UNIVERSIDAD DEL BÍO-BÍO

22 Tutorial comunicación serial con robot

Estudiantes Practicantes:
Luis Pereira

Profesor:
Luis Vera

Laboratorio CIMUBB

2023-2

Introducción:

A continuación se explicara el programa con implementación en Python utilizando la biblioteca Tkinter para crear una interfaz gráfica de usuario (GUI) que interactúa con un robot Scorbot a través de una conexión serial. Aquí hay un tutorial paso a paso para entender y modificar el código:

1. Importar bibliotecas:

```
import tkinter as tk
from tkinter import ttk, messagebox
import serial
import threading
import time
```

A diferencia de los tutoriales anteriores, utilizaremos ahora la biblioteca threading para estar leyendo continuamente datos que se recibirán del robot, en otras palabras nos permite manejar la lectura continua de datos en un hilo separado.

2. Clase ScorbotController:

Utilizaremos clases en esta ocasión porque el uso de clases en este caso mejora la claridad, la organización y la mantenibilidad del código, al tiempo que facilita la reutilización y la expansión de la funcionalidad en el futuro.
La clase **ScorbotController** encapsula toda la lógica relacionada con la interacción del Scorbot. Esto significa que las variables y métodos están encapsulados en un solo lugar, lo que facilita su mantenimiento y modificación.
La modularidad se mejora al dividir la funcionalidad en métodos específicos de la clase, como **conectar_robot, desconectar_robot, enviar_instruccion,** y **leer_datos_continuamente.** Esto hace que cada parte del código sea más fácil de entender y modificar sin afectar otras partes.

Funciones en la clase:

1) **conectar_robot(self,port):**

```
def conectar_robot(self, port):
    try:
        self.serial_port = serial.Serial(port, baudrate=9600, timeout=1)
        self.thread_leer_datos = threading.Thread(target=self.leer_datos_continuamente, daemon=True)
        self.thread_leer_datos.start()
        messagebox.showinfo(message="Conexión exitosa con el robot Scorbot.")
    except Exception as e:
        messagebox.showerror(message=f"Error al conectar con el robot Scorbot: {e}")
```

1) self.serial_port = serial.Serial(port, baudrate=9600, timeout=1)

- Crea una instancia de la clase serial.Serial del módulo serial, que se utiliza para la comunicación serial.
- **port**: Es el puerto serial al que se intentará conectar el Scorbot.
- **baudrate**=9600: Establece la velocidad de transmisión de datos en 9600 baudios.
- **timeout**=1: Configura un tiempo de espera de 1 segundo para las operaciones de lectura.

2)self.thread_leer_datos= threading.Thread(target= self.leer_datos_continuamente, daemon=True)

- Crea un hilo (**threading**.**Thread**) que ejecutará la función **self.leer_datos_continuamente.**
- target: Especifica la función que será ejecutada en el hilo.
- **daemon**=**True**: Configura el hilo como un hilo demonio, lo que significa que se cerrará cuando el programa principal termine.

3) self.thread_leer_datos.start()

- Inicia el hilo creado en el paso anterior, lo que permite que la función **leer_datos_continuamente** se ejecute en paralelo con el resto del programa.

4) messagebox.showinfo(message="Conexión exitosa con el robot Scorbot.")

- Si la conexión serial y el hilo se establecen con éxito, se muestra un mensaje informativo de éxito utilizando la función showinfo del módulo messagebox de Tkinter.

except Exception as e:

- Si algo sale mal durante el intento de conexión (por ejemplo, si el puerto no está disponible o hay algún otro error), se captura la excepción y se muestra un mensaje de error utilizando **showerror** del módulo messagebox.

En resumen, esta función intenta establecer una conexión serial con el robot Scorbot a través del puerto especificado. Si la conexión es exitosa, inicia un hilo para leer datos continuamente y muestra un mensaje de éxito. Si hay algún error durante el proceso, muestra un mensaje de error.

Observación: el baud rate en 9600 bits por segundo. Esto significa que cada segundo, el Scorbot y el dispositivo conectado intercambiarán información a una velocidad de 9600 bits por segundo. Es crucial que el Scorbot esté configurado con la misma velocidad de baudios para garantizar una comunicación efectiva.

