МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ

ЗВІТ

ЛАБОРАТОРНА РОБОТА № 2

З дисципліни «Автоматизоване проєктування комп`ютерних систем»

Виконав: ст. гр. КІ-401
Ларіонов А.О
Перевірив:
Федак П.Р.

Львів 2024

ЗМІСТ

РОЗДІЛ 1. МЕТА РОБОТИ.

Створення комунікції між сервером та клієнтом.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ВІДОМОСТІ.

Для забезпечення базової конфігурації та роботи з серійним портом (COM-портом) у Windows використовується UART-протокол. UART (Universal Asynchronous Receiver-Transmitte**r)** — це апаратний модуль або периферійний пристрій у мікроконтролерах та комп'ютерах, який забезпечує послідовну передачу та прийом даних між пристроями через дротовий інтерфейс. Він налаштовується через такі параметри(стандарт для роботи з багатьма платами):

- Швидкість передачі даних (BaudRate): 9600 біт/с.
- Розмір даних: 8

При виконання завдання було створено програму яка надсилає повідомлення на сервер та приймає повідомлення з сервера, що важливо для подальшої розробки гри, адже всі обрахунки будуть проводитися на сервері, тобто на платі.

Для автоматизації та збірки проекту було використано CMake. CMake **—** це інструмент для автоматизації процесу збірки програмного забезпечення. Тут у файлі CmakeLists буде описно як зібрати та скомпілювати проект

Для конфігурації та автоматизації робочих процесів CI в GitHub Actions**,** (CI - Continuous Integration) було використане формат серіалізації даних .yaml. YAML (абревіатура від YAML “Ain't Markup Language") — це формат серіалізації даних, який використовується для зберігання та обміну конфігураціями, даними або налаштуваннями в текстовому вигляді. У файлі ci.yalm описано як автоматично збирати ваш проект після кожної зміни в коді.

РОЗДІЛ 3. ЗАВДАННЯ.

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.

РОЗДІЛ 4. ХІД РОБОТИ.

4.1. Клієнтська чатсину коду

main.cpp використовує функції з serial.cpp для налаштування порту, надсилання та отримання даних. serial.cpp містить логіку для безпосередньої роботи з API Windows для серійного порту, налаштування з'єднання, відправки і прийому повідомлень.

main.cpp - цей файл містить основну логіку програми, яка:

- Ініціалізує серійний порт через функцію initSerialPort(), передаючи назву порту (наприклад, "COM3").
- Відправляє повідомлення на підключений пристрій через функцію sendMessage().
- Читає відповідь від пристрою через функцію receiveMessage() і виводить її на екран.
- Закриває серійний порт після завершення роботи через функцію closeSerialPort().

serial.cpp - цей файл містить реалізацію функцій для роботи з серійним портом:

- initSerialPort() — Відкриває серійний порт, налаштовує параметри зв'язку (швидкість передачі, біт даних, стоп-біти, паритет).
- sendMessage() — Відправляє повідомлення на пристрій через серійний порт.
- receiveMessage() — Читає дані з серійного порту.
- closeSerialPort() — Закриває серійний порт

4.2. Серверна частину коду

Файл server.ino містить код призначений для роботи з серійним портом на Arduino.

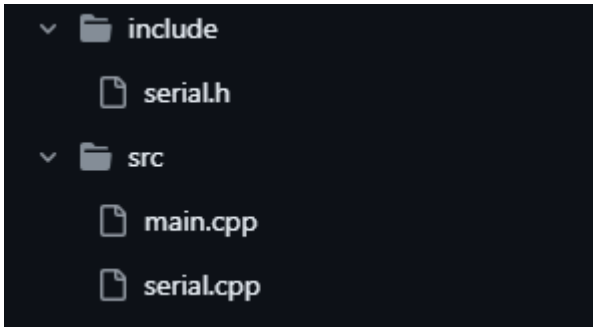
setup():

- Serial.begin(9600);: Ініціалізує серійний порт зі швидкістю 9600 бод. Це дозволяє Arduino отримувати та передавати дані через серійний порт.

