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

Ковальчук А..

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)**. 2. The client should send a message to the server. The server should modify the message and send it back to the client. 3. Create **YML file** with next features: a. build all binaries (create scripts in folder **ci/** if need); b. run tests; c. create artifacts with binaries and test reports; 4. Required steps.	1. Develop SW game. 2. Required steps.

Варіант 8:

Student number	Game	config format
8	rock paper scissors	JSON

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;

// Функція створення JSON
string createJSON(const string& message) {
    return "{\"message\":\"" + message + "\"}";
}

// Функція парсингу JSON
string parseJSON(const string& json) {
    size_t start = json.find("\"message\":\"") + 11; // 11 - довжина ключа "message" з лапками
    size_t end = json.find("\"", start);

    if (start != string::npos && end != string::npos) {
        return json.substr(start, end - start);
    }

    return ""; // Якщо JSON некоректний
}

int main() {
    /*Запит номера COM-порту у користувача*/
    int portNumber;
    cout << "Enter COM port number (e.g., 6 for COM6): ";
    cin >> portNumber;

    // Формування назви порту
    string portName_s = "COM" + to_string(portNumber);
    const char* portName = portName_s.c_str();

    // Відкриття COM-порту
    HANDLE hSerial = CreateFileA(portName,
        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;
    }

    // Налаштування параметрів порту
    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;
}

// Налаштування тайм-аутів
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);

// Цикл для безкінечної відправки повідомлень
while (true) {
    string message = "HELLO world";  // Повідомлення для відправки
    string messageJSON = createJSON(message);
    DWORD bytesWritten;

    // Відправка повідомлення до Arduino
    if (!WriteFile(hSerial, messageJSON.c_str(), messageJSON.size(), &bytesWritten, NULL)) {
        cerr << "Error writing to port" << endl;
        break;
    }
    cout << "Sent to Arduino:      " << message << " (" << messageJSON << ")" << endl;

    // Читання відповіді від 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: " << parseJSON(buffer) << " (" << buffer << ")" << endl << endl;
        }
    }
    else {
        cerr << "Error reading from port" << endl;
        break;
    }

    // Затримка перед наступним запитом
```

```
    Sleep(1000);
  }

  // Закриття COM-порту
  CloseHandle(hSerial);
  return 0;
}
```

2. Написав код для серверної частини.

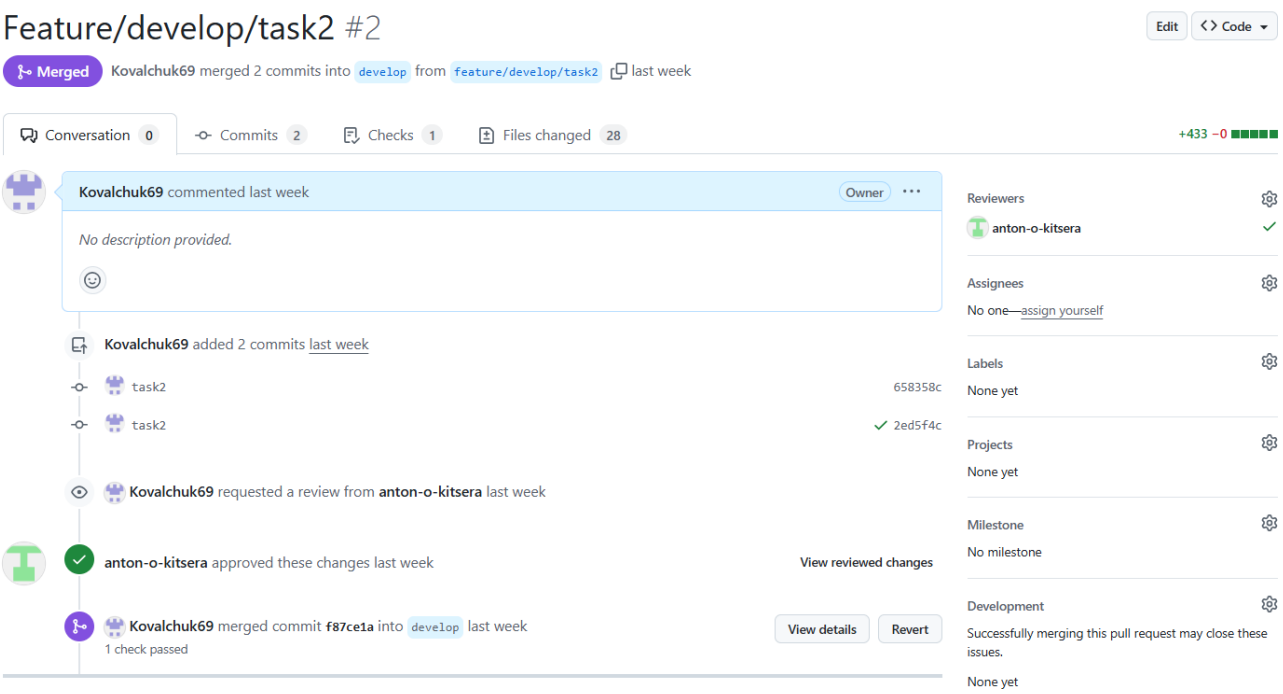
```
void setup() {
  pinMode(13, OUTPUT); // Ініціалізація піну для лампочки
  digitalWrite(13, HIGH);