2) **desconectar_robot(self):**

```
def desconectar_robot(self):
    try:
        if self.thread_leer_datos and self.thread_leer_datos.is_alive():
            self.thread_leer_datos.join()
        if self.serial_port and self.serial_port.isOpen():
            self.serial_port.close()
            messagebox.showinfo(message="Desconexión exitosa del robot Scorbot.")
    except Exception as e:
        messagebox.showerror(message=f"Error al desconectar el robot Scorbot: {e}")
```

La función **desconectar_robot** se encarga de cerrar la conexión con el robot Scorbot. En primer lugar, verifica si el hilo de lectura de datos está en ejecución y, en caso afirmativo, espera a que termine antes de continuar. Luego, comprueba si la conexión serial está abierta y la cierra. Después de realizar estas acciones, muestra un mensaje informativo indicando que la desconexión fue exitosa. En caso de que ocurra algún error durante el proceso, captura la excepción y muestra un mensaje de error detallado. En resumen, la función asegura una desconexión controlada y proporciona retroalimentación al usuario.

3) **enviar_instruccion(self,instruccion):**

```python
def enviar_instruccion(self, instruccion):
    try:
        if self.serial_port and self.serial_port.isOpen():
            # Limpiar el búfer antes de enviar una nueva instrucción
            self.response_buffer = ""

            # Agregar un carácter de retorno de carro al final de la instrucción
            instruccion = f"{instruccion}\r"
            self.serial_port.write(instruccion.encode())
            messagebox.showinfo(message="Instrucción enviada correctamente.")

            # Agregar una pausa antes de leer los datos
            time.sleep(0.23)

            # Leer datos después de la pausa
            datos = self.serial_port.read_all().decode()
            time.sleep(0.15)
            if datos:
                TextRecibidos.insert(tk.END, datos)
                if "Done" in datos or "OK" in datos:
                    TextInterpretacion.insert(tk.END,"Se recibio un Done o un OK en los datos recibidos" + "\n")
    except Exception as e:
        messagebox.showerror(message=f"Error al enviar la instrucción al robot Scorbot: {e}")
```

1) **Verificar la conexión serial:**

```python
if self.serial_port and self.serial_port.isOpen():
```

Verifica que la conexión serial (self.serial_port) esté establecida y abierta (isOpen()). Si la conexión no está abierta, la función no continuará y mostrará un mensaje de error.

2) **Limpiar el búfer y enviar la instrucción:**

```python
self.response_buffer = ""

# Agregar un carácter de retorno de carro al final de la instrucción
instruccion = f"{instruccion}\r"
self.serial_port.write(instruccion.encode())
```

- Limpia el búfer **(self.response_buffer)** antes de enviar una nueva instrucción.
- Agrega un carácter de retorno de carro **(\r)** al final de la instrucción.
- Utiliza write para enviar la instrucción codificada al Scorbot a través de la conexión serial.

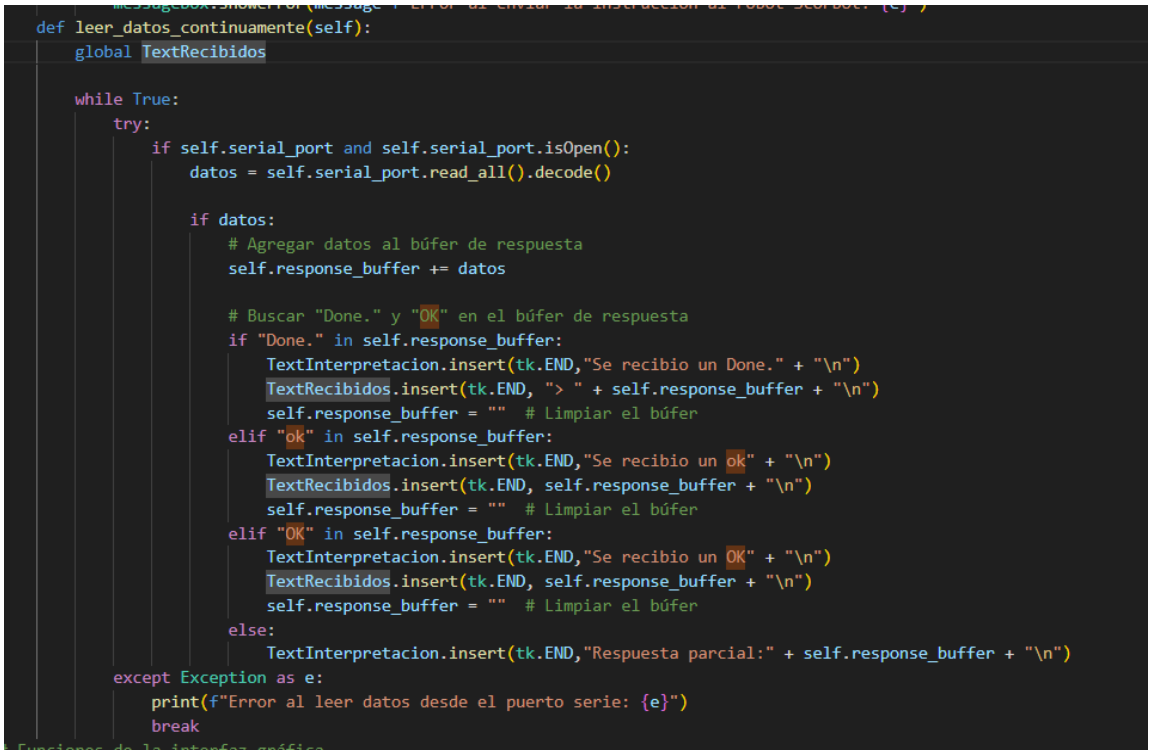
3) **Agregar pausas antes y después de leer datos:**

```python
# Agregar una pausa antes de leer los datos
time.sleep(0.23)

# Leer datos después de la pausa
datos = self.serial_port.read_all().decode()
time.sleep(0.15)
```

