МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ЛЬВІВСЬКА ПОЛІТЕХНІКА



**АВТОМАТИЗОВАНЕ ПРОЕКТУВАННЯ**

**КОМП’ЮТЕРНИХ СИСТЕМ**

Завдання 2: “Комунікація клієнта та сервера”

Виконав:

ст. гр. КІ-404

Мох М. П.

Приняв:

Федак П.Р.

Львів – 2024

ЗАВДАННЯ

1. Створіть просту комунікаційну схему SW (клієнт)<-> UART <-> HW (сервер).

2. Клієнт повинен надіслати повідомлення на сервер. Сервер повинен змінити повідомлення та надіслати його назад до клієнт.

3. Створіть файл YML із такими функціями:

a. зібрати всі бінарні файли (створити сценарії в папці ci/if

потреба);

b. запустити тести;

в. створювати артефакти з бінарними файлами та звітами про тестування;

**ТЕОРЕТИЧНИЙ МАТЕРІАЛ**

**UART (Universal Asynchronous Receiver-Transmitter)** — це протокол для асинхронної послідовної передачі даних між двома пристроями, наприклад, між комп'ютером і мікроконтролером (таким як Arduino). UART дозволяє пристроям обмінюватися інформацією через два провідники: один для передачі даних (TX) і один для прийому (RX). Головна особливість UART полягає в тому, що він не потребує синхронізуючого сигналу, що робить його простим у налаштуванні.

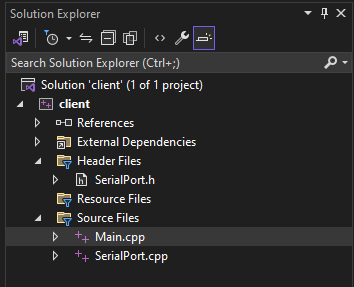
**CMake** – це кросплатформений інструмент автоматизації збірки, який використовують для генерації конфігураційних файлів (наприклад, Makefile, Ninja) для компіляції програм. Його перевага в тому, що він дозволяє легко створювати збірки на різних системах.

**YML або YAML (Yet Another Markup Language)** — це формат даних, який широко використовується для написання конфігураційних файлів. YAML є зручним через свою просту, читабельну структуру. Він застосовується для опису налаштувань у багатьох системах автоматизації, наприклад, CI/CD-процесах у GitHub Actions, GitLab CI тощо.

**Файли з розширенням .bat (batch files)** використовуються в операційній системі Windows для автоматичного виконання команд у командному рядку. Це скрипти, які містять послідовність команд, що можуть бути виконані одразу. .bat файли дуже зручні для автоматизації завдань, які часто повторюються, таких як компіляція програм, запуск тестів, переміщення файлів тощо.

**ВИКОНАННЯ РОБОТИ**

1. Створіть просту комунікаційну схему SW (клієнт)<-> UART <-> HW (сервер).



**Клієнт:** У файлі SerialPort.h визначено структуру класу, його методи та змінні, а у файлі SerialPort.cpp міститься реалізація всіх методів. Головний файл CSAD\_lab2.cpp викликає ці функції для взаємодії з сервером, що включає надсилання і отримання повідомлень, а також їх виведення у консоль.

**Сервер:** Серверна частина складається з програми, яка отримує повідомлення від клієнта, обробляє його (змінює), а потім надсилає змінене повідомлення назад до клієнта.

1. Клієнт повинен надіслати повідомлення на сервер. Сервер повинен змінити повідомлення та надіслати його назад до клієнт.

