ЗВІТ

з лабораторної роботи **№3**

з дисципліни “Автоматизоване проектування комп’ютерних систем”

Виконала:
ст.гр. КІ-401
Кріль Катерина
Перевірив:
Федак П. Р.

Львів - 2024

Ініціалізація Git репозиторію

Хід роботи:

1. Розробіть сервер та клієнт.
2. Обов'язкові кроки.

Student number	Game	config format
1	tik-tac-toe 3x3	XML
2	rock paper scissors	JSON
3	tik-tac-toe 3x3	INI
4	rock paper scissors	XML
5	tik-tac-toe 3x3	JSON
6	rock paper scissors	INI
7	tik-tac-toe 3x3	XML
8	rock paper scissors	JSON
9	tik-tac-toe 3x3	INI
10	rock paper scissors	XML
11	tik-tac-toe 3x3	JSON
12	rock paper scissors	INI
13	tik-tac-toe 3x3	XML
14	rock paper scissors	JSON
15	tik-tac-toe 3x3	INI
16	rock paper scissors	XML
17	tik-tac-toe 3x3	JSON
18	rock paper scissors	INI
19	tik-tac-toe 3x3	XML
20	rock paper scissors	JSON
21	tik-tac-toe 3x3	INI
22	rock paper scissors	XML
23	tik-tac-toe 3x3	JSON
24	rock paper scissors	INI
25	tik-tac-toe 3x3	XML
26	rock paper scissors	JSON
27	tik-tac-toe 3x3	INI
28	rock paper scissors	XML
29	tik-tac-toe 3x3	JSON
30	rock paper scissors	INI
31	tik-tac-toe 3x3	XML
32	rock paper scissors	JSON
33	tik-tac-toe 3x3	INI
34	rock paper scissors	XML
35	tik-tac-toe 3x3	JSON

Табл.1 Завдання

Виконання роботи:

Реалізація на С:

```
#ifndef COMMUNICATION_H
#define COMMUNICATION_H

#include <stdio.h>
#include <string.h>

#include "project.h"
#include "types.h"
#include "constants.h"

//! Buffer to receive communication data
static u8 communication_receive_buffer[RECEIVE_BUFFER_LENGTH];

//! Buffer to send communication data
static u8 communication_send_buffer[SEND_BUFFER_LENGTH];

/**
 * @brief Initializes the communication system.
 */
static inline void communication_start(void)
{
    UART_Start();
    setvbuf(stdin, NULL, _IONBF, 0);
}

/**
 * @brief Receives a message through the communication channel.
 *
 * @param buffer The buffer to store the received message.
 * @param length The length of the message to be received.
 */
static inline void receive_message(u8 *buffer, u8 length)
{
    if ((length + 2u > RECEIVE_BUFFER_LENGTH) || buffer == NULL)
return;

    UART_GetArrayBlocking(communication_receive_buffer, length + 2u);

    if ((communication_receive_buffer[0] == PACKET_START_VALUE)
        && (communication_receive_buffer[length + 1u] ==
PACKET_END_VALUE))
    {
        for (u8 index = 0u; index < length; index++)
            buffer[index] = communication_receive_buffer[index + 1u];

        memset(communication_receive_buffer, 0u,
RECEIVE_BUFFER_LENGTH);
    }
}

/**
 * @brief Sends a message through the communication channel.
 *
 * @param buffer The buffer containing the message to be sent.
 * @param length The length of the message to be sent.
 */
```

```
static inline void send_message(u8 *buffer, u8 length)
{
    if ((length + 2u > SEND_BUFFER_LENGTH) || buffer == NULL) return;

    communication_send_buffer[0] = PACKET_START_VALUE;
    communication_send_buffer[length + 1u] = PACKET_END_VALUE;

    for (u8 index = 1u; index < length + 1u; index++)
        communication_send_buffer[index] = buffer[index - 1u];

    UART_PutArrayBlocking(communication_send_buffer,
SEND_BUFFER_LENGTH);

    memset(communication_send_buffer, 0u, SEND_BUFFER_LENGTH);
}

#endif // COMMUNICATION_H
```

Реалізація на Python:

```
import serial

from constants import PACKET_END_VALUE, PACKET_START_VALUE, \
    RECEIVE_BUFFER_LENGTH, ACK_PACKET


class Channel:
    def __init__(self, com: str) -> None:
        self._com_ = com

        self._port_ = serial.Serial(self._com_, 115200,
timeout=0.005)

    def __del__(self) -> None:
        self._port_.close()

    def send_message(self, buffer: bytes, send_ack: bool = False,
info: bool = False) -> None:
        packet = self.receive_message()
        send_packet = PACKET_START_VALUE + buffer + PACKET_END_VALUE
        self._port_.write(send_packet)

        if send_ack:
            while packet != ACK_PACKET:
                packet = self.receive_message()
                send_packet = PACKET_START_VALUE + buffer +
PACKET_END_VALUE

                if info:
                    print(f"[INFO] Sending: {send_packet}")

                self._port_.write(send_packet)
        else:
            if info:
                print(f"[INFO] Sending: {send_packet}")

            self._port_.write(send_packet)

    def receive_message(self, decode: bool = False, info: bool =
False) -> bytes | str:
        packet =
self._port_.read(RECEIVE_BUFFER_LENGTH)[len(PACKET_START_VALUE):-
len(PACKET_END_VALUE)]

        if packet and info:
            print(f"[INFO] Received: {packet}")

        return packet.decode("windows-1252") if decode else packet
```

ВИСНОВОК:

В ході роботи №3 реалізовано схему комунікації з Лабораторної роботи №2 для мікроконтролері та ПК використовуючи мови програмування C та Python.