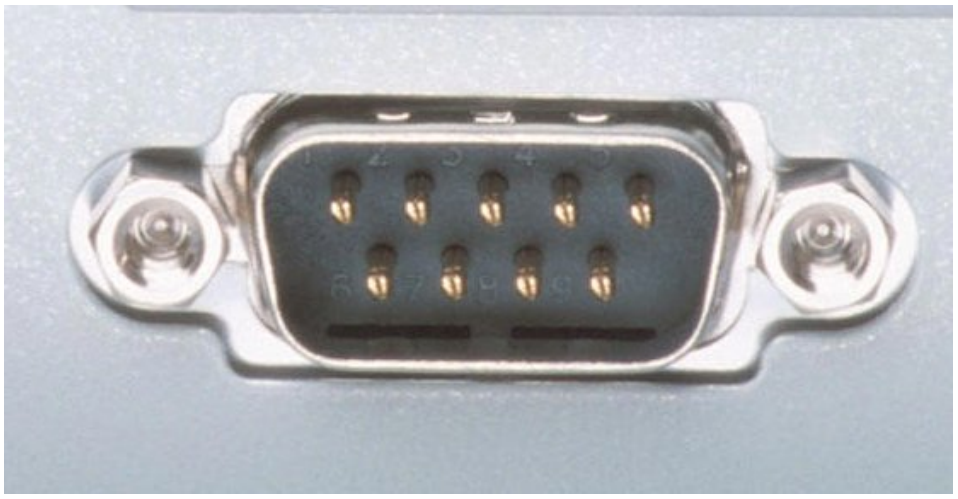
Imię i Nazwisko	Nr Indeksu:	Grupa	Data wykonania ćwiczenia:
Lidia Teleszko	325708	GI b	17.01.2024 r.
Julia Zapała	325710		

1. Cel ćwiczenia

Celem zadania było stworzenie programu do obsługi portu szeregowego na komputerze za pomocą interfejsu graficznego, który miał odczytywać i zapisywać dane wysyłane przez urządzenie (w tym przypadku niwelator).

1. Wstęp teoretyczny

Porty szeregowe stanowią istotny element współczesnych systemów komunikacyjnych, ponieważ pozwalają na przesyłanie danych między różnymi urządzeniami. Port szeregowy jest to złącze umieszczane na płycie głównej komputera PC lub urządzenia sieciowego. Przesyłanie danych odbywa się bit po bicie. Najbardziej znanymi złączami szeregowymi COM są: COM1, COM2, COM3, COM4. Porty COM w komunikacji Windows zwykle są reprezentowane przez złącza RS-232. Obecnie porty szeregowe nie są już tak popularne jak kilka lub kilkanaście lat wcześniej. Zostały zastąpione m. in. przez złącza USB oraz Bluetooth.