En resumen, la función **enviar_instruccion()** se encarga de enviar una instrucción al robot Scorbot a través de la conexión serial. Comienza verificando que la conexión esté establecida y abierta. Luego, limpia el búfer, agrega un retorno de

carro a la instrucción y la envía al Scorbot. Después de enviar la instrucción, muestra un mensaje de éxito. Añade pausas antes y después de leer datos y muestra los datos recibidos en la interfaz gráfica. Además, verifica si la respuesta contiene "**Done**" o "**OK**" y muestra un mensaje correspondiente. En caso de cualquier error durante el proceso, captura la excepción y muestra un mensaje de error detallado. En resumen, la función controla el envío de instrucciones, gestiona la recepción de datos y proporciona retroalimentación al usuario en la interfaz gráfica.

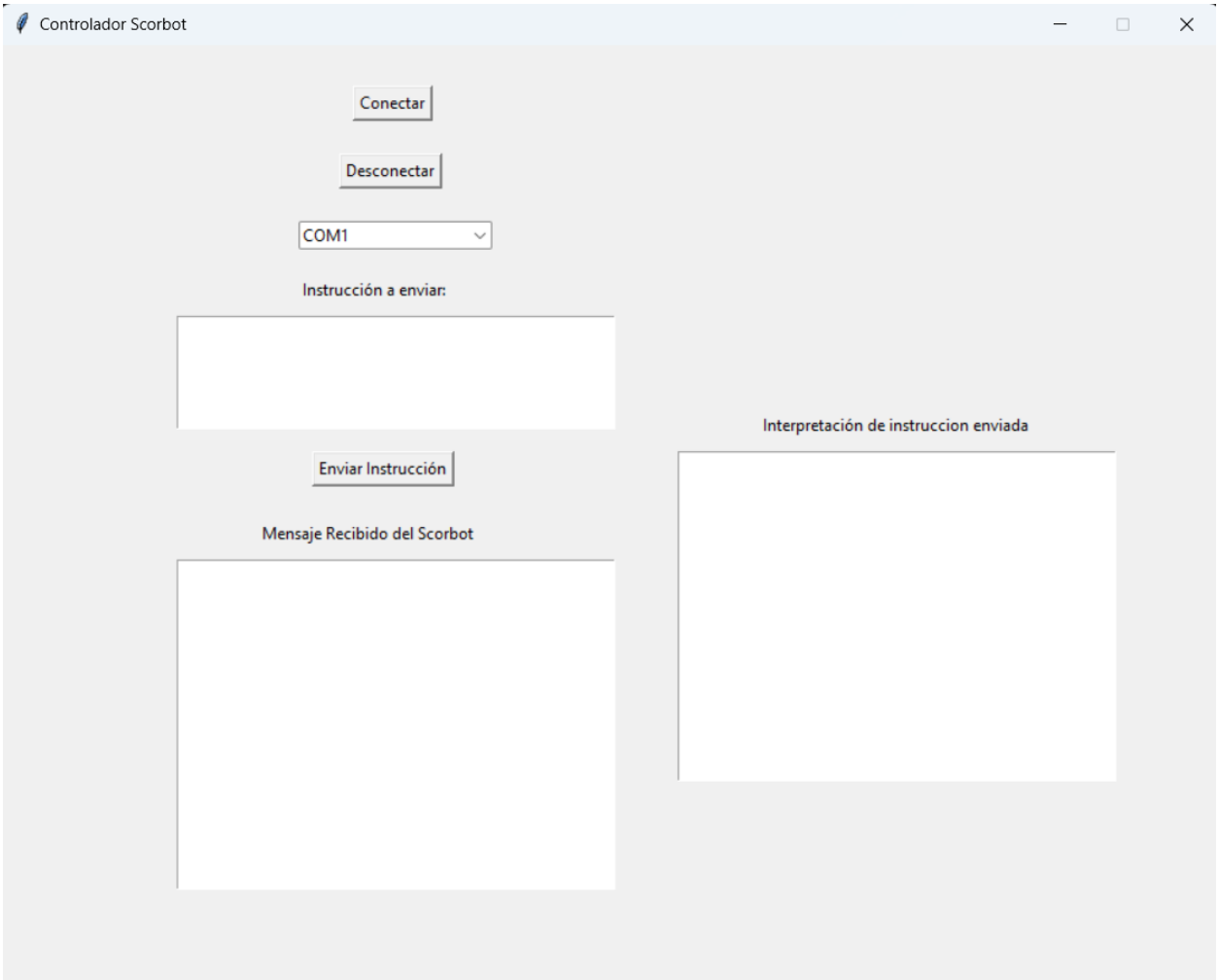
4) leer_datos_continuamente(self):



```
def leer_datos_continuamente(self):
    global TextRecibidos

    while True:
        try:
            if self.serial_port and self.serial_port.isOpen():
                datos = self.serial_port.read_all().decode()

                if datos:
                    # Agregar datos al búfer de respuesta
                    self.response_buffer += datos

                    # Buscar "Done." y "OK" en el búfer de respuesta
                    if "Done." in self.response_buffer:
                        TextInterpretacion.insert(tk.END,"Se recibio un Done." + "\n")
                        TextRecibidos.insert(tk.END, "> " + self.response_buffer + "\n")
                        self.response_buffer = ""  # Limpiar el búfer
                    elif "ok" in self.response_buffer:
                        TextInterpretacion.insert(tk.END,"Se recibio un ok" + "\n")
                        TextRecibidos.insert(tk.END, self.response_buffer + "\n")
                        self.response_buffer = ""  # Limpiar el búfer
                    elif "OK" in self.response_buffer:
