

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"



**АВТОМАТИЗОВАНЕ ПРОЕКТУВАННЯ КОМП'ЮТЕРНИХ
СИСТЕМ**

Лабораторна робота 2

Виконав:
студент групи КІ-401
Шаклеїн О.Т.
Прийняв:
Шпіцер А.С

Львів 2024

Тема: ознайомлення з комунікаційними інтерфейсами.

Порядок виконання лабораторної роботи:

1. Create a simple schema SW(client) <-> UART <-> HW(bridge) <-> HW i-fase <-> HW(server).

NOTE: that SW(client) is NOT a terminal or other downloaded SW. It is SW developed by students.

2. The client should send a message through the bridge to the server. The server should modify the message and send it back to the client through the bridge.
3. Required steps.

Виконання роботи

1 Налаштування com0com

1. Встановив com0com на Windows.
2. Створив пару віртуальних портів COM12 і COM11 за допомогою командного рядка com0com:

install PortName=COM12 PortName=COM11

3. Переконався, що порти створено та працюють коректно.

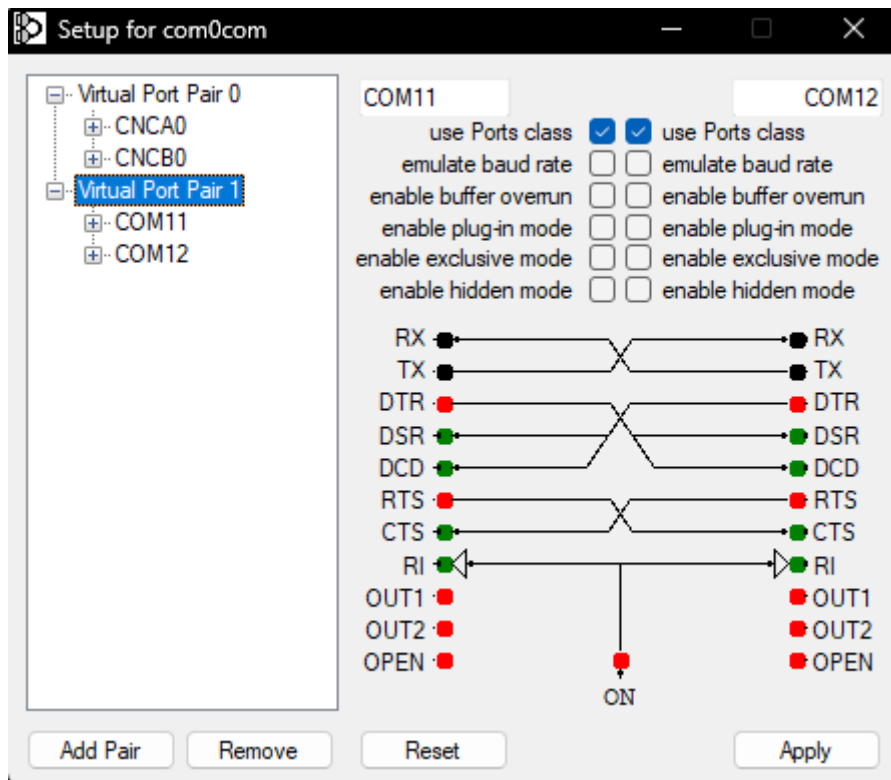


Рис. 1. Налаштування com0com та створення віртуальних портів.

2 Створення Python-сервера

1. Розробив скрипт server.py, який прослуховує порт COM11, приймає повідомлення від клієнта, змінює їх та відправляє назад.

