Міністерство освіти і науки України

Національний університет «Львівська політехніка»



**Лабораторна робота № 2**

з дисципліни: «Автоматизоване проектування комп'ютерних систем»,

на тему: «Створення схеми комунікації між клієнтом та сервером.»

Виконав:

ст. гр. КІ-410

Іванюк О.О.

Прийняв:

Кіцера А. О.

Львів – 2024

#### **Task 2.**

|  |  |
| --- | --- |
| **SW <> HW (FEF)** | **Create SW game (EEF)** |
| 1. Create a simple communication schema **SW(client)**   **<-> UART <-> HW(server)**.   1. The client should send a message to the server. The server should modify the message and send it back to the client. 2. Create **YML file** with next features:    1. build all binaries (create scripts in folder **ci/** if need);    2. run tests;    3. create artifacts with binaries and test reports; 3. Required steps. | 1. Develop SW game. 2. Required steps. |

**Варіант 7:**



**Теоретичні відомості**

UART (Universal Asynchronous Receiver/Transmitter) представляє собою стандартний протокол для обміну даними між пристроями через послідовний інтерфейс. Цей інтерфейс дозволяє пристроям передавати та приймати інформацію біт за бітом і знаходить широке застосування в мікроконтролерах, мікропроцесорах, сенсорах, модемах та інших пристроях.

Основні характеристики UART включають асинхронний режим, де дані передаються через два незалежні сигнали - TX (передача) та RX (прийом), структуру кадру, яка включає стартовий біт, біти даних, біти парності (опціонально) та стоповий біт. Ця структура дозволяє правильно ідентифікувати початок та кінець кожного байту.

Швидкість передачі (Baud Rate) грає ключову роль у визначенні того, скільки бітів передається за одну секунду, і ця швидкість повинна бути налаштована на обох пристроях для успішного обміну даними.

Парність (Parity) є опціональною функцією, яка дозволяє визначити четність чи непарність бітів даних для виявлення помилок передачі. Керівництво лінією (Flow Control) може використовуватися для управління потоком даних, особливо при великій швидкості передачі або різних швидкостях пристроїв.

Також іноді використовується ізоляція гальванічна для електричної ізоляції між пристроями та запобігання електричним помилкам.

Дистанція передачі за допомогою UART обмежена, хоча цей інтерфейс дозволяє передавати дані на значні відстані. Зазвичай це кілька метрів без спеціальних заходів.

Загалом, UART є простим та надійним інтерфейсом для обміну даними, і його широко використовують у вбудованих системах та промислових застосуваннях. Його важливі характеристики роблять його ефективним засобом для сполучення різних пристроїв у великому спектрі застосувань.

**Хід роботи**

1. Написав код для клієнтської частини.

#include <windows.h>

#include <iostream>

#include <string>

using namespace std;

string createXML(const string& message) {

    return "<message>" + message + "</message>";

}

string parseXML(const string& xml) {

    size\_t start = xml.find("<message>") + 9; // 9 - �� ������� ���� <message>

    size\_t end = xml.find("</message>");

    if (start != string::npos && end != string::npos) {

        return xml.substr(start, end - start);

    }

    return ""; // ���� XML �����������

}

int main() {

     /\*����� ������ COM-����� � �����������\*/

    int portNumber;

    cout << "Enter COM port number (e.g., 6 for COM6): ";

    cin >> portNumber;

    // ���������� ����� �����

    string portName\_s = "COM" + to\_string(portNumber);

    const char\* portName = portName\_s.c\_str();

    /\*const char\* portName = "COM6";\*/

    // ³������� COM-�����

    HANDLE hSerial = CreateFileA(portName, // portName.c\_str()

        GENERIC\_READ | GENERIC\_WRITE,

        0,

        NULL,

        OPEN\_EXISTING,

        FILE\_ATTRIBUTE\_NORMAL,

        NULL);

    if (hSerial == INVALID\_HANDLE\_VALUE) {

        cerr << "Error opening port" << endl;

        system("pause");

        return 1;

    }

    // ������������ ��������� �����

    DCB dcbSerialParams = { 0 };