**Код Main.cpp:**

#include <iostream>

#include "SerialPort.h"

// The main function initializes serial communication, sends a user-defined message to Arduino,

// and terminates automatically after receiving a response.

int main() {

SerialPort serialPort("\\\\.\\COM3");

// Attempt to configure the serial port. Exit if configuration fails.

if (!serialPort.configurePort()) {

std::cerr << "Failed to initialize the serial port. Check the connection and try again!" << std::endl;

return 1;

}

std::cout << "Serial port initialized successfully." << std::endl;

// Ask the user to input a message for Arduino.

std::string userMessage;

std::cout << "Please enter a message to send to Arduino: ";

std::getline(std::cin, userMessage);

// Inform the user that the program is waiting for a response.

std::cout << "Awaiting a response from the Arduino device..." << std::endl;

// Send the user's message and read the response from Arduino.

std::string deviceReply = serialPort.sendMessage(userMessage);

// If a response is received, display it and exit the program.

if (!deviceReply.empty()) {

std::cout << "Arduino replied: " << deviceReply << std::endl;

}

else {

std::cerr << "Error: No response received from the device." << std::endl;

}

// Program automatically exits after displaying the response.

return 0;

}

**Код server.ino:**

void setup() {

Serial.begin(9600); // Налаштування швидкості серійного зв’язку

while (!Serial) {

; // Очікуємо підключення серійного порту

}

}

void loop() {

// Очікуємо отримання повідомлення

if (Serial.available() > 0) {

String message = Serial.readStringUntil('\n'); // Читання повідомлення до символу нового рядка

message += " Copy"; // Модифікація повідомлення, додаємо "AR" в кінці

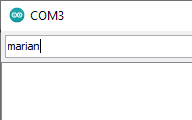
// Відправлення модифікованого повідомлення назад клієнту

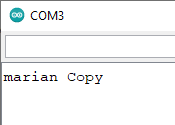
Serial.println(message); // Повертаємо змінене повідомлення

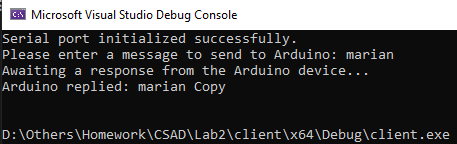
// Після одного з'єднання зупиняємося (якщо не потрібна подальша комунікація)

}

}







Клієнт надіслав повідомлення на сервер, який обробив його, додавши до тексту слово "Copy", а потім відправив змінене повідомлення назад клієнту. Клієнт отримав відповідь і відобразив результат на екрані.

3. Створіть файл YML із такими функціями: зібрати всі бінарні файли (створити сценарії в папці ci/за потреби);  
  
 **CMakeLists.txt:**

# Minimal version of CMake

cmake\_minimum\_required(VERSION 3.10)

# Name of project

project(ClientServerProject)

# Set standard C++

set(CMAKE\_CXX\_STANDARD 17)

# Specify directories with header files

include\_directories(${CMAKE\_SOURCE\_DIR}/client/include)

# add an executable file for the client

add\_executable(client

${CMAKE\_SOURCE\_DIR}/client/Main.cpp

${CMAKE\_SOURCE\_DIR}/client/SerialPort.cpp

)

# specify that the Windows library should be used

if(WIN32)

target\_link\_libraries(client PRIVATE ws2\_32)

endif()

