python

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox

import serial

import serial.tools.list\_ports

# Inicializar la conexión serial

def connect():

    global ser

    port = port\_combobox.get()

    baudrate = baudrate\_combobox.get()

    try:

        if not port:

            raise ValueError("El puerto serial no está especificado")

        ser = serial.Serial(port, int(baudrate))

        connection\_status\_label.config(text="Conectado", foreground="green")

    except Exception as e:

        messagebox.showerror("Error", f"No se pudo conectar: {e}")

        connection\_status\_label.config(text="Desconectado", foreground="red")

def disconnect():

    global ser

    if ser and ser.is\_open:

        ser.close()

    connection\_status\_label.config(text="Desconectado", foreground="red")

def led1\_on():

    if ser and ser.is\_open:

        ser.write(b'LED1\_ON')  # Cambiado a LED1\_ON

def led1\_off():

    if ser and ser.is\_open:

        ser.write(b'LED1\_OFF')  # Cambiado a LED1\_OFF

def led2\_on():

    if ser and ser.is\_open:

        ser.write(b'LED2\_ON')  # Comando para LED2

def led2\_off():

    if ser and ser.is\_open:

        ser.write(b'LED2\_OFF')  # Comando para LED2

def send\_command():

    if ser and ser.is\_open:

        option = option\_spinbox.get()

        if option == "1":

            text = input\_text.get()

            ser.write(f'OPTION1:{text}'.encode())

        elif option == "2":

            ser.write(b'OPTION2')

        elif option == "3":

            value = intensity\_entry.get()

            try:

                value = int(value)

                if 1 <= value <= 255:

                    ser.write(f'OPTION3:{value}'.encode())

                else:

                    messagebox.showwarning("Advertencia", "El valor debe estar entre 1 y 255")

            except ValueError:

                messagebox.showwarning("Advertencia", "El valor debe ser un número entero")

        else:

            messagebox.showwarning("Advertencia", "Selecciona una opción válida")

def update\_ports():

    ports = [port.device for port in serial.tools.list\_ports.comports()]

    port\_combobox['values'] = ports

    if ports:

        port\_combobox.set(ports[0])  # Selecciona el primer puerto disponible

# Crear la ventana principal

root = tk.Tk()

root.title("Interfaz de Control Arduino")

# Crear el Notebook (pestañas)

notebook = ttk.Notebook(root)

notebook.pack(fill='both', expand=True)

# Pestaña 1: Conectar al puerto

tab1 = ttk.Frame(notebook)

notebook.add(tab1, text='Conexión')

# Elementos de la Pestaña 1

ttk.Label(tab1, text="Puerto:").grid(column=0, row=0, padx=5, pady=5)

port\_combobox = ttk.Combobox(tab1)

port\_combobox.grid(column=1, row=0, padx=5, pady=5)

ttk.Label(tab1, text="Baudrate:").grid(column=0, row=1, padx=5, pady=5)

baudrate\_combobox = ttk.Combobox(tab1, values=["9600", "115200", "38400"])

baudrate\_combobox.grid(column=1, row=1, padx=15, pady=5)

baudrate\_combobox.set("9600")

connect\_button1 = ttk.Button(tab1, text="Conectar", command=connect)

connect\_button1.grid(column=0, row=2, padx=15, pady=5)

disconnect\_button1 = ttk.Button(tab1, text="Desconectar", command=disconnect)

disconnect\_button1.grid(column=1, row=2, padx=15, pady=5)

connection\_status\_label = ttk.Label(tab1, text="Desconectado", foreground="red")

connection\_status\_label.grid(column=0, row=3, columnspan=2, pady=5)

# Botones para controlar los LEDs

led\_on\_button1 = ttk.Button(tab1, text="LED ON 1", command=led1\_on)

led\_on\_button1.grid(column=0, row=4, padx=5, pady=5)

led\_off\_button1 = ttk.Button(tab1, text="LED OFF 1", command=led1\_off)

led\_off\_button1.grid(column=1, row=4, padx=5, pady=5)

led\_on\_button2 = ttk.Button(tab1, text="LED ON 2", command=led2\_on)  # Botón para LED2

led\_on\_button2.grid(column=0, row=5, padx=5, pady=5)

led\_off\_button2 = ttk.Button(tab1, text="LED OFF 2", command=led2\_off)  # Botón para LED2

led\_off\_button2.grid(column=1, row=5, padx=5, pady=5)

# Botón para actualizar la lista de puertos

update\_ports\_button = ttk.Button(tab1, text="Actualizar Puertos", command=update\_ports)

update\_ports\_button.grid(column=2, row=5, padx=5, pady=5)

# Pestaña 2: Controlar tareas

tab2 = ttk.Frame(notebook)

notebook.add(tab2, text='Control de Tareas')

# Elementos de la Pestaña 2

ttk.Label(tab2, text="Seleccionar opción:").grid(column=0, row=0, padx=5, pady=5)

option\_spinbox = ttk.Spinbox(tab2, from\_=1, to=4)

option\_spinbox.grid(column=1, row=0, padx=5, pady=5)

input\_text = tk.Entry(tab2)

input\_text.grid(column=0, row=1, columnspan=2, padx=5, pady=5)

intensity\_entry = tk.Entry(tab2)

intensity\_entry.grid(column=0, row=2, columnspan=2, padx=5, pady=5)

send\_button = ttk.Button(tab2, text="Enviar", command=send\_command)

send\_button.grid(column=0, row=3, columnspan=2, pady=5)

root.mainloop()

Arduino

#include <Arduino.h>

// Define los pines de los LEDs

const int ledPin1 = 13;  // Pin digital para el LED 1

const int ledPin2 = 12;  // Pin digital para el LED 2

void setup() {

    Serial.begin(9600);

    pinMode(ledPin1, OUTPUT);  // Configura el pin D13 como salida

    pinMode(ledPin2, OUTPUT);  // Configura el pin D12 como salida

}

void loop() {

    if (Serial.available()) {

        String command = Serial.readStringUntil('\n');

        if (command == "LED1\_ON") {

            digitalWrite(ledPin1, HIGH);  // Enciende el LED 1

        } else if (command == "LED1\_OFF") {

            digitalWrite(ledPin1, LOW);   // Apaga el LED 1

        } else if (command == "LED2\_ON") {

            digitalWrite(ledPin2, HIGH);  // Enciende el LED 2

        } else if (command == "LED2\_OFF") {

            digitalWrite(ledPin2, LOW);   // Apaga el LED 2

        } else if (command.startsWith("OPTION1:")) {

            String text = command.substring(8);

            Serial.println(text.toInt() + 1);  // Ejemplo de respuesta

        } else if (command == "OPTION2") {

            int sensorValue = analogRead(A0);

            Serial.println(sensorValue);

        } else if (command.startsWith("OPTION3:")) {

            int pwmValue = command.substring(8).toInt();

            // Nota: El pin D13 no soporta PWM, considera usar un pin adecuado para PWM

            // Por ejemplo, puedes usar el pin D9 para PWM.

            if (pwmValue >= 0 && pwmValue <= 255) {

                analogWrite(ledPin1, pwmValue);  // Ajusta el brillo del LED 1

            } else {

                Serial.println("Valor de PWM fuera de rango. Debe estar entre 0 y 255.");

            }

        }

    }

}