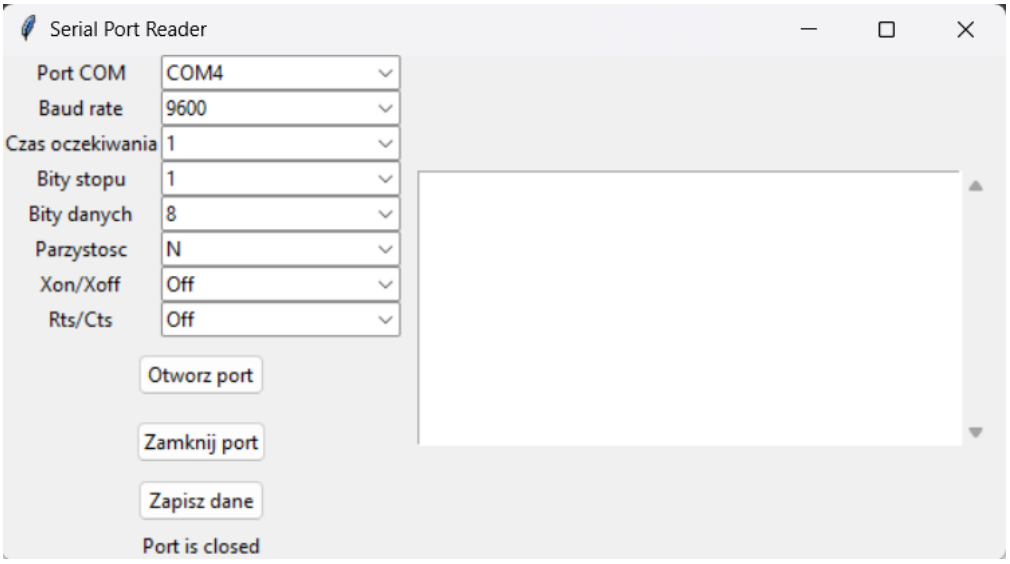
Rys. 1 Przykładowe złącze COM

Niwelator jest to instrument geodezyjny umożliwiający pomiar różnicy wysokości pomiędzy punktami pomiarowymi. Jest niezbędny w pracach pomiarowych, budowlanych oraz inżynierskich, gdzie dokładność pomiaru odgrywa kluczową rolę. Obecne niwelatory mają możliwość łączenia się z komputerem lub innym urządzeniem za pomocą m.in. portu szeregowego, wysyłanie wyników pomiarów oraz możliwość ich zapisu. Takie rozwiązanie znacznie polepszyło jakość i szybkość prac wykonanych przez geodetów. Zapisane dane mogą być archiwizowane jako zasoby internetowe, co zmniejsza biurokrację oraz umożliwia przesyłanie wyników w bardziej czytelnej wersji dla całego zespołu.

Rys. 2 Przykładowy niwelator

2. Opis przebiegu ćwiczenia

Program został zaimplementowany w języku Python z wykorzystaniem biblioteki serial oraz Tkinter. Został stworzony w celu umożliwienia łatwej obsługi portu szeregowego poprzez intuicyjny interfejs. Głównym założeniem programu jest umożliwienie użytkownikowi nawiązanie i zarządzanie połączeniem z niwelatorem poprzez port szeregowy. Poniżej przedstawiony jest wygląd interfejsu.



Rys. 4 Interfej graficzny programu

Program umożliwia użytkownikowi precyzyjną konfigurację parametrów transmisji. Poniżej przedstawione zostaną kluczowe parametry wraz z krótką charakterystyką:

- “Port COM” umożliwia wybranie jednego z 5 portów COM czyli fizycznych interfejsów szeregowej komunikacji Windows.
- “Baud Rate” pozwala użytkownikowi wybranie prędkości transmisji (musi być zgodna z ustawieniami niwelatora).
- “Timeout” pozwala określić maksymalny czas oczekiwania na pojawienie się danych w buforze. Jest przedstawiana w sekundach.
- “Bity stopu” umożliwiają wybór ile bitów stopów będzie wysyłanych po każdym bajcie podczas zakończenia transmisji.
- “Bity danych” pozwalają określić liczbę bitów reprezentujących pojedynczy bajt danych. Domyślnie wybiera się 8 bitów danych.
- “Parzystość” umożliwia wybór mechanizmu kontroli parzystości niezbędnego do wykrywania błędów transmisji.
- “Xon/Xoff” pozwala na wybór mechanizmu sterowania przepływem danych za pomocą XON czyli Start of Text oraz XOFF czyli Stop Transmission. Jeśli bufor odbiornika jest zapełniony, XOFF jest wysyłany, aby zatrzymać nadawanie, a gdy bufor jest gotowy, XON jest wysyłany, aby wznowić nadawanie.
- “Rts/Cts” umożliwia sterowanie przepływem. Urządzenie nadawcze używa sygnału RTS, aby sprawdzić, czy urządzenie odbiorcze jest gotowe do odbioru danych (sygnał CTS).

Uruchomienie przesyłu danych odbywa się po otworzeniu portu, natomiast dane są wyświetlane w czasie rzeczywistym w okienku thread. Użytkownik ma możliwość zapisu odczytanych danych w pliku tekstowym txt. Dodatkowo program obsługuje wykrywanie błędów oraz informuje o napotkanych problemach przy pomocy tzw. messagebox. Program dodatkowo sprawdza stan połączenia przed próbą zamknięcia co minimalizuje ryzyko wystąpienia błędów.

3. Wnioski

Tworzenie programów z prostym interfejsem graficznym jest kluczowe dla zapewnienia łatwej obsługi dla osób bez większym umiejętności programistycznych.
Dynamiczny odczyt danych z portu szeregowego umożliwia użytkownikowi ciągły monitoring przesyłanych wyników pomiarowych.
Funkcja zapisu danych do pliku tekstowego pozwala na archiwizację wyników pomiarów.
Program stanowi praktyczne narzędzie zarówno dla doświadczonych geodetów jak i osób spoza branży.