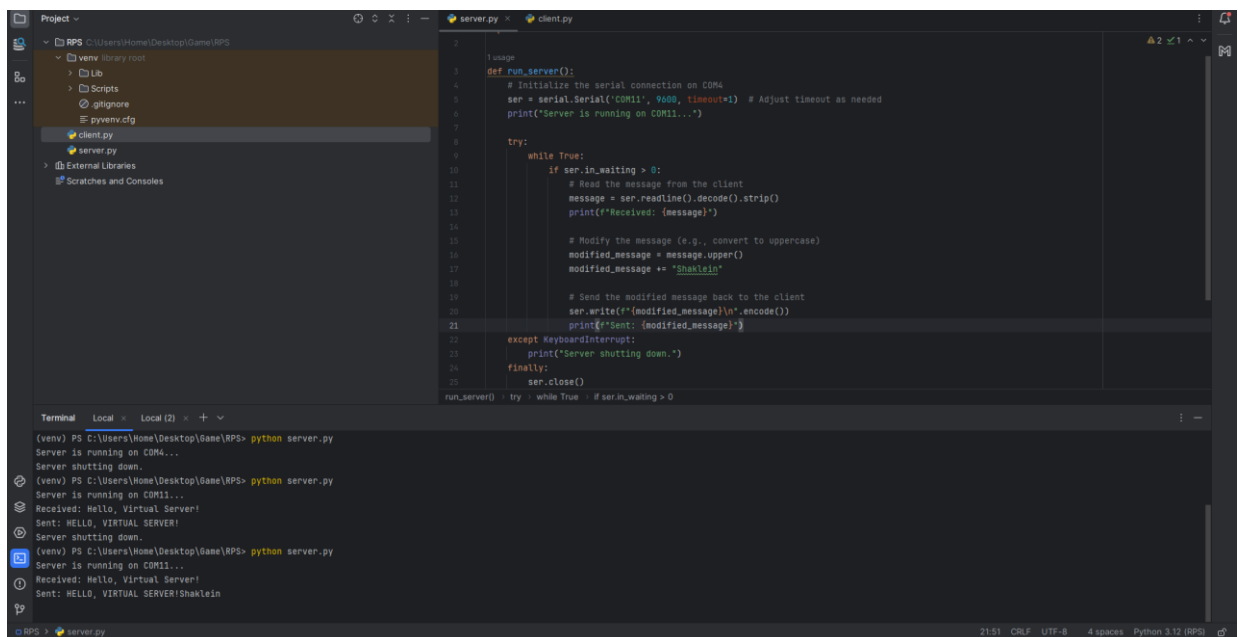
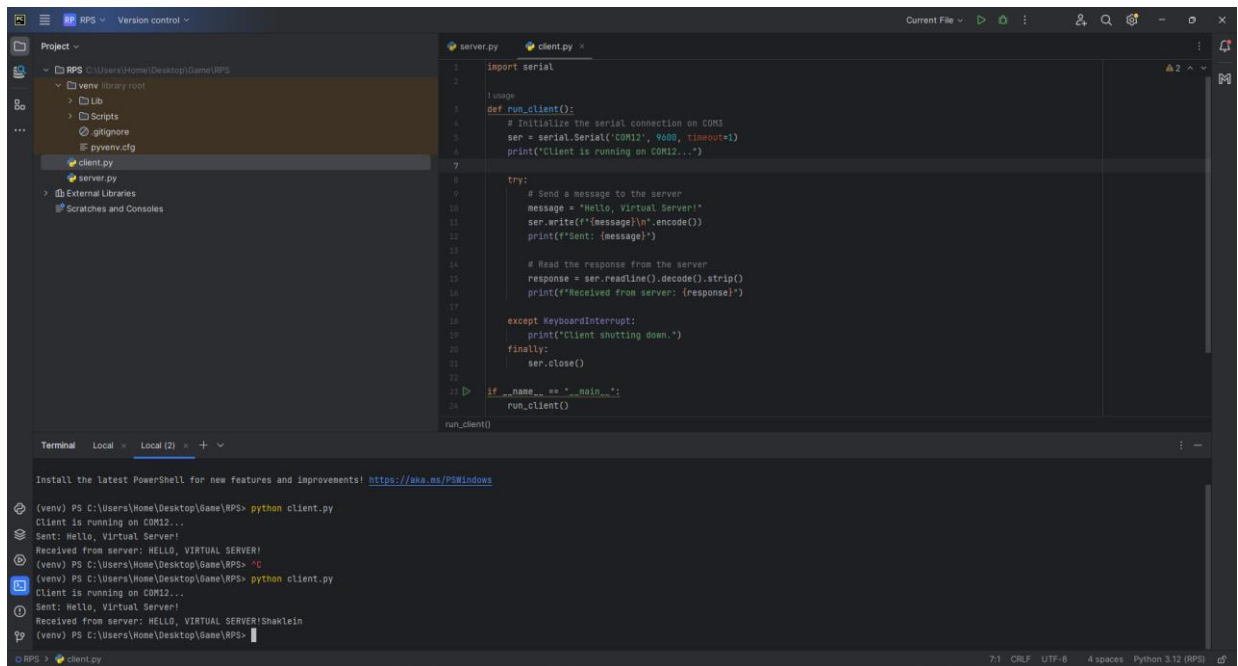


