str(V2)+ " " + str(T1)+ " " + str(T2)+"\n")

                #tiempos.append(t)
                #temperaturas.append(T)

                #plt.plot(t-t0,T,"r*")
                #plt.pause(0.5) # muestra los datos y espera 0.5
segundos
                axTc.plot(t-t0,T,"r*")
                axNv.plot(t-t0,(V1-V2)/2,"b.")
                axNt.plot(t-t0, (T1-T2)/2,"g.")
                plt.pause(0.1) # muestra los datos y espera 0.5
segundos
                #time.sleep(1)

                if(abs(T-T_fin) < 0.5):
                        seguir = False
```

```
"""
Seteamos los valores iniciales y realizamos las conecciones a los
equipos
"""

rm = visa.ResourceManager()
rm.list_resources()

tc = rm.open_resource('GPIB::10::INSTR')
nanovolt = rm.open_resource('GPIB::22::INSTR')# este hay que
habilitarlo desde el equipo

print(nanovolt.query('*IDN?;'))
print(tc.query('*IDN?;'))



print(tc.query('QSETP?1;'))# pregunta el setpoint

p = parallel.Parallel()  # open LPT1 or /dev/parport0

p.setData(0x00)
#envia el byte en hexa con los puestos 2-9
# 0xff prende todos
# 0x00 apaga todos
# dan voltaje de 0.5 cuando estan apagados y 3.4 prendidos

print(nanovolt.query('MEAS:VOLT:DC?;'))




t0 = time.time()
t = t0


fig = plt.figure()
#GRafico de temperatura
axTc = fig.add_subplot(311)
axTc.set_xlabel('Tiempo (s)')
axTc.set_ylabel('Temperatura (K)')
lineTc, = axTc.plot([],[])

#Grafico de tension
axNv = fig.add_subplot(312)
axNv.set_xlabel('Tiempo (s)')
axNv.set_ylabel('Tension 1 (V)')
lineNv, = axNv.plot([],[])

#Grafico de tension
axNt = fig.add_subplot(313)
axNt.set_xlabel('Tiempo (s)')
axNt.set_ylabel('Tension 2 (V)')
lineNt, = axNt.plot([],[])
```

```
H = handler(lineTc,tc)

#tiempos = []
#temperaturas = []


fuente = fuente()
#nanovolt.write("CONF:VOLT:DC 0.001,1E-9")

nombreArchivo = ((time.ctime(time.time()).replace(':',' '))+'
.txt').replace(' ','_')

file = open(nombreArchivo,'a')
file.write("tiempo temperatura V1 V2 T1 T2\n")




"""
Rutina:
El programa esta pensado para comenzar a temperatura ambiente, o al
menos mayor a 100 K.
baja de la temperatura actual a 100K con un paso de 5 K/min
Mide de 100 a 80 con paso de 0.1 tanto en la bajada como en la subida
Baja a 60, con paso de  5. y termina a 200K con ese mismo paso.
"""
rampa(100.,2.5,tc,nanovolt,fuente,t0,file,axTc,axNv,axNt)

rampa(80.,.1,tc,nanovolt,fuente,t0,file,axTc,axNv,axNt)

rampa(60.,2.5,tc,nanovolt,fuente,t0,file,axTc,axNv,axNt)

rampa(80.,2.5,tc,nanovolt,fuente,t0,file,axTc,axNv,axNt)

rampa(100.,.1,tc,nanovolt,fuente,t0,file,axTc,axNv,axNt)

rampa(200.,2.5,tc,nanovolt,fuente,t0,file,axTc,axNv,axNt)

file.close()
```