                        TextInterpretacion.insert(tk.END,"Se recibio un OK" + "\n")
                        TextRecibidos.insert(tk.END, self.response_buffer + "\n")
                        self.response_buffer = ""  # Limpiar el búfer
                    else:
                        TextInterpretacion.insert(tk.END,"Respuesta parcial:" + self.response_buffer + "\n")
        except Exception as e:
            print(f"Error al leer datos desde el puerto serie: {e}")
            break
```

La función **leer_datos_continuamente()** se ejecuta en un hilo separado para leer continuamente los datos del puerto serie del robot Scorbot. En un bucle infinito, verifica la conexión serial, lee los datos disponibles y los procesa. Si encuentra "**Done**." o "**OK**" en la respuesta, muestra mensajes específicos en la interfaz gráfica. Si no se encuentra ninguna de estas cadenas, asume que la respuesta es parcial y la muestra en la interfaz. La función maneja excepciones para asegurar un comportamiento controlado y termina si se produce algún error durante la lectura de datos.

Resultado final



Tenemos 3 cajas de texto donde podemos enviar instrucciones, ver el mensaje recibido por el robot y además la interpretación de lo que se recibe del scorbot, también en la interpretación de la instrucción enviada podemos ver paso a paso cómo funciona la lectura continua en un hilo separado.

Advertencia: Este programa, en su estado actual, ha demostrado un comportamiento intermitente al trabajar con el robot Scorbot debido a problemas de sincronización. En algunos casos, la respuesta esperada "**Done**" puede no ser devuelta consistentemente. Sin embargo, en el contexto específico de la aplicación que estamos considerando, que implica la recepción de la confirmación "**OK**", el programa ha demostrado un funcionamiento adecuado y fiable. Se recomienda tener en cuenta esta limitación y considerar posibles ajustes o mejoras según las necesidades específicas de su implementación con el Scorbot.