# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**



# АВТОМАТИЗОВАНЕ ПРОЕКТУВАННЯ КОМП’ЮТЕРНИХ СИСТЕМ

Лабораторна робота №5

Автоматизація тестування клієнту та серверу.

Виконав: ст. гр. КІ-401

Гавриляк Д. В.

Прийняв: Федак П.Р.

Львів 2024

**Мета роботи:** Ознайомитись з написаням автоматизованих тестів з використанням власноруч створеного або існуючого фреймворку для тестування.

## ТЕОРЕТИЧНІ ВІДОМОСТІ

Тестування програмного забезпечення є важливою частиною процесу розробки, що дозволяє забезпечити стабільність та коректність роботи програм. У випадку з Arduino, де використовуються апаратні ресурси, тестування стає більш складним, тому важливо застосовувати інструменти, які дозволяють ефективно перевіряти як програмний, так і апаратний код.

У цій лабораторній роботі розглядається тестування коду для Arduino за допомогою бібліотеки **ArduinoUnit** та тестування веб-додатка, що використовує **TypeScript** і **React**, за допомогою **Vitest**.

# Тестування Arduino за допомогою ArduinoUnit

**ArduinoUnit** — це бібліотека для юніт-тестування коду для Arduino, що дозволяє перевіряти функціональність коду без необхідності завжди підключати реальні пристрої. ArduinoUnit використовує просту синтаксис для написання тестів, що дозволяє перевіряти поведінку функцій, класів і модулів, які взаємодіють з апаратними ресурсами.

Типові етапи тестування за допомогою ArduinoUnit:

1. **Написання тестів**: Створення тестів для перевірки поведінки коду, який буде виконуватися на Arduino. Це може включати перевірку значень змінних, результатів викликів функцій або взаємодії з периферійними пристроями.
2. **Виконання тестів**: Тести виконуються на реальному або емуляторі Arduino. ArduinoUnit забезпечує зручний механізм для запуску тестів та перевірки їхніх результатів.
3. **Інтерпретація результатів**: Оцінка результатів тестів, що дозволяє зрозуміти, чи працює код згідно з очікуваннями.

# Тестування TypeScript і React за допомогою Vitest

**Vitest** — це тестувальник для JavaScript/TypeScript, який надає можливість писати і виконувати юніт-тести для коду, що працює в середовищі браузера або серверного Node.js. Vitest підтримує модульне тестування, мокінг, асинхронні тести та багато іншого.

Процес тестування в **Vitest** виглядає наступним чином:

1. **Імпорт тестових бібліотек**: Для тестування React компонентів та TypeScript коду в проєкті використовуються бібліотеки, такі як @testing- library/react для компонентів і vitest для самих тестів.
2. **Написання тестів**: Тести включають перевірку функціональності компонентів (ререндеринг, події, стани), а також логіки, що реалізована в сервісах або утилітах.
3. **Виконання тестів**: За допомогою Vitest тести можуть бути запущені через команду з консолі. Усі тести виконуються в ізольованому середовищі, що забезпечує стабільність результатів.
4. **Мокінг і шпіонаж**: У випадку з інтеграційними тестами, мокінг сервісів або API є важливою частиною для контролю зовнішніх залежностей.

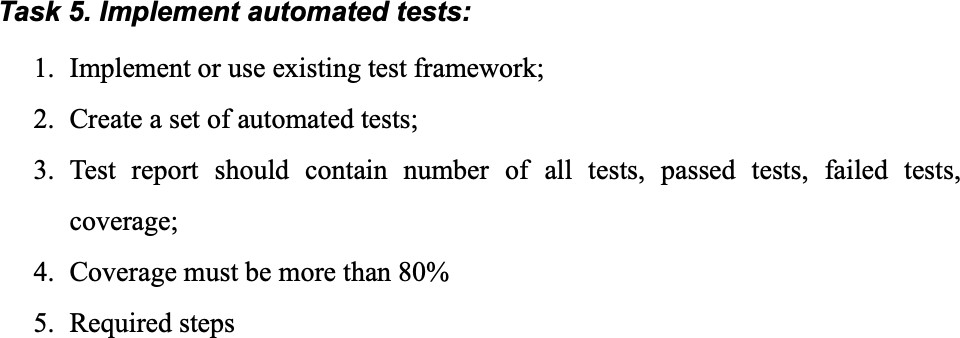
# Порівняння тестування Arduino і веб-застосунків

Основною відмінністю тестування Arduino та веб-додатків є взаємодія з апаратним забезпеченням у випадку з Arduino. У той час як у веб-додатках основним акцентом є логіка компонентів і взаємодія з API, тестування для Arduino часто включає роботу з реальними пристроями або їх емуляцією.