**ci.yaml**

name: CI for C++ and Arduino

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/\*

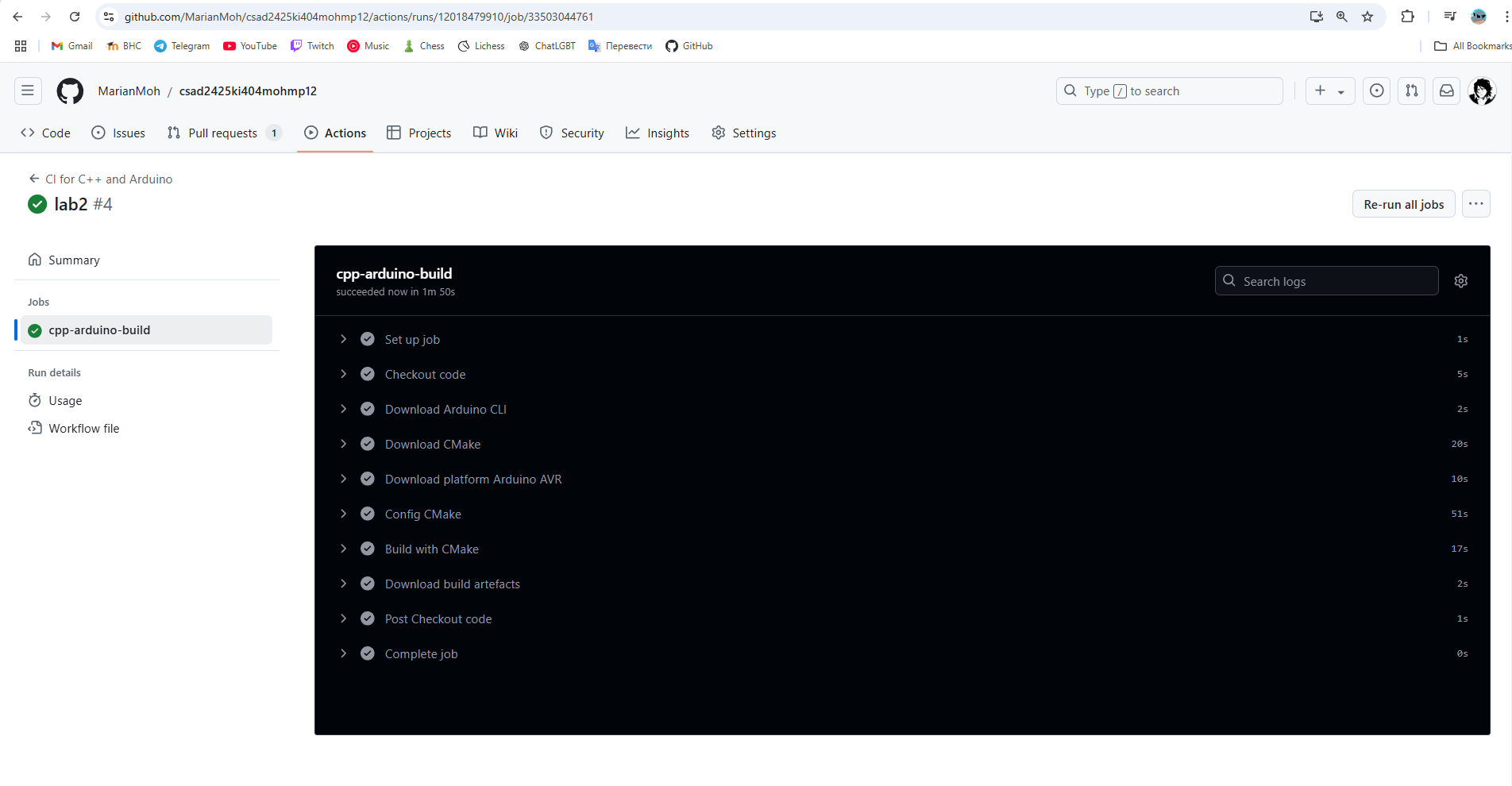


Рис 1. Тестування на git

**ВИСНОВОК**

У ході роботи було створено клієнт-серверну комунікаційну систему між клієнтом (ПК) та сервером (Arduino) через UART, яка забезпечує двосторонній обмін даними: клієнт надсилає повідомлення на сервер, сервер обробляє їх і повертає змінені дані. Також розроблено автоматизовану систему для збору та тестування коду з використанням YAML-конфігурації для CI/CD. Ця система автоматизує процес збірки бінарних файлів, розміщення скриптів та створення артефактів з бінарними файлами й результатами тестів, що сприяє стабільності та надійності процесу розробки через безперервну інтеграцію та доставку.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Arduino. "Arduino Documentation." Arduino.cc. <https://docs.arduino.cc/>
2. GitHub. "GitHub Actions Documentation." GitHub Docs. <https://docs.github.com/en/actions>
3. Microsoft Documentation on Batch Scripts: <https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb490887(v=technet.10)>
4. SparkFun. "UART Communication Basics." SparkFun Learning Resources. <https://learn.sparkfun.com/tutorials/serial-communication>
5. YAML Essentials <https://yaml.org/spec/>
6. GitLab. "CI/CD Pipelines Documentation." GitLab Docs. <https://docs.gitlab.com/ee/ci/>