    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);

    if (!GetCommState(hSerial, &dcbSerialParams)) {

        cerr << "Error getting port state" << endl;

        system("pause");

        return 1;

    }

    dcbSerialParams.BaudRate = CBR\_9600;

    dcbSerialParams.ByteSize = 8;

    dcbSerialParams.StopBits = ONESTOPBIT;

    dcbSerialParams.Parity = NOPARITY;

    if (!SetCommState(hSerial, &dcbSerialParams)) {

        cerr << "Error setting port parameters" << endl;

        system("pause");

        return 1;

    }

    // ������������ ����-����

    COMMTIMEOUTS timeouts = { 0 };

    timeouts.ReadIntervalTimeout = 50;

    timeouts.ReadTotalTimeoutConstant = 50;

    timeouts.ReadTotalTimeoutMultiplier = 10;

    if (!SetCommTimeouts(hSerial, &timeouts)) {

        cerr << "Error setting timeouts!" << endl;

        system("pause");

        return 1;

    }

    Sleep(2000);

    // ���� ��� ��������� �������� ����������

    while (true) {

        string message = "hello from pc";  // ����������� ��� ��������

        string messageXML = createXML(message);

        DWORD bytesWritten;

        // ³������� ����������� �� Arduino

        if (!WriteFile(hSerial, messageXML.c\_str(), messageXML.size(), &bytesWritten, NULL)) {

            cerr << "Error writing to port" << endl;

            break;

        }

        cout << "Sent: " << message << " (" << messageXML << ")" << endl;

        // ������� ������ �� Arduino

        char buffer[128];

        DWORD bytesRead;

        if (ReadFile(hSerial, buffer, sizeof(buffer) - 1, &bytesRead, NULL)) {

            buffer[bytesRead] = '\0';

            if (bytesRead > 0) {

                cout << "Received from Arduino: " << parseXML(buffer) << " (" << buffer << ")" << endl << endl;

            }

        }

        else {

            cerr << "Error reading from port" << endl;

            break;

        }

        // �������� ����� ��������� �������

        Sleep(1000);

    }

    // �������� COM-�����

    CloseHandle(hSerial);

    return 0;

}

1. Написав код для серверної частини.

void setup() {

  pinMode(13, OUTPUT); // Ініціалізація піну для лампочки

  digitalWrite(13, HIGH);

  Serial.begin(9600);  // Налаштування серійного зв'язку зі швидкістю 9600 біт/с

  while (!Serial);     // Очікуємо поки порт стане доступним (тільки для Leonardo і схожих плат)

  // delay(2000);

  digitalWrite(13, LOW);

}

void loop() {

  if (Serial.available() > 0) {

    String receivedMessage = Serial.readString();  // Читаємо дані з серійного порту

    digitalWrite(13, HIGH);

    String parsedMessage = parseXML(receivedMessage);

    // Перетворюємо всі літери в верхній регістр

    for (int i = 0; i < parsedMessage.length(); i++) {

      parsedMessage[i] = toupper(parsedMessage[i]);

    }

    String xmlToSend = createXML(parsedMessage);

    digitalWrite(13, LOW);

    // Відправляємо змінене повідомлення назад

    Serial.print(xmlToSend);

  }

}

// Функція створення XML

String createXML(String message) {

  String xml = "<message>";

  xml += message;

  xml += "</message>";

  return xml;

}

// Функція парсингу XML

String parseXML(String xml) {

  int start = xml.indexOf("<message>") + 9; // 9 - це довжина тегу <message>

  int end = xml.indexOf("</message>");

  if (start != -1 && end != -1) {

    return xml.substring(start, end);

  }

  return ""; // Якщо XML некоректний

}

1. Провів перевірку на працездатність.

A screenshot of a computer

Description automatically generated

1. Створив нову гілку feature/develop/task2. Створив Pull request для підтвердження змін в гілці develop.

A screenshot of a computer

Description automatically generated

**Висновок:**

Під час виконання лабораторної роботи було створено програмний код для клієнтської та серверної частини для плати Arduino. Також продемонстровано працездатність розробленої схеми комунікації.