4. Kod źródłowy

```
import tkinter as tk
from tkinter import ttk
from tkinter import scrolledtext
from tkinter import filedialog
import threading
import serial

serialInst = serial.Serial()

def open_port_command():
    try:
        port = port_var.get()
        baudrate= int(baudrate_var.get())
        timeout = float(timeout_var.get())
        stopbits = int(stopbits_var.get())
        databits = int(databits_var.get())
        parity = parity_var.get()
        xonxoff = xonxoff_var.get()
        rtscts = rtscts_var.get()
        #configuration
        serialInst.port = port
        serialInst.baudrate = baudrate
        serialInst.timeout = timeout
        serialInst.stopbits = stopbits
        serialInst.bytesize= databits
        serialInst.parity = parity
        if xonxoff == "On":
            serialInst.xonxoff = True
        else:
            serialInst.xonxoff = False
        if rtscts == "On":
            serialInst.rtscts = True
        else:
            serialInst.rtscts = False
        serialInst.open()
        port_status_var.set(f"Port {port} is open")
        start_read_thread()
    except Exception as e:
        tk.messagebox.showerror("Error", f"Error opening port: {e}")


def close_port_command():
    try:
        if serialInst.is_open:
            serialInst.close()
            port_status_var.set("Port is closed")
            stop_read_thread()
        else:
            port_status_var.set("Port is already closed")
    except Exception as e:
        tk.messagebox.showerror("Error", f"Error closing port: {e}")
```

```
def start_read_thread():
    global read_thread_running
    read_thread_running = True
    read_thread.start()

def stop_read_thread():
    global read_thread_running
    read_thread_running = False
    read_thread.join()


def read_data():
    try:
        if read_thread_running and serialInst.in_waiting > 0:
            data = serialInst.read(1000)
            if data:
                output_text.insert(tk.END, data.decode())
                output_text.yview(tk.END)
        root.after(1, read_data)
    except Exception as e:
        tk.messagebox.showerror("Błąd", f"Błąd odczytu danych: {e}")

def save_data_command():
    try:
        file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text files", "*.txt")])
        if file_path:
            with open(file_path, "w") as file:
                file.write(output_text.get(1.0, tk.END))
    except Exception as e:
        tk.messagebox.showerror("Error", f"Error saving file: {e}")


# GUI setup
root = tk.Tk()
root.title("Serial Port Reader")

# Variables
port_var = tk.StringVar(value="COM4")
baudrate_var = tk.StringVar(value="9600")
timeout_var = tk.StringVar(value="1")
stopbits_var = tk.StringVar(value="1")
databits_var = tk.StringVar(value="8")
parity_var = tk.StringVar(value='N')
xonxoff_var = tk.StringVar(value="Off")
rtscts_var = tk.StringVar(value="Off")
line_var = tk.StringVar(value="")
port_status_var = tk.StringVar(value="Port is closed")
```

```
# Labels and Comboboxes
labels = ["Port COM", "Baud rate", "Czas oczekiwania", "Bity stopu", "Bity danych", "Parzystosc", "Xon/Xoff", "Rts/Cts"]
combobox_vars = [port_var, baudrate_var, timeout_var, stopbits_var, databits_var, parity_var, xonxoff_var, rtscts_var]
combobox_options = [
    ["COM1", "COM2", "COM3", "COM4", "COM5"],
    ["9600", "19200", "38400", "57600", "115200"],
    ["1", "0"],
    ["1", "1.5", "2"],
    ["5", "6", "7", "8"],
    ['N', 'E', 'O', 'M','S'],
    ["Off", "On"],
    ["Off", "On"]
    ]

for label, combobox_var, options in zip(labels, combobox_vars, combobox_options):
    ttk.Label(root, text=label).grid(column=0, row=labels.index(label))
    ttk.Combobox(root, textvariable=combobox_var, values=options).grid(column=1, row=labels.index(label))

# Buttons
ttk.Button(root, text="Otworz port", command=open_port_command).grid(column=0, row=len(labels), columnspan=2, pady=10)
ttk.Button(root, text="Zamknij port", command=close_port_command).grid(column=0, row=len(labels)+1, columnspan=2, pady=5)
ttk.Button(root, text="Zapisz dane", command=save_data_command).grid(column=0, row=len(labels)+2, columnspan=2, pady=5)

# Output Textbox
output_text = scrolledtext.ScrolledText(root, width=40, height=10)
output_text.grid(column=2, row=0, rowspan=len(labels)+4, padx=10, pady=10)

# Status Label
ttk.Label(root, textvariable=port_status_var).grid(column=0, row=len(labels)+3, columnspan=2)

# Thread for reading data
read_thread = threading.Thread(target=read_data, daemon=True)
read_thread_running = False

root.mainloop()
```