  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)
  digitalWrite(13, LOW);
}

void loop() {
  if (Serial.available() > 0) {
    String receivedMessage = Serial.readString();  // Читаємо дані з серійного
порту
    digitalWrite(13, HIGH);

    // Парсимо JSON повідомлення
    String parsedMessage = parseJSON(receivedMessage);

    // Поділ повідомлення на перше і друге слово
    int spaceIndex = parsedMessage.indexOf(' ');
    String firstWord, secondWord;

    if (spaceIndex != -1) {
      firstWord = parsedMessage.substring(0, spaceIndex);
      secondWord = parsedMessage.substring(spaceIndex + 1);
    } else {
      firstWord = parsedMessage;  // Якщо повідомлення містить тільки одне слово
      secondWord = "";
    }

    // Перетворення регістру слів
    firstWord.toLowerCase();
    secondWord.toUpperCase();

    // Формуємо змінене повідомлення
    String modifiedMessage = firstWord + " " + secondWord;

    // Створюємо JSON для відправки
```

```
    String jsonToSend = createJSON(modifiedMessage);

    digitalWrite(13, LOW);
    // Відправляємо змінене повідомлення назад
    Serial.print(jsonToSend);
  }
}

// Функція створення JSON
String createJSON(String message) {
  return "{\"message\":\"" + message + "\"}";
}

// Функція парсингу JSON
String parseJSON(String json) {
  int start = json.indexOf("\"message\":\"") + 11; // 11 - довжина ключа "message"
з лапками
  int end = json.indexOf("\"", start);

  if (start != -1 && end != -1) {
    return json.substring(start, end);
  }

  return ""; // Якщо JSON некоректний
}
```

3. Провів перевірку на працездатність.

```
C:\Users\Andrew\Desktop\apks\lab2\Lab2_Client_Uart\x64\Debug\Lab2_Client_Uart.exe
Enter COM port number (e.g., 6 for COM6): 7
Sent to Arduino:       HELLO world ({"message":"HELLO world"})
Received from Arduino: hello WORLD ({"message":"hello WORLD"})

Sent to Arduino:       HELLO world ({"message":"HELLO world"})
Received from Arduino: hello WORLD ({"message":"hello WORLD"})

Sent to Arduino:       HELLO world ({"message":"HELLO world"})
Received from Arduino: hello WORLD ({"message":"hello WORLD"})

Sent to Arduino:       HELLO world ({"message":"HELLO world"})
Received from Arduino: hello WORLD ({"message":"hello WORLD"})

Sent to Arduino:       HELLO world ({"message":"HELLO world"})
Received from Arduino: hello WORLD ({"message":"hello WORLD"})
```

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



Висновок:

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