Universidad Nacional Autónoma de México

Facultad de Ingeniería

División de Ingeniería Mecánica e Industrial

Laboratorio de Cómputo de Ingeniería Mecatrónica (1964)

Profesor: Miguel Serrano Reyes
Semestre 2023-1

Práctica No. 4

Nombre de la práctica

Interfaz software hardware

Nombre del Estudiante:
Olmedo Guevara José Ángel

Código en Arduino

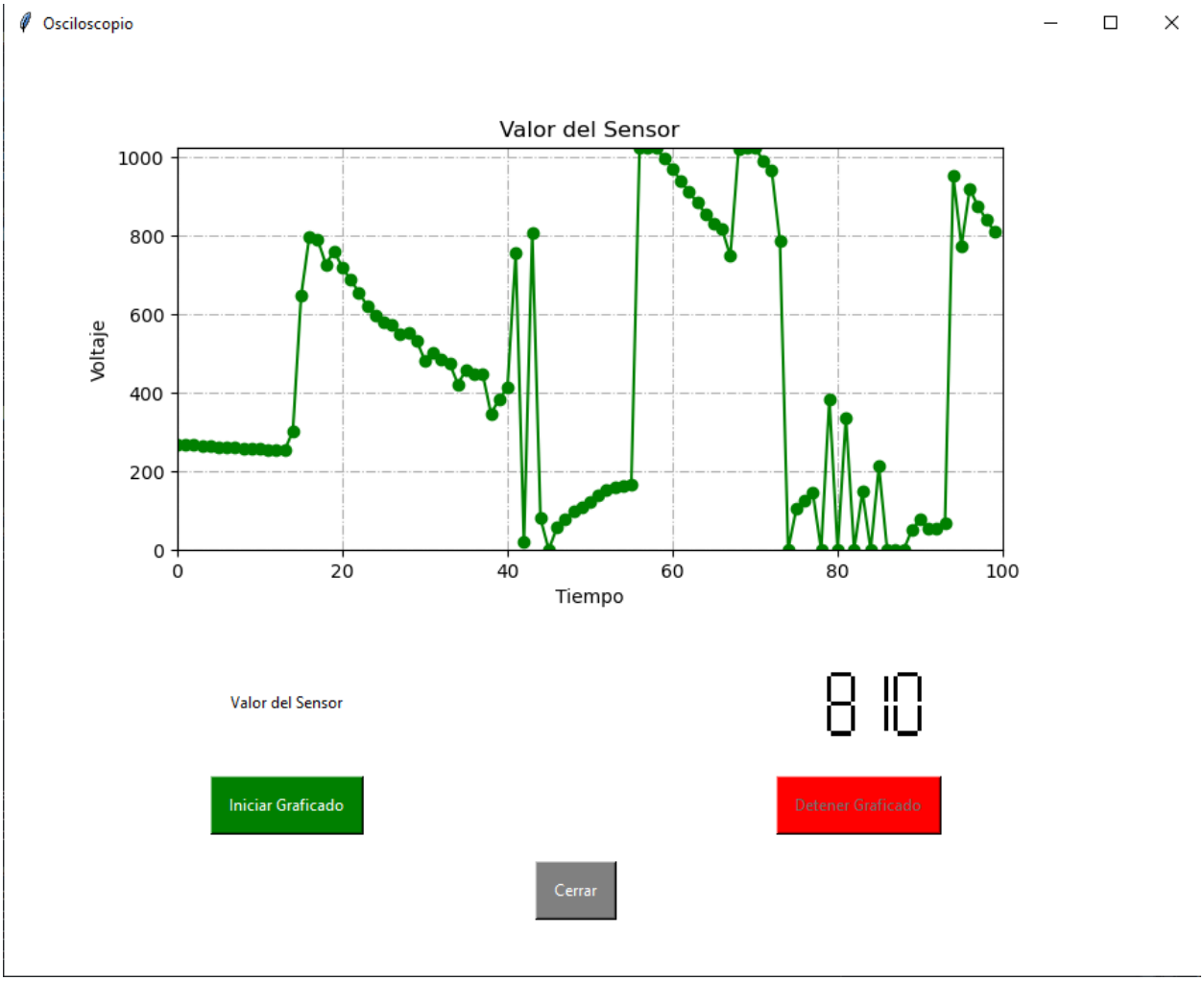
```
//Olmedo Guevara José Ángel
int dato;  //Creamos una variable entera con el nombre de 'dato'

void setup() {  //Inicialización del arduino
  // put your setup code here, to run once:
  Serial.begin(9600); //Iniciamos el puerto serial
  Serial.setTimeout(10); //Establece 10 ms como espera máxima
}

void loop() {  //Ciclo infinito bajo el que operará el arduino
  // put your main code here, to run repeatedly:
  dato=analogRead(A0); //Asignamos a 'dato' lo que registre mi entrada analógica
                       //en la terminal A0 de mi arduino
  Serial.println(dato); //Imprimimos el valor de nuestro dato
  delay(100);  //Establecemos una espera de 100 ms
}

