**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ЛЬВІВСЬКА ПОЛІТЕХНІКА**



**АВТОМАТИЗОВАНЕ**

**ПРОЕКТУВАННЯ КОМП’ЮТЕРНИХ**

**СИСТЕМ**

Лабораторна робота №2

“Створення простої комунікації SW(client) <-> UART <-> HW(server)”

**Виконав:**

студент гр. KI-401

Гавриляк Д. В.

**Прийняв:**

Федак П. Р.

Львів – 2024

**Завдання:**

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.

**Теорія:**

**Схема комунікації SW(client) <-> UART <-> HW(server)**  
UART (Universal Asynchronous Receiver-Transmitter) — це протокол для асинхронної серійної передачі даних між клієнтом (програмне забезпечення) та сервером (апаратне забезпечення). Він використовує дві лінії для передачі даних: TX (передача) і RX (прийом). У цьому контексті клієнт надсилає дані через UART на сервер, який обробляє їх і відповідає назад через той самий інтерфейс.

**Передача повідомлення від клієнта до сервера та його модифікація**  
Клієнт надсилає повідомлення через UART. Сервер приймає це повідомлення, модифікує його (наприклад, додає префікс або змінює частину тексту) та відправляє відповідь назад клієнту. Ключові етапи:

* Передача даних клієнтом
* Прийом повідомлення сервером
* Обробка/модифікація повідомлення
* Відповідь сервера клієнту

**YML-файл для збірки та тестування**  
YML — формат для конфігураційних файлів. Для автоматизації розробки можна створити YML-файл з наступними характеристиками:

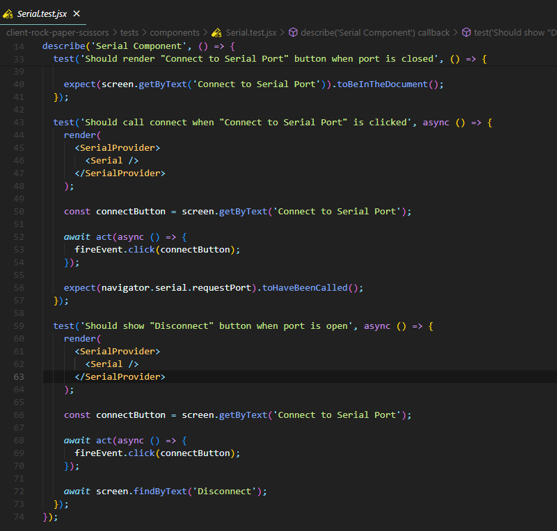
* Збірка всіх бінарних файлів (виконання скриптів із директорії ci/).
* Запуск тестів після збірки.
* Створення артефактів (бінарних файлів та звітів про тести) після виконання всіх етапів.

**Індивідуальне завдання:**

Згідно списку групи я маю 14 варіант, так як я маю такий порядковий номер у списку.



**Виконання:**

****

***Рис.1.*** Оновлена структура проекту

На даному скріншоті продемонстровано оновлена структура проекту, були добавлені папки для клієнта і сервера, а також папка, яка містить ci.yml для GitHub Actions

YAML

name: CI React Vite and PlatformIO

on:

push:

branches: [main]

pull\_request:

branches: [main]

workflow\_dispatch:

jobs:

client-build-and-test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- uses: actions/setup-node@v4

with:

node-version: 20

- run: npm install

working-directory: client-rock-paper-scissors

- run: npm run test

working-directory: client-rock-paper-scissors

- run: npm run build

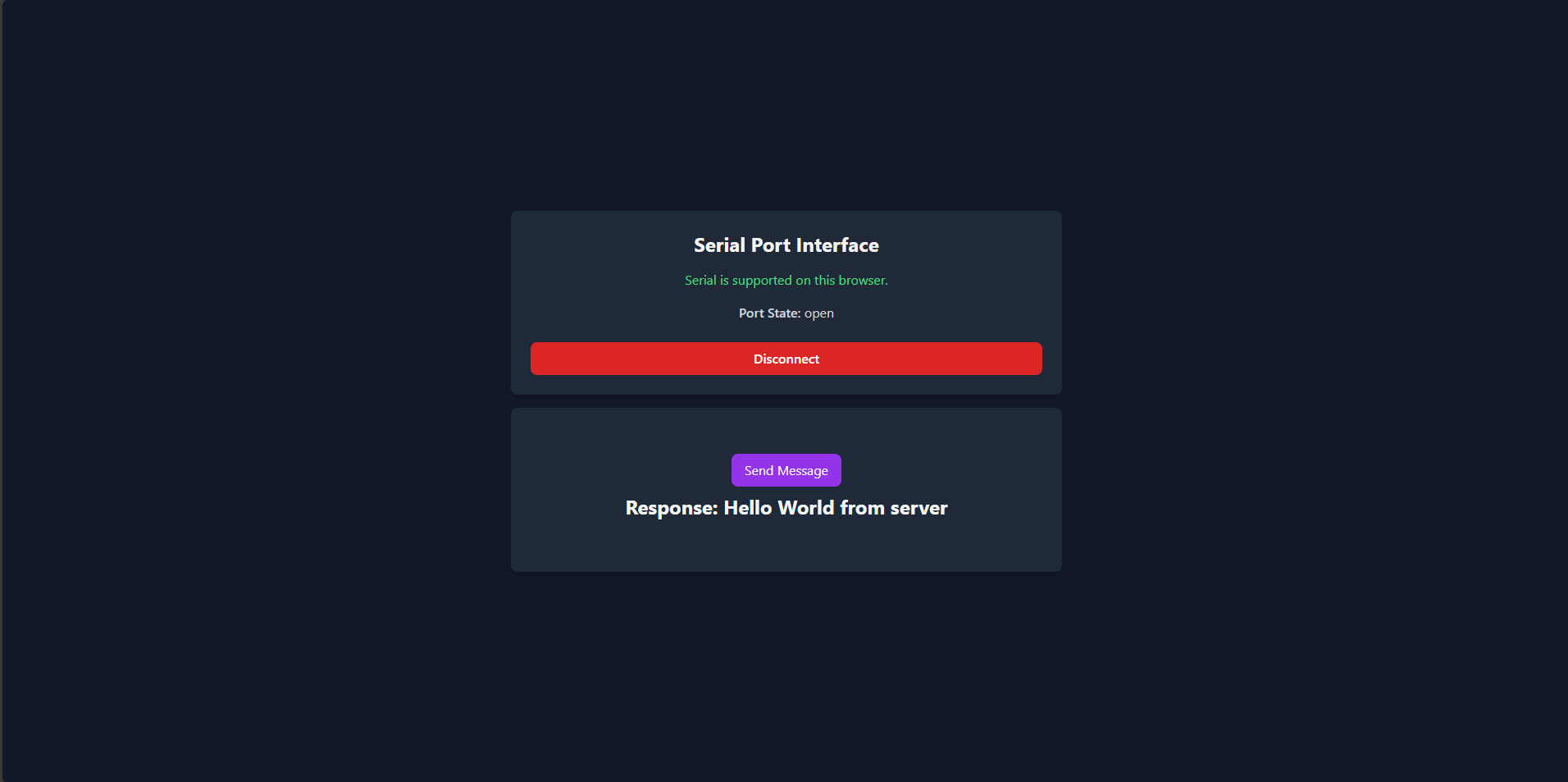
working-directory: client-rock-paper-scissors