* + **Тестування на Arduino** передбачає більш близьке взаємодія з апаратними компонентами і часто потребує використання спеціальних бібліотек для емуляції поведінки апаратного забезпечення.
  + **Тестування для React** фокусується на взаємодії компонентів, станах і ефективності відображення UI. Тести можуть бути як юнітними (перевірка окремих компонентів), так і інтеграційними (перевірка цілої логіки додатка).

## ЗАВДАННЯ

1. Створити або використати фреймворк для тестування.
2. Створити набір автоматизованих тестів.
3. Звіт про тести має містити кількість усіх тестів, пройдених тестів, невдалих тестів та покриття.
4. Покриття тестів повинне бути більше 80%.
5. Обов’язкові кроки до кожної лабораторної.



## Виконання роботи

1. Для тестування Arduino обрав бібліотеку ArduinoUnit. Її потрібно встановити в Arduino IDE.



Рис.1. Обрана бібліотека для тестування Arduino програми

1. Для тестування клієнтського коду (TypeScript + React) використовую Vitest та дві додаткові бібліотеки для тестування TSX компонентів.

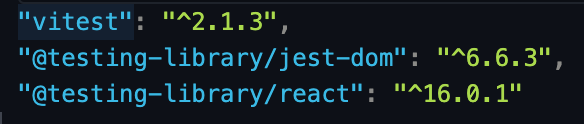


Рис.2. Обрані бібліотеки для тестування клієнтського коду

1. Написав тести для серверної частини. Протестував всі складові системи з усіма їхніми публічними методами. Для створення тестів використовується спеціальна конструкція *test(test\_name){ test\_body },* яка дуже схожа на звичайне оголошення функції.



Рис.3. Приклад тестів серверної частини

1. Під’єднав Arduino та запустив тестування. Бібліотека ArduinoUnit запускає тести на самій платі та виводить у серійний порт результати тестування.

Жодних додаткових дій окрім написання тестів не потрібно.

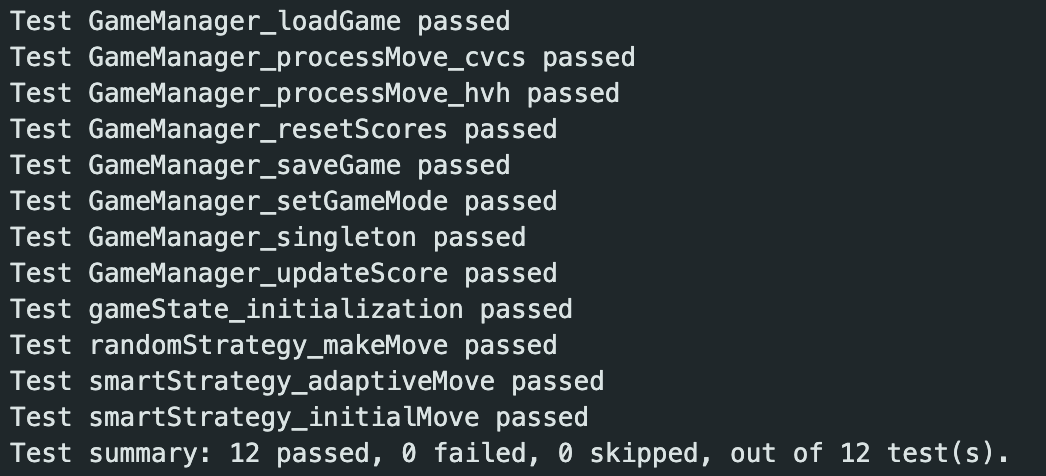


Рис.4. Підсумок тестування серверної частини.

1. Написав тести для клієнтської частини гри. Оскільки ми пишемо unit-тести, то протестувати в нашій програмі можливо все окрім основного компоненту, тому що свою логіку він інкапсулює, а також він є простою вхідною точкою програми, яка об’єднує всі компоненти для реалізації логіки гри. Написання тестів на Vitest відрізняється від написання тестів на Arduino. При використанні Vitest тести описуються спеціальними функціями, які названі так, що їх можна порівнювати зі звичайною людською мовою, цьому також допомагає сама структура виклику функцій (наприклад it("renders the Select with provided options", … expect(screen.getByText(option.label)).toBeInThe Document();). Така структура допомагає краще розуміти тест інтуїтивно. На рис. 5 продемонстрований приклад.



Рис.5. Приклад тестів для коду клієнтської частини.

1. Виконав тести командою npx vitest. В звіті про тестування можемо бачити кількість файлів з тестами та кількість самих тестів, а також скільки з них були пройдені успішно. У нашому випадку всі тести були пройдені успішно.