Рис. 2. Запуск сервера та вивід отриманих даних.

3 Створення Python-клієнта

1. Створив скрипт client.py, який підключається до порту COM12, надсилає повідомлення серверу та отримує відповідь.



The screenshot shows a code editor with a project named 'RPS' and a file named 'client.py'. The code in 'client.py' is as follows:

```
1 import serial
2
3 def run_client():
4     # Initialize the serial connection on COM3
5     ser = serial.Serial('COM12', 9600, timeout=1)
6     print("Client is running on COM12...")
7
8     try:
9         # Send a message to the server
10        message = "Hello, Virtual Server!"
11        ser.write(message.encode())
12        print(f"Sent: {message}")
13
14        # Read the response from the server
15        response = ser.readline().decode().strip()
16        print(f"Received from server: {response}")
17
18    except KeyboardInterrupt:
19        print("Client shutting down.")
20    finally:
21        ser.close()
22
23 if __name__ == "__main__":
24     run_client()
```

The terminal output shows the following commands and results:

```
(venv) PS C:\Users\Home\Desktop\Game\RPS> python client.py
Client is running on COM12...
Sent: Hello, Virtual Server!
Received from server: HELLO, VIRTUAL SERVER!
(venv) PS C:\Users\Home\Desktop\Game\RPS> ^C
(venv) PS C:\Users\Home\Desktop\Game\RPS> python client.py
Client is running on COM12...
Sent: Hello, Virtual Server!
Received from server: HELLO, VIRTUAL SERVER!Shaklein
(venv) PS C:\Users\Home\Desktop\Game\RPS> ^
```

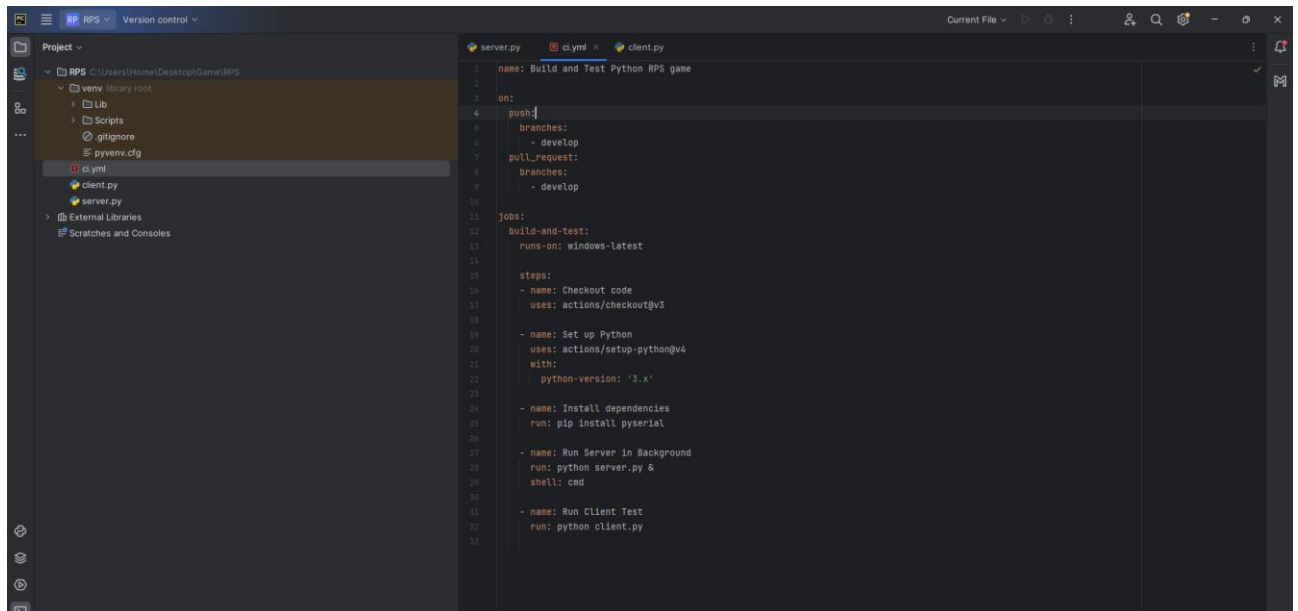
Рис. 3. Запуск клієнта та вивід отриманих даних.