- Serial.println("Arduino ready");: Виводить повідомлення "Arduino ready" у серійний монітор. Це є підтвердженням того, що Arduino готове до взаємодії.

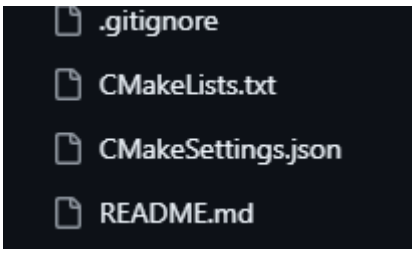
loop():

- if (Serial.available() > 0): Перевіряє, чи є вхідні дані на серійному порту (тобто чи надійшло повідомлення від комп'ютера чи іншого пристрою).
- String message = Serial.readString();: Читає вхідні дані з серійного порту у вигляді рядка. Це буде весь рядок, надісланий через серійний порт, включаючи пробіли.
- message.trim();: Видаляє зайві пробіли на початку та в кінці рядка, якщо вони є.
- message = message + " modified";: Додає до отриманого рядка текст " modified". Це демонструє модифікацію отриманого повідомлення.
- Serial.println(message);: Відправляє змінене повідомлення назад через серійний порт до підключеного пристрою.
- Serial.println("Message sent back to client");: Виводить додаткове повідомлення в серійний монітор для підтвердження того, що повідомлення було надіслано назад.

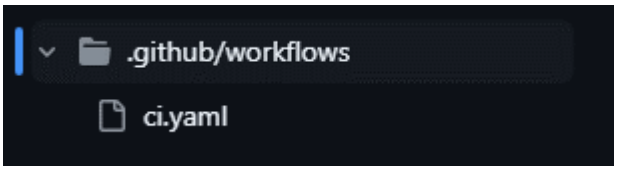


4.3. Збірка проекту

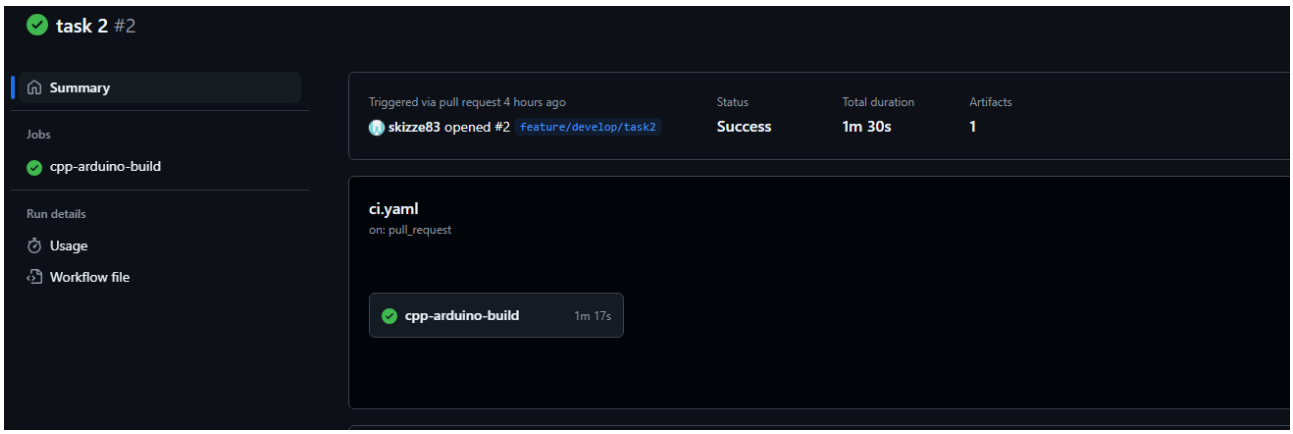
Для збірки проекту написав CMakeLists з вказаним налаштуванням збірки. Зібрав проект за допомогою build