```

Código en Spyder (Python)

```python
@author: Olmedo Guevara José Ángel
"""

import tkinter #Instalamos la biblioteca tkinter para operar código de arduino
import tk_tools # pip instal tk-tools
import serial # pip install pyserial
import numpy as np  #Importamos la biblioteca numpy como np
from matplotlib.figure import Figure  #Importamos la función 'Figure' de matplotlib.figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg #interfaz gráfica

def graficar(): #Creamos una función para graficar nuestros datos
    global condicion, datos   #Generamos dos variables globales (para utilizarlas fuera de la función)

    if condicion == True:  #Si nuestra condición tiene un valor booleano 'TRUE'
        dato=arduino.readline()   #El valor leído de nuestro arduino será asignado a la variable dato
        #dato = '500\n'
        if len(datos) < 100:  #Si la longitud de los datos va de 0 a 99
            datos=np.append(datos, float(dato[0:len(dato)-2]))  #A la variable datos se lea añadirá el elemento del ciclo en el que estemos

        else:  #Si la longitud de los datos es mayor o igual a 100
            datos[0:99]=datos[1:100]  #Reasignamos el valor de las filas 1 a 100  de mi arreglo 'datos' a mis filas 0 a 99
                                      #de mi mismo arreglo 'datos'
            datos[99]=float(dato[0:len(dato)-2])

        linea.set_xdata(np.arange(0,len(datos))) #Generamos el eje x de acuerdo a la longitud de mis datos del arduino
        linea.set_ydata(datos) #Generamos el eje y de acuerdo a la longitud de mi variable 'dato'

        canvas.draw()  #Muestra en pantalla los ejes, y los datos de nuestro arduino

        valor=int(dato[0:len(dato)-2].decode('utf-8')) #Aplicamos formato UTF-8 (Escritura)
        display.set_value(str(valor)) #Muestra en la pantalla el valor que registre mi arduino en Volts

    igu.after(1,graficar)  #Recursividad


def iniciar_graficado(): #Generamos una función para iniciar el graficado de mis valores
    global condicion #Llamamos a la variable condición, previamente declarada como variable global
    condicion=True  #Si mi condición tiene un valor booleano como 'True'
    boton_inicio.config(state='disabled') #El botón para iniciar el graficado está deshabilitado por defecto
    boton_detencion.config(state='normal')
    arduino.reset_input_buffer() #Establece la comunicación entre arduino y python

def detener_graficado(): #Generamos una función para detener el graficado de mis valores
    global condicion   #LLamamos a la variable condición, previamente declarada como variable global
    condicion=False  #Si mi condición tiene un valor booleano como 'True'
    boton_inicio.config(state='normal') #El botón de inicio tiene su configuración por defecto (Establecida previamente)
    boton_detencion.config(state='disabled')  #El botón de detener graficado está deshabilitado por defecto
    arduino.reset_input_buffer() #Establece la comunicación entre arduino y python

def cerrar():  #Generamos la función para detener la pantalla en ejecución
    arduino.close()   #Cerramos la función del arduino
    igu.destroy()   #Cierre total


igu=tkinter.Tk() # Objeto de tkinter

igu.title("Osciloscopio") # título de interfaz
```

```
    igu.title("Osciloscopio") # titulo de interfaz
    igu.geometry("900x700") # tamaño de interfaz
    igu.configure(background="white") # fondo de la interfaz

    #GRÁFICA
    fig=Figure(figsize=(8,4), dpi=100)  #Establecemos el tamaño de nuestro espacio para graficar
    ax=fig.add_subplot(111)