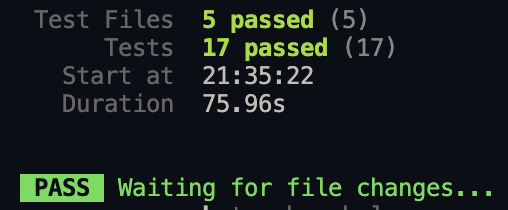


Рис.6. Результати тестування клієнтської частини

1. Аналіз покриття тестами коду. Для аналізу покриття тестами клієнтської частини встановив додатковий пакет Vitest Coverage. Для виведення інформації

про покриття разом з тестами, використовується команда npx vitest run -- coverage.

Результат:

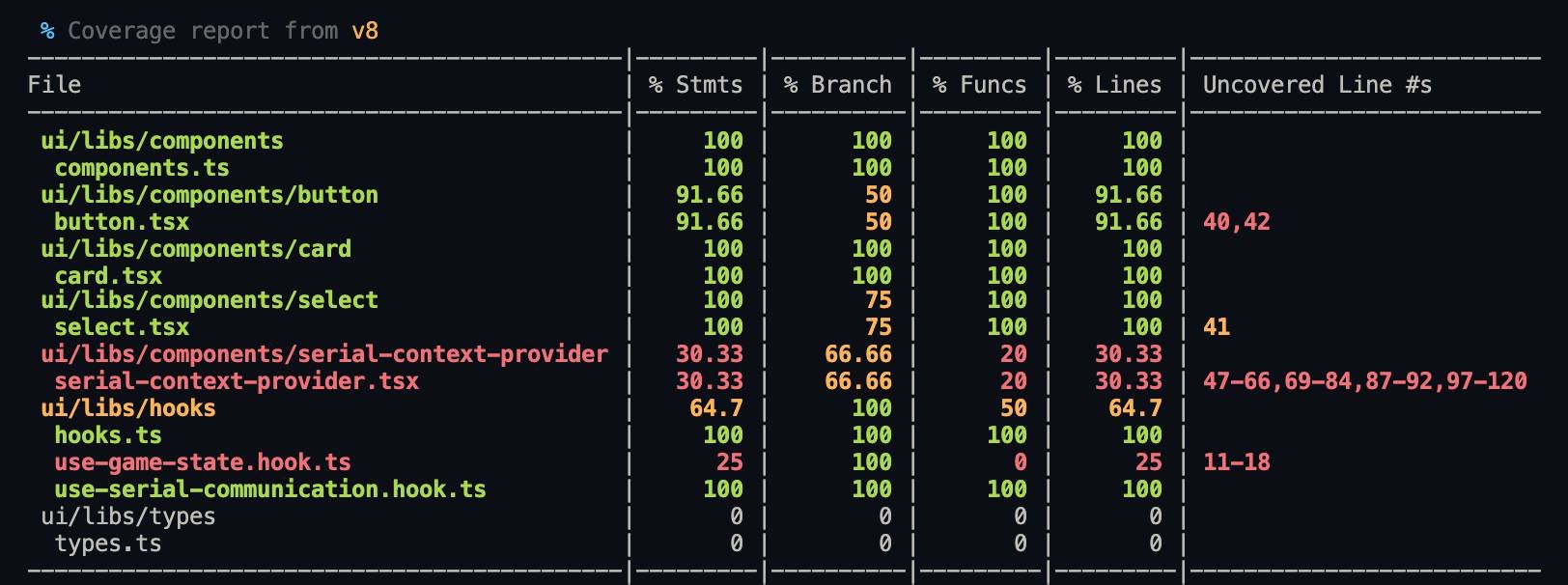


Рис.7. Результати тестування клієнтської частини

Для аналізу покриття тестами серверної частини, в бібліотеки ArduinoUnit немає спеціальних інструменів тому покриття будемо визначати вручну.

Оскільки тестами покриті всі методи розроблених класів, окрім основних методів Arduino **setup** i **loop** можемо зробити висновок що покрито більше ніж 90% коду.

Отже, весь код покритий тестами більше ніж на 80%, що відповідає вимогам до даної лабораторної роботи.

**Висновок**: У процесі виконання лабораторної роботи я набув практичного досвіду у тестуванні Arduino-коду з використанням бібліотеки ArduinoUnit та тестуванні веб-додатка на TypeScript і React за допомогою Vitest. Я ознайомився з принципами юніт-тестування для апаратних проектів та веб- розробки, а також навчився організовувати та проводити тести для перевірки коректності функціональності на різних платформах.