4.4. Створення ci.yalm для Github Action

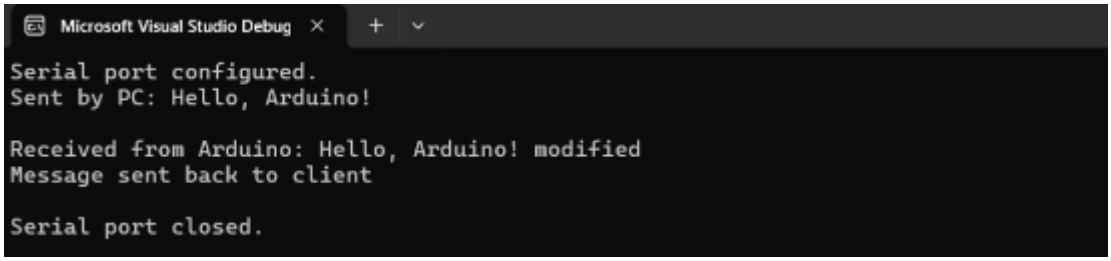


Написав ci.yalm для тестування та компіляції на Github Action. В подальшому можна переглянути на GitHub про успішне або незадовільне опрацювання проекту



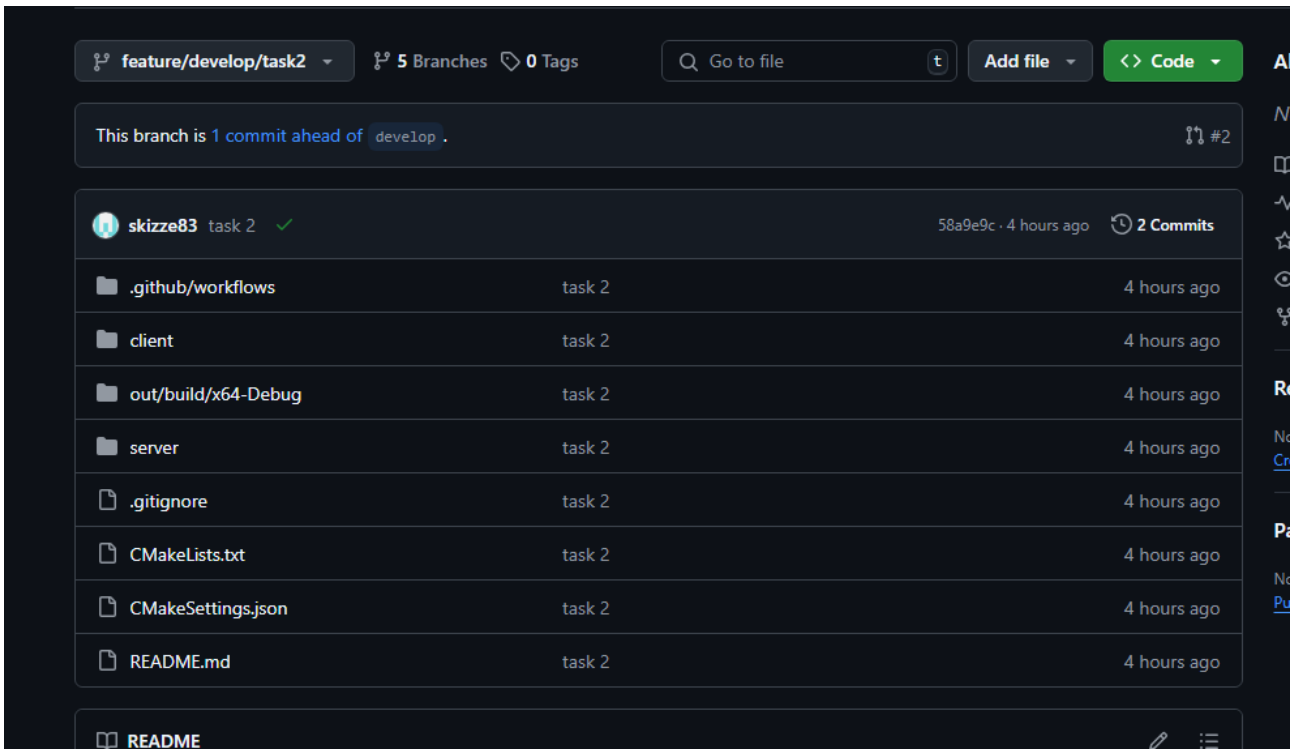
4.5. Тестування

Запустив RockPaperScissors.exe. Бачимо на екрані повідомлення що повідомлення надіслалося на сервер, та отримали відповідь від сервера



4.6. Взаємодія з репозиторієм

Відправив на віддалений репозиторій всі зміни в новій гілці feature/develop/task2. Та відредагував README.MD



csad2425ki401larionovao10

Repository Information

This repository is part of the CSAD course, where we will implement tasks related to computer systems and digital circuits. The main task - Rock Paper Scissors game

Student Information

Group: KI-401
Full Name: Larionov Artom

Task Details

Student Number: 10
Game: Rock Paper Scissors
Config: .xml

Task 1:

Initialize the GIT repository for the project.

Task 2:

Create comunication server\client. Server - Arduino Uno ci.yalm - for Continuous Integration CMakeLists - configuration for building the project

Technologies

- Language: C/C++
- Platform: Arduino
- Hardware: Arduino Uno

ВИСНОВКИ

Навчився створювати комунікації між платою та клієнтською частиною. Також писати конфігураційні файли для збірки проекту та подальших перевірок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *GitHub Docs - https://docs.github.com/en*
2. *CSAD Instructions for practical tasks and coursework from "Computer systems automated design"*
3. *https://github.com/skizze83/csad2425ki401larionovao10*
4. *https://cmake.org/download/*

ДОДАТОК А. MAIN.CPP

```
 #include <iostream>
#include <string>
#include "serial.h"

using namespace std;

int main() {
    // Use the correct COM port (replace "COM3" with your port)
    initSerialPort("\\\\.\\COM3");

    // Send message
    sendMessage("Hello, Arduino!\n");

    // Wait to allow Arduino to process the message
    Sleep(1000);

    // Attempt to receive response
    string response = receiveMessage();
    if (!response.empty()) {
        cout << "Received from Arduino: " << response << endl;
    }