- uses: actions/upload-artifact@v4

with:

name: client-build

Файл з написаним flow для роботи тестів і компіляції проектів з подальшим виведенням скомпільованих файлів у артефакти через GitHub Actions



***Рис.3.*** Ui клієнту

Details about repository This repository contains a project for a hardware-based Rock Paper Scissors game using Arduino Nano and JavaScript React application. The game logic and decision-making will run primarily on the Arduino Nano, while the React application will serve as a graphical interface for users to interact with the game.

Student details

Student number Game Config format 2 rock paper scissors JSON Technology Stack and Hardware Used

Hardware

Arduino Nano: The Arduino will handle most of the game logic, including managing inputs, processing the current game state, and sending data to the React application. Software

PlatformIO: To write and upload the logic code to the Arduino Nano, primarily using C/C++ for low-level control. Programming Languages

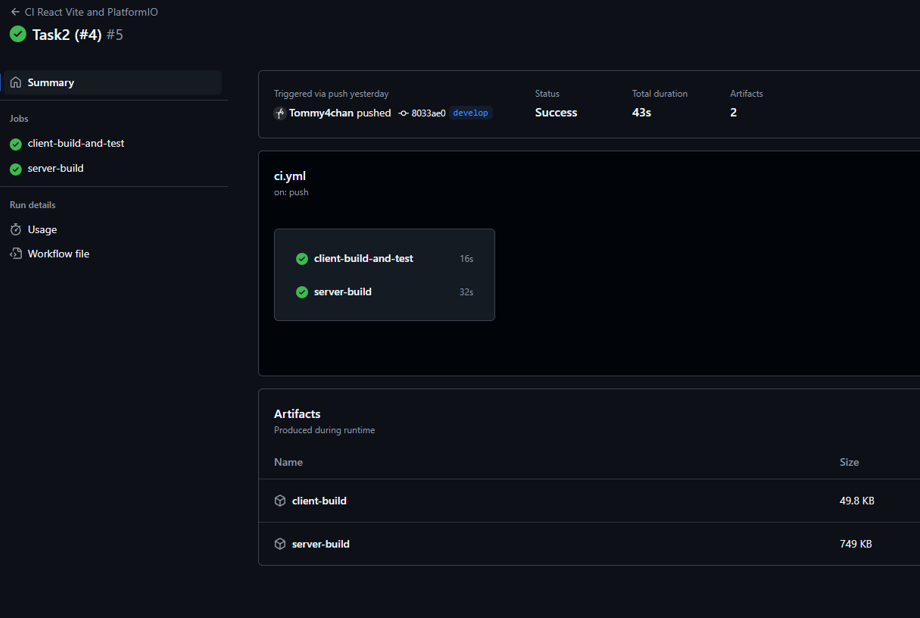
C/C++: Used in the Arduino environment to develop the Rock Paper Scissors game logic. JavaScript: Used for client app with React. Communication

Serial Communication: The Arduino will communicate with the app through a UART serial port to send and receive game status and input data. To Build and Run the Client (JS/React Application):

Fetch feature/develop/task2 branch. Navigate to csad2425ki404panurinas14/client-rock-paper-scissors. Install dependencies by typing: bash Copy code npm install Run the application by typing: bash Copy code npm run dev To Build and Run the Server (Arduino Sketch):

Open folder csad2425ki401havryliak2/server-rock-paper-scissors with PlatformIO extension. Plug into your computer your Arduino Nano board (board can be changed in platformio.ini file). Program the controller by pressing the upload button.

Оновлений файл README



***Рис.5.*** Успішно відпрацьований Action

Створив pull-запит з іменем task2 і дочекався апруву від викладача

Після того, як рев’ювер схвалив запит, переніс його у гілку розробки

**Висновок:**

Виконуючи дану лабораторну роботу я створив клієнт і сервер та налаштував комунікацію між ними використовуючи серіал порт. Також, я налаштував ci.yml для роботи з GitHub Actions в якому прописаний flow для запуску тестів і білду проектів. Останніми кроками, які я зробив стало додавання тегу та створення pull request, який після апруву змерджив.