    ax.set_title('Valor del Sensor') #Establecemos el título de la graficadora
    ax.set_xlabel('Tiempo')  #Al eje x establecemos el título de 'Tiempo'
    ax.set_ylabel('Voltaje') #Al eje y establecemos el título de 'Voltaje'
    ax.grid(True,linestyle='-.')  #Establecemos el estilo de línea para graficar
    ax.set_xlim(0,100) #Establecemos límites de espacio para el eje x
    ax.set_ylim(0,1024) #Establecemos límites de espacio para el eje y

    linea=ax.plot([], [], color='green', marker='o', markersize=6)[0]  #Establecemos el color de la línea con el marcador deseado
    canvas = FigureCanvasTkAgg(fig, master=igu)  #Generamos el dibujo
    canvas.draw()


    #CONTROLES
    # Constuir objeto botón, padx y pady es pixeles de espacio del componente
    boton_inicio = tkinter.Button(master=igu, text="Iniciar Graficado", #Establecemos el texto que tendrá el botón de iniciar graficado
                                  bg="green", fg="white", padx=10, pady=10,  #Será de color verde con letras blancas
                                  command=lambda:iniciar_graficado())  #Cuando lo presionemos iniciará el graficado

    boton_detencion = tkinter.Button(master=igu, text="Detener Graficado", #Establecemos el texto que tendrá el botón de detener graficado
                                     bg="red", fg="white", padx=10, pady=10, #Será de color rojo con letras blancas
                                     command=lambda:detener_graficado()) #Cuando lo presionemos se detendrá el graficado

    boton_cerrar = tkinter.Button(master=igu, text="Cerrar", #Establecmos el texto que tendrá el botón para cerrar la pantalla de ejecución
                                  bg="gray", fg="white", padx=10, pady=10, #Será de color gris con letras blancas
                                  command=cerrar)  #Cuando lo precionemos se cerrará

    etiqueta = tkinter.Label(master=igu, text='Valor del Sensor', bg='white')  #Generamos una etiqueta que diga 'Valor del sensor'
    display = tk_tools.SevenSegmentDigits(igu, digits=4, background='white', digit_color='black', height=50) #Será de fondo blanco con
                                                                                                              #números negros
    display.set_value('0')

    #Posicionar mis botones
    canvas.get_tk_widget().grid(row=0, column=0, columnspan=2, rowspan=2, padx=30, pady=30) #Establecemos la posición de mis botones
    boton_inicio.grid(row=3, column=0, pady=20)  #Establezco coordenadas para colocar mi botón de inicio
    boton_detencion.grid(row=3, column=1, pady=20)  #Establezco coordenadas para colocar mi botón de detener graficado
    boton_cerrar.grid(row=4, column=0, columnspan=2) #Va a ocupar dos columnas

    etiqueta.grid(row=2, column=0, pady=10)  #Establezco la posición de mi etiqueta
    display.grid(row=2, column=1, pady=10)

    boton_detencion.config(state='disabled') #El botón de detener graficado por defecto estará deshabilitado
    datos=np.array([]) #Datos será un arreglo de tipo numpy
    condicion=False  #Mi condición en un inicio será un tipo booleano 'False'
    arduino = serial.Serial('COM5', 9600)  #Mi arduino se encuentra en el puerto 'COM5' e inicio comunicación serial
    igu.after(1,graficar)  #Comienza el graficado

    igu.mainloop() # Renderizar el componente
```

Resultado de ejecución




```
In [1]: runfile('C:/Users/josea/Documents/TSPI/ejercicio_24_practica4/osciloscopio.py',
wdir='C:/Users/josea/Documents/TSPI/ejercicio_24_practica4')
```