    else {
        cout << "No response received." << endl;
    }

    // Close the port
    closeSerialPort();

    return 0;
}
```

ДОДАТОК Б. SERIAL.CPP

```
  #include <iostream>
#include <windows.h>
#include <string>
#include "serial.h"

using namespace std;

HANDLE hSerial;
DCB dcbSerialParams = { 0 };
COMMTIMEOUTS timeouts = { 0 };

// Helper function to convert std::string to std::wstring
std::wstring stringToWString(const std::string& str) {
    return std::wstring(str.begin(), str.end());
}

void initSerialPort(const std::string& portName) {
    // Convert std::string to std::wstring
    std::wstring widePortName = stringToWString(portName);

    // Use CreateFileW for wide strings
    hSerial = CreateFileW(widePortName.c_str(),
        GENERIC_READ | GENERIC_WRITE,
        0,
        0,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL,
        0);
    if (hSerial == INVALID_HANDLE_VALUE) {
        cerr << "Failed to open port!" << endl;
        exit(1);
    }

    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
    GetCommState(hSerial, &dcbSerialParams);
    dcbSerialParams.BaudRate = CBR_9600;
    dcbSerialParams.ByteSize = 8;
    dcbSerialParams.StopBits = ONESTOPBIT;
    dcbSerialParams.Parity = NOPARITY;
    SetCommState(hSerial, &dcbSerialParams);

    timeouts.ReadIntervalTimeout = 50;
    timeouts.ReadTotalTimeoutConstant = 50;
    timeouts.ReadTotalTimeoutMultiplier = 10;
    timeouts.WriteTotalTimeoutConstant = 50;
    timeouts.WriteTotalTimeoutMultiplier = 10;
    SetCommTimeouts(hSerial, &timeouts);

    cout << "Serial port configured." << endl;
}

void sendMessage(const char* message) {
    DWORD bytesWritten;
    WriteFile(hSerial, message, strlen(message), &bytesWritten, NULL);
    cout << "Sent by PC: " << message << endl;
}

std::string receiveMessage() {
    char buffer[1024] = { 0 };
    DWORD bytesRead;
    bool readSuccess = ReadFile(hSerial, buffer, sizeof(buffer), &bytesRead, NULL);
```

```
    if (readSuccess && bytesRead > 0) {
        return string(buffer);
    }
    else {
        return "";
    }
}

void closeSerialPort() {
    CloseHandle(hSerial);
    cout << "Serial port closed." << endl;
}
```