4. Результати

- Реалізував емуляцію зв'язку між програмним клієнтом та сервером за допомогою віртуальних портів com0com.
- Клієнт успішно надіслав повідомлення на COM12, сервер обробив його на COM11 та повернув змінене повідомлення.
- Забезпечив коректну роботу системи

5. Налаштування YML

Після цього налаштував YML-файл для автоматизації тестування та побудови проекту. У файлі .github/workflows/ci.yml було описано кроки, які включають налаштування середовища, запуск серверного скрипта у фоновому режимі, запуск клієнта та автоматичне тестування зв'язку.



Висновок: на цій лабораторній роботі я ознайомилася з комунікаційними інтерфейсами.

Додатки

client.py

```

import serial

def run_client():
    # Initialize the serial connection on COM3
    ser = serial.Serial('COM12', 9600, timeout=1)
    print("Client is running on COM12...")

    try:
        # Send a message to the server
        message = "Hello, Virtual Server!"
        ser.write(f"{message}\n".encode())
        print(f"Sent: {message}")

        # Read the response from the server
        response = ser.readline().decode().strip()
        print(f"Received from server: {response}")

    except KeyboardInterrupt:
        print("Client shutting down.")
    finally:
        ser.close()

if __name__ == "__main__":
    run_client()

server.py
import serial

def run_server():
    # Initialize the serial connection on COM4
    ser = serial.Serial('COM11', 9600, timeout=1) # Adjust timeout as needed

```

```

print("Server is running on COM11...")

try:
    while True:
        if ser.in_waiting > 0:
            # Read the message from the client
            message = ser.readline().decode().strip()
            print(f"Received: {message}")

            # Modify the message (e.g., convert to uppercase)
            modified_message = message.upper()
            modified_message += "Shaklein"

            # Send the modified message back to the client
            ser.write(f"{modified_message}\n".encode())
            print(f"Sent: {modified_message}")
except KeyboardInterrupt:
    print("Server shutting down.")
finally:
    ser.close()

if __name__ == "__main__":
    run_server()

```

ci.yml

name: Build and Test Python Client-Server

```

on:
  push:
    branches:
      - develop
  pull_request:
    branches:
      - develop

```

```

jobs:
  build-and-test:
    runs-on: windows-latest

```

```

steps:
  - name: Checkout code
    uses: actions/checkout@v3

  - name: Set up Python
    uses: actions/setup-python@v4
    with:
      python-version: '3.x'

  - name: Install dependencies
    run: pip install pyserial

  - name: Run Server in Background
    run: python server.py &
    shell: cmd

  - name: Run Client Test
    run: python client.py

```