ДОДАТОК В. SERIAL.H

```
 #pragma once
#ifndef SERIAL_H
#define SERIAL_H

#include <windows.h>
#include <string>

// Ãëîáàëüíà çì³ííà äëÿ ðîáîòè ç ñåð³éíèì ïîðòîì
extern HANDLE hSerial;

// Îãîëîøåííÿ ôóíêö³é äëÿ ðîáîòè ç ñåð³éíèì ïîðòîì
void initSerialPort(const std::string& portName);
void sendMessage(const char* message);
std::string receiveMessage();
void closeSerialPort();

#endif // SERIAL_H
```

ДОДАТОК Г. CI.YALM

```
 name: CI for C++ and Arduino Uno

on:
  push:
    branches:
      - '**'
  pull_request:
    branches:
      - '**'

jobs:
  cpp-arduino-build:
    runs-on: windows-latest

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

      - name: Download Arduino CLI
        uses: arduino/setup-arduino-cli@v1
        with:
          version: '0.19.2'

      - name: Download CMake
        run: choco install cmake

      - name: Download platform Arduino AVR
        run: arduino-cli core install arduino:avr

      - name: Config CMake
        run: cmake -S . -B build
        shell: cmd

      - name: Build with CMake
        run: cmake --build build
        shell: cmd

      - name: Download build artefacts
        uses: actions/upload-artifact@v3
        with:
          name: build-artifacts
          path: build/*
```

ДОДАТОК Д. CMAKELISTS.TXT

```
 # Minimum required version of CMake
cmake_minimum_required(VERSION 3.10)

# Set project name and version
project(RockPaperScissors VERSION 1.0)

# Set the C++ standard to C++17
set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

# Include directories for client
include_directories(${CMAKE_SOURCE_DIR}/client/include)

# Add executable for the RockPaperScissors application
add_executable(RockPaperScissors
    ${CMAKE_SOURCE_DIR}/client/src/main.cpp
    ${CMAKE_SOURCE_DIR}/client/src/serial.cpp
)

# Define preprocessor macros
add_definitions(-DUNICODE -D_UNICODE)

# Link the Windows specific library if on a Windows system
if(WIN32)
    target_link_libraries(RockPaperScissors PRIVATE ws2_32)
endif()
```

ДОДАТОК Е. SERVER.INO

```
 void setup() {
  // Set up the serial port
  Serial.begin(9600);
  // Additional output to confirm the start of operation
  Serial.println("Arduino ready");
}

void loop() {
  // Check if there are incoming data
  if (Serial.available() > 0) {
    // Receive the message
    String message = Serial.readString();

    message.trim();

    message = message + " modified";

    // Send the modified message back
    Serial.println(message);

    // Additional output to confirm the message has been sent
    Serial.println("Message sent back to client");
  }
}
```